SQL Connector For Data Science

User Documentation (Version 1.0.1)

May 2025

Contents

Introduction to UnifyML

1. Enterprise Data Analytics Platform for Scale

UnifyML™ is a comprehensive unified data analytics platform designed for machine learning and artificial intelligence applications. This enterprise-grade solution enables organizations to execute advanced analytics at scale using SQL interfaces combined with industry-leading open source algorithms. The platform abstracts complex algorithmic implementations and underlying infrastructure complexities, allowing users to concentrate on solving critical business challenges and deriving actionable insights. Key features of the UnifyML platform include:

- Streamlined analytics platform utilizing SQL interfaces designed for business analysts and citizen data scientists
- Comprehensive support for both standalone and distributed multi-node cluster deployments
- Multi-tenant architecture supporting on-premises and public cloud deployments through Kubernetes orchestration
- Production-ready analytical workflows and domain-specific solutions
- Zero-code implementation approach for rapid deployment

2. Core Features of UnifyML Data Analytics Platform

The UnifyML Data Analytics Platform provides comprehensive capabilities designed to address enterprise-scale analytical requirements:

- SQL-based orchestration engine for executing popular open source machine learning and artificial intelligence algorithms at scale, including Apache Spark, Scikit-learn, Dask, Apache Flink, TensorFlow, and numerous other frameworks
- Seamless integration and federation of multiple heterogeneous data sources for comprehensive analytical processing
- Native connectivity to relational databases, NoSQL systems, and unstructured file formats
- Automated data preparation pipelines, intelligent insight discovery mechanisms, and collaborative insight sharing capabilities
- Comprehensive analytical capabilities spanning descriptive, predictive, and prescriptive analytics through SQL interfaces
- Production-grade analytical model lifecycle management including versioning, deployment, and monitoring
- Industry-specific analytical solutions and pre-built templates for accelerated implementation
- Universal compatibility with SQL client tools including DBeaver, Apache Zeppelin, and other JDBC-compliant applications for executing UnifyML SQL operations

II Installation

3. Server Setup and Configuration

3.1 Prerequisites and System Requirements

3.1.1 Docker Installation and Configuration

The UnifyML server deployment utilizes Docker containerization technology to deliver software components in portable, self-contained packages. Docker provides consistent deployment across diverse hardware and operating system environments. Download and install Docker software from the official Docker documentation portal:
`https://docs.docker.com/engine/install/`

3.1.2 Minimum System Requirements

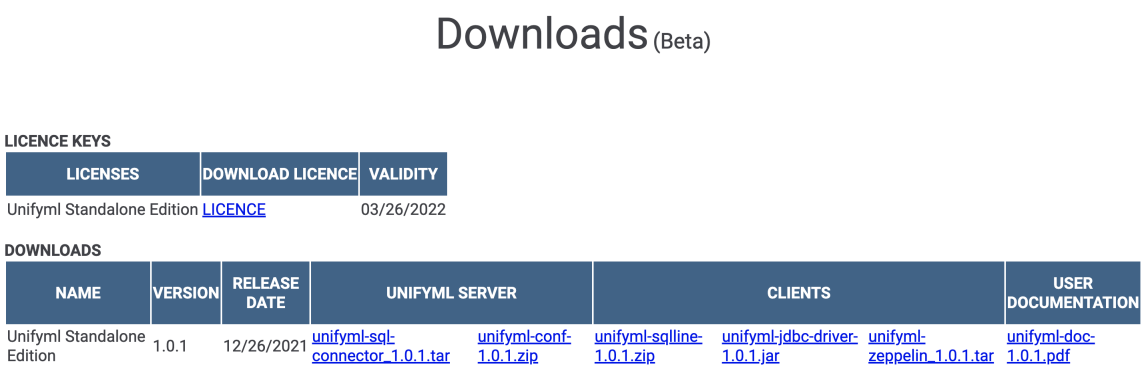
Docker containers require a minimum system configuration of 8GB RAM and 30GB local storage to ensure optimal performance when executing data science algorithms on datasets containing up to one billion records. These specifications provide adequate computational resources for enterprise-scale analytical workloads.

3.2 Standalone Edition Installation and Deployment

The UnifyML SQL connector server supports deployment on single server machines or desktop environments using Docker images. This deployment model enables UnifyML server execution across various hardware configurations and operating systems supported by Docker container runtime.

3.2.1 Docker Image Acquisition

Users can register for a complimentary account at www.unifyml.com to access and download the software distribution. After account creation and authentication, users can access the download section through the website portal to obtain the required server module components.



Downloads (Beta)

LICENCE KEYS

LICENSES	DOWNLOAD LICENCE	VALIDITY
Unifyml Standalone Edition	LICENCE	03/26/2022

DOWNLOADS

NAME	VERSION	RELEASE DATE	UNIFYML SERVER		CLIENTS			USER DOCUMENTATION
Unifyml Standalone Edition	1.0.1	12/26/2021	unifyml-sql-connector_1.0.1.tar	unifyml-conf-1.0.1.zip	unifyml-sqlline-1.0.1.zip	unifyml-jdbc-driver-1.0.1.jar	unifyml-zeppelin_1.0.1.tar	unifyml-doc-1.0.1.pdf

The following files must be downloaded from the website for server module deployment:

- `unifyml-sql-connector-1.0.1-ae1ee008.tar`
- `unifyml-conf-1.0.1-ae1ee008.zip`
- `unifyml-zeppelin-1.0.1-ae1ee008.tar`
- `LICENCE`

Alternatively, use command-line utilities such as wget from Linux, Windows, or macOS command interfaces to download files programmatically:

```
wget www.unifyml.com/unifyml-sql-connector-1.0.1-ae1ee008.tar
wget www.unifyml.com/unifyml-conf-1.0.1-ae1ee008.zip
wget www.unifyml.com/unifyml-zeppelin-1.0.1-ae1ee008.tar
```

After download completion, copy files to the /etc/datanalyt directory on your Docker host machine. Note that this directory path can be customized to any location on your Docker host system.

```
mkdir -p /etc/datanalyt
cp unifyml-sql-connector-1.0.1-ae1ee008.tar /etc/datanalyt
cp  unifyml-conf-1.0.1-ae1ee008.zip /etc/datanalyt
cp unifyml-zeppelin-1.0.1-ae1ee008.tar /etc/datanalyt
cp LICENCE /etc/datanalyt
```

3.2.2 Server Initialization and Startup

After acquiring the required UnifyML server modules and configuration files, load the Docker images and initialize the Docker containers following the procedures outlined below.

3.2.2.1 Configuration File Extraction

```
cd /etc/datanalyt
unzip unifyml-conf-1.0.1-ae1ee008.zip
cp /etc/datanalyt/LICENCE /etc/datanalyt/unifyml
```

3.2.2.2 Docker Image Loading

cd /etc/datanalyt
docker load -i `unifyml-sql-connector-1.0.1-ae1ee008.tar`
docker load -i `unifyml-zeppelin-1.0.1-ae1ee008.tar`

3.2.2.3 Docker Image Verification

docker images

```
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
unifyml-zeppelin        1.0.1     ae52dbc1a9be   18 hours ago    743MB
unifyml-sql-connector   1.0.1     177703d87fd5   18 hours ago    1.99GB
```

3.2.2.4 Container Deployment and Execution

docker run --privileged -it --cpus=".7" -d -p 4301:4301 -p 4302:4302 -v /etc/datanalyt/unifyml:/etc/datanalyt/unifyml/:rw unifyml-sql-connector:`1.0.1-ae1ee008`

docker run --privileged -it --cpus=".7" -d -p 5301:5301 -v /etc/datanalyt/unifyml:/etc/datanalyt/unifyml/:rw unifyml-zeppelin:`1.0.1-ae1ee008`

3.2.2.5 Container Status Verification

docker container list

```
CONTAINER ID   IMAGE                          COMMAND
0480880edcc0   unifyml-zeppelin:1.0.1         "/bin/sh -c "sh /opt..."
aa0340473ae3   unifyml-sql-connector:1.0.1    "/bin/sh -c "sh /opt..."
```

3.2.2.6 Server Startup Confirmation

grep "Started UnifymlServerApplication" /etc/datanalyt/unifyml/logs/Unifyml-spark.out
2022-02-08 18:47:00.668 INFO 24 — [main] c.d.u.UnifymlServerApplication : Started UnifymlServerApplication in 47.181 seconds (JVM running for 53.179)

4. Client Configuration and Setup

4.1 Client Software Acquisition

SQL client software facilitates the submission and execution of analytical SQL queries against the UnifyML platform. The UnifyML server provides comprehensive support for open JDBC drivers, enabling compatibility with any JDBC-compliant SQL client interface tools. The latest client software distributions are available from the UnifyML.com website following user authentication and accessing the download section for client utilities. UnifyML supports various client interfaces including command-line tools such as SQLLine, Calcite JDBC drivers, and web-based interfaces such as Apache Zeppelin for SQL query execution. The Zeppelin installation was completed during the server setup process and provides immediate web-based SQL client capabilities.

The following files are required for SQL client module deployment:

- `unifyml-sqlline-1.0.1-ae1ee008.tar`
- `unifyml-jdbc-driver-1.0.1-ae1ee008.zip`

Utilize command-line utilities such as wget from Linux, Windows, or macOS command interfaces for programmatic file download:

```
wget www.unifyml.com/unifyml-sqlline-1.0.1-ae1ee008.tar
wget www.unifyml.com/unifyml-jdbc-driver-1.0.1-ae1ee008.zip
```

4.2 UnifyML Client Connection Configuration

Client tools utilize JSON configuration files that specify database connection parameters and enable user connectivity to data sources for analytical query execution.

4.2.1 MySQL Database Connection Configuration

The following example demonstrates JSON configuration for MySQL database connectivity with UnifyML server host specification. The JSON schema requires JDBC username and password credentials for data source authentication:

```
{
  "version": "1.0",
  "defaultSchema": "MYDATABASE",
  "clientProperties": [
    {
       "unifymlHost":"unifyml-server-host-ip"
    }
  ],
  "schemas": [
    {
      "name": "mydatabase",
      "type": "custom",
      "factory": "org.apache.calcite.adapter.jdbc.JdbcSchema$Factory",
      "operand": {
        "jdbcDriver": "com.mysql.cj.jdbc.Driver",
        "jdbcUrl": "jdbc:mysql://mysql-server-host/testdb",
        "jdbcUser": "testuser",
        "jdbcPassword": "testuserpasswds"
      }
    }
  ]
}
```

4.2.2 Amazon Web Services S3 Connection Configuration

The following example demonstrates JSON configuration for AWS S3 bucket connectivity with UnifyML server integration. The JSON schema requires S3 access keys and secret keys for secure data source authentication:

```
{
  "version": "1.0",
  "defaultSchema": "MYDATABASE",
  "clientProperties": [
    {
      "unifymlHost":"unifyml-server-host-ip"
    }
  ],
  "schemas": [
    {
      "name": "mydatabase",
      "type": "custom",
      "factory": "com.unifyml.adapter.file.FileSchemaFactory",
      "operand": {
        "directory": "s3://s3_bucket_name",
        "s3_access_key": "s3_access_key",
        "s3_secret_key": "s3_secret_key"
      }
    }
  ]
}
```

4.2.3 Microsoft Azure Storage Connection Configuration

The following example demonstrates JSON configuration for Microsoft Azure storage connectivity with UnifyML server integration. The JSON schema requires Azure account name and account key credentials for secure data source authentication:

```
{
  "version": "1.0",
  "defaultSchema": "MYDATABASE",
  "clientProperties": [
    {
      "unifymlHost":"unifyml-server-host-ip"
    }
  ],
  "schemas": [
    {
      "name": "mydatabase",
      "type": "custom",
      "factory": "com.unifyml.adapter.file.FileSchemaFactory",
      "operand": {
        "directory":"az://azure_name",
  "Account_Name":"azure_account_name",
  "Account_Key":"azure_account_key"
      }
    }
  ]
}
```

4.2.4 Google Cloud Storage Connection Configuration

The following example demonstrates JSON configuration for Google Cloud Storage connectivity with UnifyML server integration. The JSON schema requires project ID and service account JSON credentials for secure data source authentication:

```
{
  "version": "1.0",
  "defaultSchema": "MYDATABASE",
  "clientProperties": [
    {
      "unifymlHost":"unifyml-server-host-ip"
    }
  ],
  "schemas": [
    {
      "name": "mydatabase",
      "type": "custom",
      "factory": "com.unifyml.adapter.file.FileSchemaFactory",
      "operand": {
        "directory": "gs_bucket_name",
        "Project_Id": "project_id",
        "Service_Account_Json": "service_account_json"
      }
    }
  ]
}
```

4.2.5 Local CSV File Connection Configuration

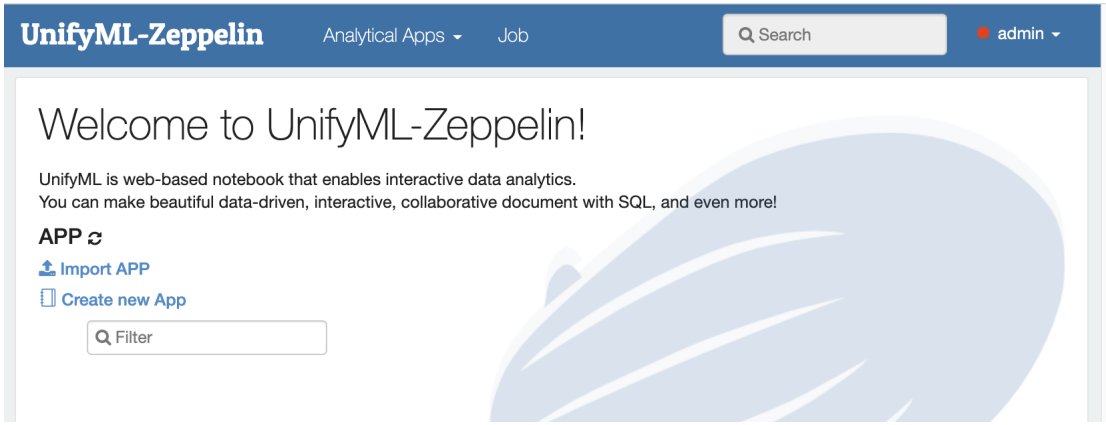
The following example demonstrates JSON configuration for local CSV file connectivity. The directory path specifies the folder location where users can place CSV files for analytical query execution:

```
{
  "version": "1.0",
  "defaultSchema": "MYDATABASE",
  "clientProperties": [
    {
      "unifymlHost":"unifyml-server-host-ip"
    }
  ],
  "schemas": [
    {
      "name": "mydatabase",
      "type": "custom",
      "factory": "com.unifyml.adapter.file.FileSchemaFactory",
      "operand": {
      "directory":"fs://"
      }
    }
  ]
}
```

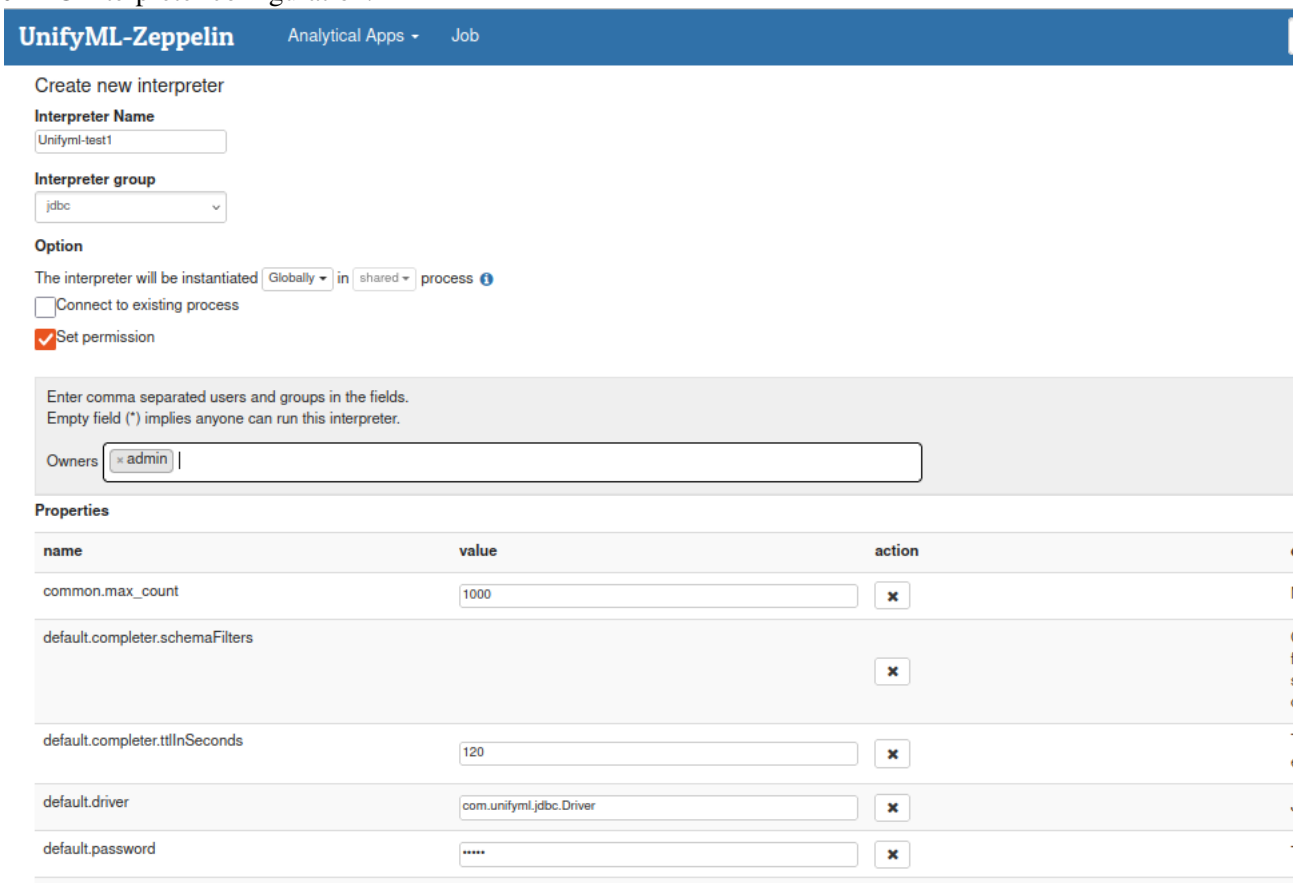
4.3 Web-Based Apache Zeppelin SQL Client

The Apache Zeppelin web-based SQL client interface provides comprehensive support for submitting UnifyML analytical SQL queries through an intuitive web interface. The Zeppelin Docker image was deployed during the UnifyML server setup process. The following steps outline the configuration and utilization of Zeppelin as a UnifyML SQL client:

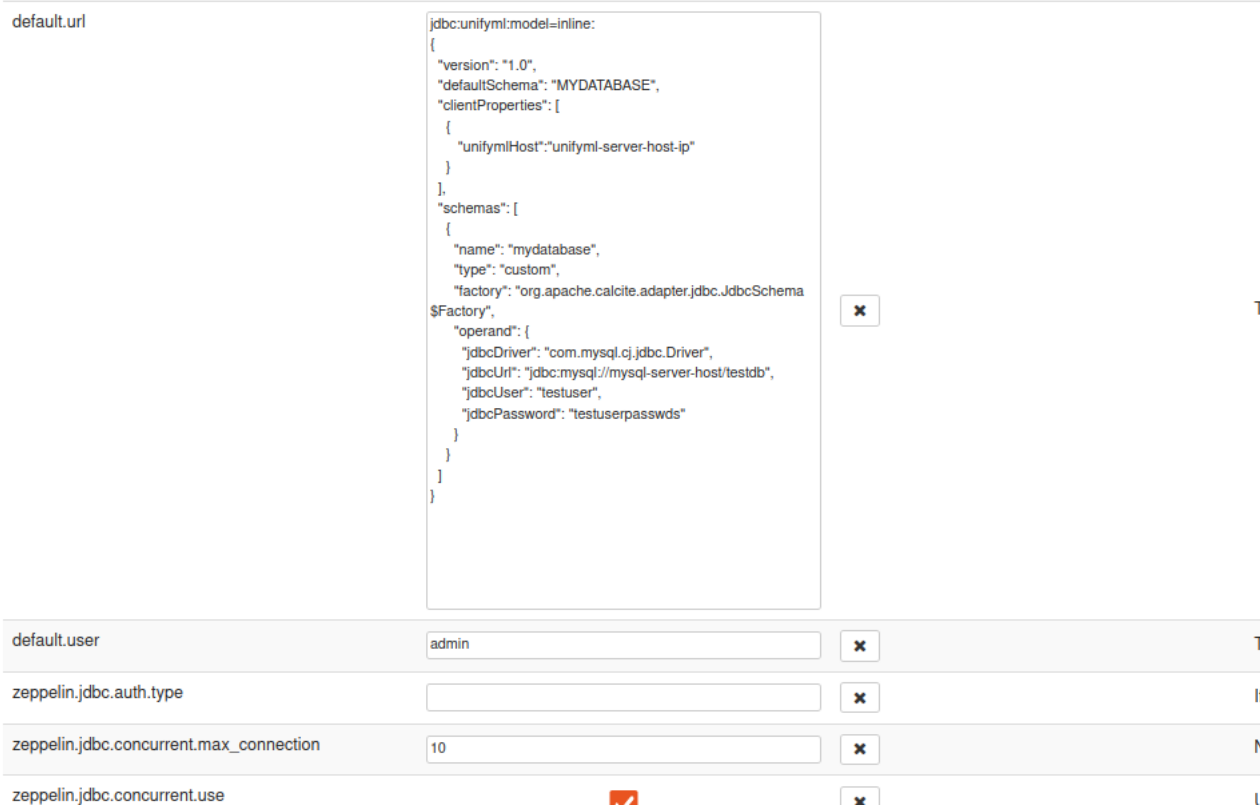
- Access the UnifyML server through your web browser using the URL: Unifyml-server-host-ip:5301. Port 5301 serves as the default listening port for the SQL web client interface. Use the default authentication credentials: username 'admin' and password 'admin'.



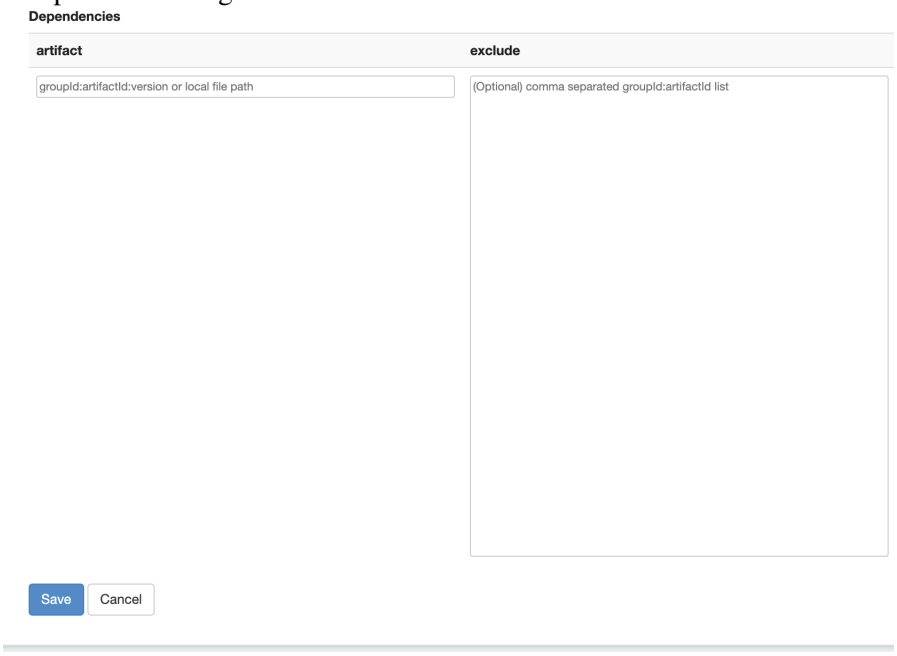
- Navigate to the admin user profile and access the interpreter menu settings to create a new JDBC interpreter configuration.



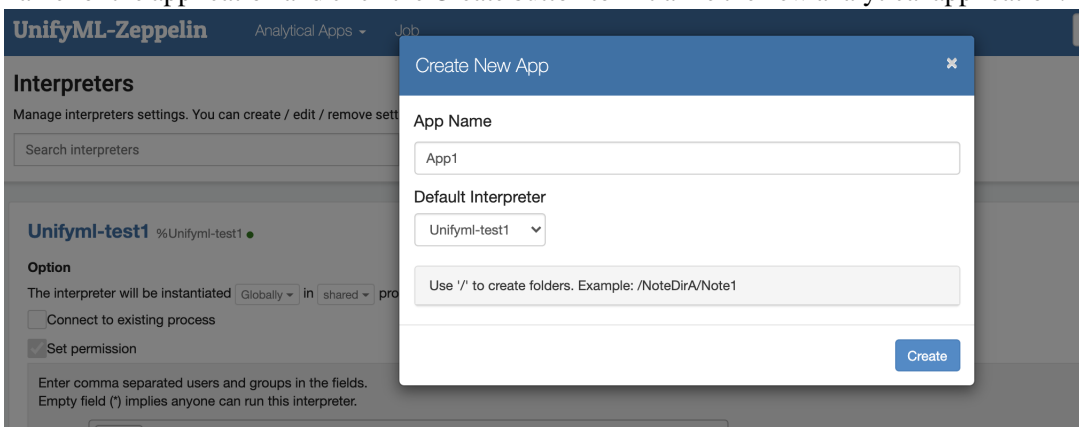
- Configure the default.url parameter with the appropriate connection details as specified in the database connection profiles above. Modify the driver class specifications according to your specific database or CSV file connection requirements. Enter 'admin' as the default password and click the Save button to persist the configuration.



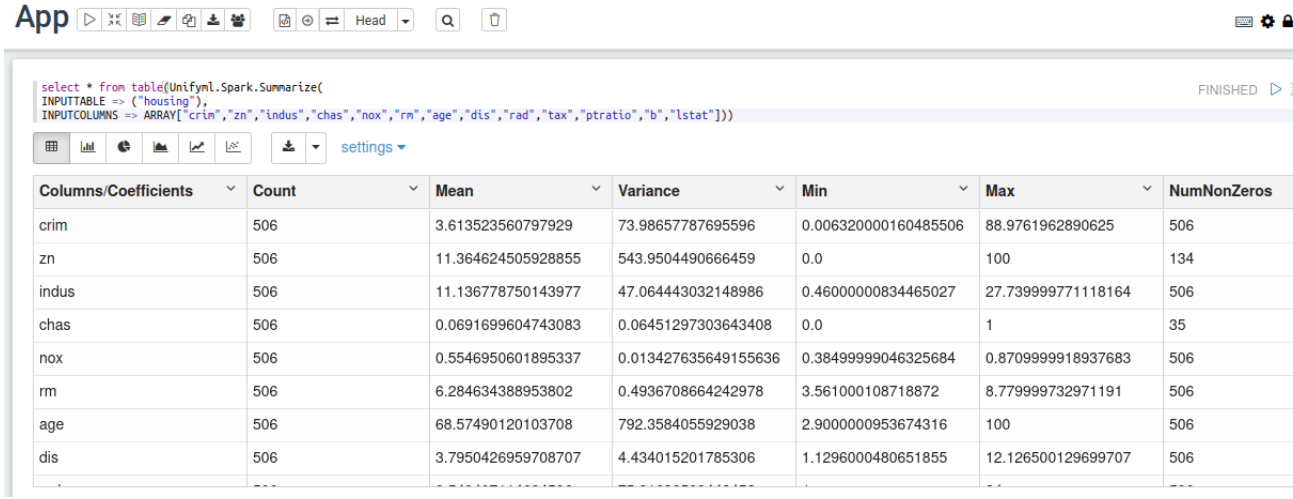
- Enter 'admin' as the default password for the admin username and click the Save button to complete the configuration.



- Navigate to the Analytical App menu and select 'Create New App'. Provide a descriptive name for the application and click the Create button to initialize the new analytical application.

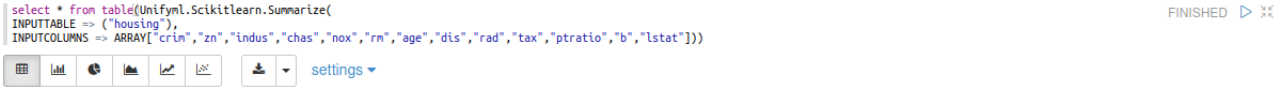


- Execute a sample Apache Spark summarize function using SQL syntax. The example demonstrates using a housing table with specified column inputs as function arguments for statistical summarization.



Columns/Coefficients	Count	Mean	Variance	Min	Max	NumNonZeros
crim	506	3.613523560797929	73.98657787695596	0.006320000160485506	88.9761962890625	506
zn	506	11.364624505928855	543.9504490666459	0.0	100	134
indus	506	11.136778750143977	47.064443032148986	0.46000000834465027	27.739999771118164	506
chas	506	0.0691699604743083	0.0645129730364340 8	0.0	1	35
nox	506	0.5546950601895337	0.013427635649155636	0.38499999046325684	0.8709999918937683	506
rm	506	6.284634388953802	0.4936708664242978	3.561000108718872	8.779999732971191	506
age	506	68.57490120103708	792.3584055929038	2.9000000953674316	100	506
dis	506	3.7950426959708707	4.434015201785306	1.1296000480651855	12.126500129699707	506

- Execute a sample Scikit-learn summarize function using SQL syntax. The example demonstrates using a housing table with specified column inputs as function arguments for comprehensive data summarization.

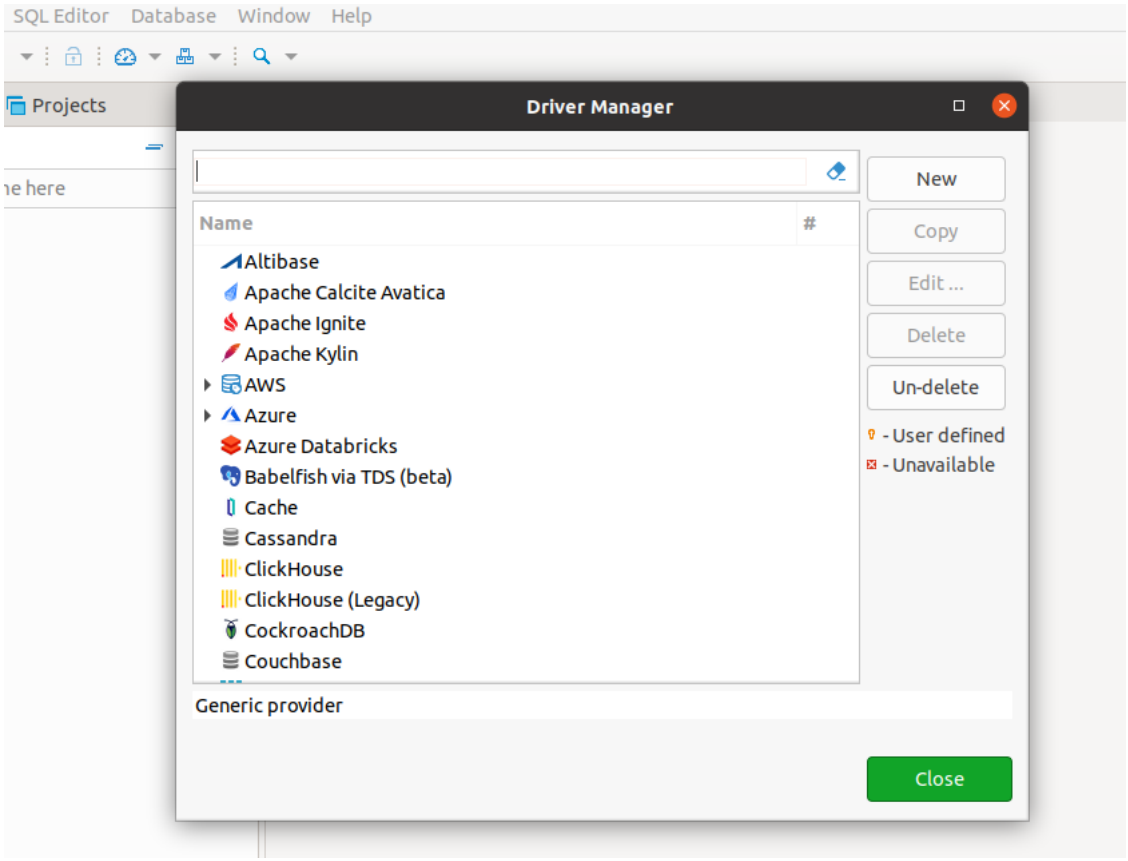


Columns/Coefficients	count	mean	std	min	25%	50%	75%
crim	506	3.613523557312254	8.60154510533249	0.00632	0.0820449999999999	0.25651	3.6770825
zn	506	11.364624505928854	23.3227453158209	0.0	0.0	0.0	12.5
indus	506	11.13677865612648	6.860352940897585	0.46	5.19	9.69	18.1
chas	506	0.0691699604743083	0.2539940413404103 7	0.0	0.0	0.0	0.0
nox	506	0.5546950592885376	0.1158776756675559 5	0.385	0.449	0.538	0.624
rm	506	6.284634387351779	0.7026171434153233	3.561	5.8854999999999995	6.2085	6.6235

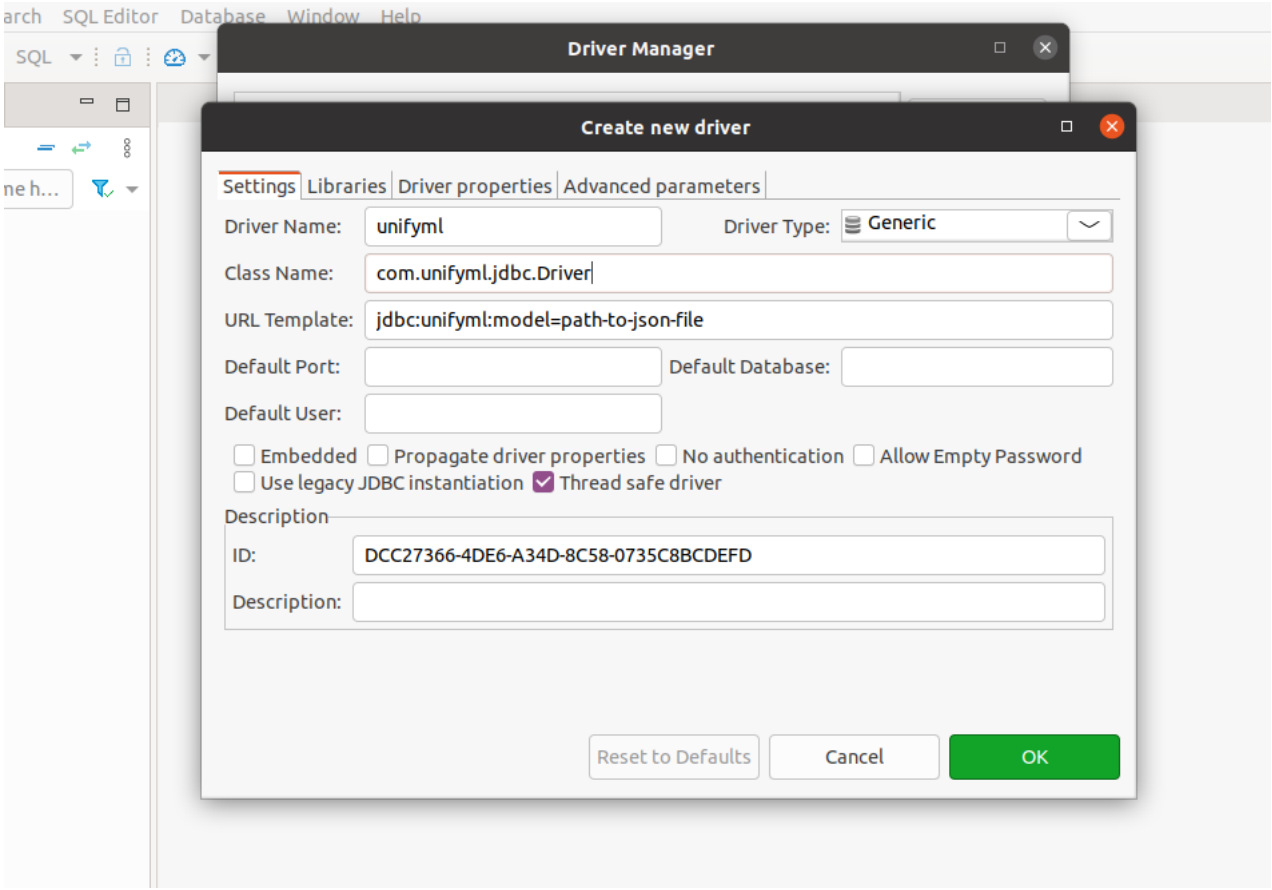
4.4 Web-Based DBeaver SQL Client Integration

DBeaver provides a comprehensive SQL client interface for submitting UnifyML analytical SQL queries through a feature-rich desktop application. The following steps outline the configuration and utilization of DBeaver as a UnifyML SQL client:

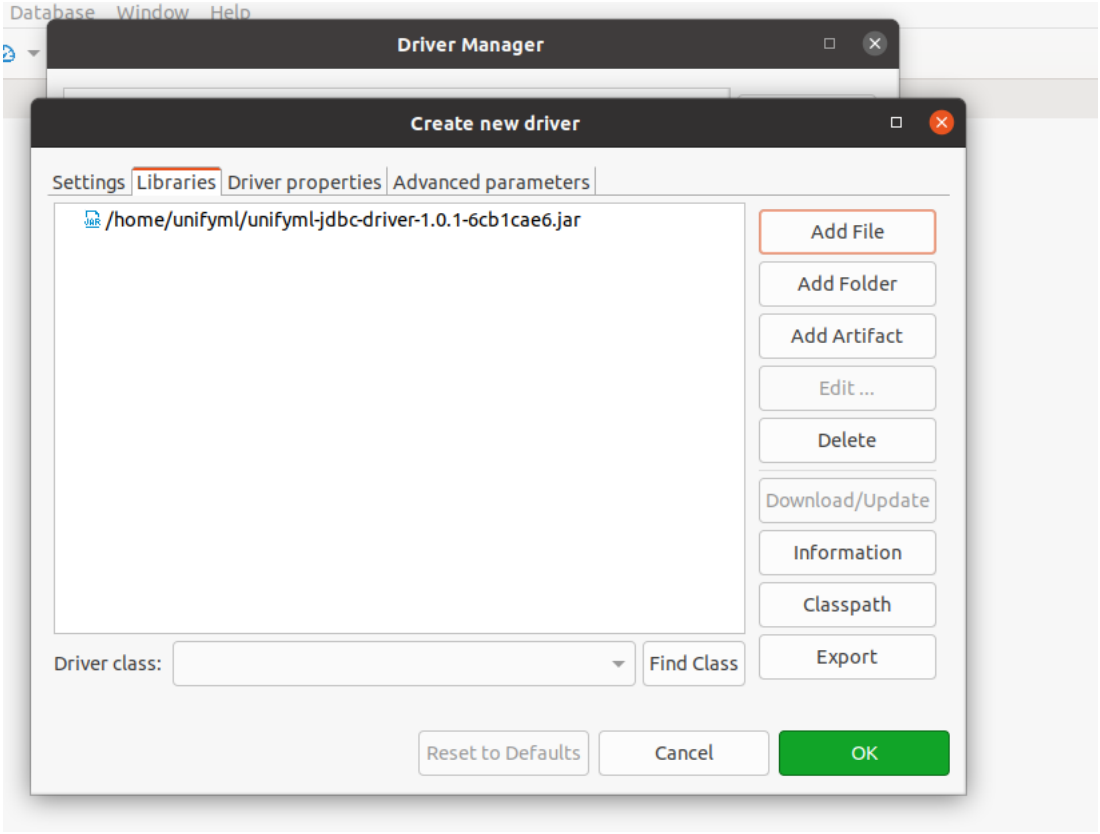
- Launch the DBeaver application on your desktop or laptop system. After application initialization, navigate to the Database menu and select Driver Manager to configure the UnifyML database driver.
- Click 'New' to create a new driver configuration and specify the driver name, class name, and URL template parameters.



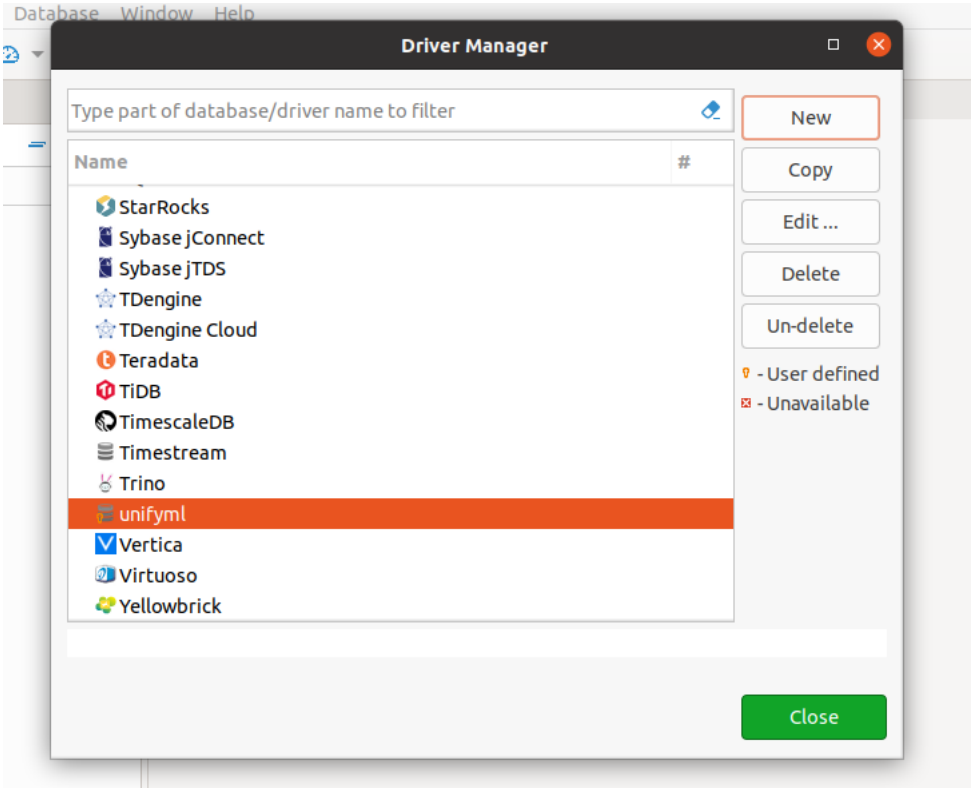
- Configure the following driver parameters:
- Class Name: com.unifyml.jdbc.Driver
- URL Template: jdbc:unifyml:model=path-to-json-file



- Navigate to the Libraries tab and click 'Add Files' to select the unifyml-jdbc-driver JAR file, then click OK to create the UnifyML driver configuration.

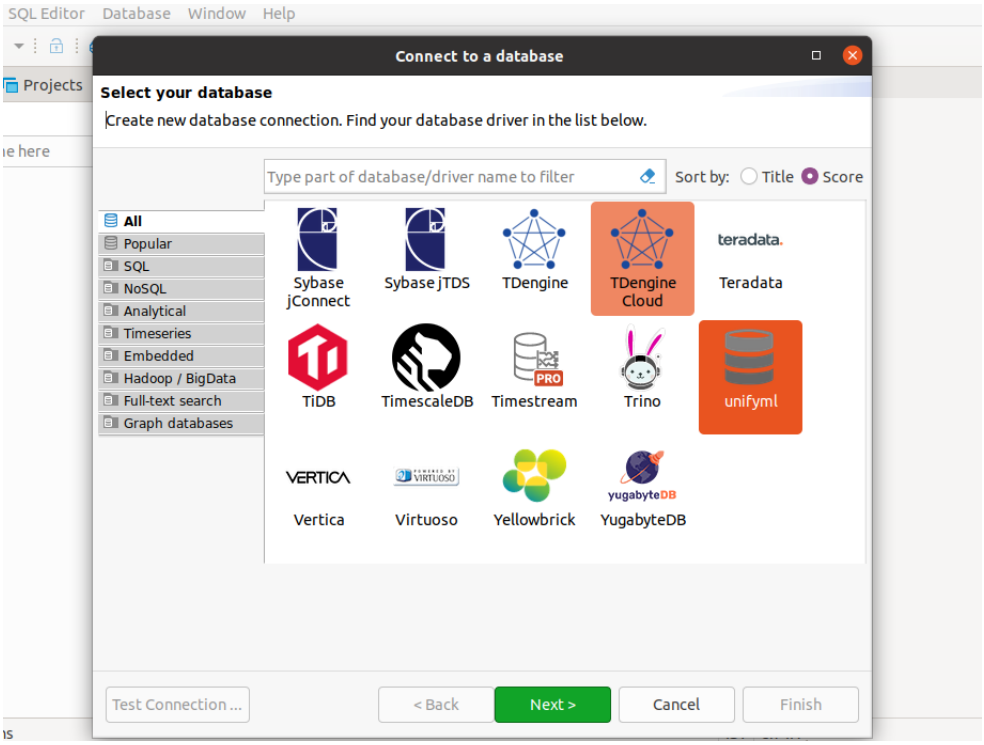


- The UnifyML driver configuration is now successfully created and available for use.

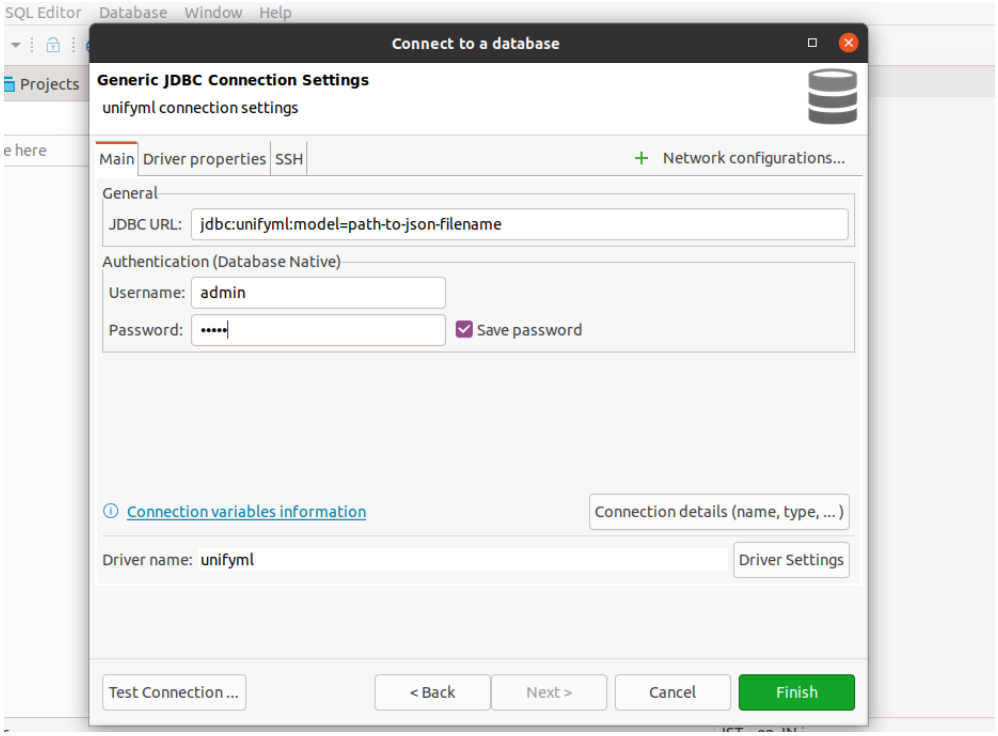


- Navigate to the Database menu and select 'New Database Connection' to establish a connection using the UnifyML driver.

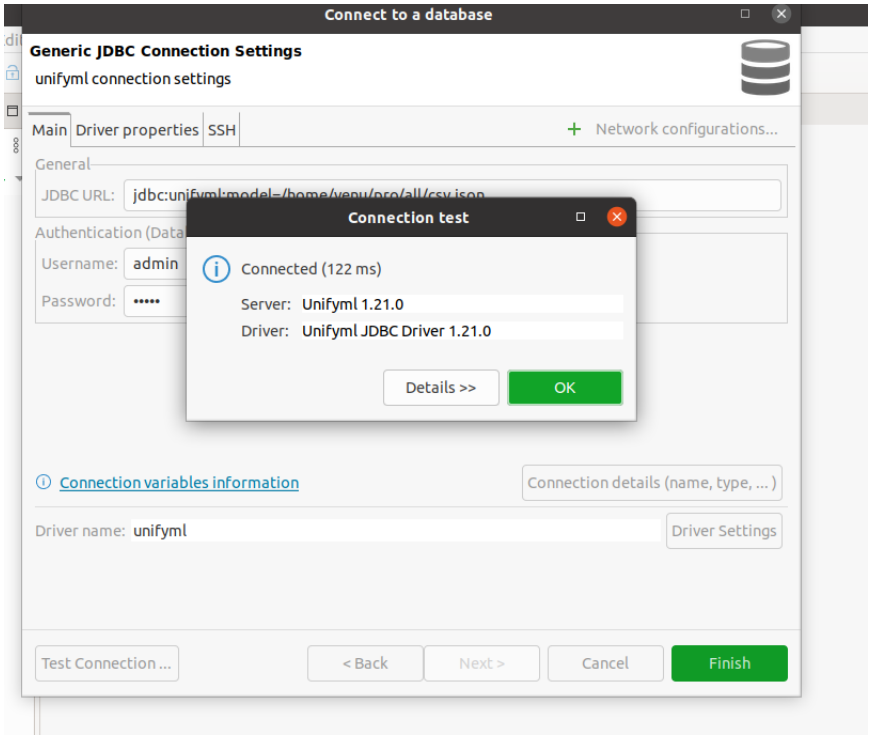
- Create a new database connection by locating and selecting your configured UnifyML driver from the available driver list.



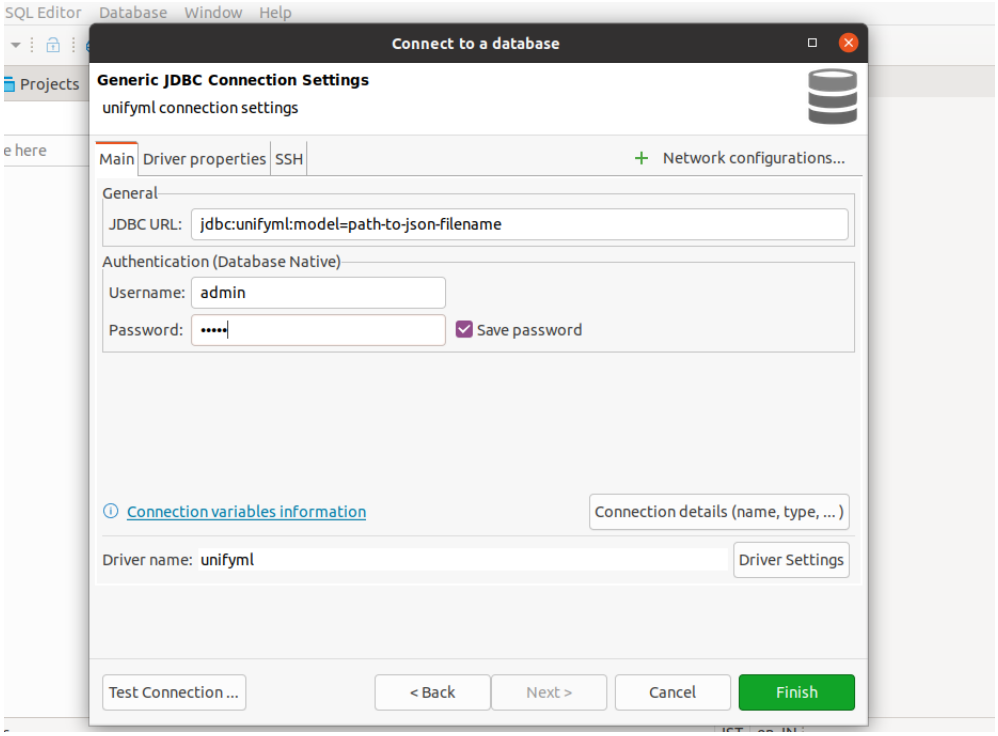
- Click Next and provide the JDBC URL, username, and password credentials. Use 'admin' as both the default username and password. Click 'Test Connection' to verify successful connectivity.



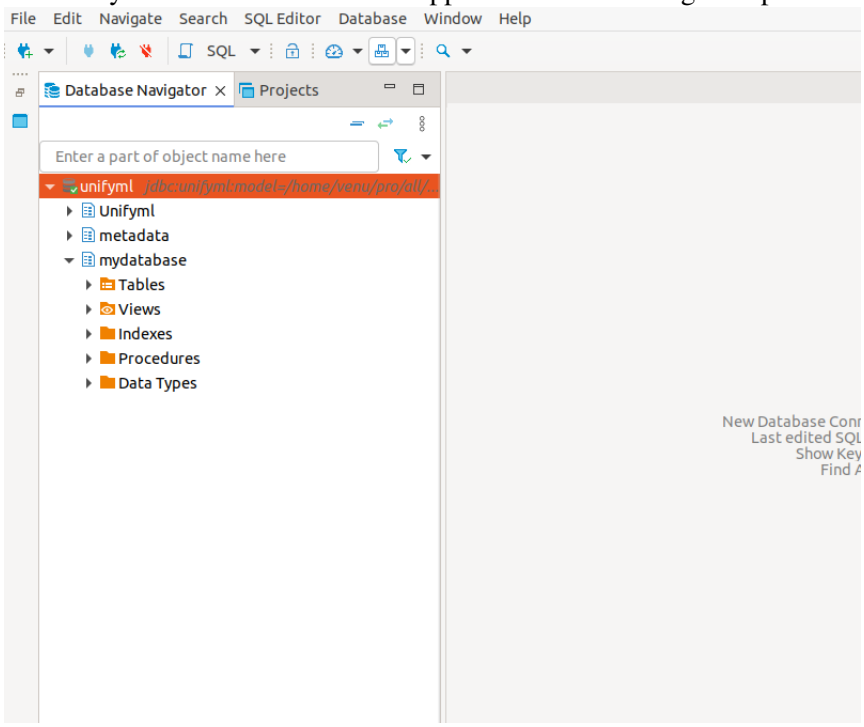
If the connection is established successfully, click OK to proceed.



- Click Finish to complete the connection setup process.



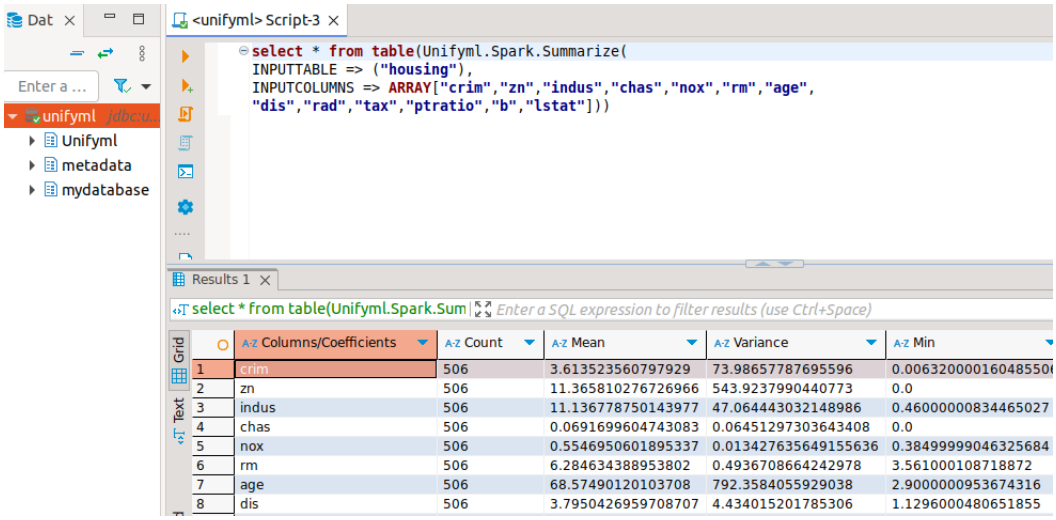
- The UnifyML database connection appears in the left navigation panel and is ready for use.



- Execute a sample Apache Spark summarize function using SQL syntax. The example demonstrates using a housing table with specified column inputs as function arguments for comprehensive statistical analysis.

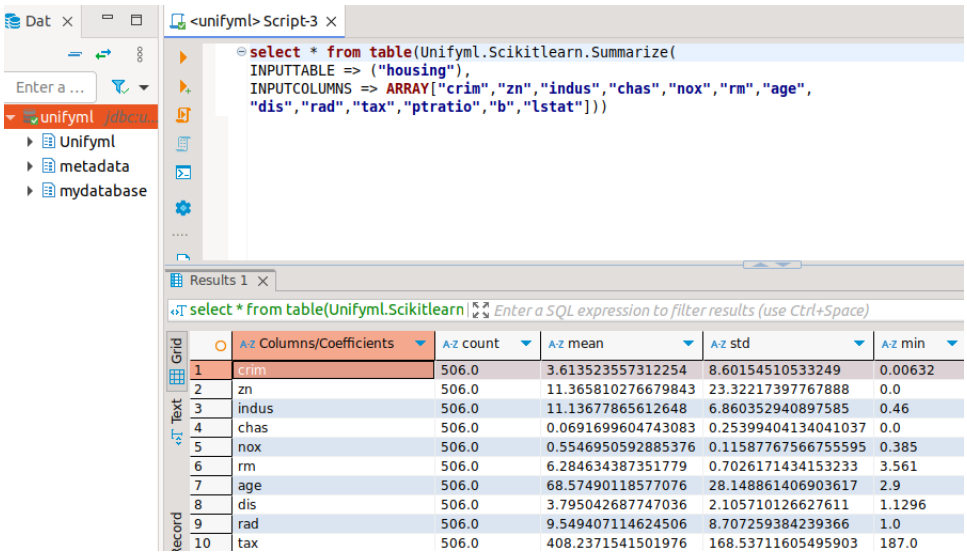
- Apache Spark SQL Query Example:
  select * from table(Unifyml.Spark.Summarize(INPUTTABLE => ("housing"), INPUT-

COLUMNS => ARRAY["crim", "zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat"]))



- Execute a sample Scikit-learn summarize function using SQL syntax. The example demonstrates using a housing table with specified column inputs as function arguments for advanced data summarization.

- Scikit-learn SQL Query Example:
  select * from table(Unifyml.Scikitlearn.Summarize(INPUTTABLE => ("housing"), INPUTCOLUMNS => ARRAY["crim", "zn", "indus", "chas", "nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat"]))



4.5 Command-Line Interface

4.5.1 SQLLine Command-Line Tool

The UnifyML SQLLine is a cross-platform command-line interface tool compatible with Linux, Windows, and macOS systems. This tool enables connectivity to various data sources including

relational databases and unstructured files for executing analytical SQL queries.

4.5.1.1 Linux and macOS System Configuration

```
unzip unifyml-sqlline-1.0.1.zip -d sqlline
cd sqlline/
chmod 777 ./sqlline
./sqlline
sqlline version 1.8.0
sqlline> !connect jdbc:unifyml:model=./mysql.json admin admin
sqlline> !q
```

4.5.1.2 Windows System Configuration

```
unzip unifyml-sqlline-1.0.1.zip -d sqlline
cd sqlline/
sqlline.bat
sqlline version 1.8.0
sqlline> !connect jdbc:unifyml:model=./csv.json admin admin
sqlline> !q
```

4.5.2 JDBC Drivers

The UnifyML JDBC drivers provide comprehensive connectivity solutions enabling users to establish connections to various databases and unstructured file systems using Java applications, facilitating direct execution of analytical queries. The following code example demonstrates JDBC driver connectivity implementation:

```
Sample.java:

import java.sql.*;
Class.forName(" com.unifyml.jdbc.Driver");
Connection jdbccon = DriverManager.getConnection(
                "jdbc:unifyml:model=mysql.json", "admin","admin");
Statement stmt=con.createStatement();
String sql =
"select * from table(Unifyml.Spark.DecisionTreeClassifierModel(
INPUTTABLE => ("diabetes"),
INPUTCOLUMNS => ARRAY["nooftimespregent", "plasmaglucose",
"bloodpressure", "skinthickness", "seruminsuline", "bodymassindex",
"pedigreefunction", "age"],
TARGETCOLUMN => ("class") ));";

stmt.executeQuery(sql);
```

```
javac Sample.java
java -cp "unifyml-jdbc-driver-1.0.1.jar:." Sample
```

III Reference Datasets

4.6 Regression Datasets

Regression datasets comprise structured collections of data specifically designed for training and evaluating regression models. In regression analysis, the primary objective is to predict continuous target variables based on one or more input features, enabling quantitative forecasting and trend analysis. The following section provides detailed descriptions of representative regression datasets.

4.6.1 Housing Dataset

The Housing dataset (exemplified by the Boston Housing dataset) contains comprehensive features describing various aspects of residential properties and their surrounding environmental factors. Each column represents specific characteristics that influence housing valuations:

CRIM	Per capita crime rate by town. **DataType**: DOUBLE
ZN	Proportion of residential land zoned for lots over 25,000 sq.ft. **DataType**: DOUBLE
INDUS	Proportion of non-retail business acres per town. **DataType**: DOUBLE
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise). **DataType**: INTEGER
NOX	Nitric oxides concentration (parts per 10 million). **DataType**: DOUBLE
RM	Average number of rooms per dwelling. **DataType**: DOUBLE
AGE	Proportion of owner-occupied units built prior to 1940. **DataType**: DOUBLE
DIS	Weighted distances to five Boston employment centres. **DataType**: DOUBLE
RAD	Index of accessibility to radial highways. **DataType**: DOUBLE
TAX	Full-value property-tax rate per dollar 10,000. **DataType**: DOUBLE
PTRATIO	Pupil-teacher ratio by town. **DataType**: DOUBLE
B	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town. **DataType**: DOUBLE
LSTAT	Percentage lower status of the population. **DataType**: DOUBLE
MEDV	Median value of owner-occupied homes in thousands of dollars. (This is the target variable for prediction). **DataType**: DOUBLE

4.6.2 Wine Quality Dataset

The Wine Quality Dataset is extensively utilized for both regression and classification tasks in machine learning applications. This dataset contains comprehensive information about physico-chemical properties of wine and their correlation with quality assessments.

The primary objective is to predict wine quality scores (target variable) based on chemical composition and physical properties.

FIXEDACIDITY	Amount of non-volatile acids in wine (affects tartness and stability). **DataType**: DOUBLE
VOLATILEACIDITY	Amount of acetic acid (high levels produce vinegar taste). **DataType**: DOUBLE
CITRICACID	Natural acid in wine that contributes freshness and flavor complexity. **DataType**: DOUBLE
RESIDUALSUGAR	Remaining sugar after fermentation (determines sweetness level). **DataType**: DOUBLE
CHLORIDES	Salt content in wine (excessive levels negatively impact quality). **DataType**: DOUBLE
FREESULFUR	Free sulfur dioxide that protects wine from spoilage and oxidation. **DataType**: DOUBLE
TOTALSULFUR	Total sulfur dioxide content (affects preservation and taste profile). **DataType**: DOUBLE
DENSITY	Mass per volume ratio (correlated with alcohol and sugar content). **DataType**: DOUBLE
PH	Acidity level of the wine (lower pH indicates higher acidity). **DataType**: DOUBLE
SULPHATES	Contributes to bitterness and preservative properties of wine. **DataType**: DOUBLE
ALCOHOL	Alcohol percentage by volume (higher levels often correlate with quality). **DataType**: DOUBLE
QUALITY	Wine quality score (0-10), assigned by expert sommelier evaluations (target variable). **DataType**: DOUBLE

4.7 Classification Datasets

Classification datasets are structured collections used to train machine learning models for predicting categorical labels rather than continuous values. The objective is to classify data points into predefined categories or classes based on feature patterns and relationships.

4.7.1 Banknote Authentication Dataset

The Banknote Authentication Dataset is designed for binary classification tasks to distinguish genuine banknotes from counterfeit ones based on statistical features extracted from digital image analysis.

VARIANCE	Measures the spread and variability of pixel intensity values. **DataType**: DOUBLE
SKEWNESS	Measures the asymmetry of pixel intensity distribution. **DataType**: DOUBLE
CURTOSIS	Measures the sharpness and peakedness of the distribution. **DataType**: DOUBLE
ENTROPY	Measures randomness and information content in the image. **DataType**: DOUBLE
CLASS	Binary classification: 0 (counterfeit), 1 (genuine) (Target variable). **DataType**: INTEGER

4.7.2 German Credit Risk Dataset

The German Credit Dataset is a widely-utilized financial dataset for predicting creditworthiness and assessing default risk based on comprehensive personal and financial factors. This dataset is

extensively used in credit scoring models and risk assessment applications.

ACCOUNTSTATUS	Describes the current status of the applicant's checking account balance. **DataType**: STRING
DURATION	The requested loan duration period (in months). **DataType**: INTEGER
CREDITHISTORY	The applicant's credit repayment history and reliability. **DataType**: STRING
PURPOSE	The stated purpose for the loan application. **DataType**: STRING
CREDITAMOUNT	The total loan amount requested by the applicant. **DataType**: INTEGER
BONDS	Savings account or bonds available as collateral security. **DataType**: STRING
EMPLOYEMENTSINCE	Duration of current employment (employment stability indicator). **DataType**: STRING
INSTALLMENTRATE	Percentage of disposable income allocated to loan repayment. **DataType**: INTEGER
PERSONALSTATUS	Personal status and gender information. **DataType**: STRING
GUARANTORS	Presence of co-applicants or guarantors for the loan. **DataType**: STRING
AGE	Age of the loan applicant (in years). **DataType**: INTEGER
RESIDENCESINCE	Duration of residence at current address (stability indicator). **DataType**: STRING
TELEXIST	Indicates whether the applicant has a telephone (contact reliability). **DataType**: INTEGER
PROPERTY	Description of the applicant's property ownership type. **DataType**: STRING
INSTALLMENTPLANS	Existing installment plans with other financial institutions. **DataType**: STRING
EXTCREDIT	Number of existing credit accounts at other banks. **DataType**: INTEGER
HOUSING	Housing situation (own, rent, or free accommodation). **DataType**: STRING
NOOFEXISTINGCREDIT	Number of existing credit accounts with this bank. **DataType**: INTEGER
NOOFPEOPLE	Number of people financially dependent on the applicant. **DataType**: STRING
TELEPHONE	Telephone availability (reliability and contactability indicator). **DataType**: STRING
CLASS	Credit risk classification: good or bad credit risk (target variable). **DataType**: INTEGER

4.7.3 Thyroid Disease Dataset

The Thyroid Disease Dataset is designed for predicting thyroid dysfunction based on comprehensive medical diagnostic features. This dataset contains attributes derived from blood tests, medical examinations, and patient characteristics. The objective is to predict thyroid-related medical conditions based on clinical measurements and laboratory results.

T3RESIN	Triiodothyronine (T3) resin uptake test results. **DataType**: INTEGER
THYROXIN	Thyroxine (T4) hormone level in blood serum. **DataType**: DOUBLE
TRIIODOTHYRONINE	Triiodothyronine (T3) hormone level affecting metabolism and energy. **DataType**: DOUBLE
TSH	Thyroid Stimulating Hormone produced by pituitary gland. **DataType**: DOUBLE
MAXTSH	Maximum recorded TSH level during testing period. **DataType**: INTEGER
CLASS	Thyroid disease classification (presence or absence of thyroid dysfunction). **DataType**: INTEGER

4.7.4 Indian Diabetes Dataset

The Indian Diabetes Dataset is specifically designed for predicting Type 2 Diabetes occurrence based on medical attributes and lifestyle factors. This dataset contains features related to medical history, diagnostic test results, and physical characteristics relevant for diabetes diagnosis.

NOOFTIMESPREGENT	Number of pregnancies (relevant for gestational diabetes risk). **DataType**: INTEGER
PLASMAGLUCOSE	Plasma glucose concentration after 2-hour oral glucose tolerance test (mg/dL). **DataType**: INTEGER
BLOODPRESSURE	Diastolic blood pressure measurement (mm Hg). **DataType**: INTEGER
SKINTHICKNESS	Triceps skinfold thickness measurement (mm) as body fat indicator. **DataType**: INTEGER
SERUMINSULINE	Serum insulin level measurement (mu U/mL). **DataType**: INTEGER
BODYMASSINDEX	Body Mass Index calculated from weight and height (kg/m²). **DataType**: DOUBLE
PEDIGREEFUNCTION	Diabetes pedigree function representing genetic predisposition. **DataType**: DOUBLE
AGE	Age of the individual (in years). **DataType**: INTEGER
CLASS	Diabetes classification: 0 (non-diabetic), 1 (diabetic) (target variable). **DataType**: INTEGER

4.8 Clustering Datasets

Clustering datasets are utilized in unsupervised learning tasks where the objective is to identify natural groupings or patterns within data without predefined class labels. Unlike supervised classification, clustering focuses on discovering inherent data structures and relationships based on feature similarity and distance metrics.

4.8.1 E-commerce Shopping Dataset

The E-commerce Shopping Dataset is designed to analyze customer behavior patterns, transaction details, and website interaction metrics within online retail environments. This dataset provides comprehensive insights into customer purchasing patterns and digital engagement metrics across e-commerce platforms:

YEAR	Calendar year when the transaction occurred. **DataType**: INTEGER
MONTH	Calendar month when the transaction was completed. **DataType**: INTEGER
DAY	Calendar day when the purchase transaction occurred. **DataType**: INTEGER
PURCHASEORDER	Unique order identifier for each completed transaction. **DataType**: INTEGER
COUNTRY	Country code or identifier for customer location. **DataType**: INTEGER
SESSIONID	Unique session identifier for website visit tracking. **DataType**: INTEGER
PAGEMAINCAT	Primary product category being browsed by the customer. **DataType**: INTEGER
PAGECLOTHING	Clothing subcategory specification (if applicable). **DataType**: STRING
COLOR	Product color attribute for browsed or purchased items. **DataType**: INTEGER
LOCATION	Geographical region or specific location within country. **DataType**: INTEGER
MODELPHOTO	Product image or model photo identifier. **DataType**: INTEGER
PRICE	Standard price of the purchased item. **DataType**: INTEGER
PRICETWO	Alternative price (discounted or promotional price). **DataType**: INTEGER
PAGE	Specific webpage identifier during customer session. **DataType**: INTEGER

IV DataCache

5. Introduction

DataCache functionality enables efficient storage of data tables and views from various data sources into optimized Parquet file format. For applications requiring repeated data access from external data sources such as databases, utilizing DataCache SQL functions significantly improves performance by storing data locally and reducing network traffic. This approach eliminates redundant data transfers as cached data is stored on local disks where the UnifyML server operates.

6. UnifyML Framework

This section provides comprehensive details about UnifyML DataCache functionality using the native UnifyML framework.

6.1 DataCache

6.1.1 Introduction

UnifyML DataCache refers to the systematic storage of data in temporary cache storage to enhance performance by minimizing data access latency for frequently accessed datasets. In the context of Parquet file storage, this involves persisting structured data in a columnar storage format optimized for performance and efficiency in big data environments. The UnifyML cache utilizes local disk storage on the UnifyML server infrastructure for optimal performance.

6.1.2 Syntax

The following syntax demonstrates the UnifyML DataCache algorithm implementation:

```
SELECT * FROM table(Unifyml.DataCache(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  OUTPUTCACHEID => (outputCacheid) [,...]
  [ PARTITIONCOLUMN => ( partitionColumn )] [,...]
  [ INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => (numPartition) ] [,...]
  [ CACHETYPE => ( cacheType ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

6.1.3 Input Arguments

The following table describes the input parameters for the UnifyML DataCache algorithm:

INPUTTABLE	Specifies the name of the source table containing the data columns. **DataType**: STRING
OUTPUTCACHEID	Unique identifier for the output cache storage. **DataType**: STRING
PARTITIONCOLUMN	[Optional] Column-based partitioning specification for optimized storage. **DataType**: STRING
INPUTCOLUMNS	[Optional] Specifies specific table columns for caching. **DataType**: ARRAY
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed storage. **DataType**: INTEGER
CACHETYPE	[Optional] Cache storage type specification. **DataType**: STRING, **Default**: diskonly
BYTESPERCHUNK	[Optional] Data chunk size in bytes for storage optimization. **DataType**: INTEGER, **Default**: 64000000

6.1.4 Output

The following table describes the expected output structure of the UnifyML DataCache algorithm:

INPUTTABLE	Name of the source table containing the cached columns. **DataType**:STRING
INPUTCOLUMNS	Input columns of the table (if specifically specified). **DataType**:ARRAY
DATACACHED	Boolean status indicator of data caching operation. **DataType**:BOOLEAN

6.1.5 Examples

The following example demonstrates the execution of the UnifyML DataCache SQL function:

```
select * from table(Unifyml.DataCache(
INPUTTABLE => ("housing"),
NUMPARTITION => 5,
PARTITIONCOLUMN => ("page"),
OUTPUTCACHEID => ("datacacheid")))

SQL Results:
+----------------+----------+
| InputTableName |DataCached|
+----------------+----------+
|   housing   |    true     |
+----------------+----------+
```

7. Apache Spark Framework

This section provides comprehensive details about Spark DataCache functionality utilizing the Apache Spark distributed computing framework.

7.1 DataCache

7.1.1 Introduction

Spark DataCache implements distributed data caching mechanisms designed to store data temporarily across cluster nodes, significantly improving performance by reducing data access latency for iterative algorithms and repeated computations. In the context of Parquet file storage, this functionality stores structured data in columnar format optimized for distributed processing and analytical workloads. The Spark cache utilizes local disk storage across the UnifyML server cluster infrastructure.

7.1.2 Syntax

The following syntax demonstrates the Spark DataCache algorithm implementation:

```
SELECT * FROM table(Unifyml.Spark.DataCache(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  OUTPUTCACHEID => (outputCacheid) [,...]
  [ PARTITIONCOLUMN => ( partitionColumn )] [,...]
  [ INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => (numPartition) ] [,...]
  [ CACHETYPE => ( cacheType ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

7.1.3 Input Arguments

The following table describes the input parameters for the Spark DataCache algorithm:

INPUTTABLE	Specifies the name of the source table containing the data columns. **DataType**: STRING
OUTPUTCACHEID	Unique identifier for the output cache storage. **DataType**: STRING
PARTITIONCOLUMN	[Optional] Column-based partitioning specification for distributed storage. **DataType**: STRING
INPUTCOLUMNS	[Optional] Specifies specific table columns for caching. **DataType**: ARRAY
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
CACHETYPE	[Optional] Cache storage type specification. **DataType**: STRING, **Default**: diskonly
BYTESPERCHUNK	[Optional] Data chunk size in bytes for storage optimization. **DataType**: INTEGER, **Default**: 64000000

7.1.4 Output

The following table describes the expected output structure of the Spark DataCache algorithm:

INPUTTABLE	Specifies the name of the source table containing the cached columns. **DataType**: STRING
INPUTCOLUMNS	Input columns of the table (if specifically specified). **DataType**:ARRAY
DATACACHED	Boolean status indicator of data caching operation. **DataType**:BOOLEAN

7.1.5 Examples

The following example demonstrates the execution of the Spark DataCache SQL function:

```
select * from table(Unifyml.Spark.DataCache(
INPUTTABLE => ("housing"),
NUMPARTITION => 5,
PARTITIONCOLUMN => ("page"),
OUTPUTCACHEID => ("datacacheid")))

SQL Results:
+----------------+----------+
| InputTableName |DataCached|
+----------------+----------+
|   housing   |    true  |
+----------------+----------+
```

8. Scikit-learn Framework

This section provides comprehensive details about DataCache functionality using the Python Scikit-learn machine learning library integration.

8.1 DataCache

8.1.1 Introduction

Scikit-learn DataCache implements data caching mechanisms designed to store datasets temporarily for machine learning workflows, significantly improving performance by reducing data loading overhead for iterative model training and evaluation processes. In the context of Parquet file storage, this functionality stores structured data in columnar format optimized for scientific computing and machine learning applications. The Scikit-learn cache utilizes local disk storage on the UnifyML server infrastructure.

8.1.2 Syntax

The following syntax demonstrates the Scikit-learn DataCache algorithm implementation:

```
SELECT * FROM table(Unifyml.Scikitlearn.DataCache(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  OUTPUTCACHEID => (outputCacheid)  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => (parallelism) ] [,...]
  [ NUMPARTITION => (numPartition) ] [,...]
  [ CACHETYPE => ( cacheType ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

8.1.3 Input Arguments

The following table describes the input parameters for the Scikit-learn DataCache algorithm:

INPUTTABLE	Specifies the name of the source table containing the data columns. **DataType**: STRING
PARTITIONCOLUMN	[Optional] Column-based partitioning specification for optimized storage. **DataType**: STRING
INPUTCOLUMNS	[Optional] Specifies specific table columns for caching. **DataType**: ARRAY
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[Optional] Number of data partitions for distributed storage. **DataType**: INTEGER
CACHETYPE	[Optional] Cache storage type specification. **DataType**: STRING, **Default**: diskonly
OUTPUTCACHEID	Unique identifier for the output cache storage. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for storage optimization. **DataType**: INTEGER, **Default**: 64000000

8.1.4 Output

The following table describes the expected output structure of the Scikit-learn DataCache algorithm:

INPUTTABLE	Specifies the name of the source table containing the cached columns. **DataType**: STRING
INPUTCOLUMNS	Input columns of the table (if specifically specified). **DataType**:ARRAY
DATACACHED	Boolean status indicator of data caching operation. **DataType**:BOOLEAN

8.1.5 Examples

The following example demonstrates the execution of the Scikit-learn DataCache SQL function:

```
select * from table(Unifyml.Scikitlearn.DataCache(
INPUTTABLE => ("eshoppinglimit"),
PARTITIONCOLUMN => ("pricetwo"),
OUTPUTCACHEID => ("datacacheid1")))

SQL Results:
+----------------+----------+
| InputTableName |DataCached|
+----------------+----------+
| eshoppinglimit |    true  |
+----------------+----------+
```

V

Data Preprocessing

9. UnifyML Framework

This section provides comprehensive details about UnifyML data preprocessing capabilities and methodologies.

9.1 Data Preprocessing

9.1.1 Introduction

UnifyML Data Preprocessing represents a critical component in the data analytics and machine learning pipeline. This process encompasses the systematic preparation and transformation of raw data to ensure optimal suitability for advanced analytical modeling and statistical analysis. Comprehensive data preprocessing eliminates data inconsistencies, handles missing values, and standardizes data formats, ultimately enhancing the accuracy and performance of machine learning models and analytical insights.

9.1.2 Syntax

The following syntax demonstrates the UnifyML Data Preprocessing algorithm implementation:

```
SELECT * FROM table(Unifyml.DataPreprocessing(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  OUTPUTCACHEID => (cacheid) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => ( inputCacheId) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ SAVECACHE => (saveCache) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ DATASCALING => (datascaling) ] [,...]
  [ SCALINGMETHOD => (scalingmethod) ] [,...]
  [ MINSCALER => (minscaler) ] [,...]
  [ MAXSCALER => (maxscaler) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

9.1.3 Input Arguments

The following table describes the input parameters for the UnifyML Data Preprocessing algorithm:

INPUTCOLUMNS	Specifies the names of columns containing the dependent variables. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTCACHEID	[Optional] Specifies the cache identifier for data retrieval. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
SAVECACHE	[Optional] Cache processed dataset for subsequent operations. **DataType**: BOOLEAN, **Default**: false
OUTPUTCACHEID	Unique identifier for output cache storage. **DataType**: STRING
SAVETODB	[Optional] Persist processed dataset to database. **DataType**: BOOLEAN, **Default**: false
OVERWRITEOUTPUTTABLE	[Optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[Optional] Export processed dataset to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[Optional] Destination table name for database storage. **DataType**: STRING
OUTPUTCSVNAME	[Optional] Output CSV file name specification. **DataType**: STRING
DATASCALING	[Optional] Apply data scaling transformations. **DataType**: BOOLEAN, **Default**: false
SCALINGMETHOD	[Optional] Scaling methodology specification (standardscaler, minmaxscaler, normalizer, binarizer). **DataType**: STRING, **Default**: standardscaler
MINSCALER	[Optional] Minimum scaling value for range normalization. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum scaling value for range normalization. **DataType**: INTEGER
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

9.1.4 Output

The following table describes the expected output structure of the UnifyML Data Preprocessing algorithm:

INPUTTABLE	Name of the source table containing the processed columns. **DataType**: STRING
INPUTCOLUMNS	Input columns processed by the algorithm. **DataType**:ARRAY
CLEANINGMETHOD	Applied data cleaning methodologies. **DataType**:ARRAY
DATACACHED	Boolean status indicator of data caching operation. **DataType**:BOOLEAN

9.1.5 Examples

The following examples demonstrate the execution of UnifyML data preprocessing SQL functions:

```
select * from table(Unifyml.DataPreprocessing(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness"],
INPUTDATATYPES => ARRAY["variance:double","skewness:double",
"curtosis:double","entropy:double","class:int"],
NUMPARTITION => 10,
PARTITIONCOLUMN => ("entropy"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
OUTPUTCACHEID => ("cacheid2"),
SAVETODB => false,
OUTPUTTABLENAME => ("outputTableName"),
BYTESPERCHUNK => 14000000))

SQL Results:
+----------------+------------------+---------------+---------------+
| InputTableName |InputColumns      | OutputCacheId |CleaningMethod |
+----------------+--------------------------------------------------+
| banknotes      |variance, skewness|    cacheid2   | [removerow]   |
+----------------+------------------+---------------+---------------+
```

10. Apache Spark Framework

This section provides comprehensive details about Spark data preprocessing capabilities utilizing distributed computing methodologies.

10.1 Data Preprocessing

10.1.1 Introduction

Spark Data Preprocessing leverages distributed computing capabilities to handle large-scale data preparation and transformation tasks across cluster environments. This approach enables efficient processing of massive datasets through parallel computation, ensuring scalable data cleaning, transformation, and feature engineering operations. The distributed architecture of Spark preprocessing optimizes performance for enterprise-scale analytical workflows.

10.1.2 Syntax

The following syntax demonstrates the Spark Data Preprocessing algorithm implementation:

```
SELECT * FROM table(Unifyml.Spark.DataPreprocessing(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  OUTPUTCACHEID => (cacheid) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => ( inputCacheId) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ SAVECACHE => (saveCache) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ DATASCALING => (datascaling) ] [,...]
  [ SCALINGMETHOD => (scalingmethod) ] [,...]
  [ MINSCALER => (minscaler) ] [,...]
  [ MAXSCALER => (maxscaler) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

10.1.3 Input Arguments

The following table describes the input parameters for the Spark Data Preprocessing algorithm:

INPUTCOLUMNS	Specifies the names of columns containing the dependent variables. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTCACHEID	[Optional] Specifies the cache identifier for data retrieval. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
SAVECACHE	[Optional] Cache processed dataset for subsequent operations. **DataType**: BOOLEAN, **Default**: false
OUTPUTCACHEID	Unique identifier for output cache storage. **DataType**: STRING
SAVETODB	[Optional] Persist processed dataset to database. **DataType**: BOOLEAN, **Default**: false
OVERWRITEOUTPUTTABLE	[Optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[Optional] Export processed dataset to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[Optional] Destination table name for database storage. **DataType**: STRING
OUTPUTCSVNAME	[Optional] Output CSV file name specification. **DataType**: STRING
DATASCALING	[Optional] Apply data scaling transformations. **DataType**: BOOLEAN, **Default**: false
SCALINGMETHOD	[Optional] Scaling methodology specification (standardscaler, minmaxscaler, normalizer, binarizer). **DataType**: STRING, **Default**: standardscaler
MINSCALER	[Optional] Minimum scaling value for range normalization. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum scaling value for range normalization. **DataType**: INTEGER
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

10.1.4 Output

The following table describes the expected output structure of the Spark Data Preprocessing algorithm:

INPUTTABLE	Name of the source table containing the processed columns. **DataType**:ARRAY
INPUTCOLUMNS	Input columns processed by the algorithm. **DataType**:ARRAY
CLEANINGMETHOD	Applied data cleaning methodologies. **DataType**:ARRAY
DATACACHED	Boolean status indicator of data caching operation. **DataType**:BOOLEAN

10.1.5 Examples

The following examples demonstrate the execution of Spark data preprocessing SQL functions:

```
select * from table(Unifyml.Spark.DataPreprocessing(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness"],
INPUTDATATYPES => ARRAY["variance:double","skewness:double",
"curtosis:double","entropy:double","class:int"],
NUMPARTITION => 10,
PARTITIONCOLUMN => ("entropy"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
SAVECACHE => true,
OUTPUTCACHEID => ("cacheid2"),
SAVETODB => false,
OUTPUTTABLENAME => ("outputTableName"),
BYTESPERCHUNK => 14000000))

SQL Results:
+---------------+-----------------+--------------+--------------+
| InputTableName |InputColumns      | OutputCacheId |CleaningMethod |
+---------------+-----------------------------------------------+
| banknotes      |variance, skewness|    cacheid2   | [removerow]   |
+---------------+-----------------+--------------+--------------+
```

11. Scikit-learn Framework

This section provides comprehensive details about Scikit-learn data preprocessing capabilities utilizing Python-based machine learning methodologies.

11.1 Data Preprocessing

11.1.1 Introduction

Scikit-learn Data Preprocessing leverages the comprehensive Python machine learning ecosystem to provide advanced data preparation and transformation capabilities. This approach integrates industry-standard preprocessing techniques including feature scaling, normalization, and advanced data cleaning methodologies. The Scikit-learn preprocessing pipeline ensures optimal data preparation for machine learning model training and statistical analysis workflows.

11.1.2 Syntax

The following syntax demonstrates the Scikit-learn Data Preprocessing algorithm implementation:

```
SELECT * FROM table(Unifyml.Scikitlearn.DataPreprocessing(
   INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
   OUTPUTCACHEID => (cacheid) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => ( inputCacheId) ] [,...]
  [ PARTITIONCOLUMN => (partitionColumn) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => ( parallelism ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ SAVECACHE => saveCache ] [,...]
  [ SAVETODB => saveToDb ] [,...]
  [ OUTPUTTABLENAME => (outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

11.1.3 Input Arguments

The following table describes the input parameters for the Scikit-learn Data Preprocessing algorithm:

INPUTCOLUMNS	Specifies the names of columns containing the dependent variables. **DataType**: ARRAY
OUTPUTCACHEID	Unique identifier for output cache storage. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTCACHEID	[Optional] Specifies the cache identifier for data retrieval. **DataType**: STRING
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
SAVECACHE	[Optional] Cache processed dataset for subsequent operations. **DataType**: BOOLEAN, **Default**: false
SAVETODB	[Optional] Persist processed dataset to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[Optional] Destination table name for database storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[Optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[Optional] Export processed dataset to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[Optional] Output CSV file name specification. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

11.1.4 Output

The following table describes the expected output structure of the Scikit-learn Data Preprocessing algorithm:

INPUTTABLE	Name of the source table containing the processed columns. **DataType**: STRING
INPUTCOLUMNS	Input columns processed by the algorithm. **DataType**:Array
CLEANINGMETHOD	Applied data cleaning methodologies. **DataType**:ARRAY
SCALINGMETHOD	Applied scaling transformation methodologies. **DataType**:ARRAY
DATACACHED	Boolean status indicator of data caching operation. **DataType**:BOOLEAN

11.1.5 Examples

The following examples demonstrate the execution of Scikit-learn data preprocessing SQL functions:

```
select * from table(Unifyml.Scikitlearn.DataPreprocessing(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness"],
INPUTDATATYPES => ARRAY["variance:double","skewness:double",
"curtosis:double","entropy:double","class:int"],
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1,
SAVECACHE => true,
OUTPUTCACHEID => ("cacheid"),
SAVETODB => false,
OUTPUTTABLENAME => ("outputTableName"),
BYTESPERCHUNK => 15000000))

SQL Results:
+----------------+-----------------+--------------+---------------+
| InputTableName |InputColumns     | OutputCacheId |CleaningMethod |
+----------------+-------------------------------------------------+
| banknotes      |variance, skewness|    cacheid2   | [removerow]   |
+----------------+-----------------+--------------+---------------+
```

VI Statistical Analysis

12. UnifyML Framework

This section provides comprehensive details about UnifyML statistical analysis capabilities and methodologies.

12.1 Correlation Analysis

12.1.1 Introduction

UnifyML Correlation Analysis provides comprehensive statistical measures for quantifying the strength and direction of relationships between multiple variables within datasets. Correlation analysis enables data scientists to identify linear and monotonic relationships, understand variable dependencies, and detect multicollinearity issues that may impact machine learning model performance. This functionality supports both Pearson and Spearman correlation coefficients for comprehensive relationship analysis.

For additional technical references, please consult Apache Spark correlation documentation.

12.1.2 Syntax

The following syntax demonstrates the UnifyML statistical correlation function implementation:

```
SELECT * FROM table(Unifyml.CorRelation(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TYPE => ( type ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

12.1.3 Input Arguments

The following table describes the input parameters for the UnifyML statistical correlation function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for correlation analysis. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
TYPE	[Optional] Correlation coefficient methodology (Pearson, Spearman). **DataType**: STRING, **Default**: Pearson
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

12.1.4 Output

The following table describes the expected output structure of the UnifyML statistical correlation function:

INPUTCOLUMNS	Correlation matrix for input columns showing pairwise relationships. **DataType**:Matrix

12.1.5 Examples

The following examples demonstrate the execution of UnifyML correlation analysis SQL functions:

```
select * from table(Unifyml.CorRelation(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "MEDV", "ZN", "RAD"],
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
INPUTDATATYPES => ARRAY["CRIM:double","MEDV:double"
,"ZN:double","RAD:double"],
TYPE => ("spearman"),
DATASCALING => true,
SCALINGMETHOD => ("Normalizer"),
BYTESPERCHUNK => 18000000))

SQL Results:
+------------+----------------------+----------------------+
| CorrMatrix |         CRIM         |         MEDV         |
+------------+----------------------+----------------------+
| CRIM       | 1.0                  | -0.23161833034345367 |
| MEDV       | -0.23161833034345367 | 1.0                  |
+------------+----------------------+----------------------+
```

12.2 Hypothesis Testing

12.2.1 Introduction

Hypothesis testing represents a fundamental statistical methodology for making data-driven inferences about population parameters based on sample data. This statistical framework enables researchers to formulate testable predictions about relationships between variables and systematically evaluate evidence to support or reject these hypotheses. Hypothesis testing provides a rigorous scientific approach for validating theoretical concepts, comparing group differences, and establishing statistical significance in experimental and observational studies.

For comprehensive technical references, please consult Apache Spark hypothesis testing documentation.

12.2.2 Syntax

The following syntax demonstrates the UnifyML statistical hypothesis testing function implementation:

```
SELECT * FROM table(Unifyml.Hypothesis(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

12.2.3 Input Arguments

The following table describes the input parameters for the UnifyML statistical hypothesis testing function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target dependent variable column. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

12.2.4 Output

The following table describes the expected output structure of the UnifyML statistical hypothesis testing function:

PValues	P-values for statistical significance testing. **DataType**:ARRAY
DegreesOfFreedom	Degrees of freedom for the statistical test. **DataType**:ARRAY
Statistics	Test statistics for hypothesis evaluation. **DataType**:ARRAY

12.2.5 Examples

The following examples demonstrate the execution of hypothesis testing SQL functions:

```
select * from table(Unifyml.Hypothesis(
INPUTTABLE => ("select * from banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------+----------------------------------------+
| CoefficientName  |         Value                          |
+------------------+----------------------------------------+
| PValues          | [0.2593395718830708,0.2067853651807845] |
| DegreesOfFreedom | [1337, 1255]                           |
| Statistics       | [1369.9751473689698,1295.73055089431]  |
+------------------+----------------------------------------+
```

12.3 Statistical Summarization

12.3.1 Introduction

UnifyML Statistical Summarization provides comprehensive descriptive statistics for DataFrame columns, including count, mean, standard deviation, minimum, maximum, and additional summary metrics for each numeric variable. This functionality enables rapid understanding of data distribution characteristics, central tendencies, and variability patterns within datasets, supporting exploratory data analysis and data quality assessment workflows.

For comprehensive technical references, please consult Apache Spark summarizer documentation.

12.3.2 Syntax

The following syntax demonstrates the UnifyML statistical summarization function implementation:

```
SELECT * FROM table(Unifyml.Summarize(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

12.3.3 Input Arguments

The following table describes the input parameters for the UnifyML statistical summarization function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for statistical analysis. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

12.3.4 Output

The following table describes the expected output structure of the UnifyML statistical summarization function:

Mean	Arithmetic mean values for specified columns. **DataType**:DOUBLE
Variance	Variance measures for specified columns. **DataType**:DOUBLE
Max	Maximum values for specified columns. **DataType**:DOUBLE
Count	Total count of observations for specified columns. **DataType**:INTEGER
Min	Minimum values for specified columns. **DataType**:DOUBLE
NumNonZeros	Count of non-zero values for specified columns. **DataType**:DOUBLE

12.3.5 Examples

The following examples demonstrate the execution of UnifyML statistical summarization SQL functions:

```
select * from table(Unifyml.Summarize(
INPUTCACHEID => ("cacheid1"),
INPUTCOLUMNS => ARRAY["crim", "zn", "indus", "chas"]))

SQL Results:
+----------------------+-------+--------------------+
| Columns/Coefficients | Count |        Mean        |
+----------------------+-------+--------------------+
| crim                 | 506   | 3.613523557312253  |
| zn                   | 506   | 11.363636363636363 |
| indus                | 506   | 11.136778656126484 |
| chas                 | 506   | 0.0691699604743083 |
+----------------------+-------+--------------------+
+--------------------+---------+---------+-------------+
|     Variance       |   Min   |   Max   | NumNonZeros |
+--------------------+---------+---------+-------------+
| 73.98657819906933  |0.00632  | 88.9762 | 506.0       |
|543.9368136813679   | 0.0     | 100.0   | 134.0       |
|47.06444247368219   |0.46     | 27.74   | 506.0       |
|0.06451297303643408 | 0.0     | 1.0     | 35.0        |
+--------------------+---------+---------+-------------+
```

12.4 Outlier Detection and Visualization

12.4.1 Introduction

UnifyML outlier detection capabilities provide essential functionality for identifying and visualizing data points that deviate significantly from the expected distribution patterns within datasets. Outliers represent anomalous observations that can substantially impact machine learning model performance and statistical analysis accuracy. The platform implements multiple statistical methodologies including Z-score analysis, Interquartile Range (IQR) techniques, and advanced graphical approaches such as boxplot visualizations for comprehensive outlier identification and assessment.

12.4.2 Syntax

The following syntax demonstrates the UnifyML outlier detection function implementation:

```
SELECT * FROM table(Unifyml.ShowOutliers(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

12.4.3 Input Arguments

The following table describes the input parameters for the UnifyML outlier detection function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for outlier analysis. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

12.4.4 Output

The following table describes the expected output structure of the UnifyML outlier detection function:

INPUTCOLUMNNAMES	Input column names containing identified outlier values. **DataType**: MIXED

12.4.5 Examples

The following examples demonstrate the execution of outlier detection SQL functions:

```
select * from table(Unifyml.ShowOutliers(
INPUTTABLE => ("germencredit"),
INPUTCOLUMNS => ARRAY["accountstatus", "duration", "credithistory",
 "purpose"]))
 limit 5

SQL Results:
+----------+---------------+---------------+---------+
| duration | accountstatus | credithistory | purpose |
+----------+---------------+---------------+---------+
| 6        | 0             | 4             | 4       |
| 12       | 3             | 4             | 7       |
| 42       | 0             | 2             | 3       |
| 24       | 0             | 3             | 0       |
| 36       | 3             | 2             | 7       |
+----------+---------------+---------------+---------+
```

12.5 Skewness Analysis

12.5.1 Introduction

UnifyML skewness analysis provides comprehensive statistical measures for quantifying the asymmetry of dataset distributions. Skewness represents a fundamental statistical property that describes the degree and direction of distribution asymmetry, enabling data scientists to understand

data shape characteristics, identify distribution patterns, and determine appropriate analytical methodologies for subsequent modeling tasks.

12.5.2 Syntax

The following syntax demonstrates the UnifyML skewness analysis function implementation:

```
SELECT * FROM table(Unifyml.ShowSkewness(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn )] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

12.5.3 Input Arguments

The following table describes the input parameters for the UnifyML skewness analysis function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for skewness analysis. **DataType**: ARRAY
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 1
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

12.5.4 Output

The following table describes the expected output structure of the skewness analysis function:

INPUTCOLUMNNAMES	Input column names with corresponding skewness coefficient values. **DataType**: NUMERIC

12.5.5 Examples

The following examples demonstrate the execution of UnifyML skewness analysis SQL functions:

```
select * from table(Unifyml.ShowSkewness(
INPUTCACHEID => ("cacheid1"),
INPUTCOLUMNS => ARRAY["ZN", "INDUS", "CHAS", "MEDV", "RAD",
 "AGE", "DIS"]))

SQL Results:
+----------+----------+----------+----------+----------+
|    zn    |  indus   |   chas   |   medv   |   rad    |
+----------+----------+----------+----------+----------+
| 2.219063 | 0.294146 | 3.395799 | 1.104811 | 1.001833 |
+----------+----------+----------+----------+----------+
```

13. Apache Spark Framework

This section provides comprehensive details about Apache Spark statistical analysis capabilities utilizing distributed computing methodologies.

13.1 Correlation Analysis

13.1.1 Introduction

Apache Spark Correlation Analysis leverages distributed computing capabilities to provide comprehensive statistical measures for quantifying relationships between variables across large-scale datasets. This implementation utilizes Spark's distributed processing architecture to efficiently compute correlation matrices for massive datasets, supporting both Pearson and Spearman correlation coefficients for comprehensive relationship analysis in big data environments.

For comprehensive technical references, please consult Apache Spark correlation documentation.

13.1.2 Syntax

The following syntax demonstrates the Spark statistical correlation function implementation:

```
SELECT * FROM table(Unifyml.Spark.CorRelation(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TYPE => ( type ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

13.1.3 Input Arguments

The following table describes the input parameters for the Spark statistical correlation function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for correlation analysis. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
TYPE	[Optional] Correlation coefficient methodology (Pearson, Spearman). **DataType**: STRING, **Default**: Pearson
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

13.1.4 Output

The following table describes the expected output structure of the Spark statistical correlation function:

INPUTCOLUMNS	Correlation matrix for input columns showing pairwise relationships. **DataType**:Matrix

13.1.5 Examples

The following examples demonstrate the execution of Spark statistical correlation SQL functions:

```
select * from table(Unifyml.Spark.CorRelation(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "MEDV", "ZN", "RAD"],
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
INPUTDATATYPES => ARRAY["CRIM:double","MEDV:double"
,"ZN:double","RAD:double"],
TYPE => ("spearman"),
DATASCALING => true,
SCALINGMETHOD => ("Normalizer"),
BYTESPERCHUNK => 18000000))

SQL Results:
+------------+----------------------+----------------------+
| CorrMatrix |         CRIM         |         MEDV         |
+------------+----------------------+----------------------+
| CRIM       | 1.0                  | -0.23161833034345367 |
| MEDV       | -0.23161833034345367 | 1.0                  |
+------------+----------------------+----------------------+
```

13.2 Hypothesis Testing

13.2.1 Introduction

Hypothesis testing represents a fundamental statistical methodology for making data-driven inferences about population parameters based on sample data. This statistical framework enables researchers to formulate testable predictions about relationships between variables and systematically evaluate evidence to support or reject these hypotheses. Hypothesis testing provides a rigorous scientific approach for validating theoretical concepts, comparing group differences, and establishing statistical significance in experimental and observational studies within distributed computing environments.

For comprehensive technical references, please consult Apache Spark hypothesis testing documentation.

13.2.2 Syntax

The following syntax demonstrates the Spark statistical hypothesis testing function implementation:

```
SELECT * FROM table(Unifyml.Spark.Hypothesis(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

13.2.3 Input Arguments

The following table describes the input parameters for the Spark statistical hypothesis testing function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target dependent variable column. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

13.2.4 Output

The following table describes the expected output structure of the Spark statistical hypothesis testing function:

PValues	P-values for statistical significance testing. **DataType**:ARRAY
DegreesOfFreedom	Degrees of freedom for the statistical test. **DataType**:ARRAY
Statistics	Test statistics for hypothesis evaluation. **DataType**:ARRAY

13.2.5 Examples

The following examples demonstrate the execution of Spark hypothesis testing SQL functions:

```
select * from table(Unifyml.Spark.Hypothesis(
INPUTTABLE => ("select * from banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------+----------------------------------------+
| CoefficientName  |         Value                          |
+------------------+----------------------------------------+
| PValues          | [0.2593395718830708,0.2067853651807845] |
| DegreesOfFreedom | [1337, 1255]                           |
| Statistics       | [1369.9751473689698,1295.73055089431]   |
+------------------+----------------------------------------+
```

13.3 Statistical Summarization

13.3.1 Introduction

Apache Spark Statistical Summarization leverages distributed computing capabilities to provide comprehensive descriptive statistics for DataFrame columns across large-scale datasets. This functionality delivers count, mean, standard deviation, minimum, maximum, and additional summary metrics for each numeric variable through parallel processing. The distributed summarization enables rapid understanding of data distribution characteristics, central tendencies, and variability patterns within massive datasets, supporting enterprise-scale exploratory data analysis and data quality assessment workflows.

For comprehensive technical references, please consult Apache Spark summarizer documentation.

13.3.2 Syntax

The following syntax demonstrates the Spark statistical summarization function implementation:

```
SELECT * FROM table(Unifyml.Spark.Summarize(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

13.3.3 Input Arguments

The following table describes the input parameters for the Spark statistical summarization function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for statistical analysis. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

13.3.4 Output

The following table describes the expected output structure of the Spark statistical summarization function:

Mean	Arithmetic mean values for specified columns. **DataType**:DOUBLE
Variance	Variance measures for specified columns. **DataType**:DOUBLE
Max	Maximum values for specified columns. **DataType**:DOUBLE
Count	Total count of observations for specified columns. **DataType**:INTEGER
Min	Minimum values for specified columns. **DataType**:DOUBLE
NumNonZeros	Count of non-zero values for specified columns. **DataType**:DOUBLE

13.3.5 Examples

The following examples demonstrate the execution of Spark statistical summarization SQL functions:

```
select * from table(Unifyml.Spark.Summarize(
INPUTCACHEID => ("cacheid1"),
INPUTCOLUMNS => ARRAY["crim", "zn", "indus", "chas"]))

SQL Results:
+----------------------+-------+--------------------+
| Columns/Coefficients | Count |        Mean        |
+----------------------+-------+--------------------+
| crim                 | 506   | 3.613523557312253  |
| zn                   | 506   | 11.363636363636363 |
| indus                | 506   | 11.136778656126484 |
| chas                 | 506   | 0.0691699604743083 |
+----------------------+-------+--------------------+
+--------------------+---------+---------+-------------+
|     Variance       |   Min   |   Max   | NumNonZeros |
+--------------------+---------+---------+-------------+
| 73.98657819906933  |0.00632  | 88.9762 | 506.0       |
|543.9368136813679   | 0.0     | 100.0   | 134.0       |
|47.06444247368219   |0.46     | 27.74   | 506.0       |
|0.06451297303643408 | 0.0     | 1.0     | 35.0        |
+--------------------+---------+---------+-------------+
```

13.4 Outlier Detection and Visualization

13.4.1 Introduction

Apache Spark outlier detection capabilities provide essential functionality for identifying and visualizing anomalous data points that deviate significantly from expected distribution patterns within large-scale datasets. The distributed computing architecture enables efficient outlier detection across massive datasets using statistical methodologies including Z-score analysis, Interquartile Range (IQR) techniques, and advanced graphical approaches such as boxplot visualizations for comprehensive anomaly identification and assessment.

13.4.2 Syntax

The following syntax demonstrates the Spark outlier detection function implementation:

```
SELECT * FROM table(Unifyml.Spark.ShowOutliers(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

13.4.3 Input Arguments

The following table describes the input parameters for the Spark outlier detection function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for outlier analysis. **DataType**: ARRAY
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

13.4.4 Output

The following table describes the expected output structure of the Spark outlier detection function:

INPUTCOLUMNNAMES	Input column names containing identified outlier values. **DataType**: MIXED

13.4.5 Examples

The following examples demonstrate the execution of Spark outlier detection SQL functions:

```
select * from table(Unifyml.Spark.ShowOutliers(
INPUTTABLE => ("germencredit"),
INPUTCOLUMNS => ARRAY["accountstatus", "duration", "credithistory",
 "purpose"]))
 limit 5

SQL Results:
+----------+---------------+---------------+---------+
| duration | accountstatus | credithistory | purpose |
+----------+---------------+---------------+---------+
| 6        | 0             | 4             | 4       |
| 12       | 3             | 4             | 7       |
| 42       | 0             | 2             | 3       |
| 24       | 0             | 3             | 0       |
| 36       | 3             | 2             | 7       |
+----------+---------------+---------------+---------+
```

14. Scikit-learn Framework

This section provides comprehensive details about Scikit-learn statistical analysis capabilities utilizing Python-based machine learning methodologies.

14.1 Outlier Detection and Visualization

14.1.1 Introduction

Scikit-learn outlier detection capabilities provide essential functionality for identifying and visualizing anomalous data points that deviate significantly from expected distribution patterns within datasets using Python-based machine learning algorithms. This implementation leverages advanced statistical methodologies including Z-score analysis, Interquartile Range (IQR) techniques, and sophisticated graphical approaches such as boxplot visualizations for comprehensive anomaly identification and assessment in machine learning workflows.

14.1.2 Syntax

The following syntax demonstrates the Scikit-learn outlier detection function implementation:

```
SELECT * FROM table(Unifyml.Scikitlearn.ShowOutliers(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn )] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

14.1.3 Input Arguments

The following table describes the input parameters for the Scikit-learn outlier detection function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for outlier analysis. **DataType**: ARRAY
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 1
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

14.1.4 Output

The following table describes the expected output structure of the Scikit-learn outlier detection function:

INPUTCOLUMNNAMES	Input column names containing identified outlier values. **DataType**: MIXED

14.1.5 Examples

The following examples demonstrate the execution of Scikit-learn outlier detection SQL functions:

```
select * from table(Unifyml.Scikitlearn.ShowOutliers(
INPUTTABLE => ("germencredit"),
INPUTCOLUMNS => ARRAY["accountstatus", "duration", "credithistory",
"purpose"]))
 limit 5

SQL Results:
+----------+---------------+---------------+---------+
| duration | accountstatus | credithistory | purpose |
+----------+---------------+---------------+---------+
| 6        | 0             | 4             | 4       |
| 12       | 3             | 4             | 7       |
| 42       | 0             | 2             | 3       |
| 24       | 0             | 3             | 0       |
| 36       | 3             | 2             | 7       |
+----------+---------------+---------------+---------+
```

14.2 Skewness Analysis

14.2.1 Introduction

Scikit-learn skewness analysis provides comprehensive statistical measures for quantifying the asymmetry of dataset distributions using Python-based statistical computing methodologies. Skewness represents a fundamental statistical property that describes the degree and direction of distribution asymmetry, enabling data scientists to understand data shape characteristics, identify

distribution patterns, and determine appropriate analytical methodologies for subsequent machine learning model development and evaluation.

14.2.2 Syntax

The following syntax demonstrates the Scikit-learn skewness analysis function implementation:

```
SELECT * FROM table(Unifyml.Scikitlearn.ShowSkewness(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn )] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

14.2.3 Input Arguments

The following table describes the input parameters for the Scikit-learn skewness analysis function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for skewness analysis. **DataType**: ARRAY
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 1
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

14.2.4 Output

The following table describes the expected output structure of the Scikit-learn skewness analysis function:

INPUTCOLUMNNAMES	Input column names with corresponding skewness coefficient values. **DataType**: NUMERIC

14.2.5 Examples

The following examples demonstrate the execution of Scikit-learn skewness analysis SQL functions:

```
select * from table(Unifyml.Scikitlearn.ShowSkewness(
INPUTCACHEID => ("cacheid1"),
INPUTCOLUMNS => ARRAY["ZN", "INDUS", "CHAS", "MEDV", "RAD", "AGE",
"DIS"]))

SQL Results:
+----------+----------+----------+----------+----------+
|    zn    |  indus   |   chas   |   medv   |   rad    |
+----------+----------+----------+----------+----------+
| 2.219063 | 0.294146 | 3.395799 | 1.104811 | 1.001833 |
+----------+----------+----------+----------+----------+
```

14.3 Statistical Summarization

14.3.1 Introduction

Scikit-learn Statistical Summarization leverages Python-based data science libraries to provide comprehensive descriptive statistics for DataFrame columns, including count, mean, standard deviation, minimum, maximum, quartile distributions, and additional summary metrics for each numeric variable. This functionality enables rapid understanding of data distribution characteristics, central tendencies, and variability patterns within datasets, supporting comprehensive exploratory data analysis and data quality assessment workflows in machine learning pipelines.

For comprehensive technical references, please consult Dask DataFrame summarization documentation.

14.3.2 Syntax

The following syntax demonstrates the Scikit-learn statistical summarization function implementation:

```
SELECT * FROM table(Unifyml.Scikitlearn.Summarize(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ PARTITIONCOLUMN => (partitionColumn)] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => ( parallelism ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

14.3.3 Input Arguments

The following table describes the input parameters for the Scikit-learn statistical summarization function:

INPUTCOLUMNS	Specifies the names of columns containing the variables for statistical analysis. **DataType**: ARRAY
PARTITIONCOLUMN	Column specification for indexing and partitioning operations. Should be an indexed column in the SQL server with orderable data type. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

14.3.4 Output

The following table describes the expected output structure of the Scikit-learn statistical summarization function:

Mean	Arithmetic mean values for specified columns. **DataType**:DOUBLE
Std	Standard deviation values for specified columns. **DataType**:DOUBLE
Max	Maximum values for specified columns. **DataType**:DOUBLE
Count	Total count of observations for specified columns. **DataType**:INTEGER
Min	Minimum values for specified columns. **DataType**:DOUBLE
25percentage	25th percentile values (first quartile). **DataType**:DOUBLE
50percentage	50th percentile values (median). **DataType**:DOUBLE
75percentage	75th percentile values (third quartile). **DataType**:DOUBLE

14.3.5 Examples

The following examples demonstrate the execution of Scikit-learn statistical summarization SQL functions:

```
select * from table(Unifyml.Scikitlearn.Summarize(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN"],
PARTITIONCOLUMN => ("LSTAT"),
DATASCALING => true))

SQL Results:
+----------------------+-------+--------------+---------+-----------+
| Columns/Coefficients | count |     mean     |   std   |    min    |
+----------------------+-------+--------------+---------+-----------+
| crim                 | 506   | -8.51317e-17 | 1.00099 | -0.419782 |
| zn                   | 506   | 3.30653e-16  | 1.00099 | -0.487722 |
+----------------------+-------+--------------+---------+-----------+
+-----------+-----------+------------+---------+
|    25%    |    50%    |    75%     |   max   |
+-----------+-----------+------------+---------+
| -0.41097  | -0.390667 | 0.00739656 | 9.93393 |
| -0.487722 | -0.487722 | 0.0487722  | 3.80423 |
+-----------+-----------+------------+---------+
```

VII

Automated Machine Learning Algorithms

15. UnifyML Framework

This section provides comprehensive details about UnifyML automated machine learning algorithms for regression and classification tasks.

15.1 Automated Regression Analysis

15.1.1 Introduction

UnifyML Automated Regression Analysis provides comprehensive model performance evaluation across multiple regression algorithms, enabling data scientists to identify the optimal modeling approach for specific datasets. This functionality automatically benchmarks various regression techniques and provides accuracy metrics to guide algorithm selection. By executing automated regression analysis prior to model development, users can make informed decisions about the most suitable regression methodology based on empirical performance comparisons across their specific data characteristics.

15.1.2 Syntax

The following syntax demonstrates the UnifyML Automated Regression function implementation:

```
SELECT * FROM table(Unifyml.AutoRegression(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ INPUTALGORITHMS => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

15.1.3 Input Arguments

The following table describes the input parameters for the UnifyML Automated Regression function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target dependent variable column for prediction. **DataType**: STRING
INPUTALGORITHMS	[Optional] Specifies specific regression algorithms for evaluation. **DataType**: ARRAY, **Default**: All available regression algorithms
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[Optional] Proportion of data allocated for training versus testing. **DataType**: DOUBLE, **Default**: 0.7
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
APPLYPCAMODEL	[Optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[Optional] Identifier for existing PCA model application. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

15.1.4 Output

The following table describes the expected output structure of the UnifyML Automated Regression function:

InputAlgorithms	Name of the evaluated regression algorithm. **DataType**: STRING
Accuracy	Performance accuracy metric for the algorithm. **DataType**: DOUBLE

15.1.5 Examples

The following examples demonstrate the execution of UnifyML Automated Regression SQL functions:

```
select * from table(Unifyml.AutoRegression(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity",
"citricacid", "residualsugar", "chlorides",
"freesulfur", "totalsulfur", "density", "pH",
"sulphates", "alcohol"],
TARGETCOLUMN =>  ("quality")))

SQL Results:
+-------------------+-------------------+
|     ModelName     |     Accuracy      |
+-------------------+-------------------+
| LinReg            | 99.41362839216885 |
| DecisionTreeReg   | 99.40908536422711 |
| Glr               | 99.41362839216885 |
| GradientboostTree | 99.43958090355727 |
| RandomForestReg   | 99.4300566315211  |
+-------------------+-------------------+
```

15.2 Automated Classification Analysis

15.2.1 Introduction

UnifyML Automated Classification Analysis provides comprehensive model performance evaluation across multiple classification algorithms, enabling data scientists to identify the optimal classification approach for specific datasets. This functionality automatically benchmarks various classification techniques and provides accuracy metrics to guide algorithm selection. By executing automated classification analysis prior to model development, users can make informed decisions about the most suitable classification methodology based on empirical performance comparisons across their specific data characteristics and class distributions.

15.2.2 Syntax

The following syntax demonstrates the UnifyML Automated Classification function implementation:

```
SELECT * FROM table(Unifyml.AutoClassification(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ INPUTALGORITHMS => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

15.2.3 Input Arguments

The following table describes the input parameters for the UnifyML Automated Classification function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target class variable column for prediction. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTALGORITHMS	[Optional] Specifies specific classification algorithms for evaluation. **DataType**: ARRAY, **Default**: All available classification algorithms
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[Optional] Proportion of data allocated for training versus testing. **DataType**: DOUBLE, **Default**: 0.7
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
APPLYPCAMODEL	[Optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[Optional] Identifier for existing PCA model application. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

15.2.4 Output

The following table describes the expected output structure of the UnifyML Automated Classification function:

InputAlgorithms	Name of the evaluated classification algorithm. **DataType**: STRING
Accuracy	Performance accuracy metric for the algorithm. **DataType**: DOUBLE

15.2.5 Examples

The following examples demonstrate the execution of UnifyML Automated Classification SQL functions:

```
select * from table(Unifyml.AutoClassification(
INPUTTABLE => ("select * from banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN => ("class"),
INPUTALGORITHMS => ARRAY["DecisionTreeClassifier",
"GradientboostedClassifier"]))

SQL Results:
+---------------------------+-------------------+
|         ModelName         |     Accuracy      |
+---------------------------+-------------------+
| DecisionTreeClassifier    | 99.01643835616439 |
| GradientboostedClassifier | 99.01095890410959 |
+---------------------------+-------------------+
```

16. Apache Spark Framework

This section provides comprehensive details about Apache Spark automated machine learning algorithms utilizing distributed computing capabilities.

16.1 Automated Regression Analysis

16.1.1 Introduction

Apache Spark Automated Regression Analysis leverages distributed computing capabilities to provide comprehensive model performance evaluation across multiple regression algorithms on large-scale datasets. This functionality automatically benchmarks various regression techniques using Spark's distributed processing architecture and provides accuracy metrics to guide algorithm selection for big data environments. By executing automated regression analysis prior to model development, users can make informed decisions about the most suitable regression methodology based on empirical performance comparisons across massive datasets and distributed computing resources.

16.1.2 Syntax

The following syntax demonstrates the Spark Automated Regression function implementation:

```
SELECT * FROM table(Unifyml.Spark.AutoRegression(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ INPUTALGORITHMS => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

16.1.3 Input Arguments

The following table describes the input parameters for the Spark Automated Regression function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target dependent variable column for prediction. **DataType**: STRING
INPUTALGORITHMS	[Optional] Specifies specific regression algorithms for evaluation. **DataType**: ARRAY, **Default**: All available regression algorithms
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[Optional] Proportion of data allocated for training versus testing. **DataType**: DOUBLE, **Default**: 0.7
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
APPLYPCAMODEL	[Optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[Optional] Identifier for existing PCA model application. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

16.1.4 Output

The following table describes the expected output structure of the Spark Automated Regression function:

InputAlgorithms	Name of the evaluated regression algorithm. **DataType**: STRING
Accuracy	Performance accuracy metric for the algorithm. **DataType**: DOUBLE

16.1.5 Examples

The following examples demonstrate the execution of Spark Automated Regression SQL functions:

```
select * from table(Unifyml.Spark.AutoRegression(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity",
"citricacid", "residualsugar", "chlorides",
"freesulfur", "totalsulfur", "density", "pH",
"sulphates", "alcohol"],
TARGETCOLUMN =>  ("quality")))

SQL Results:
+-------------------+-------------------+
|     ModelName     |     Accuracy      |
+-------------------+-------------------+
| LinReg            | 99.41362839216885 |
| DecisionTreeReg   | 99.40908536422711 |
| Glr               | 99.41362839216885 |
| GradientboostTree | 99.43958090355727 |
| RandomForestReg   | 99.4300566315211  |
+-------------------+-------------------+
```

VIII Predictive Machine Learning Algorithms

17. UnifyML Framework

This section provides comprehensive details about UnifyML predictive machine learning algorithms for regression and prediction tasks.

17.1 Decision Tree Regression

17.1.1 Introduction

UnifyML Decision Tree Regression represents a powerful machine learning algorithm designed for predicting continuous numerical values through hierarchical decision-making structures. This regression technique constructs decision trees to model complex relationships between input features and target variables, enabling accurate predictions through recursive data partitioning based on feature values and optimal splitting criteria.

For comprehensive technical references, please consult Apache Spark Decision Tree Regression documentation.

17.1.2 Syntax

The following syntax demonstrates the UnifyML Decision Tree Regression function implementation:

```
SELECT * FROM table(Unifyml.DecisionTreeReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.1.3 Input Arguments

The following table describes the input parameters for the UnifyML Decision Tree Regression function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target dependent variable column for prediction. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
MAXBINS	[Optional] Maximum number of bins for discretizing continuous features and determining optimal splits at tree nodes. **DataType**: INTEGER, **Default**: 32, **Range**: 2-32
MAXDEPTH	[Optional] Maximum depth of the decision tree (non-negative). **DataType**: INTEGER, **Default**: 5, **Range**: 0-5
MININFOGAIN	[Optional] Minimum information gain threshold for considering splits at tree **DataType**: DOUBLE, **Default**: 0.0, **Range**: 0.0-0.1
SEED	[Optional] Random seed parameter for reproducible results. **DataType**: LONG, **Default**: 159147643, **Range**: 0-159147643
TRAININGSAMPLESIZE	[Optional] Proportion of data allocated for training versus testing. **DataType**: DOUBLE, **Default**: 0.7
HYPERPARAMETERTUNING	[Optional] Enable automated hyperparameter optimization for optimal mode **DataType**: BOOLEAN, **Default**: false
TUNINGTYPE	[Optional] Validation methodology: CrossValidation (cv) or TrainValidation **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[Optional] Hyperparameter tuning range for maximum bins parameter. **DataType**: ARRAY, **Default**: 32, **Range**: 2-32
MAXDEPTHPARAMS	[Optional] Hyperparameter tuning range for maximum depth parameter. **DataType**: ARRAY, **Default**: 5, **Range**: 0-5
MININFOGAINPARAMS	[Optional] Hyperparameter tuning range for minimum information gain para **DataType**: ARRAY, **Default**: 0.0, **Range**: 0.0-0.1
SEEDPARAMS	[Optional] Hyperparameter tuning range for random seed parameter. **DataType**: ARRAY, **Default**: 159147643, **Range**: 0-159147643
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[Optional] Number of cross-validation folds for model evaluation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING

APPLYPCAMODEL	[Optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[Optional] Identifier for existing PCA model application. **DataType**: STRING
SAVEMODEL	[Optional] Enable model persistence for future predictions. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[Optional] Output model identifier for saved models. **DataType**: STRING
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

17.1.4 Output

The following table describes the expected output structure of the UnifyML Decision Tree Regression function:

Rmse	Root Mean Square Error measuring standard deviation of prediction residuals. **DataType**: DOUBLE
R2	R-squared coefficient of determination measuring model fit quality. **DataType**: DOUBLE
Mse	Mean Squared Error measuring average squared prediction errors. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction errors. **DataType**: DOUBLE
Accuracy	Overall model accuracy percentage. **DataType**: DOUBLE

17.1.5 Examples

The following examples demonstrate the execution of UnifyML Decision Tree Regression SQL functions:

```
select * from table(Unifyml.DecisionTreeReg(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["crim", "zn", "indus", "chas",
"nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat"],
TARGETCOLUMN =>  ("medv"),
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates","removeoutliers"]
))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

17.2 Decision Tree Regression Prediction

17.2.1 Introduction

UnifyML Decision Tree Regression Prediction enables the application of trained decision tree models to generate predictions on new datasets. This functionality utilizes the hierarchical structure of trained decision trees, where each leaf node represents a predicted value for specific data subsets based on the learned decision rules and feature relationships.

For comprehensive technical references, please consult Apache Spark Decision Tree Regression documentation.

17.2.2 Syntax

The following syntax demonstrates the UnifyML Decision Tree Regression Prediction function implementation:

```
SELECT * FROM table(Unifyml.DecisionTreeRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.2.3 Input Arguments

The following table describes the input parameters for the UnifyML Decision Tree Regression Prediction function:

INPUTMODELNAME	Specifies the trained model identifier for prediction generation. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing prediction data. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
SAVETODB	[Optional] Persist prediction results to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[Optional] Destination table name for prediction results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[Optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[Optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[Optional] Output CSV file name specification. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

17.2.4 Output

The following table describes the expected output structure of the UnifyML Decision Tree Regression Prediction function:

INPUTCOLUMNNAMES	Original column names as specified in the trained model. **DataType**: MIXED
PREDICTCOLUMN	Generated prediction values for the target variable. **DataType**: DOUBLE

17.2.5 Examples

The following examples demonstrate the execution of Decision Tree Regression Prediction SQL functions:

```
select * from table(Unifyml.DecisionTreeRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelDT")))

SQL Results:
+---------+------+-------+------+-------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

17.3 Generalized Linear Regression

17.3.1 Introduction

Generalized Linear Regression (GLR) represents an advanced statistical modeling framework that extends traditional linear regression by accommodating diverse target variable distributions and flexible link functions. This comprehensive regression approach enables modeling of various data types beyond normally distributed continuous variables, supporting different probability distributions and enabling sophisticated statistical analysis for complex datasets.

For comprehensive technical references, please consult Apache Spark Generalized Linear Regression documentation.

17.3.2 Syntax

The following syntax demonstrates the UnifyML Generalized Linear Regression function implementation:

```
SELECT * FROM table(Unifyml.Glr(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ FAMILY => ( family ) ] [,...]
  [ LINK => ( link ) ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ VARIANCEPOWER => variancePower ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ VARIANCEPOWERPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.3.3 Input Arguments

The following table describes the input parameters for the UnifyML Generalized Linear Regression function:

INPUTCOLUMNS	Specifies the names of columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the name of the target dependent variable column for prediction. **DataType**: STRING
INPUTTABLE	[Optional] Specifies the name of the source table containing the columns. **DataType**: STRING
INPUTDATATYPES	[Optional] Data type specifications for input columns. **DataType**: ARRAY
FAMILY	[Optional] Probability distribution family specification for the target variable **DataType**: STRING, **Default**: gaussian
LINK	[Optional] Link function specification connecting linear predictor to mean. **DataType**: STRING, **Default**: Identity
MAXITERNUM	[Optional] Maximum number of optimization iterations. **DataType**: INTEGER, **Default**: 25, **Range**: 2-25
REGPARAM	[Optional] Regularization parameter for model complexity control. **DataType**: DOUBLE, **Default**: 0.0, **Range**: 0.0-0.1
VARIANCEPOWER	[Optional] Variance power parameter for Tweedie family distributions. **DataType**: DOUBLE, **Default**: 0.0, **Range**: 0.0-0.1
TRAININGSAMPLESIZE	[Optional] Proportion of data allocated for training versus testing. **DataType**: DOUBLE, **Default**: 0.7
HYPERPARAMETERTUNING	[Optional] Enable automated hyperparameter optimization for optimal mode **DataType**: BOOLEAN, **Default**: false
TUNINGTYPE	[Optional] Validation methodology: CrossValidation (cv) or TrainValidation **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[Optional] Hyperparameter tuning range for maximum iterations parameter. **DataType**: ARRAY, **Default**: 25, **Range**: 2-25
REGPARAMPARAMS	[Optional] Hyperparameter tuning range for regularization parameter. **DataType**: ARRAY, **Default**: 0.0, **Range**: 0.0-0.1
VARIANCEPOWERPARAMS	[Optional] Hyperparameter tuning range for variance power parameter. **DataType**: ARRAY, **Default**: 0.0, **Range**: 0.0-0.1
PARALLELISM	[Optional] Parallel processing capability for parameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[Optional] Number of cross-validation folds for model evaluation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[Optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[Optional] Column-based partitioning specification. **DataType**: STRING
APPLYPCAMODEL	[Optional] Apply Principal Component Analysis for dimensionality reductio **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[Optional] Identifier for existing PCA model application. **DataType**: STRING
SAVEMODEL	[Optional] Enable model persistence for future predictions. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[Optional] Output model identifier for saved models. **DataType**: STRING

DATACLEANING	[Optional] Enable comprehensive dataset cleaning operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[Optional] Apply specific cleaning methodologies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[Optional] Value specifications for fill cleaning method. **DataType**: ARRAY
DATASCALING	[Optional] Enable dataset scaling transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[Optional] Apply specific scaling methodologies (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[Optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[Optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[Optional] Cache identifier for data retrieval. **DataType**: STRING
BYTESPERCHUNK	[Optional] Data chunk size in bytes for processing optimization. **DataType**: INTEGER, **Default**: 64000000

17.3.4 Output

The following table describes the expected output structure of the UnifyML Generalized Linear Regression function:

Coefficients	Regression coefficients representing feature importance and direction. **DataType**: ARRAY
Intercept	Intercept term representing expected mean value when all predictors are zero **DataType**: DOUBLE
Rmse	Root Mean Square Error measuring standard deviation of prediction residuals **DataType**: DOUBLE
R2	R-squared coefficient of determination measuring model fit quality. **DataType**: DOUBLE
Dispersion	Dispersion parameter measuring variability in the fitted model. **DataType**: DOUBLE
NullDeviance	Null deviance measuring baseline model performance. **DataType**: DOUBLE
ResidualDegreeOfFreedomNull	Degrees of freedom for null model (intercept-only model). **DataType**: DOUBLE
Deviance	Model deviance measuring goodness of fit. **DataType**: DOUBLE
ResidualDegreeOfFreedom	Residual degrees of freedom for the fitted model. **DataType**: DOUBLE
AIC	Akaike Information Criterion for model selection and comparison. **DataType**: DOUBLE
Mse	Mean Squared Error measuring average squared prediction errors. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction errors. **DataType**: DOUBLE
Accuracy	Overall model accuracy percentage. **DataType**: DOUBLE

17.3.5 Examples

The following examples demonstrate the execution of Generalized Linear Regression SQL functions:

```
select * from table(Unifyml.Glr(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+---------------------------------+----------------------+
|         CoefficientName         |        Value         |
+---------------------------------+----------------------+
| Mean Squared Error(MSE)         | 0.7495913109460369   |
| Mean Absolute Error(MAE)        | 0.6505508427404258   |
| Root Mean Squared Error(RMSE)   | 0.8657894148960456   |
| Regression Score(R2)            | 0.059324001570318585 |
| Accuracy                        | 99.34944915725957    |
| Intercept                       | 7.185730896280032    |
| fixedacidity                    | -0.11587705201451137 |
| volatileacidity                 | -1.6457566314599963  |
| citricacid                      | -0.11541540224980984 |
| Dispersion                      | 0.7426126449072332   |
| Null_Deviance                   | 2723.237496427457    |
| Residual_Degree_Of_Freedom_Null | 3498                 |
| Deviance                        | 2595.43119395078     |
| Residual_Degree_Of_Freedom      | 3495                 |
| AIC                             | 8894.494664716532    |
+---------------------------------+----------------------+
```

17.4 GeneralizedLinearRegressionPredict

17.4.1 Introduction

The GeneralizedLinearRegressionPredict function enables prediction capabilities using previously trained generalized linear regression models. This function applies the fitted model parameters to new data instances, generating predictions based on the established statistical relationships between predictor variables and the target variable. The prediction process follows the same mathematical framework as traditional linear regression but extends to accommodate various probability distributions and link functions inherent in generalized linear models.

For comprehensive technical references, please consult GeneralizedLinearRegression.

17.4.2 Syntax

The following syntax specification defines the UnifyML GlrPredict function interface:

```
SELECT * FROM table(Unifyml.GlrPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.4.3 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML GlrPredict function:

INPUTMODELNAME	Specifies the trained model identifier for prediction execution. **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial **DataType**: STRING
INPUTDATATYPES	[optional] Specifies data type mappings for input feature columns. **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and cleansing operatio **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing techniques including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization operations. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of prediction results to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table name for result persistence. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table overwrite behavior for existing output tables. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export of prediction results. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename for exported results. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for optimizati **DataType**: INTEGER, **Default**: 64000000

17.4.4 Output

The following table describes the prediction output structure from the UnifyML GlrPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained model.
PREDICTCOLUMN	Generated prediction values based on model inference.

17.4.5 Examples

The following example demonstrates practical implementation of the GlrPredict SQL function:

```
select * from table(Unifyml.GlrPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeGLR7")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

17.5 GradientBoostTree

17.5.1 Introduction

The UnifyML Gradient Boosted Trees (GBT) implementation represents a sophisticated ensemble learning methodology designed for both classification and regression analytical tasks. This algorithm employs sequential boosting techniques that iteratively combine multiple weak learners, typically decision trees, to construct a robust predictive model with enhanced accuracy and generalization capabilities. The gradient boosting framework optimizes model performance by minimizing prediction errors through gradient descent optimization, making it particularly effective for complex data patterns and non-linear relationships.

The algorithm constructs models in a stage-wise fashion, where each successive tree attempts to correct the residual errors of the previous ensemble. This iterative refinement process enables the capture of intricate data dependencies while maintaining computational efficiency through careful regularization and early stopping mechanisms.

For comprehensive technical documentation and implementation details, please refer to GradientBoostedTree.

17.5.2 Syntax

The following syntax specification defines the UnifyML GradientBoostTree function interface:

```
SELECT * FROM table(Unifyml.GradientBoostTree(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MAXITERS => maxiters ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.5.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML GradientBoostTree function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors. **DataType**: ARRAY
TARGETCOLUMN	Defines the dependent variable column for prediction or classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training data. **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns. **DataType**: ARRAY
MAXBINS	[optional] Controls feature discretization granularity and optimal split point selection for continuous variables. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Constrains individual tree complexity to prevent overfitting. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERS	[optional] Defines maximum boosting iterations for ensemble construction. **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAIN	[optional] Sets minimum information gain threshold for tree node splitting. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Controls random number generation for reproducible results. **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Configures training/validation data split ratio. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced performa **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation or TrainValidati **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for automated bin count optimization during hyperparameter tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for automated depth optimization during tuning. **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERNUMPARAMS	[optional] Parameter grid for automated iteration count optimization. **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAINPARAMS	[optional] Parameter grid for automated information gain threshold optimiza **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed optimization during tuning. **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for model evaluation. **DataType**: INTEGER, **Default**: 2

NUMPARTITION	[optional] Specifies data partitioning count for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for data distribution. **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained gradient boost model. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved model persistence. **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing techniques including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer transformations. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary for MinMaxScaler transformation range. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary for MinMaxScaler transformation range. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance **DataType**: INTEGER, **Default**: 64000000

17.5.4 Output

The following table describes the comprehensive evaluation metrics returned by the UnifyML GradientBoostTree function:

Rmse	Root Mean Square Error: Measures standard deviation of prediction residuals, providing scale-dependent accuracy assessment. **DataType**: DOUBLE
R2	Coefficient of Determination: Quantifies proportion of target variable variance explained by the model, ranging from 0 to 1. **DataType**: DOUBLE
Mse	Mean Squared Error: Calculates average squared differences between predicted and actual values. **DataType**: DOUBLE
Mae	Mean Absolute Error: Computes average absolute differences between predictions and ground truth values. **DataType**: DOUBLE
Accuracy	Model Accuracy: Represents overall prediction correctness as percentage of correctly classified or predicted instances. **DataType**: DOUBLE

17.5.5 Examples

The following example demonstrates comprehensive implementation of the GradientBoostTree SQL function with data preprocessing and partitioning:

```
select * from table(Unifyml.GradientBoostTree(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
 "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"],
TARGETCOLUMN =>  ("MEDV"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

17.6 GradientBoostTreePredict

17.6.1 Introduction

The UnifyML GradientBoostTreePredict function implements ensemble-based prediction inference utilizing pre-trained gradient boosting models. This function leverages the sequential aggregation methodology inherent in gradient boosting, where predictions are generated through the weighted combination of multiple decision tree outputs. Each constituent tree in the ensemble contributes to

the final prediction by addressing residual errors from previous iterations, resulting in a sophisticated prediction mechanism that captures complex non-linear patterns and feature interactions within the data.

The prediction process maintains the same mathematical framework established during model training, ensuring consistency in feature scaling, data preprocessing, and ensemble aggregation methodologies.

For comprehensive technical documentation and implementation details, please refer to GradientBoostedTree.

17.6.2 Syntax

The following syntax specification defines the UnifyML GradientBoostTreePredict function interface:

```
SELECT * FROM table(Unifyml.GradientBoostPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.6.3 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML GradientBoostTreePredict function:

INPUTMODELNAME	Specifies the trained gradient boosting model identifier for prediction executi **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns. **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of prediction results to database infrastr **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for result persistence and storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar **DataType**: INTEGER, **Default**: 64000000

17.6.4 Output

The following table describes the prediction output structure from the UnifyML GradientBoostTreePredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained gradient boosting model.
PREDICTCOLUMN	Generated prediction values based on ensemble model inference and aggregation.

17.6.5 Examples

The following example demonstrates practical implementation of the GradientBoostPredict SQL function:

```
select * from table(Unifyml.GradientBoostPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeGBT")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

17.7 LinReg

17.7.1 Introduction

The UnifyML Linear Regression implementation provides a comprehensive statistical modeling framework for analyzing linear relationships between dependent and independent variables. This fundamental machine learning technique employs ordinary least squares optimization to determine optimal coefficients that minimize the sum of squared residuals between predicted and observed values. The algorithm supports both simple linear regression for single predictor scenarios and multiple linear regression for complex multivariate analysis.

The implementation incorporates advanced regularization techniques including Ridge (L2) and Lasso (L1) regularization through the ElasticNet framework, enabling effective handling of multicollinearity and feature selection. The linear regression model assumes linear relationships, independence of observations, homoscedasticity, and normally distributed residuals, making it particularly suitable for interpretable predictive modeling and statistical inference applications.

For comprehensive technical references and mathematical foundations, please consult Linear-Regression.

17.7.2 Syntax

The following syntax specification defines the UnifyML LinReg function interface:

```
SELECT * FROM table(Unifyml.LinReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ ELASTICPARAM => elasticparam ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ ELASTICPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.7.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML LinReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors in the r **DataType**: ARRAY
TARGETCOLUMN	Defines the dependent variable column for continuous value prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset. **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns. **DataType**: ARRAY
MAXITERNUM	[optional] Controls maximum optimization iterations for coefficient converg **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAM	[optional] Configures regularization strength to prevent overfitting and impro **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAM	[optional] Controls ElasticNet mixing ratio between Ridge and Lasso regula **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
TRAININGSAMPLESIZE	[optional] Defines proportional split ratio for training and validation datasets **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced model pe **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation or TrainValidati **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[optional] Parameter grid for automated iteration count optimization during t **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Parameter grid for automated regularization strength optimization **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAMPARAMS	[optional] Parameter grid for automated ElasticNet mixing parameter optimi **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation eff **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust model evaluation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed processing optimi **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for data distribution. **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reducti **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for feature space transf **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained linear regression model paran **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved model persistence and future inference **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement operation **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for improved conver **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation range.

17.7.4 Output

The following table describes the comprehensive statistical output metrics from the UnifyML LinReg function:

Coefficients	Linear regression coefficients representing feature importance and directional impact on **DataType**: ARRAY
Intercept	Y-intercept representing the expected target value when all predictors equal zero. **DataType**: DOUBLE
Rmse	Root Mean Square Error: Standard deviation of prediction residuals, measuring average **DataType**: DOUBLE
R2	Coefficient of Determination: Proportion of target variance explained by the linear model **DataType**: DOUBLE
NumInstances	Total number of training instances utilized in model estimation and validation. **DataType**: DOUBLE
MeanSquaredError	Mean Squared Error: Average squared differences between predicted and actual values. **DataType**: DOUBLE
MeanAbsoluteError	Mean Absolute Error: Average absolute differences between predictions and ground truth **DataType**: DOUBLE
TotalIterations	Total optimization iterations required for coefficient convergence and model training com **DataType**: DOUBLE
ExplainedVariance	Explained Variance Score: Proportion of target variance captured by the regression mode **DataType**: DOUBLE
DegreesOfFreedom	Statistical degrees of freedom representing available information for parameter estimatio **DataType**: DOUBLE
R2adj	Adjusted R-squared: Modified coefficient of determination accounting for predictor coun **DataType**: DOUBLE
Mse	Mean Squared Error: Duplicate metric measuring average squared prediction errors. **DataType**: DOUBLE
Mae	Mean Absolute Error: Duplicate metric measuring average absolute prediction errors. **DataType**: DOUBLE
Accuracy	Model Accuracy: Overall prediction correctness expressed as percentage of acceptable p **DataType**: DOUBLE

17.7.5 Examples

The following example demonstrates comprehensive implementation of the LinReg SQL function with data preprocessing and partitioning:

```
select * from table(Unifyml.LinReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.7495913109460369   |
| Mean Absolute Error(MAE)     | 0.6505508427404258   |
| Root Mean Squared Error(RMSE)| 0.8657894148960456   |
| Regression Score(R2)         | 0.059324001570318585 |
| Accuracy                     | 99.34944915725957    |
| Intercept                    | 7.185730896280032    |
| fixedacidity                 | -0.11587705201451137 |
| volatileacidity              | -1.6457566314599963  |
| citricacid                   | -0.11541540224980984 |
| NumInstances                 | 3499                 |
| MeanSquaredError             | 0.7417637021865621   |
| MeanAbsoluteError            | 0.6491153898723525   |
| TotalIterations              | 0                    |
| ExplainedVariance            | 0.0365265225712546   |
| DegreesOfFreedom             | 3495.0               |
| R2adj                        | 0.04611366607091327  |
+------------------------------+----------------------+
```

17.8 LinRegPredict

17.8.1 Introduction

The UnifyML LinRegPredict function enables prediction inference using pre-trained linear regression models for continuous target variable estimation. This function applies the learned linear coefficients and intercept parameters to new data instances, generating predictions based on the established linear relationships between predictor variables and the target variable. The prediction process maintains mathematical consistency with the training phase, ensuring that feature scaling, data preprocessing, and coefficient application follow the same statistical framework established during model development.

The function supports real-time prediction scenarios and batch processing workflows, enabling seamless integration of trained linear regression models into production environments and analytical pipelines.

For comprehensive technical references and implementation guidelines, please consult Linear-Regression.

17.8.2 Syntax

The following syntax specification defines the UnifyML LinRegPredict function interface:

```
SELECT * FROM table(Unifyml.LinRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.8.3 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML LinRegPredict function:

INPUTMODELNAME	Specifies the trained linear regression model identifier for prediction executio **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor variat **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns du **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consist **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of prediction results to database infrastr **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar **DataType**: INTEGER, **Default**: 64000000

17.8.4 Output

The following table describes the prediction output structure from the UnifyML LinRegPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained linear regression model for predic
PREDICTCOLUMN	Generated continuous prediction values based on linear model coefficients and fea

17.8.5 Examples

The following example demonstrates practical implementation of the LinRegPredict SQL function:

```
select * from table(Unifyml.LinRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelLR4")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

17.9 GradientBoostTreePredict

17.9.1 Introduction

The UnifyML GradientBoostTreePredict function implements ensemble-based prediction inference utilizing pre-trained gradient boosting models. This function leverages the sequential aggregation methodology inherent in gradient boosting, where predictions are generated through the weighted combination of multiple decision tree outputs. Each constituent tree in the ensemble contributes to the final prediction by addressing residual errors from previous iterations, resulting in a sophisticated prediction mechanism that captures complex non-linear patterns and feature interactions within the data.

The prediction process maintains the same mathematical framework established during model training, ensuring consistency in feature scaling, data preprocessing, and ensemble aggregation methodologies.

For comprehensive technical documentation and implementation details, please refer to GradientBoostedTree.

17.9.2 Syntax

The following syntax specification defines the UnifyML GradientBoostTreePredict function interface:

```
SELECT * FROM table(Unifyml.GradientBoostPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.10 RandomForestReg

17.10.1 Introduction

The UnifyML Random Forest Regression implementation represents a sophisticated ensemble learning methodology that constructs multiple decision trees through bootstrap aggregating (bagging) techniques to generate robust and accurate predictions for continuous target variables. This algorithm leverages the principle of ensemble diversity by training numerous decision trees on different subsets of both training instances and feature variables, subsequently combining their outputs through averaging to produce final predictions with enhanced generalization capabilities and reduced overfitting.

Random Forest Regression incorporates two primary sources of randomness: bootstrap sampling of training instances for each tree and random feature subset selection at each node split. This dual randomization strategy creates decorrelated trees that collectively capture complex non-linear relationships while maintaining computational efficiency and interpretability. The algorithm excels in handling high-dimensional datasets, mixed data types, and provides built-in feature importance metrics for analytical insights.

The ensemble methodology effectively reduces prediction variance compared to individual decision trees while preserving low bias, making it particularly suitable for complex regression tasks with intricate feature interactions and non-linear patterns.

For comprehensive technical documentation and implementation details, please refer to RandomForestRegression.

17.10.2 Syntax

The following syntax specification defines the UnifyML RandomForestReg function interface:

```
SELECT * FROM table(Unifyml.RandomForestReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.10.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML RandomForestReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors for ens **DataType**: ARRAY
TARGETCOLUMN	Defines the continuous dependent variable column for regression prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset f **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns dur **DataType**: ARRAY
MAXBINS	[optional] Controls feature discretization granularity and optimal split point selection for continuous variables in decision tree construction. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Constrains individual tree complexity to prevent overfitting while maintaining ensemble diversity. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Sets minimum information gain threshold for tree node splitting decisions and pruning control. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Controls random number generation for reproducible bootstrap sa and feature selection. **DataType**: Long, **Default**: 235498149, **Min**: 0, **Max**: 235498149
TRAININGSAMPLESIZE	[optional] Configures proportional split ratio for training and validation data **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced ensembl **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation or TrainValidati for parameter optimization. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for automated bin count optimization during hyperparameter tuning processes. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for automated tree depth optimization during ense **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Parameter grid for automated information gain threshold optimiza **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed optimization during ensemble tun **DataType**: ARRAY, **Default**: 235498149, **Min**: 0, **Max**: 235498149
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation an **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust ensemble evaluation and parameter selection. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed ensemble processi **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data pr **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reducti before ensemble training. **DataType**: BOOLEAN, **Default**: false

INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained random forest ensemble model. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved ensemble model persistence and future inf **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement o **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for enhance **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation rang **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation rang **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and proce **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance **DataType**: INTEGER, **Default**: 64000000

17.10.4 Output

The following table describes the comprehensive evaluation metrics returned by the UnifyML RandomForestReg function:

Rmse	Root Mean Square Error: Measures standard deviation of ensemble prediction residuals, providing scale-dependent accuracy assessment for regression performance. **DataType**: DOUBLE
R2	Coefficient of Determination: Quantifies proportion of target variable variance explained by the random forest ensemble, ranging from 0 to 1. **DataType**: DOUBLE
Mse	Mean Squared Error: Calculates average squared differences between ensemble predictions and actual continuous values. **DataType**: DOUBLE
Mae	Mean Absolute Error: Computes average absolute differences between ensemble predictions and ground truth values. **DataType**: DOUBLE
Accuracy	Model Accuracy: Represents overall ensemble prediction correctness as percentage of accurately predicted instances within acceptable tolerance. **DataType**: DOUBLE

17.10.5 Examples

The following example demonstrates comprehensive implementation of the RandomForestReg SQL function with data preprocessing and model persistence:

```
select * from table(Unifyml.RandomForestReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
SAVEMODEL => true,
OUTPUTMODEL => ("modeRF")))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |        Value       |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 0.6991760258990803 |
| Mean Absolute Error(MAE)     | 0.6318749865613316 |
| Root Mean Squared Error(RMSE)| 0.8361674628321054 |
| Regression Score(R2)         | 0.12259107511444745|
| Accuracy                     | 99.36812501343867  |
| OutputModelName              | modelRF            |
+------------------------------+--------------------+
```

17.10.6 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML Gradient-BoostTreePredict function:

INPUTMODELNAME	Specifies the trained gradient boosting model identifier for prediction executi **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns. **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of prediction results to database infrastr **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for result persistence and storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar **DataType**: INTEGER, **Default**: 64000000

17.10.7 Output

The following table describes the prediction output structure from the UnifyML GradientBoostTreePredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained gradient boosting model.
PREDICTCOLUMN	Generated prediction values based on ensemble model inference and aggregation.

17.10.8 Examples

The following example demonstrates practical implementation of the GradientBoostPredict SQL function:

```
select * from table(Unifyml.GradientBoostPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeGBT")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

17.11 LinReg

17.11.1 Introduction

The UnifyML Linear Regression implementation provides a comprehensive statistical modeling framework for analyzing linear relationships between dependent and independent variables. This fundamental machine learning technique employs ordinary least squares optimization to determine optimal coefficients that minimize the sum of squared residuals between predicted and observed values. The algorithm supports both simple linear regression for single predictor scenarios and multiple linear regression for complex multivariate analysis.

The implementation incorporates advanced regularization techniques including Ridge (L2) and Lasso (L1) regularization through the ElasticNet framework, enabling effective handling of multicollinearity and feature selection. The linear regression model assumes linear relationships, independence of observations, homoscedasticity, and normally distributed residuals, making it particularly suitable for interpretable predictive modeling and statistical inference applications.

For comprehensive technical references and mathematical foundations, please consult Linear-Regression.

17.11.2 Syntax

The following syntax specification defines the UnifyML LinReg function interface:

```
SELECT * FROM table(Unifyml.LinReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ ELASTICPARAM => elasticparam ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ ELASTICPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.11.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML LinReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors in the r **DataType**: ARRAY
TARGETCOLUMN	Defines the dependent variable column for continuous value prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset. **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns. **DataType**: ARRAY
MAXITERNUM	[optional] Controls maximum optimization iterations for coefficient converg **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAM	[optional] Configures regularization strength to prevent overfitting and impro **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAM	[optional] Controls ElasticNet mixing ratio between Ridge and Lasso regula **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
TRAININGSAMPLESIZE	[optional] Defines proportional split ratio for training and validation datasets **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced model pe **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation or TrainValidati **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[optional] Parameter grid for automated iteration count optimization during t **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Parameter grid for automated regularization strength optimization **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAMPARAMS	[optional] Parameter grid for automated ElasticNet mixing parameter optimi **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation eff **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust model evaluation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed processing optimi **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for data distribution. **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reducti **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for feature space transf **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained linear regression model paran **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved model persistence and future inference **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement operation **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for improved conver **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation range.

17.11.4 Output

The following table describes the comprehensive statistical output metrics from the UnifyML LinReg function:

Coefficients	Linear regression coefficients representing feature importance and directional impact on **DataType**: ARRAY
Intercept	Y-intercept representing the expected target value when all predictors equal zero. **DataType**: DOUBLE
Rmse	Root Mean Square Error: Standard deviation of prediction residuals, measuring average **DataType**: DOUBLE
R2	Coefficient of Determination: Proportion of target variance explained by the linear model **DataType**: DOUBLE
NumInstances	Total number of training instances utilized in model estimation and validation. **DataType**: DOUBLE
MeanSquaredError	Mean Squared Error: Average squared differences between predicted and actual values. **DataType**: DOUBLE
MeanAbsoluteError	Mean Absolute Error: Average absolute differences between predictions and ground truth **DataType**: DOUBLE
TotalIterations	Total optimization iterations required for coefficient convergence and model training com **DataType**: DOUBLE
ExplainedVariance	Explained Variance Score: Proportion of target variance captured by the regression mode **DataType**: DOUBLE
DegreesOfFreedom	Statistical degrees of freedom representing available information for parameter estimatio **DataType**: DOUBLE
R2adj	Adjusted R-squared: Modified coefficient of determination accounting for predictor coun **DataType**: DOUBLE
Mse	Mean Squared Error: Duplicate metric measuring average squared prediction errors. **DataType**: DOUBLE
Mae	Mean Absolute Error: Duplicate metric measuring average absolute prediction errors. **DataType**: DOUBLE
Accuracy	Model Accuracy: Overall prediction correctness expressed as percentage of acceptable p **DataType**: DOUBLE

17.11.5 Examples

The following example demonstrates comprehensive implementation of the LinReg SQL function with data preprocessing and partitioning:

```
select * from table(Unifyml.LinReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.7495913109460369   |
| Mean Absolute Error(MAE)     | 0.6505508427404258   |
| Root Mean Squared Error(RMSE)| 0.8657894148960456   |
| Regression Score(R2)         | 0.059324001570318585 |
| Accuracy                     | 99.34944915725957    |
| Intercept                    | 7.185730896280032    |
| fixedacidity                 | -0.11587705201451137 |
| volatileacidity              | -1.6457566314599963  |
| citricacid                   | -0.11541540224980984 |
| NumInstances                 | 3499                 |
| MeanSquaredError             | 0.7417637021865621   |
| MeanAbsoluteError            | 0.6491153898723525   |
| TotalIterations              | 0                    |
| ExplainedVariance            | 0.0365265225712546   |
| DegreesOfFreedom             | 3495.0               |
| R2adj                        | 0.04611366607091327  |
+------------------------------+----------------------+
```

17.12 LinRegPredict

17.12.1 Introduction

The UnifyML LinRegPredict function enables prediction inference using pre-trained linear regression models for continuous target variable estimation. This function applies the learned linear coefficients and intercept parameters to new data instances, generating predictions based on the established linear relationships between predictor variables and the target variable. The prediction process maintains mathematical consistency with the training phase, ensuring that feature scaling, data preprocessing, and coefficient application follow the same statistical framework established during model development.

The function supports real-time prediction scenarios and batch processing workflows, enabling seamless integration of trained linear regression models into production environments and analytical pipelines.

For comprehensive technical references and implementation guidelines, please consult Linear-Regression.

17.12.2 Syntax

The following syntax specification defines the UnifyML LinRegPredict function interface:

```
SELECT * FROM table(Unifyml.LinRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.12.3 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML LinRegPredict function:

INPUTMODELNAME	Specifies the trained linear regression model identifier for prediction executio **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor variat **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns du **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consist **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of prediction results to database infrastr **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar **DataType**: INTEGER, **Default**: 64000000

17.12.4 Output

The following table describes the prediction output structure from the UnifyML LinRegPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained linear regression model for predic
PREDICTCOLUMN	Generated continuous prediction values based on linear model coefficients and fea

17.12.5 Examples

The following example demonstrates practical implementation of the LinRegPredict SQL function:

```
select * from table(Unifyml.LinRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelLR4")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

17.13 GradientBoostTreePredict

17.13.1 Introduction

The UnifyML GradientBoostTreePredict function implements ensemble-based prediction inference utilizing pre-trained gradient boosting models. This function leverages the sequential aggregation methodology inherent in gradient boosting, where predictions are generated through the weighted combination of multiple decision tree outputs. Each constituent tree in the ensemble contributes to the final prediction by addressing residual errors from previous iterations, resulting in a sophisticated prediction mechanism that captures complex non-linear patterns and feature interactions within the data.

The prediction process maintains the same mathematical framework established during model training, ensuring consistency in feature scaling, data preprocessing, and ensemble aggregation methodologies.

For comprehensive technical documentation and implementation details, please refer to GradientBoostedTree.

17.13.2 Syntax

The following syntax specification defines the UnifyML GradientBoostTreePredict function interface:

```
SELECT * FROM table(Unifyml.GradientBoostPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.14 RandomForestRegPredict

17.14.1 Introduction

The UnifyML RandomForestRegPredict function enables prediction inference using pre-trained Random Forest ensemble models for continuous target variable estimation. This function leverages the aggregated prediction capabilities of multiple decision trees, where individual tree predictions are combined through averaging to generate robust final predictions. The ensemble prediction process maintains the same bootstrap sampling methodology and feature randomization strategies established during training, ensuring consistent mathematical framework and enhanced prediction reliability through variance reduction.

The function supports real-time inference scenarios and batch processing workflows, enabling seamless deployment of trained Random Forest models into production environments while preserving the ensemble's inherent robustness against overfitting and outlier sensitivity.

For comprehensive technical documentation and implementation guidelines, please refer to RandomForestRegPredict.

17.14.2 Syntax

The following syntax specification defines the UnifyML RandomForestRegPredict function interface:

```
SELECT * FROM table(Unifyml.RandomForestRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.14.3 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML RandomForestRegPredict function:

INPUTMODELNAME	Specifies the trained Random Forest ensemble model identifier for prediction **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns du **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consist **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of ensemble prediction results to databa **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performan **DataType**: INTEGER, **Default**: 64000000

17.14.4 Output

The following table describes the prediction output structure from the UnifyML RandomForestRegPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained Random Forest ensemble model.
PREDICTCOLUMN	Generated continuous prediction values based on ensemble aggregation of individ

17.14.5 Examples

The following example demonstrates practical implementation of the RandomForestRegPredict SQL function with feature scaling:

```
select * from table(Unifyml.RandomForestRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeRF"),
DATASCALING => true,
SCALINGMETHOD => ("Normalizer")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.756806220503233 |
| 6.3          | 0.3             | 0.34       | 5.756806220503233 |
| 8.1          | 0.28            | 0.4        | 5.756806220503233 |
| 7.2          | 0.23            | 0.32       | 5.756806220503233 |
| 7.2          | 0.23            | 0.32       | 5.756806220503233 |
+--------------+-----------------+------------+-------------------+
```

17.15 Pca

17.15.1 Introduction

The UnifyML Principal Component Analysis (PCA) implementation provides a sophisticated dimensionality reduction technique that employs orthogonal linear transformations to convert potentially correlated feature variables into a reduced set of linearly uncorrelated components called principal components. This statistical procedure maximizes variance capture while minimizing information loss, enabling effective feature space compression and noise reduction for improved computational efficiency and model performance.

PCA operates through eigenvalue decomposition of the feature covariance matrix, identifying principal components that represent the directions of maximum variance in the high-dimensional feature space. The algorithm ranks components by their explained variance, allowing practitioners to select an optimal subset that preserves the most significant data patterns while reducing computational complexity. This technique proves particularly valuable for high-dimensional datasets, visualization applications, and preprocessing steps for downstream machine learning algorithms.

The implementation supports both feature standardization and scaling operations to ensure appropriate treatment of variables with different scales and units, maintaining mathematical rigor in the orthogonal transformation process.

For comprehensive technical references and mathematical foundations, please consult Pca.

17.15.2 Syntax

The following syntax specification defines the UnifyML PCA function interface:

```
SELECT * FROM table(Unifyml.Pca(
  NUMBEROFREDUCTIONS => ( numberOfReductions ) [,...]
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ INPUTMODELNAME => ( inputModelName ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ]  [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.15.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML PCA function configuration:

NUMBEROFREDUCTIONS	Specifies the target dimensionality for principal component reduction and varia **DataType**: INTEGER
INPUTCOLUMNS	Defines feature variable columns for principal component analysis and transfor **DataType**: ARRAY
INPUTTABLE	[optional] Specifies source table, view, or query containing feature variables for **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns during **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Configures proportional split ratio for training and validation dataset **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Specifies data partitioning count for distributed processing optimizat **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data proce **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained PCA transformation model and **DataType**: BOOLEAN, **Default**: false
INPUTMODELNAME	[optional] Specifies model identifier for PCA transformation persistence and re **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization for improved PCA perform **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScale StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation ran **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation ran **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and proc **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance **DataType**: INTEGER, **Default**: 64000000

17.15.4 Output

The following table describes the comprehensive output structure from the UnifyML PCA function:

InputColumnNames	Original feature column identifiers subjected to principal component analysis. **DataType**: ARRAY
NumberOfReductions	Specified target dimensionality representing the number of principal components retain **DataType**: INTEGER
PcaModelName	Identifier for the saved PCA transformation model enabling future applications. **DataType**: STRING

17.15.5 Examples

The following example demonstrates practical implementation of the PCA SQL function with model persistence:

```
select * from table(Unifyml.Pca(
INPUTTABLE => ("indiandiabetes"),
INPUTCOLUMNS => ARRAY["nooftimespregent", "age"],
NUMbEROFREDUCTIONS => 2,
SAVEMODEL => true,
INPUTMODELNAME => ("modelClassIDPca1")))

SQL Results:

+-----------------------+--------------------+------------------+
|   InputColumnNames    | NumberOfReductions |   PcaModelName   |
+-----------------------+--------------------+------------------+
| nooftimespregent, age | 2                  | modelClassIDPca1 |
+-----------------------+--------------------+------------------+
```

17.16 Tokenizer

17.16.1 Introduction

The UnifyML Tokenizer represents a comprehensive natural language processing (NLP) component designed for sophisticated text preprocessing and linguistic analysis. This advanced tokenization system decomposes textual data into meaningful linguistic units called tokens, which can include words, subwords, characters, or custom-defined segments depending on the specified tokenization strategy. The tokenizer serves as a fundamental preprocessing step for downstream NLP applications, including text classification, sentiment analysis, machine translation, and information extraction tasks.

The implementation incorporates multiple text processing capabilities including stopword removal, n-gram generation, and advanced linguistic preprocessing techniques. The tokenizer supports configurable text cleaning operations, normalization procedures, and feature extraction methodologies that prepare textual data for machine learning algorithms and statistical analysis. This functionality enables efficient handling of large-scale text corpora while maintaining linguistic integrity and semantic preservation.

17.16.2 Syntax

The following syntax specification defines the UnifyML Tokenizer function interface:

```
SELECT * FROM table(Unifyml.Tokenizer(
  INPUTTABLE => ( { table | view  }) [,...]
  [ INPUTTEXTCOLUMNNAME => ( inputTextcolumn ) ] [,...]
  [ STOPWORDS => ( stopWords ) ] [,...]
  [ NGRAMS => (stemming ) ] [,...]
  [ SETNGRAMS => ( partsOfSpeech ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

17.16.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML Tokenizer function configuration:

INPUTTABLE	Specifies the source table or view containing textual data for tokenization pro **DataType**: STRING
INPUTTEXTCOLUMNNAME	[optional] Defines the specific text column identifier for tokenization analysi **DataType**: STRING
PARALLELISM	[optional] Configures parallel processing degree for distributed tokenization **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Specifies data partitioning count for distributed text processing opt **DataType**: INTEGER
STOPWORDS	[optional] Enables automatic removal of common stopwords for improved se **DataType**: BOOLEAN, **Default**: true
NGRAMS	[optional] Activates n-gram generation for capturing multi-word linguistic pa **DataType**: BOOLEAN,**Default**: false
SETNGRAMS	[optional] Configures specific n-gram size for linguistic pattern extraction. **DataType**: INTEGER,**Default**: false
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized text data access. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performan **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Enables persistent storage of tokenized results to database infrastru **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for tokenized result persistence. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for tokenized text analys **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported tokenized data. **DataType**: STRING

17.16.4 Output

The following table describes the tokenization output structure from the UnifyML Tokenizer function:

17.16.5 Examples

The following example demonstrates practical implementation of the Tokenizer SQL function with text processing:

```
select * from table(Unifyml.Tokenizer(
INPUTTABLE => ("medicaldrama"),
INPUTTEXTCOLUMNNAME => ("compliant"),
PARTITIONCOLUMN => ("id"),
OUTPUTCACHEID => ("tokencache")))

SQL Results:

+----------------+------------+--------------+
| InputTableName | DataCached | InputColumns |
+----------------+------------+--------------+
| medicaldrama   | true       | compliant    |
+----------------+------------+--------------+
```

18. Spark

This section provides comprehensive documentation for the supported Apache Spark-based machine learning algorithms and advanced predictive analytics implementations available within the UnifyML framework.

18.1 DecisionTreeReg

18.1.1 Introduction

The Apache Spark Decision Tree Regression implementation represents a sophisticated tree-based machine learning algorithm engineered for predicting continuous numerical values through hierarchical decision-making processes and recursive data partitioning strategies. This algorithm constructs a binary tree structure where each internal node represents a feature-based decision rule optimized for variance reduction, each branch corresponds to the outcome of that decision criterion, and each leaf node contains the predicted continuous value derived from training data patterns.

The decision tree learning process employs advanced recursive partitioning methodologies to identify optimal split points that minimize prediction variance and maximize information gain across the multidimensional feature space. The algorithm systematically evaluates potential split criteria for both numerical and categorical features, selecting the optimal thresholds that produce the most homogeneous child nodes in terms of target variable distribution.

The Spark implementation leverages distributed computing capabilities to efficiently handle large-scale datasets while maintaining statistical rigor in tree construction processes. The algorithm incorporates various impurity measures (variance reduction, mean absolute error), sophisticated pruning techniques, and regularization methods to prevent overfitting and ensure robust generalization performance across unseen data instances.

Decision trees provide inherent interpretability through their transparent decision-making structure and human-readable rule extraction capabilities, making them valuable for both predictive modeling applications and exploratory data analysis where model explainability is paramount.

For comprehensive technical documentation and implementation details, please refer to DecisionTreeRegression.

18.1.2 Syntax

The following syntax specification defines the Spark DecisionTreeReg function interface:

```
SELECT * FROM table(Unifyml.Spark.DecisionTreeReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.1.3 Input Arguments

The following comprehensive parameter specifications define the Spark DecisionTreeReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as decision tree splitting criteria a **DataType**: ARRAY
TARGETCOLUMN	Defines the continuous dependent variable column for regression prediction **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset f **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns dur **DataType**: ARRAY
MAXBINS	[optional] Controls feature discretization granularity and optimal split point selection for continuous variables in tree construction. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Constrains tree complexity by limiting maximum depth to prevent overfitting and maintain generalization. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Sets minimum information gain threshold for tree node splitting decisions and pruning control. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Controls random number generation for reproducible tree construc **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Configures proportional split ratio for training and validation data **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced tree perfo **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation or TrainValidatio for parameter optimization. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for automated bin count optimization during hyperparameter tuning processes. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for automated tree depth optimization during mode **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Parameter grid for automated information gain threshold optimiza **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed optimization during tree tuning pr **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation an **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust tree evaluation and parameter selection. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed tree processing op **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data pro **DataType**: STRING

APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reduction b **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained decision tree model and paramete **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved decision tree model persistence and future **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement c **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for improve **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation rang **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation rang **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and proce **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance **DataType**: INTEGER, **Default**: 64000000

18.1.4 Output

The following table describes the comprehensive evaluation metrics returned by the Spark DecisionTreeReg function:

Rmse	Root Mean Square Error: Measures standard deviation of decision tree prediction residuals, providing scale-dependent accuracy assessment for regression performance. **DataType**: DOUBLE
R2	Coefficient of Determination: Quantifies proportion of target variable variance explained by the decision tree model, ranging from 0 to 1. **DataType**: DOUBLE
Mse	Mean Squared Error: Calculates average squared differences between decision tree predictions and actual continuous values. **DataType**: DOUBLE
Mae	Mean Absolute Error: Computes average absolute differences between tree predictions and ground truth values. **DataType**: DOUBLE
Accuracy	Model Accuracy: Represents overall decision tree prediction correctness as percentage of accurately predicted instances within acceptable tolerance. **DataType**: DOUBLE

18.1.5 Examples

The following example demonstrates comprehensive implementation of the DecisionTreeReg SQL function with data preprocessing and scaling:

```
select * from table(Unifyml.Spark.DecisionTreeReg(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["crim", "zn", "indus", "chas",
"nox", "rm", "age", "dis", "rad", "tax", "ptratio", "b", "lstat"],
TARGETCOLUMN =>  ("medv"),
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates","removeoutliers"]
))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

18.2 RandomForestReg

18.2.1 Introduction

The UnifyML Random Forest Regression implementation represents an advanced ensemble learning methodology that employs bootstrap aggregating (bagging) techniques combined with sophisticated randomization strategies to construct multiple decision trees for robust continuous variable prediction. This state-of-the-art algorithm leverages ensemble diversity principles by training numerous decision trees on different bootstrap samples of training instances while incorporating random feature subset selection at each node split, subsequently aggregating their outputs through statistical averaging to produce final predictions with superior generalization capabilities and significantly reduced overfitting risk.

The Random Forest architecture incorporates dual randomization mechanisms: bootstrap sampling of training instances for each constituent tree and stochastic feature subset selection at every internal node split decision. This sophisticated randomization strategy creates decorrelated trees that collectively capture complex non-linear relationships, intricate feature interactions, and high-dimensional patterns while maintaining computational efficiency and providing interpretable feature importance metrics for analytical insights.

The ensemble methodology effectively reduces prediction variance compared to individual decision trees while preserving low bias characteristics, making it exceptionally suitable for complex regression tasks involving high-dimensional feature spaces, mixed data types, missing

values, and non-linear relationships. The algorithm provides inherent robustness against outliers and demonstrates superior performance across diverse domains including finance, healthcare, environmental modeling, and predictive analytics applications.

For comprehensive technical documentation and implementation details, please refer to RandomForestRegression.

18.2.2 Syntax

The following syntax specification defines the UnifyML RandomForestReg function interface:

```
SELECT * FROM table(Unifyml.RandomForestReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.2.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML RandomForestReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors for ens **DataType**: ARRAY
TARGETCOLUMN	Defines the continuous dependent variable column for regression prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset f **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns dur **DataType**: ARRAY
MAXBINS	[optional] Controls feature discretization granularity and optimal split point selection for continuous variables in decision tree construction. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Constrains individual tree complexity to prevent overfitting while maintaining ensemble diversity. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Sets minimum information gain threshold for tree node splitting decisions and pruning control. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Controls random number generation for reproducible bootstrap sa and feature selection. **DataType**: Long, **Default**: 235498149, **Min**: 0, **Max**: 235498149
TRAININGSAMPLESIZE	[optional] Configures proportional split ratio for training and validation data **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced ensembl **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation or TrainValidati for parameter optimization. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for automated bin count optimization during hyperparameter tuning processes. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for automated tree depth optimization during ense **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Parameter grid for automated information gain threshold optimiza **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed optimization during ensemble tun **DataType**: ARRAY, **Default**: 235498149, **Min**: 0, **Max**: 235498149
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation an **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust ensemble evaluation and parameter selection. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed ensemble processi **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data pr **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reducti before ensemble training. **DataType**: BOOLEAN, **Default**: false

INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained random forest ensemble model. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved ensemble model persistence and future inf **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement c **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for enhance **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation rang **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation rang **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and proce **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance **DataType**: INTEGER, **Default**: 64000000

18.2.4 Output

The following table describes the comprehensive evaluation metrics returned by the UnifyML RandomForestReg function:

Rmse	Root Mean Square Error: Measures standard deviation of ensemble prediction residuals, providing scale-dependent accuracy assessment for regression performance. **DataType**: DOUBLE
R2	Coefficient of Determination: Quantifies proportion of target variable variance explained by the random forest ensemble, ranging from 0 to 1. **DataType**: DOUBLE
Mse	Mean Squared Error: Calculates average squared differences between ensemble predictions and actual continuous values. **DataType**: DOUBLE
Mae	Mean Absolute Error: Computes average absolute differences between ensemble predictions and ground truth values. **DataType**: DOUBLE
Accuracy	Model Accuracy: Represents overall ensemble prediction correctness as percentage of accurately predicted instances within acceptable tolerance. **DataType**: DOUBLE

18.2.5 Examples

The following example demonstrates comprehensive implementation of the RandomForestReg SQL function with data preprocessing and model persistence:

```
select * from table(Unifyml.RandomForestReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
SAVEMODEL => true,
OUTPUTMODEL => ("modeRF")))

SQL Results:
+------------------------------+---------------------+
|        CoefficientName       |        Value        |
+------------------------------+---------------------+
| Mean Squared Error(MSE)      | 0.6991760258990803  |
| Mean Absolute Error(MAE)     | 0.6318749865613316  |
| Root Mean Squared Error(RMSE) | 0.8361674628321054  |
| Regression Score(R2)         | 0.12259107511444745 |
| Accuracy                     | 99.36812501343867   |
| OutputModelName              | modelRF             |
+------------------------------+---------------------+
```

18.3 GradientBoostTreePredict

18.3.1 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML GradientBoostTreePredict function:

INPUTMODELNAME	Specifies the trained gradient boosting ensemble model identifier for predicti **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns du **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consist **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of ensemble prediction results to databa **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar **DataType**: INTEGER, **Default**: 64000000

18.3.2 Output

The following table describes the prediction output structure from the UnifyML GradientBoostTreePredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained gradient boosting ensemble mode
PREDICTCOLUMN	Generated prediction values based on ensemble model inference and sequential ag

18.3.3 Examples

The following example demonstrates practical implementation of the GradientBoostTreePredict SQL function with real-time inference:

```
select * from table(Unifyml.GradientBoostPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeGBT")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+--------------------+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+--------------------+
```

18.4 LinReg

18.4.1 Introduction

The UnifyML Linear Regression implementation provides a comprehensive statistical modeling framework designed for analyzing and quantifying linear relationships between dependent and independent variables through sophisticated mathematical optimization techniques. This fundamental machine learning algorithm employs ordinary least squares (OLS) optimization to determine optimal regression coefficients that minimize the sum of squared residuals between predicted and observed values, ensuring statistically optimal parameter estimation under standard linear model assumptions.

The implementation supports both simple linear regression for single predictor variable scenarios and multiple linear regression for complex multivariate analysis involving multiple independent variables. The algorithm incorporates advanced regularization techniques including Ridge regression (L2 regularization), Lasso regression (L1 regularization), and Elastic Net regularization (combined L1/L2) to address multicollinearity issues, prevent overfitting, and enable automatic feature selection in high-dimensional datasets.

The linear regression framework assumes fundamental statistical properties including linear relationships between predictors and response variables, independence of observations, homoscedasticity (constant variance of residuals), and normality of residual distributions. These assumptions make linear regression particularly suitable for interpretable predictive modeling, statistical inference, hypothesis testing, and causal analysis applications where coefficient interpretation and statistical significance are paramount.

The implementation provides comprehensive diagnostic metrics, confidence intervals, and statistical significance tests, enabling thorough model validation and interpretation for research and business intelligence applications.

For comprehensive technical references and mathematical foundations, please consult Linear-Regression.

18.4.2 Syntax

The following syntax specification defines the UnifyML LinReg function interface:

```
SELECT * FROM table(Unifyml.LinReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ ELASTICPARAM => elasticparam ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ ELASTICPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.4.3 Input Arguments

The following comprehensive parameter specifications define the UnifyML LinReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors in the l **DataType**: ARRAY
TARGETCOLUMN	Defines the continuous dependent variable column for regression prediction **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset f **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns dur **DataType**: ARRAY
MAXITERNUM	[optional] Controls maximum optimization iterations for regression coefficie and parameter estimation stability. **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAM	[optional] Configures regularization strength parameter to control model com prevent overfitting, and improve generalization performance. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAM	[optional] Controls ElasticNet mixing parameter balancing Ridge (L2) and L regularization for optimal feature selection and coefficient shrinkage. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
TRAININGSAMPLESIZE	[optional] Defines proportional split ratio for training and validation dataset and model evaluation framework. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization through systematic gri for enhanced model performance and statistical significance. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation for robust evalua or TrainValidation for computational efficiency. **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[optional] Parameter grid for automated iteration count optimization during hyperparameter tuning and convergence analysis. **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Parameter grid for automated regularization strength optimization and overfitting prevention strategies. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAMPARAMS	[optional] Parameter grid for automated ElasticNet mixing parameter optimi and regularization technique selection. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation eff and computational performance optimization. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust statistical model eva and generalization performance assessment. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed processing optimi and scalable model training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data pro and computational load balancing. **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reducti and multicollinearity mitigation before regression analysis. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for consistent feature s transformation and dimensionality reduction. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained linear regression model para and coefficients for future inference. **DataType**: BOOLEAN, **Default**: false

18.4.4 Output

The following table describes the comprehensive statistical output metrics and diagnostic information from the UnifyML LinReg function:

Coefficients	Linear regression coefficients representing feature importance, directional impact, and quantitative relationship strength between predictors and target variable. **DataType**: ARRAY
Intercept	Y-intercept term representing the expected target value when all predictor variables equal zero, providing baseline prediction capability. **DataType**: DOUBLE
Rmse	Root Mean Square Error: Standard deviation of prediction residuals, measuring average prediction accuracy and model performance quality. **DataType**: DOUBLE
R2	Coefficient of Determination: Proportion of target variable variance explained by the linear regression model, ranging from 0 to 1. **DataType**: DOUBLE
NumInstances	Total number of training instances utilized in model estimation, parameter calculation, and statistical validation procedures. **DataType**: DOUBLE
MeanSquaredError	Mean Squared Error: Average squared differences between predicted and actual values, providing scale-dependent accuracy assessment. **DataType**: DOUBLE
MeanAbsoluteError	Mean Absolute Error: Average absolute differences between predictions and ground truth values, offering robust accuracy measurement. **DataType**: DOUBLE
TotalIterations	Total optimization iterations required for coefficient convergence and successful model training completion. **DataType**: DOUBLE
ExplainedVariance	Explained Variance Score: Proportion of target variable variance captured and explained by the regression model's predictive capability. **DataType**: DOUBLE
DegreesOfFreedom	Statistical degrees of freedom representing available information for parameter estimation and statistical inference procedures. **DataType**: DOUBLE
R2adj	Adjusted R-squared: Modified coefficient of determination accounting for predictor count and sample size to prevent overfitting bias. **DataType**: DOUBLE
Mse	Mean Squared Error: Alternative representation measuring average squared prediction errors for model evaluation. **DataType**: DOUBLE
Mae	Mean Absolute Error: Alternative representation measuring average absolute prediction errors for robust performance assessment. **DataType**: DOUBLE
Accuracy	Model Accuracy: Overall prediction correctness expressed as percentage of predictions within acceptable tolerance thresholds. **DataType**: DOUBLE

18.4.5 Examples

The following example demonstrates comprehensive implementation of the LinReg SQL function with advanced data preprocessing and distributed processing:

```
select * from table(Unifyml.LinReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.7495913109460369   |
| Mean Absolute Error(MAE)     | 0.6505508427404258   |
| Root Mean Squared Error(RMSE) | 0.8657894148960456   |
| Regression Score(R2)         | 0.059324001570318585 |
| Accuracy                     | 99.34944915725957    |
| Intercept                    | 7.185730896280032    |
| fixedacidity                 | -0.11587705201451137 |
| volatileacidity              | -1.6457566314599963  |
| citricacid                   | -0.11541540224980984 |
| NumInstances                 | 3499                 |
| MeanSquaredError             | 0.7417637021865621   |
| MeanAbsoluteError            | 0.6491153898723525   |
| TotalIterations              | 0                    |
| ExplainedVariance            | 0.0365265225712546   |
| DegreesOfFreedom             | 3495.0               |
| R2adj                        | 0.04611366607091327  |
+------------------------------+----------------------+
```

18.5 LinRegPredict

18.5.1 Introduction

The UnifyML LinRegPredict function provides advanced prediction capabilities utilizing pre-trained linear regression models for continuous target variable estimation and statistical inference. This sophisticated prediction engine applies the learned linear coefficients, intercept parameters, and statistical relationships to new data instances, generating precise predictions based on the established mathematical framework between predictor variables and the continuous target variable. The prediction process maintains rigorous mathematical consistency with the training phase, ensuring that feature scaling transformations, data preprocessing methodologies, and coefficient application procedures follow the identical statistical framework established during model development and validation.

The function architecture supports both real-time prediction scenarios and high-throughput batch processing workflows, enabling seamless integration of trained linear regression models into production environments, analytical pipelines, and decision support systems. The implementation preserves statistical properties including confidence intervals, prediction intervals, and uncertainty

quantification where applicable, providing comprehensive prediction capabilities for enterprise applications.

The prediction engine maintains computational efficiency while ensuring numerical stability and statistical accuracy, making it suitable for large-scale deployment scenarios requiring consistent and reliable linear model inference.

For comprehensive technical references and implementation guidelines, please consult Linear-Regression.

18.5.2 Syntax

The following syntax specification defines the UnifyML LinRegPredict function interface:

```
SELECT * FROM table(Unifyml.LinRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteouputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.5.3 Input Arguments

The following table provides comprehensive parameter specifications for the UnifyML LinRegPredict function:

INPUTMODELNAME	Specifies the trained linear regression model identifier for prediction executio **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns du **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme for improved prediction reliability. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and value imputation strategies. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consist with training phase methodology. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches for feature normalizatio **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and pr **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of linear regression prediction results to **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performan **DataType**: INTEGER, **Default**: 64000000

18.5.4 Output

The following table describes the prediction output structure from the UnifyML LinRegPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained linear regression model for predic
PREDICTCOLUMN	Generated continuous prediction values based on linear regression coefficients and

18.5.5 Examples

The following example demonstrates practical implementation of the LinRegPredict SQL function with real-time inference:

```
select * from table(Unifyml.LinRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelLR4")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

18.6 GradientBoostTree

18.6.1 Introduction

The Apache Spark Gradient Boosted Trees (GBT) implementation represents a sophisticated ensemble learning methodology designed for both classification and regression tasks through sequential model refinement and error minimization. This advanced machine learning technique belongs to the boosting family of ensemble algorithms, which systematically combine predictions from multiple weak learners—typically shallow decision trees—to construct a robust and highly accurate predictive model through iterative error correction and gradient-based optimization.

The Gradient Boosting framework employs a forward-stage additive modeling approach where each subsequent tree is trained to correct the residual errors of the previous ensemble, utilizing gradient descent optimization to minimize a specified loss function. This sequential learning process enables the algorithm to capture complex non-linear patterns, intricate feature interactions, and subtle data relationships that individual models might miss, resulting in superior predictive performance compared to single-model approaches.

The Spark implementation leverages distributed computing capabilities to efficiently handle large-scale datasets while maintaining mathematical rigor in the boosting process. The algorithm incorporates sophisticated regularization techniques including tree depth constraints, learning rate controls, and early stopping mechanisms to prevent overfitting and ensure robust generalization across diverse datasets and application domains.

For comprehensive technical documentation and mathematical foundations, please refer to GradientBoostedTree.

18.6.2 Syntax

The following syntax specification defines the Spark GradientBoostTree function interface:

```
SELECT * FROM table(Unifyml.Spark.GradientBoostTree(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MAXITERS => maxiters ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.6.3 Input Arguments

The following comprehensive parameter specifications define the Spark GradientBoostTree function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors for grad **DataType**: ARRAY
TARGETCOLUMN	Defines the target variable column for prediction, supporting both continuou **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset f **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns dur **DataType**: ARRAY
MAXBINS	[optional] Controls feature discretization granularity for continuous variable and optimal split point selection in tree construction. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Constrains individual tree complexity by limiting maximum depth to prevent overfitting while maintaining ensemble diversity. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERS	[optional] Specifies maximum boosting iterations controlling ensemble size and learning progression for optimal performance. **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAIN	[optional] Sets minimum information gain threshold for tree node splitting decisions and pruning control mechanisms. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Controls random number generation for reproducible model const and deterministic training processes. **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Configures proportional split ratio for training and validation dataset preparation and evaluation framework. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced ensemble performance and model selection. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation for robust evalua or TrainValidation for computational efficiency. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for automated bin count optimization during hyperparameter tuning and model selection processes. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for automated tree depth optimization during ense **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERNUMPARAMS	[optional] Parameter grid for automated boosting iteration optimization and ensemble size selection. **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAINPARAMS	[optional] Parameter grid for automated information gain threshold optimiza and splitting criteria refinement. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed optimization during ensemble tun **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation and ensemble construction efficiency. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust ensemble evaluation and statistical validation procedures. **DataType**: INTEGER, **Default**: 2

NUMPARTITION	[optional] Specifies data partitioning count for distributed ensemble processing c **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data proces and computational load balancing. **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reduction before ensemble training and feature transformation. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for consistent feature trans **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained gradient boosting ensemble mode and parameters for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved ensemble model persistence and deployme **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement c for improved model reliability. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and missing value treatment. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies and data completeness enhancement. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for improve numerical stability and convergence. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler StandardScaler, Normalizer, or Binarizer approaches. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation rang and feature value constraints. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation rang and scaling optimization. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and computational performance enhancement. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance and resource management efficiency. **DataType**: INTEGER, **Default**: 64000000

18.6.4 Output

The following table describes the comprehensive evaluation metrics returned by the Spark GradientBoostTree function:

Rmse	Root Mean Square Error: Measures standard deviation of ensemble prediction residuals, providing scale-dependent accuracy assessment and performance evaluation. **DataType**: DOUBLE
R2	Coefficient of Determination: Quantifies proportion of target variable variance explained by the gradient boosting ensemble model, ranging from 0 to 1. **DataType**: DOUBLE
Mse	Mean Squared Error: Calculates average squared differences between ensemble predictions and actual target values for accuracy assessment. **DataType**: DOUBLE
Mae	Mean Absolute Error: Computes average absolute differences between ensemble predictions and ground truth values for robust evaluation. **DataType**: DOUBLE
Accuracy	Model Accuracy: Represents overall ensemble prediction correctness as percentage of accurately predicted instances within acceptable tolerance. **DataType**: DOUBLE

18.6.5 Examples

The following example demonstrates comprehensive implementation of the GradientBoostTree SQL function with distributed processing and data preprocessing:

```
select * from table(Unifyml.Spark.GradientBoostTree(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
 "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"],
TARGETCOLUMN =>  ("MEDV"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

18.7 GradientBoostTreePredict

18.7.1 Introduction

The Apache Spark Gradient Boosting Trees prediction functionality provides advanced inference capabilities through sophisticated ensemble aggregation of multiple decision tree outputs, where each constituent tree contributes to the final prediction by systematically improving upon the residual errors of preceding models. This prediction framework leverages the sequential learning

methodology established during the gradient boosting training process, applying the learned ensemble structure to generate accurate predictions for new data instances through weighted combination of individual tree predictions.

The prediction engine maintains mathematical consistency with the training framework, ensuring that feature preprocessing, scaling transformations, and ensemble aggregation procedures follow identical methodologies established during model development. This approach guarantees reliable and consistent prediction quality while preserving the ensemble's ability to capture complex non-linear relationships and intricate feature interactions within the data.

For comprehensive technical documentation and implementation details, please refer to GradientBoostedTree.

18.7.2 Syntax

The following syntax specification defines the Spark GradientBoostTreePredict function interface:

```
SELECT * FROM table(Unifyml.Spark.GradientBoostPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.7.3 Input Arguments

The following comprehensive parameter specifications define the Spark GradientBoostTreePredict function configuration:

INPUTMODELNAME	Specifies the trained gradient boosting ensemble model identifier for predicti **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor varial for ensemble inference and prediction generation. **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns du **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme for improved prediction reliability and accuracy. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and missing value treatment strategie **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie during inference and data completeness enhancement. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consist with ensemble training methodology. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches for feature preprocessir **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r during prediction preprocessing. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation during prediction preprocessing. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and computational performance enhancement. **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of ensemble prediction results to databa for analysis and reporting purposes. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc and data management policies. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and external analysis capabilities. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar and resource management efficiency. **DataType**: INTEGER, **Default**: 64000000

18.7.4 Output

The following table describes the prediction output structure from the Spark GradientBoostTreePredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained gradient boosting ensemble mode
PREDICTCOLUMN	Generated prediction values based on ensemble model inference and sequential tre

18.7.5 Examples

The following example demonstrates practical implementation of the GradientBoostTreePredict SQL function with real-time inference:

```
select * from table(Unifyml.Spark.GradientBoostPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeGBT")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+--------------------+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+--------------------+
```

18.8 LinReg

18.8.1 Introduction

The Apache Spark Linear Regression implementation represents a fundamental yet sophisticated statistical modeling framework designed for analyzing and quantifying linear relationships between dependent and independent variables through advanced mathematical optimization and distributed computing capabilities. This comprehensive machine learning algorithm employs ordinary least squares (OLS) optimization combined with distributed processing to determine optimal regression coefficients that minimize the sum of squared residuals between predicted and observed values, ensuring statistically robust parameter estimation under standard linear model assumptions.

The implementation supports both simple linear regression for univariate analysis and multiple linear regression for complex multivariate modeling scenarios involving multiple independent variables. The algorithm incorporates advanced regularization techniques including Ridge regression (L2 regularization), Lasso regression (L1 regularization), and Elastic Net regularization (combined L1/L2 penalties) to effectively address multicollinearity issues, prevent overfitting in high-dimensional datasets, and enable automatic feature selection capabilities.

The Spark-based linear regression framework leverages distributed computing architecture to handle large-scale datasets efficiently while maintaining statistical rigor and numerical stability. The implementation assumes fundamental statistical properties including linearity of relationships, independence of observations, homoscedasticity of residuals, and normality of error distributions, making it particularly suitable for interpretable predictive modeling, statistical inference, hypothesis testing, and causal analysis applications where coefficient interpretation and statistical significance are paramount.

For comprehensive technical references and mathematical foundations, please consult Linear-Regression.

18.8.2 Syntax

The following syntax specification defines the Spark LinReg function interface:

```
SELECT * FROM table(Unifyml.Spark.LinReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ ELASTICPARAM => elasticparam ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ ELASTICPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.8.3 Input Arguments

The following comprehensive parameter specifications define the Spark LinReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors in the l **DataType**: ARRAY
TARGETCOLUMN	Defines the continuous dependent variable column for regression prediction **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset f **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns dur **DataType**: ARRAY
MAXITERNUM	[optional] Controls maximum optimization iterations for regression coefficie and numerical stability in distributed environments. **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAM	[optional] Configures regularization strength parameter to control model cor prevent overfitting, and improve generalization performance in high-dimensi **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAM	[optional] Controls ElasticNet mixing parameter balancing Ridge (L2) and L regularization for optimal feature selection and coefficient shrinkage strategi **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
TRAININGSAMPLESIZE	[optional] Defines proportional split ratio for training and validation dataset and statistical evaluation framework. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization through systematic gri for enhanced statistical performance and model selection. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation for robust statist or TrainValidation for computational efficiency in large-scale scenarios. **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[optional] Parameter grid for automated iteration count optimization during hyperparameter tuning and convergence analysis. **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Parameter grid for automated regularization strength optimization and overfitting prevention strategies. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
ELASTICPARAMPARAMS	[optional] Parameter grid for automated ElasticNet mixing parameter optimi and regularization technique selection. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation eff and distributed computation optimization. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust statistical model eva and generalization performance assessment. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed processing optimi and scalable model training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Defines column-based partitioning strategy for distributed data pro and computational load balancing. **DataType**: STRING
APPLYPCAMODEL	[optional] Enables Principal Component Analysis for dimensionality reducti and multicollinearity mitigation before regression analysis. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for consistent feature t and dimensionality reduction preprocessing. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained linear regression model para and coefficients for future inference and deployment.

DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement opera for improved statistical reliability and model validity. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and missing value treatment strategies. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies and data completeness enhancement procedures. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for improved numerical stability and convergence acceleration in distributed environments. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer approaches for feature preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation range and feature value constraints. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation range and feature scaling optimization. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and computational performance enhancement. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performance optim and resource management efficiency. **DataType**: INTEGER, **Default**: 64000000

18.8.4 Output

The following table describes the comprehensive statistical output metrics and diagnostic information from the Spark LinReg function:

Coefficients	Linear regression coefficients representing feature importance, directional impact, and quantitative relationship strength between predictors and target variable. **DataType**: ARRAY
Intercept	Y-intercept term representing the expected target value when all predictor variables equal zero, providing baseline prediction capability. **DataType**: DOUBLE
Rmse	Root Mean Square Error: Standard deviation of prediction residuals, measuring average prediction accuracy and model performance quality. **DataType**: DOUBLE
R2	Coefficient of Determination: Proportion of target variable variance explained by the linear regression model, ranging from 0 to 1. **DataType**: DOUBLE
NumInstances	Total number of training instances utilized in model estimation, parameter calculation, and statistical validation procedures. **DataType**: DOUBLE
MeanSquaredError	Mean Squared Error: Average squared differences between predicted and actual values, providing scale-dependent accuracy assessment. **DataType**: DOUBLE
MeanAbsoluteError	Mean Absolute Error: Average absolute differences between predictions and ground truth values, offering robust accuracy measurement. **DataType**: DOUBLE
TotalIterations	Total optimization iterations required for coefficient convergence and successful model training completion. **DataType**: DOUBLE
ExplainedVariance	Explained Variance Score: Proportion of target variable variance captured and explained by the regression model's predictive capability. **DataType**: DOUBLE
DegreesOfFreedom	Statistical degrees of freedom representing available information for parameter estimation and statistical inference procedures. **DataType**: DOUBLE
R2adj	Adjusted R-squared: Modified coefficient of determination accounting for predictor count and sample size to prevent overfitting bias. **DataType**: DOUBLE
Mse	Mean Squared Error: Alternative representation measuring average squared prediction errors for model evaluation and comparison. **DataType**: DOUBLE
Mae	Mean Absolute Error: Alternative representation measuring average absolute prediction errors for robust performance assessment. **DataType**: DOUBLE
Accuracy	Model Accuracy: Overall prediction correctness expressed as percentage of predictions within acceptable tolerance thresholds. **DataType**: DOUBLE

18.8.5 Examples

The following example demonstrates comprehensive implementation of the Spark LinReg SQL function with distributed processing and data preprocessing:

```
select * from table(Unifyml.Spark.LinReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.7495913109460369   |
| Mean Absolute Error(MAE)     | 0.6505508427404258   |
| Root Mean Squared Error(RMSE)| 0.8657894148960456   |
| Regression Score(R2)         | 0.059324001570318585 |
| Accuracy                     | 99.34944915725957    |
| Intercept                    | 7.185730896280032    |
| fixedacidity                 | -0.11587705201451137 |
| volatileacidity              | -1.6457566314599963  |
| citricacid                   | -0.11541540224980984 |
| NumInstances                 | 3499                 |
| MeanSquaredError             | 0.7417637021865621   |
| MeanAbsoluteError            | 0.6491153898723525   |
| TotalIterations              | 0                    |
| ExplainedVariance            | 0.0365265225712546   |
| DegreesOfFreedom             | 3495.0               |
| R2adj                        | 0.04611366607091327  |
+------------------------------+----------------------+
```

18.9 LinRegPredict

18.9.1 Introduction

The Apache Spark Linear Regression Prediction functionality provides enterprise-grade inference capabilities utilizing pre-trained linear regression models for continuous target variable estimation and statistical prediction in distributed computing environments. This sophisticated prediction engine applies the learned linear coefficients, intercept parameters, and statistical relationships established during training to new data instances, generating precise and statistically valid predictions based on the mathematical framework between predictor variables and the continuous target variable.

The prediction framework maintains rigorous mathematical consistency with the distributed training methodology, ensuring that feature scaling transformations, data preprocessing operations, and coefficient application procedures follow identical statistical frameworks established during model development and validation phases. This approach guarantees prediction reliability and statistical validity while leveraging Spark's distributed computing capabilities for scalable inference scenarios.

The implementation supports both real-time prediction workflows and high-throughput batch processing operations, enabling seamless integration of trained linear regression models into production environments, analytical pipelines, business intelligence systems, and decision support frameworks requiring statistical rigor and computational scalability.

For comprehensive technical references and implementation guidelines, please consult Linear-Regression.

18.9.2 Syntax

The following syntax specification defines the Spark LinRegPredict function interface:

```
SELECT * FROM table(Unifyml.Spark.LinRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.9.3 Input Arguments

The following comprehensive parameter specifications define the Spark LinRegPredict function configuration:

INPUTMODELNAME	Specifies the trained linear regression model identifier for prediction executio and statistical inference in distributed environments. **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor variat for inference and prediction generation. **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns during prediction processing and validation. **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhanceme for improved prediction reliability and statistical validity. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and missing value treatment strategie **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategie during inference and data completeness enhancement. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consiste with training phase methodology and distributed processing requirements. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxSc StandardScaler, Normalizer, or Binarizer approaches for feature preprocessin **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation r during distributed prediction processing. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation during distributed prediction processing. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and computational performance enhancement. **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of linear regression prediction results to database infrastructure for analysis and reporting. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and distributed storage management. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table struc and data management policies. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and external analytical system integration. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and statistical analysis results. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for performar and distributed resource management efficiency. **DataType**: INTEGER, **Default**: 64000000

18.9.4 Output

The following table describes the prediction output structure from the Spark LinRegPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained linear regression model for predic
PREDICTCOLUMN	Generated continuous prediction values based on linear regression coefficients and

18.9.5 Examples

The following example demonstrates practical implementation of the Spark LinRegPredict SQL function with distributed inference:

```
select * from table(Unifyml.Spark.LinRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelLR4")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

18.10 RandomForestReg

18.10.1 Introduction

The Apache Spark Random Forest Regression implementation represents a sophisticated ensemble learning methodology that employs bootstrap aggregating (bagging) techniques combined with advanced randomization strategies to construct multiple decision trees for robust continuous variable prediction in distributed computing environments. This state-of-the-art algorithm leverages ensemble diversity principles by training numerous decision trees on different bootstrap samples of training instances while incorporating stochastic feature subset selection at each node split, subsequently aggregating their outputs through statistical averaging to produce final predictions with superior generalization capabilities and significantly reduced overfitting risk.

The Random Forest architecture incorporates dual randomization mechanisms that create decorrelated trees capable of capturing complex non-linear relationships, intricate feature interactions, and high-dimensional patterns while maintaining computational efficiency in distributed environments. The algorithm provides inherent robustness against outliers, missing values, and noisy data while delivering interpretable feature importance metrics for analytical insights and model explainability.

The Spark implementation leverages distributed computing capabilities to handle large-scale datasets efficiently while maintaining statistical rigor in the ensemble construction process. The ensemble methodology effectively reduces prediction variance compared to individual decision trees while preserving low bias characteristics, making it exceptionally suitable for complex regression tasks involving high-dimensional feature spaces, mixed data types, and non-linear relationships across diverse application domains including finance, healthcare, environmental modeling, and predictive analytics.

For comprehensive technical documentation and implementation details, please refer to RandomForestRegression.

18.10.2 Syntax

The following syntax specification defines the Spark RandomForestReg function interface:

```
SELECT * FROM table(Unifyml.Spark.RandomForestReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.10.3 Input Arguments

The following comprehensive parameter specifications define the Spark RandomForestReg function configuration:

INPUTCOLUMNS	Specifies feature variable columns serving as independent predictors for ens in distributed Random Forest construction. **DataType**: ARRAY
TARGETCOLUMN	Defines the continuous dependent variable column for regression prediction and ensemble optimization. **DataType**: STRING
INPUTTABLE	[optional] Specifies source table, view, or query containing training dataset for distributed ensemble construction. **DataType**: STRING
INPUTDATATYPES	[optional] Defines explicit data type mappings for input feature columns during distributed processing and validation. **DataType**: ARRAY
MAXBINS	[optional] Controls feature discretization granularity and optimal split point selection for continuous variables in distributed tree construction. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Constrains individual tree complexity to prevent overfitting while maintaining ensemble diversity in distributed environments. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Sets minimum information gain threshold for tree node splitting decisions and pruning control in ensemble construction. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Controls random number generation for reproducible bootstrap sa and feature selection across distributed computing nodes. **DataType**: Long, **Default**: 235498149, **Min**: 0, **Max**: 235498149
TRAININGSAMPLESIZE	[optional] Configures proportional split ratio for training and validation datasets in distributed ensemble evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enables automated parameter optimization for enhanced ensembl performance through distributed grid search methodologies. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Selects validation methodology: CrossValidation for robust evalua or TrainValidation for computational efficiency in distributed scenarios. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for automated bin count optimization during distributed hyperparameter tuning and model selection processes. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for automated tree depth optimization during distributed ensemble tuning and complexity control. **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Parameter grid for automated information gain threshold optimiza and splitting criteria refinement in distributed environments. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed optimization during distributed ensemble tuning and reproducibility control. **DataType**: ARRAY, **Default**: 235498149, **Min**: 0, **Max**: 235498149
PARALLELISM	[optional] Configures parallel processing degree for parameter evaluation and distributed ensemble construction efficiency. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Defines cross-validation fold count for robust ensemble evaluation and statistical validation in distributed processing. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Specifies data partitioning count for distributed ensemble processi optimization and computational load balancing. **DataType**: INTEGER
	[optional] Defines column-based partitioning strategy for distributed data pr

INPUTPCAMODELNAME	[optional] Specifies pre-trained PCA model identifier for consistent feature transformation across distributed ensemble construction. **DataType**: STRING
SAVEMODEL	[optional] Enables persistent storage of trained Random Forest ensemble model and parameters for distributed inference deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Defines identifier for saved ensemble model persistence and distributed deployment across computing infrastructure. **DataType**: STRING
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement o for improved ensemble reliability in distributed environments. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and missing value treatment strategies. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies and data completeness enhancement in distributed processing. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations for improve ensemble convergence in distributed computing environments. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler StandardScaler, Normalizer, or Binarizer approaches for distributed preprocessin **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation rang in distributed feature preprocessing operations. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation rang in distributed feature preprocessing operations. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and distributed computational performance enhancement. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for distributed pe optimization and resource management efficiency. **DataType**: INTEGER, **Default**: 64000000

18.10.4 Output

The following table describes the comprehensive evaluation metrics returned by the Spark RandomForestReg function:

Rmse	Root Mean Square Error: Measures standard deviation of ensemble prediction residuals, providing scale-dependent accuracy assessment for distributed regression performance. **DataType**: DOUBLE
R2	Coefficient of Determination: Quantifies proportion of target variable variance explained by the distributed Random Forest ensemble, ranging from 0 to 1. **DataType**: DOUBLE
Mse	Mean Squared Error: Calculates average squared differences between ensemble predictions and actual continuous values in distributed evaluation. **DataType**: DOUBLE
Mae	Mean Absolute Error: Computes average absolute differences between ensemble predictions and ground truth values for robust distributed assessment. **DataType**: DOUBLE
Accuracy	Model Accuracy: Represents overall ensemble prediction correctness as percentage of accurately predicted instances within acceptable tolerance in distributed evaluation. **DataType**: DOUBLE

18.10.5 Examples

The following example demonstrates comprehensive implementation of the Spark RandomForestReg SQL function with distributed processing and model persistence:

```
select * from table(Unifyml.Spark.RandomForestReg(
INPUTTABLE => ("select * from winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
NUMPARTITION => 10,
PARTITIONCOLUMN => ("LSTAT"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
SAVEMODEL => true,
OUTPUTMODEL => ("modeRF")))

SQL Results:
+-------------------------------+---------------------+
|        CoefficientName        |        Value        |
+-------------------------------+---------------------+
| Mean Squared Error(MSE)       | 0.6991760258990803  |
| Mean Absolute Error(MAE)      | 0.6318749865613316  |
| Root Mean Squared Error(RMSE) | 0.8361674628321054  |
| Regression Score(R2)          | 0.12259107511444745 |
| Accuracy                      | 99.36812501343867   |
| OutputModelName               | modelRF             |
+-------------------------------+---------------------+
```

18.11 RandomForestRegPredict

18.11.1 Introduction

The Apache Spark Random Forest Regression Prediction functionality provides enterprise-grade inference capabilities utilizing pre-trained Random Forest ensemble models for continuous target

variable estimation in distributed computing environments. This sophisticated prediction engine applies the learned ensemble structure, bootstrap aggregation methodology, and feature randomization strategies established during distributed training to new data instances, generating robust and accurate predictions through democratic averaging of individual tree outputs across the distributed computing infrastructure.

The prediction framework maintains mathematical consistency with the distributed ensemble training methodology, ensuring that feature preprocessing operations, scaling transformations, and ensemble aggregation procedures follow identical statistical frameworks established during model development and validation phases. This approach guarantees prediction reliability and ensemble consistency while leveraging Spark's distributed computing capabilities for scalable, high-throughput inference scenarios.

The implementation supports both real-time prediction workflows and large-scale batch processing operations, enabling seamless integration of trained Random Forest ensemble models into production environments, analytical pipelines, business intelligence systems, and decision support frameworks requiring robust predictive capabilities and computational scalability across distributed infrastructure.

For comprehensive technical documentation and implementation details, please refer to RandomForestRegPredict.

18.11.2 Syntax

The following syntax specification defines the Spark RandomForestRegPredict function interface:

```
SELECT * FROM table(Unifyml.Spark.RandomForestRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.11.3 Input Arguments

The following comprehensive parameter specifications define the Spark RandomForestRegPredict function configuration:

INPUTMODELNAME	Specifies the trained Random Forest ensemble model identifier for distributed prediction execution and inference processing. **DataType**: STRING
INPUTTABLE	[optional] Defines the source table, view, or query containing predictor variables for distributed ensemble inference and prediction generation. **DataType**: STRING
INPUTDATATYPES	[optional] Specifies explicit data type mappings for input feature columns during distributed prediction processing and validation. **DataType**: ARRAY
DATACLEANING	[optional] Enables comprehensive data preprocessing and quality enhancement for improved prediction reliability in distributed environments. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Configures data cleansing methodologies including row removal, duplicate elimination, outlier detection, and missing value treatment strategies. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Specifies replacement values for missing data imputation strategies during distributed inference and data completeness enhancement. **DataType**: ARRAY
DATASCALING	[optional] Enables feature scaling and normalization transformations consistent with ensemble training methodology in distributed processing environments. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Defines scaling transformation methodology including MinMaxScaler, StandardScaler, Normalizer, or Binarizer approaches for distributed preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Sets minimum boundary value for MinMaxScaler transformation during distributed prediction preprocessing operations. **DataType**: INTEGER
MAXSCALER	[optional] Sets maximum boundary value for MinMaxScaler transformation during distributed prediction preprocessing operations. **DataType**: INTEGER
INPUTCACHEID	[optional] Specifies cached dataset identifier for optimized data access and distributed computational performance enhancement. **DataType**: STRING
SAVETODB	[optional] Enables persistent storage of ensemble prediction results to distributed database infrastructure for analysis and reporting purposes. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Defines target table identifier for prediction result persistence and distributed storage management across computing infrastructure. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Controls table replacement behavior for existing output table structures and distributed data management policies. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Enables CSV format export functionality for prediction results and external analytical system integration in distributed environments. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Specifies CSV filename identifier for exported prediction data and distributed analytical results processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Configures memory allocation per processing chunk for distributed optimization and resource management efficiency. **DataType**: INTEGER, **Default**: 64000000

18.11.4 Output

The following table describes the prediction output structure from the Spark RandomForestRegPredict function:

INPUTCOLUMNNAMES	Feature column identifiers utilized in the trained Random Forest ensemble model
PREDICTCOLUMN	Generated continuous prediction values based on ensemble aggregation and distrib

18.11.5 Examples

The following example demonstrates practical implementation of the Spark RandomForestRegPredict SQL function with distributed inference and feature scaling:

```
select * from table(Unifyml.Spark.RandomForestRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modeRF"),
DATASCALING => true,
SCALINGMETHOD => ("Normalizer")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.756806220503233 |
| 6.3          | 0.3             | 0.34       | 5.756806220503233 |
| 8.1          | 0.28            | 0.4        | 5.756806220503233 |
| 7.2          | 0.23            | 0.32       | 5.756806220503233 |
| 7.2          | 0.23            | 0.32       | 5.756806220503233 |
+--------------+-----------------+------------+-------------------+
```

18.12 Pca

18.12.1 Introduction

Spark PCA is a dimensionality reduction technique that employs orthogonal transformation to convert observations of potentially correlated variables into linearly independent components known as principal components.

For further references please see Pca.

18.12.2 Syntax

Below is the syntax for the Spark pca function.

```
SELECT * FROM table(Unifyml.Spark.Pca(
  NUMBEROFREDUCTIONS => ( numberOfReductions ) [,...]
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ INPUTMODELNAME => ( inputModelName ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ]  [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.12.3 Input Arguments

Below are the input arguments for the Spark pca function.

NUMBEROFREDUCTIONS	Number of principal components to extract. **DataType**: INTEGER
INPUTCOLUMNS	Names of columns containing the feature variables. **DataType**: ARRAY
INPUTTABLE	[optional] Name of the source table containing the data.
INPUTDATATYPES	[optional] Data type specifications for input columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Proportion of data allocated for training versus testing. **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Number of data partitions to create. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column used as the basis for data partitioning. **DataType**: STRING
SAVEMODEL	[optional] Whether to persist the trained model. **DataType**: BOOLEAN, **Default**: false
INPUTMODELNAME	[optional] Identifier for the model instance. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing and cleaning. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning techniques to apply (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Replacement values when using fill cleaning method. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling technique to use (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000

18.12.4 Output

Below is the expected output for the Spark pca function.

InputColumnNames	List of feature columns processed. **DataType**: ARRAY
NumberOfReductions	Count of principal components generated. **DataType**: INTEGER
PcaModelName	Identifier of the saved PCA model. **DataType**: STRING

18.12.5 Examples

Below are the some of the examples for running Pca SQL function.

```
select * from table(Unifyml.Spark.Pca(
INPUTTABLE => ("indiandiabetes"),
INPUTCOLUMNS => ARRAY["nooftimespregent", "age"],
NUMbEROFREDUCTIONS => 2,
SAVEMODEL => true,
INPUTMODELNAME => ("modelClassIDPca1")))

SQL Results:

+----------------------+--------------------+------------------+
|   InputColumnNames   | NumberOfReductions |   PcaModelName   |
+----------------------+--------------------+------------------+
| nooftimespregent, age | 2                  | modelClassIDPca1 |
+----------------------+--------------------+------------------+
```

18.13 Tokenizer

18.13.1 Introduction

A tokenizer is a natural language processing component that segments text documents into smaller linguistic units called tokens. These tokens may represent words, characters, or subword elements, depending on the tokenization strategy employed. Tokenization serves as a fundamental preprocessing step in various NLP applications, including text analysis, machine learning model preparation, and computational linguistics tasks.

18.13.2 Syntax

Below is the syntax for the Spark Tokenizer function.

```
SELECT * FROM table(Unifyml.Spark.Tokenizer(
  INPUTTABLE => ( { table | view  }) [,...]
  [ INPUTTEXTCOLUMNNAME => ( inputTextcolumn ) ] [,...]
  [ STOPWORDS => ( stopWords ) ] [,...]
  [ NGRAMS => (stemming ) ] [,...]
  [ SETNGRAMS => ( partsOfSpeech ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

18.13.3 Input Arguments

Below are the input arguments for the Spark Tokenizer function.

INPUTTABLE	Name of the source table containing text data.
INPUTTEXTCOLUMNNAME	[optional] Column containing text content for tokenization. **DataType**: STRING
PARALLELISM	[optional] Number of concurrent processing threads. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions to create. **DataType**: INTEGER
STOPWORDS	[optional] Enable removal of common stop words. **DataType**: BOOLEAN, **Default**: true
NGRAMS	[optional] Enable n-gram token generation. **DataType**: BOOLEAN,**Default**: false
SETNGRAMS	[optional] Specify the n-gram size parameter. **DataType**: INTEGER,**Default**: false
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for saved results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export. **DataType**: STRING

18.13.4 Output

Below is the expected output for the Spark Tokenizer function.

18.13.5 Examples

Below are the some of the examples for running Tokenizer SQL function.

```
select * from table(Unifyml.Spark.Tokenizer(
INPUTTABLE => ("medicaldrama"),
INPUTTEXTCOLUMNNAME => ("compliant"),
PARTITIONCOLUMN => ("id"),
OUTPUTCACHEID => ("tokencache")))

SQL Results:

+----------------+------------+--------------+
| InputTableName | DataCached | InputColumns |
+----------------+------------+--------------+
| medicaldrama   | true       | compliant    |
+----------------+------------+--------------+
```

19. ScikitLearn

This section provides details about the supported Scikitlearn Predictive algorithms.

19.1 AdaBoostRegressor

19.1.1 Introduction

Scikit-learn AdaBoost Regression (Adaptive Boosting Regression) is an ensemble machine learning method that enhances predictive accuracy by iteratively combining multiple weak regression models. The algorithm assigns greater weight to previously misclassified instances, enabling the ensemble to concentrate on challenging prediction cases and improve overall model performance.

For further references please see AdaBoostRegressor.

19.1.2 Syntax

Below is the syntax for the Scikitlearn AdaBoost Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.AdaBoostReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.1.3 Input Arguments

Below are the input arguments for the Scikitlearn AdaBoost Regression function.

INPUTCOLUMNS	Names of columns containing the feature variables. **DataType**: ARRAY
TARGETCOLUMN	Name of the dependent variable column for prediction. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used as the basis for data partitioning. **DataType**: STRING
INPUTTABLE	[optional] Name of the source table containing the data.
INPUTDATATYPES	[optional] Data type specifications for input columns. **DataType**: ARRAY
PARALLELISM	[optional] Number of concurrent processing threads. **DataType**: INTEGER, **Default**: 1
TRAININGSAMPLESIZE	[optional] Proportion of data allocated for training versus testing. **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Number of data partitions to create. **DataType**: INTEGER
SAVEMODEL	[optional] Whether to persist the trained model. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model instance. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing and cleaning. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning techniques to apply (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Replacement values when using fill cleaning method. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling technique to use (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000

19.1.4 Output

Below is the expected output for the Scikitlearn AdaBoost Regression function.

Rmse	Root Mean Square Error measuring prediction accuracy. **DataType**: DOUBLE
R2	Coefficient of determination indicating model fit quality. **DataType**: DOUBLE
Mse	Mean Squared Error of the regression model. **DataType**: DOUBLE
Mae	Mean Absolute Error of the regression model. **DataType**: DOUBLE
Accuracy	Overall predictive accuracy of the model. **DataType**: DOUBLE

19.1.5 Examples

Below are the some of the examples for running AdaBoost Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.AdaBoostReg(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
 "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B"],
TARGETCOLUMN =>  ("MEDV"),
PARTITIONCOLUMN => ("LSTAT"),
INPUTDATATYPES => ARRAY["CRIM:float","ZN:float","INDUS:float",
"CHAS:int","NOX:float","RM:float","AGE:float",
"DIS:float","RAD:float","TAX:float","PTRATIO:float",
"B:float","MEDV:float"],
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler")))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

19.2 AdaBoostRegPredict

19.2.1 Introduction

After training a Scikit-learn AdaBoost Regression model, it can be deployed to generate continuous target value predictions from new input datasets.

For further references please see AdaBoostRegressor.

19.2.2 Syntax

Below is the syntax for the Scikitlearn AdaBoostRegPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.AdaBoostRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.2.3 Input Arguments

Below are the input arguments for the Scikitlearn AdaBoostRegPredict function.

INPUTMODELNAME	Identifier of the trained model for prediction tasks. **DataType**: STRING
INPUTTABLE	[optional] Name of the source table containing the data.
PARTITIONCOLUMN	[optional] Column used as the basis for data partitioning. **DataType**: STRING
INPUTDATATYPES	[optional] Data type specifications for input columns. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing and cleaning. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning techniques to apply (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Replacement values when using fill cleaning method. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling technique to use (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
SAVETODB	[optional] Persist results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for saved results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000

19.2.4 Output

Below is the expected output for the Scikitlearn AdaBoostRegPredict function.

INPUTCOLUMNNAMES	Feature column names used in the model.
PREDICTCOLUMN	Generated prediction values.

19.2.5 Examples

Below are the some of the examples for running AdaBoostRegPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.AdaBoostRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelABR"),
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

19.3 BayesianRidge

19.3.1 Introduction

Bayesian Ridge Regression is a probabilistic linear regression approach that incorporates Bayesian statistical principles to estimate model parameters. This technique excels in scenarios involving multicollinearity among predictor variables and limited sample sizes.

For further references please see BayesianRidgeRegression.

19.3.2 Syntax

Below is the syntax for the Scikitlearn BayesianRidge Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.BayesianRidge(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.3.3 Input Arguments

Below are the input arguments for the Scikitlearn BayesianRidge Regression function.

INPUTCOLUMNS	Names of columns containing the feature variables. **DataType**: ARRAY
TARGETCOLUMN	Name of the dependent variable column for prediction. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used as the basis for data partitioning. **DataType**: STRING
INPUTTABLE	[optional] Name of the source table containing the data.
INPUTDATATYPES	[optional] Data type specifications for input columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Proportion of data allocated for training versus testing. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Number of concurrent processing threads. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions to create. **DataType**: INTEGER
SAVEMODEL	[optional] Whether to persist the trained model. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model instance. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing and cleaning. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning techniques to apply (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Replacement values when using fill cleaning method. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling technique to use (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000

19.3.4 Output

Below is the expected output for the Scikitlearn BayesianRidge Regression function.

Rmse	Root Mean Square Error measuring prediction accuracy. **DataType**: DOUBLE
R2	Coefficient of determination indicating model fit quality. **DataType**: DOUBLE
Mse	Mean Squared Error of the regression model. **DataType**: DOUBLE
Mae	Mean Absolute Error of the regression model. **DataType**: DOUBLE
Intercept	Y-intercept parameter of the linear model. **DataType**: DOUBLE
Coefficients	Feature weight coefficients of the model. **DataType**: DOUBLE
Accuracy	Overall predictive accuracy of the model. **DataType**: DOUBLE

19.3.5 Examples

Below are the some of the examples for running BayesianRidge Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.BayesianRidge(
INPUTTABLE => ("winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity",
"citricacid", "residualsugar", "chlorides",
"freesulfur", "totalsulfur", "density", "pH", "sulphates"],
TARGETCOLUMN =>  ("quality"),
PARTITIONCOLUMN => ("alcohol"),
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates",
"removeoutliers"]))

SQL Results:
+------------------------------+------------------------+
|        CoefficientName       |         Value          |
+------------------------------+------------------------+
| Mean Squared Error(MSE)      | 0.6645199319523911     |
| Mean Absolute Error(MAE)     | 0.6184913294740219     |
| Root Mean Squared Error(RMSE)| 0.8151809197671344     |
| Regression Score(R2)         | 0.07392676328661973    |
| Accuracy                     | 99.38150867052597      |
| fixedacidity                 | -0.002997609267413372  |
| volatileacidity              | -0.0003114103239641918 |
| citricacid                   | 0.0006857813002126994  |
| residualsugar                | -0.008991820664448046  |
| chlorides                    | -0.0002353503027216344 |
| freesulfur                   | 0.0129944275213258     |
| totalsulfur                  | -0.005867823024759452  |
| density                      | -6.656575212821852e-05 |
| pH                           | 0.0014835070931907548  |
| sulphates                    | 0.0005131486916260405  |
| Intercept                    | 6.37299740413461       |
+------------------------------+------------------------+
```

19.4 BayesianRidgePredict

19.4.1 Introduction

After training a Scikit-Learn Bayesian Ridge Regression model, it can be utilized to generate continuous target value predictions from new input datasets.

For further references please see BayesianRidgeRegression.

19.4.2 Syntax

Below is the syntax for the Scikitlearn BayesianRidgePredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.BayesianRidgePredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.4.3 Input Arguments

Below are the input arguments for the Scikitlearn BayesianRidgePredict function.

INPUTMODELNAME	Identifier of the trained model for prediction tasks. **DataType**: STRING
INPUTTABLE	[optional] Name of the source table containing the data.
PARTITIONCOLUMN	[optional] Column used as the basis for data partitioning. **DataType**: STRING
INPUTDATATYPES	[optional] Data type specifications for input columns. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing and cleaning. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning techniques to apply (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Replacement values when using fill cleaning method. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling technique to use (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
SAVETODB	[optional] Persist results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for saved results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000

19.4.4 Output

Below is the expected output for the Scikitlearn BayesianRidgePredict function.

INPUTCOLUMNNAMES	Feature column names used in the model.
PREDICTCOLUMN	Generated prediction values.

19.4.5 Examples

Below are the some of the examples for running BayesianRidgePredict SQL function.

```
select * from table(Unifyml.Scikitlearn.BayesianRidgePredict(
INPUTTABLE => ("winequality"),
INPUTMODELNAME => ("model12"),
PARTITIONCOLUMN => ("citricacid"))) limit 5

SQL Results:
+--------------+-----------------+------------+-----------------+
| fixedacidity | volatileacidity | citricacid | quality_Predict |
+--------------+-----------------+------------+-----------------+
| 7.000000     | 0.270000        | 0.360000   | 5.907366        |
| 6.300000     | 0.300000        | 0.340000   | 5.944923        |
| 8.100000     | 0.280000        | 0.400000   | 5.799947        |
| 7.2000       | 0.2300          | 0.3200     | 5.9026          |
| 7.2000       | 0.2300          | 0.3200     | 5.9026          |
+--------------+-----------------+------------+-----------------+
```

19.5 DecisionTreeReg

19.5.1 Introduction

Decision Tree Regression is a machine learning algorithm designed for predicting continuous numerical outcomes. This regression methodology constructs a hierarchical decision tree structure to model the relationships between input features and target variables.

For further references please see DecisionTreeRegressor.

19.5.2 Syntax

Below is the syntax for the Scikitlearn DecisionTreeReg function.

```
SELECT * FROM table(Unifyml.Scikitlearn.DecisionTreeReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => ( parallelism ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.5.3 Input Arguments

Below are the input arguments for the Scikitlearn DecisionTreeReg function.

INPUTCOLUMNS	Names of columns containing the feature variables. **DataType**: ARRAY
TARGETCOLUMN	Name of the dependent variable column for prediction. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used as the basis for data partitioning. **DataType**: STRING
INPUTTABLE	[optional] Name of the source table containing the data.
INPUTDATATYPES	[optional] Data type specifications for input columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Proportion of data allocated for training versus testing. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Number of concurrent processing threads. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions to create. **DataType**: INTEGER
SAVEMODEL	[optional] Whether to persist the trained model. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model instance. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing and cleaning. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning techniques to apply (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Replacement values when using fill cleaning method. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling technique to use (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data storage. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk. **DataType**: INTEGER, **Default**: 64000000

19.5.4 Output

Below is the expected output for the Scikitlearn DecisionTreeReg function.

Rmse	Root Mean Square Error measuring prediction accuracy. **DataType**: DOUBLE
R2	Coefficient of determination indicating model fit quality. **DataType**: DOUBLE
Mse	Mean Squared Error of the regression model. **DataType**: DOUBLE
Mae	Mean Absolute Error of the regression model. **DataType**: DOUBLE
Accuracy	Overall predictive accuracy of the model. **DataType**: DOUBLE

19.5.5 Examples

Below are the some of the examples for running DecisionTree Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.DecisionTreeReg(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
 "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B"],
TARGETCOLUMN =>  ("MEDV"),
PARTITIONCOLUMN => ("LSTAT"),
INPUTDATATYPES => ARRAY["CRIM:float","ZN:float","INDUS:float",
"CHAS:int","NOX:float","RM:float","AGE:float",
"DIS:float","RAD:float","TAX:float","PTRATIO:float",
"B:float","MEDV:float"],
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+-------------------------------+--------------------+
|        CoefficientName        |       Value        |
+-------------------------------+--------------------+
| Mean Squared Error(MSE)| Mean Squared Error(MSE)       | 7.281684313725489  |
| Mean Absolute Error(MAE)      | 2.24843137254902   |
| Root Mean Squared Error(RMSE) | 2.6984596186946153 |
| Regression Score(R2)          | 0.1593268057930256 |
| Accuracy                      | 97.75156862745098  |
+-------------------------------+--------------------+
```

19.6 DecisionTreeRegPredict

19.6.1 Introduction

In Decision Tree Regression inference, predictions are generated through traversal of the trained tree structure, where each input observation follows a path of binary decision nodes until reaching a terminal leaf node that contains the predicted continuous value for that specific region of the feature space. This prediction mechanism leverages the hierarchical partitioning learned during training to provide interpretable and deterministic regression estimates.

For further references please see DecisionTreeRegressor.

19.6.2 Syntax

Below is the syntax for the Scikitlearn DecisionTreeRegPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.DecisionTreeRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.6.3 Input Arguments

Below are the input arguments for the Scikitlearn DecisionTreeRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained decision tree regression model for inferen **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for prediction gen
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure tree **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during inf **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with training preprocessin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching training co **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and processin **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and r **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and result archiv **DataType**: STRING
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and r **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale inference **DataType**: INTEGER, **Default**: 64000000

19.6.4 Output

Below is the expected output for the Scikitlearn DecisionTreeRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during model training and inference pipelin
PREDICTCOLUMN	Continuous regression predictions generated through decision tree traversal and le

19.6.5 Examples

Below are the some of the examples for running DecisionTreeRegPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.DecisionTreeRegPredict(
INPUTTABLE => ("housing"),
INPUTMODELNAME => ("modelDTR1"),
INPUTDATATYPES => ARRAY["CRIM:float","ZN:float","INDUS:float",
"CHAS:int","NOX:float","RM:float","AGE:float",
"DIS:float","RAD:float","TAX:float","PTRATIO:float",
"B:float"],
PARTITIONCOLUMN => ("MEDV"))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

19.7 LassoLars

19.7.1 Introduction

Scikit-Learn LassoLars (Least Angle Regression with L1 regularization) represents a sophisticated algorithmic approach that combines the computational efficiency of Least Angle Regression (LARS) with the feature selection capabilities of Lasso regularization. This hybrid methodology provides an optimal solution for high-dimensional regression problems where feature sparsity is desired, offering superior computational performance compared to traditional coordinate descent methods, particularly when the number of features significantly exceeds the number of observations. The algorithm systematically adds features to the active set while maintaining the equiangular condition, making it exceptionally well-suited for scenarios requiring both predictive accuracy and model interpretability.

For further references please see LassoLarsRegression.

19.7.2 Syntax

Below is the syntax for the Scikitlearn LassoLars Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.LassoLars(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ ALPHA => ARRAYlpha ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => tuningType ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.7.3 Input Arguments

Below are the input arguments for the Scikitlearn LassoLars Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for spars **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent varia **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for mode
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure nur **DataType**: ARRAY
MAXITER	[optional] Maximum iteration limit for LARS algorithm convergence contro **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
ALPHA	[optional] L1 regularization strength parameter controlling feature sparsity a **DataType**: DOUBLE, **Default**: 1.0, **Min**: 0.0, **Max**: 1.0
TRAININGSAMPLESIZE	[optional] Train-test split ratio for model validation and generalization perfo **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal model **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Hyperparameter search strategy selection between exhaustive grid **DataType**: STRING, **Default**: Gridsearchcv
MAXITERNUMPARAMS	[optional] Hyperparameter grid for maximum iteration values during automa **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Regularization parameter search space for optimal sparsity-accura **DataType**: ARRAY, **Default**: 1.0, **Min**: 0.0, **Max**: 1.0
PARALLELISM	[optional] Degree of parallelization for computational optimization during h **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing and scalabilit **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and interpretability ana **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted LassoLars regression model art **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling missing val **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during pr **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved alg **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal perform **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computa **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale dataset op **DataType**: INTEGER, **Default**: 64000000

19.7.4 Output

Below is the expected output for the Scikitlearn LassoLars Regression function.

Rmse	Root Mean Square Error quantifying prediction accuracy and residual variance magnitude. **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and sparse model goodness-of-fit perf **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across validation samples. **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of prediction deviations from true values. **DataType**: DOUBLE
Intercept	Regression intercept parameter representing the baseline prediction value in the sparse model. **DataType**: DOUBLE
Coefficients	Sparse feature weight coefficients indicating selected features and their linear relationship streng **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing sparse model effectiveness on validation data **DataType**: DOUBLE

19.7.5 Examples

Below are the some of the examples for running LassoLars Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.LassoLars(
INPUTTABLE => ("winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
PARTITIONCOLUMN => ("alcohol"),
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates",
"removeoutliers"],
SAVEMODEL => true,
OUTPUTMODEL => ("model123")))

SQL Results:
+------------------------------+-----------------------+
|        CoefficientName       |         Value         |
+------------------------------+-----------------------+
| Mean Squared Error(MSE)      | 0.7813470302351285    |
| Mean Absolute Error(MAE)     | 0.6641990920881969    |
| Root Mean Squared Error(RMSE) | 0.8839383633688089    |
| Regression Score(R2)         | -0.001664759100461044 |
| Intercept                    | 5.885214007782102     |
| fixedacidity                 | 0.0                   |
| volatileacidity              | 0.0                   |
| citricacid                   | 0.0                   |
| Accuracy                     | 99.3358009079118      |
| OutputModelName              | model123              |
+------------------------------+-----------------------+
```

19.8 LassoLarsPredict

19.8.1 Introduction

After training a Scikit-Learn LassoLars regression model, the trained sparse model can be deployed for inference to generate continuous target value predictions on new datasets while maintaining the feature selection properties learned during training. This prediction function leverages the sparse coefficient structure to provide computationally efficient predictions with automatic feature selection.

For further references please see LassoLarsRegression.

19.8.2 Syntax

Below is the syntax for the Scikitlearn LassoLarsPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.LassoLarsPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.8.3 Input Arguments

Below are the input arguments for the Scikitlearn LassoLarsPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained LassoLars sparse regression model for in **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for sparse predicti
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during inf **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with training preprocessin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching training co **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computat **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and prediction r **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale sparse inf **DataType**: INTEGER, **Default**: 64000000

19.8.4 Output

Below is the expected output for the Scikitlearn LassoLarsPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during sparse model training and inference
PREDICTCOLUMN	Continuous regression predictions generated through sparse linear combination of

19.8.5 Examples

Below are the some of the examples for running LassoLarsPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.LassoLarsPredict(
INPUTTABLE => ("winequality"),
INPUTMODELNAME => ("model123"),
PARTITIONCOLUMN => ("citricacid"))) limit 5

SQL Results:
+--------------+-----------------+------------+-----------------+
| fixedacidity | volatileacidity | citricacid | quality_Predict |
+--------------+-----------------+------------+-----------------+
| 7.000000     | 0.270000        | 0.360000   | 5.885214        |
| 6.300000     | 0.300000        | 0.340000   | 5.885214        |
| 8.100000     | 0.280000        | 0.400000   | 5.885214        |
| 7.200000     | 0.230000        | 0.320000   | 5.885214        |
| 7.200000     | 0.230000        | 0.320000   | 5.885214        |
+--------------+-----------------+------------+-----------------+
```

19.9 LinReg

19.9.1 Introduction

Scikit-Learn Linear Regression represents the foundational statistical method for modeling linear relationships between predictor variables and a continuous response variable. This parametric approach seeks to identify the optimal hyperplane that minimizes the sum of squared residuals through ordinary least squares estimation, providing interpretable coefficients that quantify the marginal effect of each feature on the target variable. Linear regression serves as both a standalone predictive model and a baseline benchmark for evaluating more complex machine learning algorithms, offering computational efficiency and statistical interpretability crucial for many data science applications.

For further references please see LinearRegression.

19.9.2 Syntax

Below is the syntax for the Scikitlearn Linear Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.LinReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.9.3 Input Arguments

Below are the input arguments for the Scikitlearn Linear Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for linear regr **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent variable f **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize distribu **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for regression
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure numerica **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Train-test split ratio for model validation and generalization performanc **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelization for computational optimization during matrix **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing and linear algebra **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and coefficient interpretabili **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted linear regression model artifact and r **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust linear **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling missing values a **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during preproc **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved numeric **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal performance. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computational **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale dataset optimiz **DataType**: INTEGER, **Default**: 64000000

19.9.4 Output

Below is the expected output for the Scikitlearn Linear Regression function.

Rmse	Root Mean Square Error quantifying prediction accuracy and residual variance magnitude. **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and linear model goodness-of-fit perf **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across validation samples. **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of prediction deviations from observed value **DataType**: DOUBLE
Intercept	Regression intercept parameter representing the baseline prediction value when all features equal **DataType**: DOUBLE
Coefficients	Feature weight coefficients quantifying the linear relationship strength and direction between pre **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing linear model effectiveness on validation datas **DataType**: DOUBLE

19.9.5 Examples

Below are the some of the examples for runningLinear Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.LinReg(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
"RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B"],
TARGETCOLUMN =>  ("MEDV"),
PARTITIONCOLUMN => ("LSTAT"),
INPUTDATATYPES => ARRAY["CRIM:float","ZN:float","INDUS:float",
"CHAS:int","NOX:float","RM:float","AGE:float",
"DIS:float","RAD:float","TAX:float","PTRATIO:float",
"B:float","MEDV:float"],
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates","removeoutliers"]
))

SQL Results:
+--------------------------------+-----------------------+
|         CoefficientName        |         Value         |
+--------------------------------+-----------------------+
| Mean Squared Error(MSE)        | 33.58525563418508     |
| Mean Absolute Error(MAE)       | 3.810201235803673     |
| Root Mean Squared Error(RMSE)  | 5.7952787365393466    |
| Regression Score(R2)           | 0.6506106939715184    |
| Accuracy                       | 96.18979876419633     |
| Intercept                      | 30.344644109500997    |
| CRIM                           | -0.11928695491962961  |
| ZN                             | 0.04726546780246406   |
| INDUS                          | 0.048610970891376046  |
| CHAS                           | 2.9395751174276485    |
| NOX                            | -16.524640440134835   |
| RM                             | 4.450208055968567     |
| AGE                            | -0.006468585398538034 |
| DIS                            | -1.3586958237356161   |
| RAD                            | 0.27409583913772656   |
| TAX                            | -0.014841244923663446 |
| PTRATIO                        | -0.8649017257130465   |
| B                              | 0.007797138613616723  |
| LSTAT                          | -0.42011398849471493  |
+--------------------------------+-----------------------+
```

19.10 LinRegPredict

19.10.1 Introduction

After training a Scikit-Learn Linear Regression model, the fitted linear function can be deployed for inference to generate continuous target value predictions on new datasets by applying the learned linear transformation to input features. This prediction function leverages the estimated coefficients

and intercept to produce deterministic regression estimates through matrix multiplication operations.

For further references please see LinearRegression.

19.10.2 Syntax

Below is the syntax for the Scikitlearn LinRegPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.LinRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.10.3 Input Arguments

Below are the input arguments for the Scikitlearn LinRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained linear regression model for inference dep **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for linear predicti
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during inf **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with training preprocessin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching training co **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computat **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and prediction r **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale linear infe **DataType**: INTEGER, **Default**: 64000000

19.10.4 Output

Below is the expected output for the Scikitlearn LinRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during linear model training and inference p
PREDICTCOLUMN	Continuous regression predictions generated through linear combination of feature

19.10.5 Examples

Below are the some of the examples for running LinRegPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.LinRegPredict(
INPUTTABLE => ("housing"),
INPUTMODELNAME => ("modelLR1"),
INPUTDATATYPES => ARRAY["CRIM:float","ZN:float","INDUS:float",
"CHAS:int","NOX:float","RM:float","AGE:float",
"DIS:float","RAD:float","TAX:float","PTRATIO:float",
"B:float"],
PARTITIONCOLUMN => ("MEDV")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

19.11 RandomForestReg

19.11.1 Introduction

Scikit-Learn Random Forest Regression represents a sophisticated ensemble learning methodology that constructs multiple decision trees through bootstrap aggregating (bagging) and random feature subsampling to generate robust and accurate predictions. This ensemble approach mitigates individual tree overfitting through model averaging while capturing complex non-linear relationships and feature interactions inherent in the data. Random Forest leverages the wisdom of crowds principle, where the collective decision of multiple diverse trees typically outperforms any single tree, making it particularly effective for high-dimensional datasets with intricate patterns and providing built-in feature importance rankings for interpretability analysis.

For further references please see RandomForestRegression.

19.11.2 Syntax

Below is the syntax for the Scikitlearn RandomForest Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.RandomForestReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => tuningType ] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.11.3 Input Arguments

Below are the input arguments for the Scikitlearn RandomForest Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for ensem **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent varia **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for ensem
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure opti **DataType**: ARRAY
MAXDEPTH	[optional] Maximum tree depth constraint for controlling individual tree com **DataType**: INTEGER, **Default**: 2, **Min**: 0, **Max**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for ensemble model validation and overfitting **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal ensem **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Hyperparameter search strategy selection between exhaustive grid **DataType**: STRING, **Default**: Gridsearchcv
MAXDEPTHPARAMS	[optional] Hyperparameter grid for tree depth values during automated ensem **DataType**: ARRAY, **Default**: 2, **Min**: 0, **Max**: 2
PARALLELISM	[optional] Degree of parallelization for computational optimization during en **DataType**: INTEGER, **Default** : 1
NUMPARTITION	[optional] Number of data partitions for distributed processing and ensemble **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and ensemble interpret **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted Random Forest ensemble mode **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling missing val **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during en **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved en **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal ensembl **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computa **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale ensemble **DataType**: INTEGER, **Default**: 64000000

19.11.4 Output

Below is the expected output for the Scikitlearn RandomForest Regression function.

Rmse	Root Mean Square Error quantifying ensemble prediction accuracy and residual variance magnitud **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and ensemble model goodness-of-fit per **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across validation samples for en **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of ensemble prediction deviations from observe **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing ensemble model effectiveness on validation dat **DataType**: DOUBLE

19.11.5 Examples

Below are the some of the examples for running RandomForest Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.RandomForestReg(
INPUTTABLE => ("winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
PARTITIONCOLUMN => ("alcohol"),
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates",
"removeoutliers"],
SAVEMODEL => true,
OUTPUTMODEL => ("model1234")))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.7239800614887499   |
| Mean Absolute Error(MAE)     | 0.6319118976731708   |
| Root Mean Squared Error(RMSE)| 0.850870178986636    |
| Regression Score(R2)         | 0.045563225722054734 |
| Accuracy                     | 99.36808810232682    |
| OutputModelName              | model1234            |
+------------------------------+----------------------+
```

19.12 RandomForestRegPredict

19.12.1 Introduction

After training a Scikit-Learn Random Forest Regression ensemble, the trained model can be deployed for inference to generate continuous target value predictions by aggregating predictions from

multiple decision trees. This prediction function leverages the ensemble's collective intelligence to produce robust regression estimates while providing inherent uncertainty quantification through variance across individual tree predictions.

For further references please see RandomForestRegression.

19.12.2 Syntax

Below is the syntax for the Scikitlearn RandomForestRegPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.RandomForestRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.12.3 Input Arguments

Below are the input arguments for the Scikitlearn RandomForestRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained Random Forest ensemble model for infer **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for ensemble pred
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure tree **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during inf **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with training preprocessin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching training co **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computat **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and e **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and ensemble pr **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale ensemble **DataType**: INTEGER, **Default**: 64000000

19.12.4 Output

Below is the expected output for the Scikitlearn RandomForestRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during ensemble model training and inferen
PREDICTCOLUMN	Continuous regression predictions generated through ensemble averaging of indiv

19.12.5 Examples

Below are the some of the examples for running RandomForestRegPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.RandomForestRegPredict(
INPUTTABLE => ("winequality"),
INPUTMODELNAME => ("model1234"),
PARTITIONCOLUMN => ("citricacid"))) limit 5

SQL Results:
+--------------+-----------------+------------+-----------------+
| fixedacidity | volatileacidity | citricacid | quality_Predict |
+--------------+-----------------+------------+-----------------+
| 7.000000     | 0.270000        | 0.360000   | 5.972844        |
| 6.300000     | 0.300000        | 0.340000   | 5.980663        |
| 8.100000     | 0.280000        | 0.400000   | 5.845986        |
| 7.200000     | 0.230000        | 0.320000   | 6.016609        |
| 7.200000     | 0.230000        | 0.320000   | 6.016609        |
+--------------+-----------------+------------+-----------------+
```

19.13 Ridge

19.13.1 Introduction

Ridge Regression represents a regularized extension of ordinary least squares regression that incorporates L2 penalty terms to address multicollinearity and overfitting challenges in high-dimensional datasets. This technique shrinks coefficient estimates toward zero proportionally to the regularization strength, providing a bias-variance trade-off that often results in improved generalization performance. Ridge regression is particularly valuable when dealing with correlated predictor variables or when the number of features approaches or exceeds the number of observations, offering a principled approach to coefficient estimation under these challenging conditions.

For further references please see RidgeRegression.

19.13.2 Syntax

Below is the syntax for the Scikitlearn Ridge Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.Ridge(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ ALPHA => ARRAYlpha ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => tuningType ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.13.3 Input Arguments

Below are the input arguments for the Scikitlearn Ridge Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for regul **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent varia **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for regula
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure nur **DataType**: ARRAY
MAXITER	[optional] Maximum iteration limit for iterative solver convergence control a **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
ALPHA	[optional] L2 regularization strength parameter controlling coefficient shrink **DataType**: DOUBLE, **Default**: 1.0, **Min**: 0.0, **Max**: 1.0
TRAININGSAMPLESIZE	[optional] Train-test split ratio for regularized model validation and generali **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal regula **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Hyperparameter search strategy selection between exhaustive grid **DataType**: STRING, **Default**: Gridsearchcv
MAXITERNUMPARAMS	[optional] Hyperparameter grid for maximum iteration values during automa **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Regularization parameter search space for optimal bias-variance tr **DataType**: ARRAY, **Default**: 1.0, **Min**: 0.0, **Max**: 1.0
PARALLELISM	[optional] Degree of parallelization for computational optimization during re **DataType**: INTEGER, **Default**: 1
SAVEMODEL	[optional] Enable model serialization for deployment and regularized coeffic **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted Ridge regression model artifact **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling missing val **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during pr **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved reg **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal regulari **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computa **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale regularize **DataType**: INTEGER, **Default**: 64000000

19.13.4 Output

Below is the expected output for the Scikitlearn Ridge Regression function.

Rmse	Root Mean Square Error quantifying regularized model prediction accuracy and residual variance **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and regularized model goodness-of-fi **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across validation samples. **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of regularized prediction deviations from obs **DataType**: DOUBLE
Intercept	Regression intercept parameter representing the baseline prediction value in the regularized mode **DataType**: DOUBLE
Coefficients	Shrunk feature weight coefficients demonstrating regularization effects on parameter estimates. **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing regularized model effectiveness on validation **DataType**: DOUBLE

19.13.5 Examples

Below are the some of the examples for running Ridge Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.Ridge(
INPUTCACHEID => ("cacheid1"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
"RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B"],
TARGETCOLUMN =>  ("MEDV"),
MAXITER => 20,
ALPHA => 1.1,
TRAININGSAMPLESIZE => 0.6))

SQL Results:
+---------------------------------+-----------------------+
|         CoefficientName         |         Value         |
+---------------------------------+-----------------------+
| Mean Squared Error(MSE)         | 33.58525563418508     |
| Mean Absolute Error(MAE)        | 3.810201235803673     |
| Root Mean Squared Error(RMSE)   | 5.7952787365393466    |
| Regression Score(R2)            | 0.6506106939715184    |
| Accuracy                        | 96.18979876419633     |
| Intercept                       | 30.344644109500997    |
| CRIM                            | -0.11928695491962961  |
| ZN                              | 0.04726546780246406   |
| INDUS                           | 0.048610970891376046  |
| CHAS                            | 2.9395751174276485    |
| NOX                             | -16.524640440134835   |
| RM                              | 4.450208055968567     |
| AGE                             | -0.006468585398538034 |
| DIS                             | -1.3586958237356161   |
| RAD                             | 0.27409583913772656   |
| TAX                             | -0.014841244923663446 |
| PTRATIO                         | -0.8649017257130465   |
| B                               | 0.007797138613616723  |
| LSTAT                           | -0.42011398849471493  |
+---------------------------------+-----------------------+
```

19.14 RidgePredict

19.14.1 Introduction

After training a Scikit-Learn Ridge Regression model, the regularized linear function can be deployed for inference to generate continuous target value predictions by applying the learned shrunk coefficients to new input features. This prediction function leverages the Ridge regularization effects to provide stable and robust regression estimates, particularly effective when dealing with multicollinear features or high-dimensional datasets.

For further references please see RidgeRegression.

19.14.2 Syntax

Below is the syntax for the Scikitlearn RidgePredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.RidgePredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.14.3 Input Arguments

Below are the input arguments for the Scikitlearn RidgePredict function.

INPUTMODELNAME	Unique identifier of the pre-trained Ridge regression model for regularized in **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for regularized pre
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during inf **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with training preprocessin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching regularize **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computat **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and regularized **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale regularize **DataType**: INTEGER, **Default**: 64000000

19.14.4 Output

Below is the expected output for the Scikitlearn RidgePredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during regularized model training and infere
PREDICTCOLUMN	Continuous regression predictions generated through regularized linear combinati

19.14.5 Examples

Below are the some of the examples for running RidgePredict SQL function.

```
select * from table(Unifyml.Scikitlearn.RidgePredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelR1"),
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

19.15 SgdReg

19.15.1 Introduction

SGDRegressor in Scikit-Learn implements Stochastic Gradient Descent for linear regression, providing an efficient scalable approach to regression problems through iterative optimization. This method is particularly advantageous for large-scale datasets that cannot fit into memory, enabling incremental learning through mini-batch processing and online learning capabilities. SGD regression offers computational efficiency and memory scalability while supporting various loss functions and regularization techniques, making it an essential tool for big data regression applications and real-time learning scenarios.

For further references please see SgdRegression.

19.15.2 Syntax

Below is the syntax for the Scikitlearn Sgd Regression function.

```
SELECT * FROM table(Unifyml.Scikitlearn.SgdReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ ALPHA => ARRAYlpha ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => tuningType ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ CACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.15.3 Input Arguments

Below are the input arguments for the Scikitlearn Sgd Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for stoch **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent vari **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for stoch
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure nur **DataType**: ARRAY
MAXITER	[optional] Maximum iteration limit for stochastic gradient descent converger **DataType**: INTEGER, **Default**: 100, **Min**: 2, **Max**: 100
ALPHA	[optional] Regularization strength parameter controlling coefficient shrinkag **DataType**: DOUBLE, **Default**: 1.0, **Min**: 0.0, **Max**: 1.0
TRAININGSAMPLESIZE	[optional] Train-test split ratio for stochastic model validation and generaliza **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal stocha **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Hyperparameter search strategy selection between exhaustive grid **DataType**: STRING, **Default**: Gridsearchcv
MAXITERNUMPARAMS	[optional] Hyperparameter grid for maximum iteration values during automa **DataType**: ARRAY, **Default**: 100, **Min**: 2, **Max**: 100
REGPARAMPARAMS	[optional] Regularization parameter search space for optimal stochastic learn **DataType**: ARRAY, **Default**: 1.0, **Min**: 0.0, **Max**: 1.0
PARALLELISM	[optional] Degree of parallelization for computational optimization during st **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing and stochastic **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and stochastic coefficie **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted SGD regression model artifact a **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling missing val **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during sto **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved sto **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal stochast **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computa **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale stochastic **DataType**: INTEGER, **Default**: 64000000

19.15.4 Output

Below is the expected output for the Scikitlearn Sgd Regression function.

Rmse	Root Mean Square Error quantifying stochastic model prediction accuracy and convergence qual **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and stochastic model goodness-of-fit **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across validation samples. **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of stochastic prediction deviations from obse **DataType**: DOUBLE
Intercept	Regression intercept parameter learned through stochastic gradient descent optimization process. **DataType**: DOUBLE
Coefficients	Feature weight coefficients estimated through iterative stochastic optimization and regularization **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing stochastic model effectiveness on validation **DataType**: DOUBLE

19.15.5 Examples

Below are the some of the examples for running Sgd Regression SQL function.

```
select * from table(Unifyml.Scikitlearn.SgdReg(
INPUTTABLE => ("winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
PARTITIONCOLUMN => ("alcohol"),
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates",
"removeoutliers"],
SAVEMODEL => true,
OUTPUTMODEL => ("model12345")))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.8011462170319646   |
| Mean Absolute Error(MAE)     | 0.7119910794172286   |
| Root Mean Squared Error(RMSE)| 0.8950677164505291   |
| Regression Score(R2)         | -0.06100468448176577 |
| Accuracy                     | 99.28800892058277    |
| Intercept                    | [5.03346366]         |
| fixedacidity                 | 0.10442567037681104  |
| volatileacidity              | 0.001506042699080143 |
| citricacid                   | 0.012642533730014621 |
| Intercept                    | [5.03346366]         |
| OutputModelName              | model12345           |
+------------------------------+----------------------+
```

19.16 SgdRegPredict

19.16.1 Introduction

After training a Scikit-Learn SGDRegressor model, the stochastically optimized linear function can be deployed for inference to generate continuous target value predictions using the learned coefficients from the gradient descent optimization process. This prediction function leverages the computational efficiency of stochastic gradient descent while providing scalable inference capabilities for large-scale regression applications.

For further references please see SgdRegression.

19.16.2 Syntax

Below is the syntax for the Scikitlearn SgdRegPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.SgdRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

19.16.3 Input Arguments

Below are the input arguments for the Scikitlearn SgdRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained SGD regression model for stochastic infe **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for stochastic pred
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during inf **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with training preprocessin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching stochastic **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized data retrieval and computat **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and stochastic p **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale stochastic **DataType**: INTEGER, **Default**: 64000000

19.16.4 Output

Below is the expected output for the Scikitlearn SgdRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during stochastic model training and inferen
PREDICTCOLUMN	Continuous regression predictions generated through stochastic linear combinatio

19.16.5 Examples

Below are the some of the examples for running SgdRegPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.SgdRegPredict(
INPUTTABLE => ("winequality"),
INPUTMODELNAME => ("model12345"),
PARTITIONCOLUMN => ("citricacid"))) limit 5

SQL Results:
+--------------+-----------------+------------+-----------------+
| fixedacidity | volatileacidity | citricacid | quality_Predict |
+--------------+-----------------+------------+-----------------+
| 7.000000     | 0.270000        | 0.360000   | 5.769401        |
| 6.300000     | 0.300000        | 0.340000   | 5.696096        |
| 8.10000      | 0.28000         | 0.40000    | 5.88479         |
| 7.20000      | 0.23000         | 0.32000    | 5.78972         |
| 7.20000      | 0.23000         | 0.32000    | 5.78972         |
+--------------+-----------------+------------+-----------------+
```

20. Dask

20.1 LinReg

20.1.1 Introduction

Dask Linear Regression represents a distributed computing implementation of the fundamental statistical method for modeling linear relationships between predictor variables and continuous response variables. Built on the Dask parallel computing framework, this approach enables scalable regression analysis across distributed datasets that exceed single-machine memory limitations. Dask Linear Regression leverages parallel processing and out-of-core computation to handle big data scenarios while maintaining the interpretability and statistical foundations of traditional linear regression, making it an essential tool for large-scale data science applications requiring both computational scalability and analytical rigor.

For further references please see LinearRegression.

20.1.2 Syntax

Below is the syntax for the dask Linear Regression function.

```
SELECT * FROM table(Unifyml.Dask.LinReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ SOLVER => ( solver ) ] [,...]
  [ MAXITER => ( maxiter ) ] [,...]
  [ PENALTY => ( penalty ) ] [,...]
  [ TOL => ( tol ) ] [,...]
  [ C => ( c ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.1.3 Input Arguments

Below are the input arguments for the dask Linear Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for distributed **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent variable f **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize distribu **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for distributed
SOLVER	[optional] Optimization algorithm selection for distributed coefficient estimation a **DataType**: STRING,**Default**: 'admm'
MAXITER	[optional] Maximum iteration limit for distributed solver convergence control and **DataType**: INTEGER,**Default**: 100
PENALTY	[optional] Regularization method selection for distributed model complexity contr **DataType**: STRING,**Default**: 'l2'
TOL	[optional] Convergence tolerance threshold for distributed optimization algorithm **DataType**: DOUBLE,**Default**: 0.0001
C	[optional] Regularization strength parameter controlling distributed model comple **DataType**: DOUBLE,**Default**: 1.0
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure numerica **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed model validation and generalization **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelization for computational optimization during distribu **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing architecture and cl **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and distributed coefficient in **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted distributed linear regression model ar **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust distri **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling distributed data a **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during distribu **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved distribu **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal distributed pe **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval and co **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed data **DataType**: INTEGER, **Default**: 64000000

20.1.4 Output

Below is the expected output for the dask Linear Regression function.

Rmse	Root Mean Square Error quantifying distributed model prediction accuracy and residual variance **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and distributed model goodness-of-fit **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across distributed validation **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of distributed prediction deviations from obs **DataType**: DOUBLE
Intercept	Regression intercept parameter estimated through distributed optimization and aggregated across **DataType**: DOUBLE
Coefficients	Feature weight coefficients computed through distributed linear algebra and consensus optimizat **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing distributed model effectiveness on validation **DataType**: DOUBLE

20.1.5 Examples

Below are the some of the examples for running Linear Regression SQL function.

```
select * from table(Unifyml.Dask.LinReg(
INPUTTABLE => ("housing"),
INPUTCOLUMNS => ARRAY["CRIM", "ZN", "INDUS", "CHAS", "NOX",
"RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B"],
TARGETCOLUMN =>  ("MEDV"),
PARTITIONCOLUMN => ("LSTAT"),
INPUTDATATYPES => ARRAY["CRIM:float","ZN:float","INDUS:float",
"CHAS:int","NOX:float","RM:float","AGE:float",
"DIS:float","RAD:float","TAX:float","PTRATIO:float",
"B:float","MEDV:float"],
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates","removeoutliers"]
))

SQL Results:
+--------------------------------+----------------------+
|         CoefficientName        |         Value        |
+--------------------------------+----------------------+
| Mean Squared Error(MSE)        | 33.58525563418508    |
| Mean Absolute Error(MAE)       | 3.810201235803673    |
| Root Mean Squared Error(RMSE)  | 5.7952787365393466   |
| Regression Score(R2)           | 0.6506106939715184   |
| Accuracy                       | 96.18979876419633    |
| Intercept                      | 30.344644109500997   |
| CRIM                           | -0.11928695491962961 |
| ZN                             | 0.04726546780246406  |
| INDUS                          | 0.048610970891376046 |
| CHAS                           | 2.9395751174276485   |
| NOX                            | -16.524640440134835  |
| RM                             | 4.450208055968567    |
| AGE                            | -0.006468585398538034|
| DIS                            | -1.3586958237356161  |
| RAD                            | 0.27409583913772656  |
| TAX                            | -0.014841244923663446|
| PTRATIO                        | -0.8649017257130465  |
| B                              | 0.007797138613616723 |
| LSTAT                          | -0.42011398849471493 |
+--------------------------------+----------------------+
```

20.2 LinRegPredict

20.2.1 Introduction

After training a Dask Linear Regression model, the distributed linear function can be deployed for scalable inference to generate continuous target value predictions across large datasets using

the learned coefficients from distributed optimization. This prediction function leverages Dask's parallel computing capabilities to provide efficient and scalable regression inference for big data applications while maintaining computational consistency across distributed partitions.

For further references please see LinearRegression.

20.2.2 Syntax

Below is the syntax for the dask LinRegPredict function.

```
SELECT * FROM table(Unifyml.Dask.LinRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.2.3 Input Arguments

Below are the input arguments for the dask LinRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained distributed linear regression model for sc **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for distributed pre
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during dis **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with distributed training p **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching distributed **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval ar **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured prediction results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and distributed p **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed **DataType**: INTEGER, **Default**: 64000000

20.2.4 Output

Below is the expected output for the dask LinRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during distributed model training and infere
PREDICTCOLUMN	Continuous regression predictions generated through distributed linear combinatio

20.2.5 Examples

Below are the some of the examples for running LinRegPredict SQL function.

```
select * from table(Unifyml.Dask.LinRegPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modelLR"),
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler")))

SQL Results:
+---------+------+-------+------+--------------------+
|  crim   |  zn  | indus | chas |    medv_Predict    |
+---------+------+-------+------+-------+-------+----+
| 0.00632 | 18.0 | 2.31  | 0    | 27.20357142857143  |
| 0.02731 | 0.0  | 7.07  | 0    | 22.977777777777778 |
| 0.02729 | 0.0  | 7.07  | 0    | 31.659374999999997 |
| 0.03237 | 0.0  | 2.18  | 0    | 31.659374999999997 |
| 0.06905 | 0.0  | 2.18  | 0    | 31.659374999999997 |
+---------+------+-------+------+-------+-------+----+
```

20.3 LogisticReg

20.3.1 Introduction

Dask Logistic Regression represents a distributed computing implementation of the statistical classification method designed for binary and multinomial classification problems across large-scale datasets. Built on the Dask parallel computing framework, this approach enables scalable logistic regression analysis that can handle datasets exceeding single-machine memory limitations while maintaining the probabilistic foundations and interpretability of traditional logistic regression. The distributed implementation leverages parallel processing and out-of-core computation to provide efficient classification modeling for big data scenarios requiring both computational scalability and statistical rigor.

For further references please see LogisticRegression.

20.3.2 Syntax

Below is the syntax for the dask Logistic Regression function.

```
SELECT * FROM table(Unifyml.Dask.LogisticReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ SOLVER => ( solver ) ] [,...]
  [ MAXITER => ( maxiter ) ] [,...]
  [ PENALTY => ( penalty ) ] [,...]
  [ TOL => ( tol ) ] [,...]
  [ C => ( c ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.3.3 Input Arguments

Below are the input arguments for the dask Logistic Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for distributed **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the categorical dependent variable f **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize distribu **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for distributed
SOLVER	[optional] Optimization algorithm selection for distributed maximum likelihood e **DataType**: STRING,**Default**: 'admm'
MAXITER	[optional] Maximum iteration limit for distributed logistic solver convergence cor **DataType**: INTEGER,**Default**: 100
PENALTY	[optional] Regularization method selection for distributed model complexity contr **DataType**: STRING,**Default**: 'l2'
TOL	[optional] Convergence tolerance threshold for distributed logistic optimization al **DataType**: DOUBLE,**Default**: 0.0001
C	[optional] Regularization strength parameter controlling distributed logistic mode **DataType**: DOUBLE,**Default**: 1.0
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure numerica **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed model validation and generalization **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelization for computational optimization during distribu **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing architecture and cl **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and distributed logistic coef **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted distributed logistic regression model **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust distri **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling distributed data a **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during distribu **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved distribu **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal distributed pe **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval and co **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed data **DataType**: INTEGER, **Default**: 64000000

20.3.4 Output

Below is the expected output for the dask Logistic Regression function.

Rmse	Root Mean Square Error quantifying distributed logistic model prediction accuracy across partiti **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance in distributed logistic model performa **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across distributed validation **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of distributed classification prediction deviati **DataType**: DOUBLE
Intercept	Logistic regression intercept parameter estimated through distributed maximum likelihood optim **DataType**: DOUBLE
Coefficients	Feature weight coefficients computed through distributed logistic optimization and consensus alg **DataType**: DOUBLE
Accuracy	Overall classification performance metric representing distributed model effectiveness on validat **DataType**: DOUBLE

20.3.5 Examples

Below are the some of the examples for running Logistic Regression SQL function.

```
select * from table(Unifyml.Dask.LogisticReg(
INPUTTABLE => ("winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
PARTITIONCOLUMN => ("alcohol"),
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates",
"removeoutliers"],
SAVEMODEL => true,
OUTPUTMODEL => ("model23")))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |        Value       |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 24.811248808388942 |
| Mean Absolute Error(MAE)     | 4.902764537654909  |
| Root Mean Squared Error(RMSE)| 4.981089118695724  |
| Regression Score(R2)         | -31.049719794241867|
| Accuracy                     | 95.0972354623451   |
| fixedacidity                 | 13836.044973175562 |
| volatileacidity              | -222.22517993854592|
| citricacid                   | -3390.564674533608 |
| Intercept                    | 523.5981376204364  |
| OutputModelName              | model23            |
+------------------------------+--------------------+
```

20.4 LogisticRegPredict

20.4.1 Introduction

After training a Dask Logistic Regression model, the distributed classification function can be deployed for scalable inference to generate class probability predictions and categorical classifications across large datasets using the learned coefficients from distributed maximum likelihood estimation. This prediction function leverages Dask's parallel computing capabilities to provide efficient and scalable classification inference for big data applications while maintaining probabilistic consistency across distributed partitions.

For further references please see LogisticRegression.

20.4.2 Syntax

Below is the syntax for the dask LogisticRegPredict function.

```
SELECT * FROM table(Unifyml.Dask.LogisticRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.4.3 Input Arguments

Below are the input arguments for the dask LogisticRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained distributed logistic regression model for s **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for distributed cla
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during dis **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with distributed training p **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching distributed **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval ar **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured classification predictio **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and d **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and distributed d **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed **DataType**: INTEGER, **Default**: 64000000

20.4.4 Output

Below s the expected output for the dask LogisticRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during distributed classification model train
PREDICTCOLUMN	Binary or multinomial classification predictions generated through distributed logi

20.4.5 Examples

Below are the some of the examples for running LogisticRegPredict SQL function.

```
select * from table(Unifyml.Dask.LogisticRegPredict(
INPUTTABLE => ("winequality"),
INPUTMODELNAME => ("model23"),
PARTITIONCOLUMN => ("citricacid"))) limit 5

SQL Results:
+--------------+-----------------+------------+-----------------+
| fixedacidity | volatileacidity | citricacid | quality_Predict |
+--------------+-----------------+------------+-----------------+
| 7            | 0.27            | 0.36       | True            |
| 6.3          | 0.3             | 0.34       | True            |
| 8.1          | 0.28            | 0.4        | True            |
| 7.2          | 0.23            | 0.32       | True            |
| 7.2          | 0.23            | 0.32       | True            |
+--------------+-----------------+------------+-----------------+
```

20.5 PoissonReg

20.5.1 Introduction

Dask Poisson Regression represents a distributed computing implementation of the generalized linear model specifically designed for count data and rate data that follows a Poisson distribution. Built on the Dask parallel computing framework, this approach enables scalable analysis of count phenomena across large-scale datasets that exceed single-machine memory limitations. Poisson regression in Dask is particularly valuable for modeling discrete events occurring within fixed time intervals or spatial regions, providing both the statistical rigor of traditional Poisson modeling and the computational scalability necessary for big data applications in epidemiology, quality control, and event analysis.

For further references please see PoissonRegression.

20.5.2 Syntax

Below is the syntax for the dask Poisson Regression function.

```
SELECT * FROM table(Unifyml.Dask.PoissonReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ SOLVER => ( solver ) ] [,...]
  [ MAXITER => ( maxiter ) ] [,...]
  [ PENALTY => ( penalty ) ] [,...]
  [ TOL => ( tol ) ] [,...]
  [ C => ( c ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.5.3 Input Arguments

Below are the input arguments for the dask Poisson Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for distributed **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the count or rate dependent variable **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize distribu **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for distributed
SOLVER	[optional] Optimization algorithm selection for distributed maximum likelihood e **DataType**: STRING,**Default**: 'admm'
MAXITER	[optional] Maximum iteration limit for distributed Poisson solver convergence co **DataType**: INTEGER,**Default**: 100
PENALTY	[optional] Regularization method selection for distributed Poisson model complex **DataType**: STRING,**Default**: 'l2'
TOL	[optional] Convergence tolerance threshold for distributed Poisson optimization a **DataType**: DOUBLE,**Default**: 0.0001
C	[optional] Regularization strength parameter controlling distributed Poisson mode **DataType**: DOUBLE,**Default**: 1.0
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure numerica **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed Poisson model validation and gener **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelization for computational optimization during distribu **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing architecture and cl **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and distributed Poisson coef **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted distributed Poisson regression model **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust distri **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling distributed coun **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during distribu **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved distribu **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal distributed Po **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval and co **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed cou **DataType**: INTEGER, **Default**: 64000000

20.5.4 Output

Below is the expected for the dask Poisson Regression function.

Rmse	Root Mean Square Error quantifying distributed Poisson model prediction accuracy for count dat **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance in distributed Poisson model performa **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across distributed count valid **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of distributed count prediction deviations fro **DataType**: DOUBLE
Intercept	Poisson regression intercept parameter estimated through distributed maximum likelihood optimi **DataType**: DOUBLE
Coefficients	Feature weight coefficients computed through distributed Poisson optimization representing log-l **DataType**: DOUBLE
Accuracy	Overall count prediction performance metric representing distributed Poisson model effectivenes **DataType**: DOUBLE
Deviance	Poisson deviance statistic measuring goodness-of-fit for distributed count data modeling and mod **DataType**: DOUBLE

20.5.5 Examples

Below are the some of the examples for running Poisson Regression SQL function.

```
select * from table(Unifyml.Dask.PoissonReg(
INPUTTABLE => ("winequality"),
INPUTCOLUMNS => ARRAY["fixedacidity", "volatileacidity", "citricacid"],
TARGETCOLUMN =>  ("quality"),
PARTITIONCOLUMN => ("alcohol"),
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates",
"removeoutliers"],
SAVEMODEL => true,
OUTPUTMODEL => ("model234")))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |         Value        |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.7718038786322746   |
| Mean Absolute Error(MAE)     | 0.6686656382108017   |
| Root Mean Squared Error(RMSE)| 0.87852369269831     |
| Regression Score(R2)         | 0.0030288988150435348|
| Deviance                     | 19534.725804487127   |
| Accuracy                     | 99.3313343617892     |
| fixedacidity                 | 1.9302754497669388   |
| volatileacidity              | -0.017782512887470736|
| citricacid                   | -0.2395607288654851  |
| Intercept                    | 0.052074279629048655 |
| OutputModelName              | model234             |
+------------------------------+----------------------+
```

20.6 PoissonRegPredict

20.6.1 Introduction

After training a Dask Poisson Regression model, the distributed count prediction function can be deployed for scalable inference to generate count value predictions across large datasets using the learned coefficients from distributed maximum likelihood estimation. This prediction function leverages Dask's parallel computing capabilities to provide efficient and scalable count data inference for big data applications while maintaining statistical consistency and Poisson distribution properties across distributed partitions.

For further references please see PoissonRegressionPredict.

20.6.2 Syntax

Below is the syntax for the dask PoissonRegPredict function.

```
SELECT * FROM table(Unifyml.Dask.PoissonRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.6.3 Input Arguments

Below are the input arguments for the dask PoissonRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained distributed Poisson regression model for **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for distributed cou
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during dis **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with distributed training p **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching distributed **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval ar **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured count prediction result **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and d **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and distributed d **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed **DataType**: INTEGER, **Default**: 64000000

20.6.4 Output

Below is the expected output for the dask PoissonRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during distributed Poisson model training ar
PREDICTCOLUMN	Count value predictions generated through distributed Poisson exponential transfo

20.6.5 Examples

Below are the some of the examples for running PoissonRegPredict SQL function.

```
select * from table(Unifyml.Dask.PoissonRegPredict(
INPUTTABLE => ("winequality"),
INPUTMODELNAME => ("model234"),
PARTITIONCOLUMN => ("citricacid"))) limit 5

SQL Results:
+--------------+-----------------+------------+-----------------+
| fixedacidity | volatileacidity | citricacid | quality_Predict |
+--------------+-----------------+------------+-----------------+
| 7.000000     | 0.270000        | 0.360000   | 5.811642        |
| 6.300000     | 0.300000        | 0.340000   | 5.836216        |
| 8.100000     | 0.280000        | 0.400000   | 5.697285        |
| 7.200000     | 0.230000        | 0.320000   | 5.834603        |
| 7.200000     | 0.230000        | 0.320000   | 5.834603        |
+--------------+-----------------+------------+-----------------+
```

20.7 XgbReg

20.7.1 Introduction

Dask XGBRegressor represents a distributed computing implementation of XGBoost (Extreme Gradient Boosting) specifically optimized for regression tasks across large-scale datasets. Built on the Dask parallel computing framework, this approach combines the powerful ensemble learning capabilities of XGBoost with distributed processing to handle datasets that exceed single-machine memory limitations. The implementation leverages parallel tree construction, distributed gradient computation, and optimized communication protocols to provide scalable gradient boosting regression while maintaining the superior predictive performance and feature importance analysis capabilities that make XGBoost a leading choice for competitive machine learning applications.

For further references please see XGBRegressor.

20.7.2 Syntax

Below is the syntax for the dask XgbReg Regression function.

```
SELECT * FROM table(Unifyml.Dask.XgbReg(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.7.3 Input Arguments

Below are the input arguments for the dask XgbReg Regression function.

INPUTCOLUMNS	Feature matrix column identifiers containing independent variables for distributed **DataType**: ARRAY
TARGETCOLUMN	Response variable column name representing the continuous dependent variable f **DataType**: STRING
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize distribu **DataType**: STRING
INPUTTABLE	[optional] Source data table or view containing the training dataset for distributed
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure numerica **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed model validation and generalization **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelization for computational optimization during distribu **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing architecture and cl **DataType**: INTEGER
SAVEMODEL	[optional] Enable model serialization for deployment and distributed ensemble int **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Unique identifier for the persisted distributed XGBoost regression mod **DataType**: STRING
DATACLEANING	[optional] Activate automated data quality preprocessing pipeline for robust distri **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling distributed data **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during distribu **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization and standardization for improved distribu **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for optimal distributed X **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization range opt **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range opti **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval and co **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed ens **DataType**: INTEGER, **Default**: 64000000

20.7.4 Output

Below is the expected output for the dask XgbReg Regression function.

Rmse	Root Mean Square Error quantifying distributed XGBoost model prediction accuracy and ensemble **DataType**: DOUBLE
R2	Coefficient of determination measuring explained variance and distributed gradient boosting model **DataType**: DOUBLE
Mse	Mean Squared Error representing average squared prediction errors across distributed validation sa **DataType**: DOUBLE
Mae	Mean Absolute Error indicating average magnitude of distributed ensemble prediction deviations fr **DataType**: DOUBLE
Accuracy	Overall predictive performance metric representing distributed XGBoost model effectiveness on va **DataType**: DOUBLE

20.7.5 Examples

Below are the some of the examples for running Xgb Regression SQL function.

```
select * from table(Unifyml.Dask.XgbReg(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelXGB")))

SQL Results:
+------------------------------+-----------------------+
|        CoefficientName       |         Value         |
+------------------------------+-----------------------+
| Mean Squared Error(MSE)      | 0.0024154589371980675 |
| Mean Absolute Error(MAE)     | 0.0024154589371980675 |
| Root Mean Squared Error(RMSE) | 0.04914731871829904   |
| Regression Score(R2)         | 0.9902560723027678    |
| Accuracy                     | 99.9975845410628      |
| OutputModelName              | modelXGB              |
+------------------------------+-----------------------+
```

20.8 XgbRegPredict

20.8.1 Introduction

After training a Dask XGBRegressor model, the distributed gradient boosting ensemble can be deployed for scalable inference to generate continuous target value predictions across large datasets using the learned ensemble of decision trees. This prediction function leverages Dask's parallel computing capabilities to provide efficient and scalable ensemble inference for big data applications while maintaining the superior predictive accuracy and computational consistency of XGBoost across distributed partitions.

For further references please see XGBRegressorPredict.

20.8.2 Syntax

Below is the syntax for the dask XgbRegPredict function.

```
SELECT * FROM table(Unifyml.Dask.XgbRegPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

20.8.3 Input Arguments

Below are the input arguments for the dask XgbRegPredict function.

INPUTMODELNAME	Unique identifier of the pre-trained distributed XGBoost regression model fo **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing the feature data for distributed gra
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTDATATYPES	[optional] Explicit data type specifications for feature columns to ensure num **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data quality preprocessing pipeline for inferenc **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality enhancement strategies (removerow, removeduplicates, removeoutliers, fill) for handling anomalous **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data replacement during dis **DataType**: ARRAY
DATASCALING	[optional] Enable feature normalization consistent with distributed training p **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm selection (MinMaxScaler, StandardScaler, Normalizer, Binarizer) matching distributed **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary parameter for MinMaxScaler normalization rang **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary parameter for MinMaxScaler normalization range **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed data retrieval ar **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage for downstream analy **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured ensemble prediction re **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format for external analysis and d **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and distributed e **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed **DataType**: INTEGER, **Default**: 64000000

20.8.4 Output

Below is the expected output for the dask XgbRegPredict function.

INPUTCOLUMNNAMES	Feature vector column names utilized during distributed XGBoost model training
PREDICTCOLUMN	Continuous regression predictions generated through distributed ensemble aggrega

20.8.5 Examples

Below are the some of the examples for running XgbRegPredict SQL function.

```
select * from table(Unifyml.Dask.XgbRegPredict(
INPUTTABLE => ("cacheid2"),
INPUTMODELNAME => ("modelXGB")))

SQL Results:
+----------+----------+----------+----------+---------------+
| variance | skewness | curtosis | entropy  | class_Predict |
+----------+----------+----------+----------+---------------+
| 0.769004 | 0.820084 | 0.159006 | 0.679130 | 1.000000      |
| 0.835659 | 0.799148 | 0.181374 | 0.566678 | 1.000000      |
| 0.786629 | 0.345498 | 0.462516 | 0.740440 | 1.000000      |
| 0.757105 | 0.856051 | 0.081780 | 0.330464 | 1.000000      |
| 0.531578 | 0.269221 | 0.632350 | 0.619109 | 1.000000      |
+----------+----------+----------+----------+---------------+
```

IX TextProcessing Algorithms

21. Unifyml

This section provides details about the supported Unifyml Text algorithms.

21.1 AudioConversion

21.1.1 Introduction

The AudioConversion function implements advanced speech-to-text conversion algorithms within the Unifyml framework, providing automated transcription capabilities for audio data processing. This function leverages state-of-the-art automatic speech recognition (ASR) technology to convert audio files in WAV format into accurate textual representations, enabling seamless integration of voice data into text-based analytics workflows and natural language processing pipelines.

21.1.2 Syntax

Below is the syntax for the Unifyml AudioWavToText function.

```
SELECT * FROM table(Unifyml.AudioWavToText(
  [ INPUTAUDIOCOLUMNNAME => ( inputAudiocolumn ) ] [,...]
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ LANGUAGE => (language ) ] [,...]
  [ MINIMUMSILENCELENGTH => ( minsilencelen ) ] [,...]
  [ SILENCETHRESH => ( silencethresh ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.1.3 Input Arguments

Below are the input arguments for the Unifyml AudioWavToText function.

INPUTAUDIOCOLUMNNAME	[optional] Column identifier containing audio file references or binary audio **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing audio data columns for batch tran
PARALLELISM	[optional] Degree of parallelization for computational optimization during c **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed audio processing archite **DataType**: INTEGER
INPUTDIRECTORY	[optional] File system directory path containing audio files for batch transcr **DataType**: STRING
LANGUAGE	[optional] Target language specification for speech recognition model select **DataType**: STRING
MINIMUMSILENCELENGTH	[optional] Minimum duration threshold for silence detection in audio segme **DataType**: INTEGER
SILENCETHRESH	[optional] Amplitude threshold parameter for silence detection algorithm ca **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized audio data retrieval and co **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale audio da **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist transcription results to database storage for downstream te **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured transcription results a **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export transcription results to CSV format for external text analy **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and transcriptio **DataType**: STRING

21.1.4 Output

Below is the expected output for the Unifyml AudioWavToText function.

21.1.5 Examples

Below are the some of the examples for running AudioWavToText SQL function.

```
select * from table(Unifyml.AudioWavToText(
INPUTTABLE => ("audios"),
INPUTAUDIOCOLUMNNAME => ("audio"),
PARTITIONCOLUMN => ("id")))

SQL Results:

+-------+-------------------------------+
| index |             text              |
+-------+-------------------------------+
| 0     | Testing done by stress test.  |
+-------+-------------------------------+
```

21.2 ImageConversion

21.2.1 Introduction

The ImageConversion function implements advanced optical character recognition (OCR) and computer vision algorithms within the Unifyml framework, providing automated text extraction capabilities from image data. This function leverages state-of-the-art image processing and machine learning techniques to convert visual text content within images into machine-readable textual representations, enabling seamless integration of visual data into text-based analytics workflows and document processing pipelines.

21.2.2 Syntax

Below is the syntax for the Unifyml ImageToText function.

```
SELECT * FROM table(Unifyml.ImageToText(
  [ INPUTIMAGECOLUMNNAME => ( inputImagecolumn ) ] [,...]
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.2.3 Input Arguments

Below are the input arguments for the Unifyml ImageToText function.

INPUTIMAGECOLUMNNAME	[optional] Column identifier containing image file references or binary imag **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing image data columns for batch tex
PARALLELISM	[optional] Degree of parallelization for computational optimization during **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed image processing archit **DataType**: INTEGER
INPUTDIRECTORY	[optional] File system directory path containing image files for batch OCR **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized image data retrieval and c **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale image da **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist text extraction results to database storage for downstream **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured text extraction results **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export text extraction results to CSV format for external docume **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and text extrac **DataType**: STRING

21.2.4 Output

Below is the expected output for the Unifyml ImageToText function.

21.2.5 Examples

Below are the some of the examples for running ImageToText SQL function.

```
select * from table(Unifyml.FileIterator(
INPUTDIRECTORY => ("images"),
OUTPUTCACHEID => ("FileIteratorcache")))

SQL Results:

+-----------------+------------+
| InputFolderName | DataCached |
+-----------------+------------+
| images          | true       |
+-----------------+------------+

select * from table(Unifyml.ImageToText(
INPUTCACHEID => ("FileIteratorcache"),
INPUTIMAGECOLUMNNAME => ("FileName"),
PARTITIONCOLUMN => ("index"),
OUTPUTCACHEID => ("ImageToTextcachecsv")))

SQL Results:

+-------------------+------------+------------------+
|   InputCacheId    | DataCached | InputColumnNames |
+-------------------+------------+------------------+
| FileIteratorcache | true       | [FileName]       |
+-------------------+------------+------------------+
```

21.3 FileIterator

21.3.1 Introduction

The FileIterator function implements efficient file system traversal and data cataloging capabilities within the Unifyml framework, providing automated file discovery and metadata extraction for large-scale data processing workflows. This utility function systematically scans specified directories to identify and catalog files, converting distributed file collections into structured, queryable datasets stored in optimized Parquet format for subsequent processing operations and analytics pipelines.

21.3.2 Syntax

Below is the syntax for the Unifyml FileIterator function.

```
SELECT * FROM table(Unifyml.FileIterator(
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ OUTPUTCACHEID => (cacheId ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.3.3 Input Arguments

Below are the input arguments for the Unifyml FileIterator function.

INPUTDIRECTORY	[optional] File system directory path for systematic file discovery and metadata extracti **DataType**: STRING
OUTPUTCACHEID	[optional] Cache identifier for storing structured file catalog in optimized Parquet forma **DataType**: STRING
PARALLELISM	[optional] Degree of parallelization for computational optimization during distributed fi **DataType**: INTEGER, **Default**: 1
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale file system optimiza **DataType**: INTEGER, **Default**: 64000000

21.3.4 Output

Below is the expected output for the Unifyml FileIterator function.

21.3.5 Examples

Below are the some of the examples for running FileIterator SQL function.

```
select * from table(Unifyml.FileIterator(
INPUTDIRECTORY => ("images"),
OUTPUTCACHEID => ("FileIteratorcache")))

SQL Results:

+-----------------+------------+
| InputFolderName | DataCached |
+-----------------+------------+
| images          | true       |
+-----------------+------------+
```

21.4 SentimentAnalysis

21.4.1 Introduction

The SentimentAnalysis function implements advanced natural language processing algorithms within the Unifyml framework, providing automated sentiment classification and emotional tone detection for textual data. This function leverages state-of-the-art machine learning models and lexicon-based approaches to analyze text content and extract sentiment polarities, emotional intensities, and subjective opinions, enabling comprehensive text analytics for social media monitoring, customer feedback analysis, and opinion mining applications.

21.4.2 Syntax

Below is the syntax for the Unifyml SentimentAnalysis function.

```
SELECT * FROM table(Unifyml.SentimentAnalysis(
  INPUTTEXTCOLUMNNAME => ( inputTextcolumn )
  [ INPUTDOCCOLUMNNAME => ( inputDocColumnName ) ] [,...]
  [ INPUTDOCID => (inputDocId ) ] [,...]
  [ INPUTGROUPBYNAME => ( inputGroupByName ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.4.3 Input Arguments

Below are the input arguments for the Unifyml SentimentAnalysis function.

INPUTTEXTCOLUMNNAME	Column identifier containing textual content for sentiment analysis and emot **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing text data columns for batch sentim
PARALLELISM	[optional] Degree of parallelization for computational optimization during di **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed text processing architectu **DataType**: INTEGER
INPUTDOCCOLUMNNAME	[optional] Column identifier for document or record identification during sen **DataType**: STRING
INPUTDOCID	[optional] Specific document identifier for targeted sentiment analysis and fo **DataType**: INTEGER
INPUTGROUPBYNAME	[optional] Column identifier for grouping sentiment analysis results by categ **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized text data retrieval and comp **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale text datas **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist sentiment analysis results to database storage for downstrea **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured sentiment analysis resu **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export sentiment analysis results to CSV format for external text a **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and sentiment a **DataType**: STRING

21.4.4 Output

Below is the expected output for the Unifyml SentimentAnalysis function.

21.4.5 Examples

Below are the some of the examples for running Sentimentanalysis SQL function.

```
select * from table(Unifyml.SentimentAnalysis(
INPUTTABLE => ("medicaldrama"),
INPUTTEXTCOLUMNNAME => ("compliant"),
INPUTDOCCOLUMNNAME => ("id"),
INPUTDOCID => 10))

SQL Results:

+----+------------------+-----+-------+-------+----------+-----------+
| id |     compliant    | neg |  neu  |  pos  | compound | sentiment |
+----+------------------+-----+-------+-------+----------+-----------+
| 10 |  Has a good hook. | 0.0 | 0.408 | 0.592 | 0.4404   | positive  |
+----+------------------+-----+-------+-------+----------+-----------+
```

21.5 TextTranslator

21.5.1 Introduction

The TextTranslator function implements advanced machine translation capabilities within the Unifyml framework, providing automated language translation services for multilingual text processing and cross-linguistic content analysis. This function leverages state-of-the-art neural machine translation models and natural language processing techniques to convert textual content between different languages while preserving semantic meaning and contextual nuances, enabling global content localization and multilingual data analytics workflows.

21.5.2 Syntax

Below is the syntax for the Unifyml TextTranslator function.

```
SELECT * FROM table(Unifyml.TextTranslator(
  [ INPUTTEXTCOLUMNNAME => ( inputTextcolumn ) ] [,...]
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ INPUTTEXT => (inputText ) ] [,...]
  [ SOURCELANG => ( sourcelang ) ] [,...]
  [ DESTLANG => ( destlang ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.5.3 Input Arguments

Below are the input arguments for the Unifyml TextTranslator function.

INPUTTEXTCOLUMNNAME	[optional] Column identifier containing source text content for machine trans **DataType**: STRING
INPUTTABLE	[optional] Source table or view containing text data columns for batch transla
PARALLELISM	[optional] Degree of parallelization for computational optimization during di **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed translation processing arc **DataType**: INTEGER
INPUTDIRECTORY	[optional] File system directory path containing text files for batch translatio **DataType**: STRING
INPUTTEXT	[optional] Direct text input for immediate translation processing and languag **DataType**: STRING
SOURCELANG	[optional] Source language specification for translation model selection and l **DataType**: STRING
DESTLANG	[optional] Target language specification for translation output generation and **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized text data retrieval and comp **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale multilingu **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist translation results to database storage for downstream mult **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured translation results and **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export translation results to CSV format for external multilingual **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and translation r **DataType**: STRING

21.5.4 Output

Below is the expected output for the Unifyml TextTranslator function.

21.5.5 Examples

Below are the some of the examples for running TextTranslator SQL function.

```
select * from table(Unifyml.TextTranslator(
INPUTDIRECTORY => ("texts"),
INPUTTEXTCOLUMNNAME => ("FileName"),
PARTITIONCOLUMN => ("index"),
DESTLANG => ("te"),
OUTPUTCACHEID => ("TextTranslatecacheCsv")))

SQL Results:

+-----------------------+------------+------------------+
|     OutputCacheid     | DataCached | InputColumnNames |
+-----------------------+------------+------------------+
| TextTranslatecacheCsv | true       | [FileName]       |
+-----------------------+------------+------------------+
```

21.6 TweetData

21.6.1 Introduction

The TweetData function implements advanced social media data extraction and analysis capabilities within the Unifyml framework, providing automated Twitter data collection and preprocessing for social media analytics and sentiment monitoring applications. This function leverages Twitter API integration and social media mining techniques to gather real-time and historical tweet data based on specified search criteria, hashtags, and temporal parameters, enabling comprehensive social media research and opinion analysis workflows.

21.6.2 Syntax

Below is the syntax for the unifyml TweetData function.

```
SELECT * FROM table(Unifyml.TweetData(
  INPUTHASHTAG => ( inputSearchQuery ) [,...]
  [ INPUTFILTER => ( inputFilter ) ] [,...]
  [ METHOD => ( method ) ] [,...]
  [ STARTTIME => (startTime ) ] [,...]
  [ ENDTIME => ( endTime ) ] [,...]
  [ MAXTWEETSPERREQUEST => ( maxTweetsPerRequest ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.6.3 Input Arguments

Below are the input arguments for the unifyml TweetData function.

INPUTHASHTAG	Primary search query or hashtag for targeted tweet collection and social med **DataType**: STRING
INPUTFILTER	[optional] Advanced filtering criteria for refined tweet selection based on lan **DataType**: STRING
PARALLELISM	[optional] Degree of parallelization for computational optimization during di **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed social media processing a **DataType**: INTEGER
METHOD	[optional] Data collection methodology specification for different Twitter AP **DataType**: STRING
STARTTIME	[optional] Temporal boundary for historical tweet collection defining the beg **DataType**: INTEGER
ENDTIME	[optional] Temporal boundary for historical tweet collection defining the con **DataType**: INTEGER
MAXTWEETSPERREQUEST	[optional] Rate limiting parameter controlling the volume of tweets retrieved **DataType**: INTEGER,**Default**: 100
INPUTCACHEID	[optional] Memory cache identifier for optimized social media data retrieval **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale social me **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist tweet data to database storage for downstream social media **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured tweet data and social m **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export tweet data to CSV format for external social media analysi **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and social medi **DataType**: STRING

21.6.4 Output

Below is the expected output for the unifyml TweetData function.

21.6.5 Examples

Below are the some of the examples for running Tweetdata SQL function.

```
select * from table(Unifyml.TweetData(
INPUTHASHTAG => ("COVID"),
INPUTFILTER => ("hoax -is:retweet lang:en"),
OUTPUTCACHEID => ("tweetDataCache")))

SQL Results:

+------------------+------------+
| InputSearchQuery | DataCached |
+------------------+------------+
| COVID            | true       |
+------------------+------------+
```

21.7 AlsRecommendForAllItems

21.7.1 Introduction

The AlsRecommendForAllItems function implements advanced collaborative filtering recommendation algorithms using Alternating Least Squares (ALS) matrix factorization within the Unifyml framework. This function generates comprehensive item-to-user recommendations by training an ALS model on user-item interaction data and producing ranked lists of users most likely to engage with each item in the dataset. The algorithm is particularly valuable for reverse recommendation scenarios such as targeted marketing, user acquisition strategies, and audience identification for specific products, movies, or content items.

21.7.2 Syntax

Below is the syntax for the Unifyml AlsRecommendForAllItems function.

```
SELECT * FROM table(Unifyml.AlsRecommendForAllItems(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ RANK => ( rank ) ] [,...]
  [ NUMUSERS => ( numUsers ) ] [,...]
  [ MAXITER => ( maxItrnum ) ] [,...]
  [ REGPARAM => ( regParam ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.7.3 Input Arguments

Below are the input arguments for the Unifyml AlsRecommendForAllItems function.

USERCOLUMN	Column identifier containing user identifiers for collaborative filtering and re **DataType**: INTEGER
ITEMCOLUMN	Column identifier containing item identifiers for recommendation target spec **DataType**: INTEGER
RATING	Column identifier containing user preference ratings or interaction scores for **DataType**: DOUBLE
RANK	[optional] Latent factor dimensionality parameter controlling model complex **DataType**: INTEGER, **Default**: 10
NUMUSERS	[optional] Number of top users to recommend for each item in reverse recom **DataType**: INTEGER, **Default**: 5
MAXITER	[optional] Maximum iteration limit for ALS algorithm convergence control a **DataType**: INTEGER, **Default**: 10
REGPARAM	[optional] Regularization parameter for overfitting prevention and model gen **DataType**: Double, **Default**: 0.1
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source table identifier containing user-item interaction data for co **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized recommendation data retrie **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed recommendation process **DataType**: INTEGER
CACHETYPE	[optional] Cache storage format specification for optimized recommendation **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale recomme **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist recommendation results to database storage for downstrean **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured recommendation resul **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export recommendation results to CSV format for external recom **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and recommend **DataType**: STRING
OUTPUTCACHEID	[optional] Cache identifier for storing processed recommendation results in c **DataType**: STRING

21.7.4 Output

Below is the expected output for the Unifyml AlsRecommendForAllItems function.

InputTableName	Source table identifier used for collaborative filtering model training and recommendation g **DataType**: STRING
DataCached	Cache operation status indicating successful recommendation result storage and retrieval op **DataType**: STRING
OutputCacheId	Generated cache identifier for accessing processed recommendation results and collaborative **DataType**: STRING
ItemColumn	Item column identifier used for recommendation target specification and matrix factorization **DataType**: STRING

21.7.5 Examples

Below are the some of the examples for running AlsRecommendForAllItems SQL function.

```
select * from table(Unifyml.AlsRecommendForAllItems(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("item_recommend_cache")))

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | ItemColumn    |
+-------------------+------------+----------------------+-----------------+
| user_item_ratings | true       | item_recommend_cache | movie_id    |
+-------------------+------------+----------------------+-----------------+
```

21.8 AlsRecommendForAllUsers

21.8.1 Introduction

The AlsRecommendForAllUsers function implements comprehensive collaborative filtering recommendation algorithms using Alternating Least Squares (ALS) matrix factorization within the Unifyml framework. This function generates personalized item recommendations for all users in the dataset by training an ALS model on user-item interaction patterns and producing ranked lists of items most likely to appeal to each individual user. The algorithm excels in traditional recommendation scenarios such as personalized content delivery, product suggestions, and user engagement optimization across e-commerce, entertainment, and content platforms.

21.8.2 Syntax

Below is the syntax for the Unifyml AlsRecommendForAllUsers function.

```
SELECT * FROM table(Unifyml.AlsRecommendForAllUsers(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ RANK => ( rank ) ] [,...]
  [ NUMITEMS => ( numItems ) ] [,...]
  [ MAXITER => ( maxItrnum ) ] [,...]
  [ REGPARAM => ( regParam ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.8.3 Input Arguments

Below are the input arguments for the Unifyml AlsRecommendForAllUsers function.

USERCOLUMN	Column identifier containing user identifiers for personalized recommendatio **DataType**: INTEGER
ITEMCOLUMN	Column identifier containing item identifiers for recommendation candidate s **DataType**: INTEGER
RATING	Column identifier containing user preference ratings or interaction scores for **DataType**: DOUBLE
RANK	[optional] Latent factor dimensionality parameter controlling model complex **DataType**: INTEGER, **Default**: 10
NUMITEMS	[optional] Number of top items to recommend for each user in personalized r **DataType**: INTEGER, **Default**: 5
MAXITER	[optional] Maximum iteration limit for ALS algorithm convergence control a **DataType**: INTEGER, **Default**: 10
REGPARAM	[optional] Regularization parameter for overfitting prevention and model gen **DataType**: Double, **Default**: 0.1
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source table identifier containing user-item interaction data for per **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized recommendation data retrie **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed recommendation process **DataType**: INTEGER
CACHETYPE	[optional] Cache storage format specification for optimized personalized reco **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale personaliz **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist personalized recommendation results to database storage fo **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured personalized recomme **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export personalized recommendation results to CSV format for ex **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and personalized **DataType**: STRING
OUTPUTCACHEID	[optional] Cache identifier for storing processed personalized recommendatio **DataType**: STRING

21.8.4 Output

Below is the expected output for the Unifyml AlsRecommendForAllUsers function.

InputTableName	Source table identifier used for personalized collaborative filtering model training and recom **DataType**: STRING
DataCached	Cache operation status indicating successful personalized recommendation result storage and **DataType**: STRING
OutputCacheId	Generated cache identifier for accessing processed personalized recommendation results and **DataType**: STRING
UserColumn	User column identifier used for personalized recommendation target specification and matrix **DataType**: STRING

21.8.5 Examples

Below are the some of the examples for running AlsRecommendForAllUsers SQL function.

```
select * from table(Unifyml.AlsRecommendForAllUsers(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("user_recommend_cache")))

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | UserColumn    |
+-------------------+------------+----------------------+-----------------+
| user_item_ratings | true       | user_recommend_cache | user_id     |
+-------------------+------------+----------------------+-----------------+
```

21.9 Tf_Idf

21.9.1 Introduction

Term Frequency - Inverse Document Frequency (TF-IDF) represents a sophisticated statistical measure for evaluating the relative importance and semantic significance of terms within individual documents relative to an entire document corpus. This algorithm computes weighted term importance scores by combining local term frequency statistics with global document frequency information, effectively identifying discriminative terms that characterize document content while diminishing the influence of commonly occurring words across the collection. TF-IDF serves as a fundamental technique in information retrieval, text mining, and natural language processing applications, providing essential feature extraction capabilities for document classification, search relevance ranking, and content analysis workflows.

21.9.2 Syntax

Below is the syntax for the Scikitlearn TfIdf function.

```
SELECT * FROM table(Unifyml.Tf\_Idf(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  INPUTEXTCOLUMNNAME => ( inputTextcolumn )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

21.9.3 Input Arguments

Below are the input arguments for the Unifyml TfIdf function.

INPUTTABLE	Source table identifier containing textual document data for TF-IDF feature e **DataType**: STRING
INPUTTEXTCOLUMNNAME	Column identifier containing document text content for term frequency and i **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed TF-IDF processing archi **DataType**: INTEGER
INPUTCACHEID	[optional] Memory cache identifier for optimized document data retrieval an **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale document **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist TF-IDF feature vectors to database storage for downstream **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured TF-IDF feature matric **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export TF-IDF results to CSV format for external text mining and **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and TF-IDF fea **DataType**: STRING
CACHETYPE	[optional] Cache storage format specification for optimized TF-IDF result pe **DataType**: STRING, **Default**: Cachetype
OUTPUTCACHEID	[optional] Cache identifier for storing processed TF-IDF feature vectors in op **DataType**: STRING

21.9.4 Output

Below is the expected output for the Unifyml TfIdf function.

InputTableName	Source table identifier used for TF-IDF feature extraction and document corpus analysis. **DataType**: STRING
DataCached	Cache operation status indicating successful TF-IDF feature vector storage and retrieval opti **DataType**: STRING
OutputCacheId	Generated cache identifier for accessing processed TF-IDF feature matrices and term import **DataType**: STRING
InputColumns	Text column identifier used for TF-IDF computation and document feature extraction proces **DataType**: STRING

21.9.5 Examples

Below are the some of the examples for running unifyml TfIdf SQL function.

```
select * from table(Unifyml.Tf\_Idf(
INPUTTABLE => ("name_entity_data"),
INPUTTEXTCOLUMNNAME => ("text"),
OUTPUTCACHEID => ("tf_idf_cache")));

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | InputColumns    |
+-------------------+------------+----------------------+-----------------+
| name_entity_data  | true       | tf_idf_cache         | text            |
+-------------------+------------+----------------------+-----------------+
```

22. Spark

22.1 AlsRecommendForAllItems

22.1.1 Introduction

The Spark AlsRecommendForAllItems function implements distributed collaborative filtering recommendation algorithms using Apache Spark's Alternating Least Squares (ALS) matrix factorization capabilities. This function generates comprehensive item-to-user recommendations by training an ALS model on large-scale user-item interaction data and producing ranked lists of users most likely to engage with each item in the dataset. The Spark implementation leverages distributed computing to handle massive recommendation datasets while providing superior scalability and performance for enterprise-level recommendation systems, making it particularly valuable for reverse recommendation scenarios in digital marketing, audience targeting, and customer acquisition strategies.

22.1.2 Syntax

Below is the syntax for the Spark AlsRecommendForAllItems function.

```
SELECT * FROM table(Unifyml.Spark.AlsRecommendForAllItems(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ RANK => ( rank ) ] [,...]
  [ NUMUSERS => ( numUsers ) ] [,...]
  [ MAXITER => ( maxItrnum ) ] [,...]
  [ REGPARAM => ( regParam ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

22.1.3 Input Arguments

Below are the input arguments for the Spark AlsRecommendForAllItems function.

USERCOLUMN	Column identifier containing user identifiers for distributed collaborative filte **DataType**: INTEGER
ITEMCOLUMN	Column identifier containing item identifiers for recommendation target spec **DataType**: INTEGER
RATING	Column identifier containing user preference ratings or interaction scores for **DataType**: DOUBLE
RANK	[optional] Latent factor dimensionality parameter controlling model complex **DataType**: INTEGER, **Default**: 10
NUMUSERS	[optional] Number of top users to recommend for each item in distributed rev **DataType**: INTEGER, **Default**: 5
MAXITER	[optional] Maximum iteration limit for distributed ALS algorithm convergen **DataType**: INTEGER, **Default**: 10
REGPARAM	[optional] Regularization parameter for overfitting prevention and model gen **DataType**: Double, **Default**: 0.1
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source table identifier containing user-item interaction data for dis **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed recommendatio **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed recommendation process **DataType**: INTEGER
CACHETYPE	[optional] Cache storage format specification for optimized distributed recon **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distributed **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist distributed recommendation results to database storage for **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured distributed recommend **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed recommendation results to CSV format for exte **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and distributed r **DataType**: STRING
OUTPUTCACHEID	[optional] Cache identifier for storing processed distributed recommendation **DataType**: STRING

22.1.4 Output

Below is the expected output for the Spark AlsRecommendForAllItems function.

InputTableName	Source table identifier used for distributed collaborative filtering model training and recomm **DataType**: STRING
DataCached	Cache operation status indicating successful distributed recommendation result storage and r **DataType**: STRING
OutputCacheId	Generated cache identifier for accessing processed distributed recommendation results and c **DataType**: STRING
ItemColumn	Item column identifier used for recommendation target specification and distributed matrix f **DataType**: STRING

22.1.5 Examples

Below are the some of the examples for running AlsRecommendForAllItems SQL function.

```
select * from table(Unifyml.Spark.AlsRecommendForAllItems(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("item_recommend_cache")))

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | ItemColumn    |
+-------------------+------------+----------------------+-----------------+
| user_item_ratings | true       | item_recommend_cache | movie_id    |
+-------------------+------------+----------------------+-----------------+
```

22.2 AlsRecommendForAllUsers

22.2.1 Introduction

The Spark AlsRecommendForAllUsers function implements comprehensive distributed collaborative filtering recommendation algorithms using Apache Spark's Alternating Least Squares (ALS) matrix factorization capabilities. This function generates personalized item recommendations for all users in large-scale datasets by training an ALS model on distributed user-item interaction patterns and producing ranked lists of items most likely to appeal to each individual user. The Spark implementation provides exceptional scalability and performance for enterprise-level personalized recommendation systems, making it ideal for traditional recommendation scenarios such as e-commerce product suggestions, content platform recommendations, and user engagement optimization across massive user bases.

22.2.2 Syntax

Below is the syntax for the Spark AlsRecommendForAllUsers function.

```
SELECT * FROM table(Unifyml.Spark.AlsRecommendForAllUsers(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ RANK => ( rank ) ] [,...]
  [ NUMITEMS => ( numItems ) ] [,...]
  [ MAXITER => ( maxItrnum ) ] [,...]
  [ REGPARAM => ( regParam ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

22.2.3 Input Arguments

Below are the input arguments for the Spark AlsRecommendForAllUsers function.

USERCOLUMN	Column identifier containing user identifiers for distributed personalized reco **DataType**: INTEGER
ITEMCOLUMN	Column identifier containing item identifiers for recommendation candidate **DataType**: INTEGER
RATING	Column identifier containing user preference ratings or interaction scores for **DataType**: DOUBLE
RANK	[optional] Latent factor dimensionality parameter controlling model complex **DataType**: INTEGER, **Default**: 10
NUMITEMS	[optional] Number of top items to recommend for each user in distributed pe **DataType**: INTEGER, **Default**: 5
MAXITER	[optional] Maximum iteration limit for distributed ALS algorithm convergen **DataType**: INTEGER, **Default**: 10
REGPARAM	[optional] Regularization parameter for overfitting prevention and model gen **DataType**: Double, **Default**: 0.1
PARTITIONCOLUMN	[optional] Column designated for horizontal data partitioning to optimize dis **DataType**: STRING
INPUTTABLE	[optional] Source table identifier containing user-item interaction data for dis **DataType**: STRING
INPUTCACHEID	[optional] Memory cache identifier for optimized distributed recommendatio **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed recommendation process **DataType**: INTEGER
CACHETYPE	[optional] Cache storage format specification for optimized distributed perso **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for large-scale distribute **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist distributed personalized recommendation results to databas **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table identifier for storing structured distributed personalize **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table to prevent data versioning conflicts d **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed personalized recommendation results to CSV fo **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] File path specification for CSV export destination and distributed **DataType**: STRING
OUTPUTCACHEID	[optional] Cache identifier for storing processed distributed personalized reco **DataType**: STRING

22.2.4 Output

Below is the expected output for the Spark AlsRecommendForAllUsers function.

InputTableName	Source table identifier used for distributed personalized collaborative filtering model training **DataType**: STRING
DataCached	Cache operation status indicating successful distributed personalized recommendation result **DataType**: STRING
OutputCacheId	Generated cache identifier for accessing processed distributed personalized recommendation **DataType**: STRING
UserColumn	User column identifier used for personalized recommendation target specification and distrib **DataType**: STRING

22.2.5 Examples

Below are the some of the examples for running AlsRecommendForAllUsers SQL function.

```
select * from table(Unifyml.Spark.AlsRecommendForAllUsers(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("user_recommend_cache")))

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | UserColumn   |
+-------------------+------------+----------------------+-----------------+
| user_item_ratings | true       | user_recommend_cache | user_id    |
+-------------------+------------+----------------------+-----------------+
```

22.3 Tf_Idf

22.3.1 Introduction

Term Frequency - Inverse Document Frequency (TF-IDF) is a fundamental text mining and feature engineering technique used to quantify the importance of terms within documents relative to a corpus. This statistical measure transforms textual data into numerical features, enabling downstream machine learning algorithms to process and analyze text-based datasets effectively.

22.3.2 Syntax

Below is the syntax for the Spark TF-IDF vectorization function.

```
SELECT * FROM table(Unifyml.Tf\_Idf(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  INPUTTEXTCOLUMNNAME => ( inputTextcolumn )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

22.3.3 Input Arguments

Below are the input parameters for the Spark TF-IDF vectorization function.

INPUTTABLE	Specifies the input dataset containing text features for vectorization. **DataType**: STRING
INPUTTEXTCOLUMNNAME	Name of the text feature column for TF-IDF transformation. **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for data processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist vectorized features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output feature table. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export vectorized features to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
CACHETYPE	[optional] Caching strategy for feature vectors. **DataType**: STRING, **Default**: Cachetype
OUTPUTCACHEID	[optional] Identifier for cached feature vectors. **DataType**: STRING

22.3.4 Output

Below is the expected output schema for the Spark TF-IDF vectorization function.

InputTableName	Source dataset name for feature extraction. **DataType**: STRING
DataCached	Feature vector caching status. **DataType**: STRING
OutputCacheId	Identifier of cached TF-IDF features. **DataType**: STRING
InputColumns	Name of the processed text feature column. **DataType**: STRING

22.3.5 Examples

Below are examples demonstrating the Spark TF-IDF feature extraction SQL function.

```
select * from table(Unifyml.Tf\_Idf(
INPUTTABLE => ("name_entity_data"),
INPUTTEXTCOLUMNNAME => ("text"),
OUTPUTCACHEID => ("tf_idf_cache")));

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | InputColumns    |
+-------------------+------------+----------------------+-----------------+
| name_entity_data  | true       | tf_idf_cache         | text            |
+-------------------+------------+----------------------+-----------------+
```

23. Nltk

This section provides details about the supported Natural Language Processing (NLP) algorithms and text analytics capabilities.

23.1 AudioConversion

23.1.1 Introduction

The function executes NLTK-based audio-to-text transcription algorithms for speech recognition and audio data preprocessing. This function converts audio files (.wav format) into structured text data for downstream NLP analysis and feature extraction.

23.1.2 Syntax

Below is the syntax for the NLTK AudioWavToText speech recognition function.

```
SELECT * FROM table(Unifyml.Nltk.AudioWavToText(
  [ INPUTAUDIOCOLUMNNAME => ( inputAudiocolumn ) ] [,...]
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ LANGUAGE => (language ) ] [,...]
  [ MINIMUMSILENCELENGTH => ( minsilencelen ) ] [,...]
  [ SILENCETHRESH => ( silencethresh ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.1.3 Input Arguments

Below are the input parameters for the NLTK AudioWavToText speech recognition function.

INPUTAUDIOCOLUMNNAME	[optional] Name of the audio feature column for transcription. **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing audio features.
PARALLELISM	[optional] Parallel processing level for model inference. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
INPUTDIRECTORY	[optional] Directory path containing audio data files. **DataType**: STRING
LANGUAGE	[optional] Target language for speech recognition model. **DataType**: STRING
MINIMUMSILENCELENGTH	[optional] Minimum silence duration threshold for audio segmentation. **DataType**: INTEGER
SILENCETHRESH	[optional] Silence detection threshold for audio preprocessing. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached audio dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for audio data processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist transcribed text features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output text dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export transcribed text to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING

23.1.4 Output

Below is the expected output schema for the NLTK AudioWavToText speech recognition function.

23.1.5 Examples

Below are examples demonstrating the AudioWavToText speech recognition SQL function.

```
select * from table(Unifyml.Nltk.AudioWavToText(
INPUTTABLE => ("audios"),
INPUTAUDIOCOLUMNNAME => ("audio"),
PARTITIONCOLUMN => ("id")))

SQL Results:

+-------+-------------------------------+
| index |             text              |
+-------+-------------------------------+
| 0     | Testing done by stress test.  |
+-------+-------------------------------+
```

23.2 ImageConversion

23.2.1 Introduction

The function executes NLTK-based optical character recognition (OCR) algorithms for computer vision and text extraction tasks. This function converts image data into structured text features for multimodal machine learning pipelines and document analysis workflows.

23.2.2 Syntax

Below is the syntax for the NLTK ImageToText OCR function.

```
SELECT * FROM table(Unifyml.Nltk.ImageToText(
  [ INPUTIMAGECOLUMNNAME => ( inputImagecolumn ) ] [,...]
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.2.3 Input Arguments

Below are the input parameters for the NLTK ImageToText OCR function.

INPUTIMAGECOLUMNNAME	[optional] Name of the image feature column for text extraction. **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing image features.
PARALLELISM	[optional] Parallel processing level for OCR model inference. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
INPUTDIRECTORY	[optional] Directory path containing image data files. **DataType**: STRING
INPUTCACHEID	[optional] Identifier for cached image dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for image data processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist extracted text features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output text dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export extracted text to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING

23.2.4 Output

Below is the expected output schema for the NLTK ImageToText OCR function.

23.2.5 Examples

Below are examples demonstrating the ImageToText OCR SQL function.

```
select * from table(Unifyml.Nltk.FileIterator(
INPUTDIRECTORY => ("images"),
OUTPUTCACHEID => ("FileIteratorcache")))

SQL Results:

+-----------------+------------+
| InputFolderName | DataCached |
+-----------------+------------+
| images          | true       |
+-----------------+------------+

select * from table(Unifyml.Nltk.ImageToText(
INPUTCACHEID => ("FileIteratorcache"),
INPUTIMAGECOLUMNNAME => ("FileName"),
PARTITIONCOLUMN => ("index"),
OUTPUTCACHEID => ("ImageToTextcachecsv")))

SQL Results:

+-------------------+------------+------------------+
|   InputCacheId    | DataCached | InputColumnNames |
+-------------------+------------+------------------+
| FileIteratorcache | true       | [FileName]       |
+-------------------+------------+------------------+
```

23.3 FileIterator

23.3.1 Introduction

The function executes NLTK-based file system data ingestion algorithms for batch processing workflows.

File iterator aggregates multiple data files from directories into unified cached datasets (Parquet format) for efficient downstream machine learning operations and feature engineering pipelines.

23.3.2 Syntax

Below is the syntax for the NLTK FileIterator data ingestion function.

```
SELECT * FROM table(Unifyml.Nltk.FileIterator(
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ OUTPUTCACHEID => (cacheId ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.3.3 Input Arguments

Below are the input parameters for the NLTK FileIterator data ingestion function.

INPUTDIRECTORY	[optional] Directory path containing raw data files for ingestion. **DataType**: STRING
OUTPUTCACHEID	[optional] Identifier for the output cached dataset. **DataType**: STRING
PARALLELISM	[optional] Parallel processing level for data ingestion operations. **DataType**: INTEGER, **Default**: 1
BYTESPERCHUNK	[optional] Chunk size for data ingestion optimization. **DataType**: INTEGER, **Default**: 64000000

23.3.4 Output

Below is the expected output schema for the NLTK FileIterator data ingestion function.

23.3.5 Examples

Below are examples demonstrating the FileIterator data ingestion SQL function.

```
select * from table(Unifyml.Nltk.FileIterator(
INPUTDIRECTORY => ("images"),
OUTPUTCACHEID => ("FileIteratorcache")))

SQL Results:

+-----------------+------------+
| InputFolderName | DataCached |
+-----------------+------------+
| images          | true       |
+-----------------+------------+
```

23.4 SentimentAnalysis

23.4.1 Introduction

The function executes NLTK-based sentiment classification algorithms for text analytics and opinion mining tasks. This function performs sentiment scoring and polarity detection on textual data, providing quantitative sentiment metrics for business intelligence and customer analytics applications.

23.4.2 Syntax

Below is the syntax for the NLTK SentimentAnalysis classification function.

```
SELECT * FROM table(Unifyml.Nltk.SentimentAnalysis(
  INPUTTEXTCOLUMNNAME => ( inputTextcolumn )
  [ INPUTDOCCOLUMNNAME => ( inputDocColumnName ) ] [,...]
  [ INPUTDOCID => (inputDocId ) ] [,...]
  [ INPUTGROUPBYNAME => ( inputGroupByName ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.4.3 Input Arguments

Below are the input parameters for the NLTK SentimentAnalysis classification function.

INPUTTEXTCOLUMNNAME	Name of the text feature column for sentiment analysis. **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing text features.
PARALLELISM	[optional] Parallel processing level for sentiment model inference. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
INPUTDOCCOLUMNNAME	[optional] Name of the document identifier column. **DataType**: STRING
INPUTDOCID	[optional] Specific document ID for targeted analysis. **DataType**: INTEGER
INPUTGROUPBYNAME	[optional] Name of the grouping variable for aggregated analysis. **DataType**: STRING
INPUTCACHEID	[optional] Identifier for cached text dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for text processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist sentiment scores to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output sentiment dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export sentiment scores to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING

23.4.4 Output

Below is the expected output schema for the NLTK SentimentAnalysis classification function.

23.4.5 Examples

Below are examples demonstrating the SentimentAnalysis classification SQL function.

```
select * from table(Unifyml.Nltk.SentimentAnalysis(
INPUTTABLE => ("medicaldrama"),
INPUTTEXTCOLUMNNAME => ("compliant"),
INPUTDOCCOLUMNNAME => ("id"),
INPUTDOCID => 10))

SQL Results:

+----+------------------+-----+-------+-------+----------+-----------+
| id |     compliant    | neg |  neu  |  pos  | compound | sentiment |
+----+------------------+-----+-------+-------+----------+-----------+
| 10 |  Has a good hook. | 0.0 | 0.408 | 0.592 | 0.4404   | positive  |
+----+------------------+-----+-------+-------+----------+-----------+
```

23.5 TextTranslator

23.5.1 Introduction

The function executes NLTK-based neural machine translation algorithms for multilingual text processing and cross-language data preprocessing.

This function performs language translation on textual features, enabling multilingual machine learning pipelines and cross-cultural sentiment analysis workflows.

23.5.2 Syntax

Below is the syntax for the NLTK TextTranslator neural translation function.

```
SELECT * FROM table(Unifyml.Nltk.TextTranslator(
  [ INPUTTEXTCOLUMNNAME => ( inputTextcolumn ) ] [,...]
  [ INPUTDIRECTORY => ( inputDirectory ) ] [,...]
  [ INPUTTEXT => (inputText ) ] [,...]
  [ SOURCELANG => ( sourcelang ) ] [,...]
  [ DESTLANG => ( destlang ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.5.3 Input Arguments

Below are the input parameters for the NLTK TextTranslator neural translation function.

INPUTTEXTCOLUMNNAME	[optional] Name of the source text feature column for translation. **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing multilingual text features.
PARALLELISM	[optional] Parallel processing level for translation model inference. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
INPUTDIRECTORY	[optional] Directory path containing multilingual text data files. **DataType**: STRING
INPUTTEXT	[optional] Input text string for direct translation processing. **DataType**: STRING
SOURCELANG	[optional] Source language code for translation model configuration. **DataType**: STRING
DESTLANG	[optional] Target language code for translation output. **DataType**: STRING
INPUTCACHEID	[optional] Identifier for cached multilingual dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for translation processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist translated text features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output translated dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export translated text to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING

23.5.4 Output

Below is the expected output schema for the NLTK TextTranslator neural translation function.

23.5.5 Examples

Below are examples demonstrating the TextTranslator neural translation SQL function.

```
select * from table(Unifyml.Nltk.TextTranslator(
INPUTDIRECTORY => ("texts"),
INPUTTEXTCOLUMNNAME => ("FileName"),
PARTITIONCOLUMN => ("index"),
DESTLANG => ("te"),
OUTPUTCACHEID => ("TextTranslatecacheCsv")))

SQL Results:

+-----------------------+------------+------------------+
|     OutputCacheid     | DataCached | InputColumnNames |
+-----------------------+------------+------------------+
| TextTranslatecacheCsv | true       | [FileName]       |
+-----------------------+------------+------------------+
```

23.6 Tokenizer

23.6.1 Introduction

A tokenizer is a fundamental text preprocessing component in natural language processing (NLP) pipelines that segments raw text into structured tokens for downstream machine learning algorithms. These tokens serve as input features for various NLP models including word embeddings, language models, and text classification systems. Tokenization represents the initial feature engineering step that transforms unstructured text data into analyzable units, enabling advanced text analytics and deep learning applications.

23.6.2 Syntax

Below is the syntax for the NLTK Tokenizer text preprocessing function.

```
SELECT * FROM table(Unifyml.Nltk.Tokenizer(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  [ INPUTTEXTCOLUMNNAME => ( inputTextcolumn ) ] [,...]
  [ STOPWORDS => ( stopWords ) ] [,...]
  [ STEMMING => (stemming ) ] [,...]
  [ PARTSOFSPEECH => ( partsOfSpeech ) ] [,...]
  [ LEMMATIZING => ( lemmatizing ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.6.3 Input Arguments

Below are the input parameters for the NLTK Tokenizer text preprocessing function.

INPUTTABLE	Specifies the input dataset containing text features for tokenization.
INPUTTEXTCOLUMNNAME	[optional] Name of the text feature column for tokenization processing. **DataType**: STRING
PARALLELISM	[optional] Parallel processing level for tokenization operations. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
STOPWORDS	[optional] Enable stopword removal for feature dimensionality reduction. **DataType**: BOOLEAN, **Default**: true
STEMMING	[optional] Apply stemming algorithms for morphological normalization. **DataType**: BOOLEAN,**Default**: false
PARTSOFSPEECH	[optional] Include part-of-speech tagging for syntactic features. **DataType**: BOOLEAN,**Default**: false
LEMMATIZING	[optional] Apply lemmatization for lexical normalization. **DataType**: BOOLEAN,**Default**: false
INPUTCACHEID	[optional] Identifier for cached text dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for tokenization processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist tokenized features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output tokenized dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export tokenized features to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING

23.6.4 Output

Below is the expected output schema for the NLTK Tokenizer text preprocessing function.

23.6.5 Examples

Below are examples demonstrating the Tokenizer text preprocessing SQL function.

```
select * from table(Unifyml.Nltk.Tokenizer(
INPUTTABLE => ("select * from medicaldrama"),
INPUTEXTCOLUMNNAME => ("compliant"),
PARTITIONCOLUMN => ("id"),
OUTPUTCACHEID => ("tokencache1"),
STEMMING => true))

SQL Results:

+---------------------------+------------+------------------+
|       InputTableName      | DataCached | InputColumnNames |
+---------------------------+------------+------------------+
| select * from medicaldrama | true       | [compliant]      |
+---------------------------+------------+------------------+
```

23.7 TweetData

23.7.1 Introduction

The function executes NLTK-based social media data collection algorithms for social network analysis and sentiment mining applications.

This function provides real-time social media analytics capabilities, extracting structured datasets from Twitter APIs for social listening, trend analysis, and customer behavior modeling in data science workflows.

23.7.2 Syntax

Below is the syntax for the NLTK TweetData social media analytics function.

```
SELECT * FROM table(Unifyml.Nltk.TweetData(
  INPUTHASHTAG => ( inputSearchQuery ) [,...]
  [ INPUTFILTER => ( inputFilter ) ] [,...]
  [ METHOD => ( method ) ] [,...]
  [ STARTTIME => (startTime ) ] [,...]
  [ ENDTIME => ( endTime ) ] [,...]
  [ MAXTWEETSPERREQUEST => ( maxTweetsPerRequest ) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.7.3 Input Arguments

Below are the input parameters for the NLTK TweetData social media analytics function.

INPUTHASHTAG	Target hashtag or search query for social media data extraction. **DataType**: STRING
INPUTFILTER	[optional] Advanced filtering criteria for targeted data collection. **DataType**: STRING
PARALLELISM	[optional] Parallel processing level for API data collection. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
METHOD	[optional] Data collection methodology for API optimization. **DataType**: STRING
STARTTIME	[optional] Temporal window start for historical data collection. **DataType**: INTEGER
ENDTIME	[optional] Temporal window end for historical data collection. **DataType**: INTEGER
MAXTWEETSPERREQUEST	[optional] Maximum sample size per API request for rate limiting. **DataType**: INTEGER,**Default**: 100
INPUTCACHEID	[optional] Identifier for cached social media dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for social media data processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist social media features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output social media dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export social media data to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING

23.7.4 Output

Below is the expected output schema for the NLTK TweetData social media analytics function.

23.7.5 Examples

Below are examples demonstrating the TweetData social media analytics SQL function.

```
select * from table(Unifyml.Nltk.TweetData(
INPUTHASHTAG => ("COVID"),
INPUTFILTER => ("hoax -is:retweet lang:en"),
OUTPUTCACHEID => ("tweetDataCache")))

SQL Results:

+------------------+------------+
| InputSearchQuery | DataCached |
+------------------+------------+
| COVID            | true       |
+------------------+------------+
```

23.8 MatrixFactorization

23.8.1 Introduction

Matrix Factorization is a fundamental dimensionality reduction and latent feature extraction technique that decomposes large-scale user-item interaction matrices into lower-dimensional latent factor representations. This collaborative filtering method is essential for building scalable recommendation systems, enabling the discovery of hidden user preferences and item characteristics for personalized content delivery and predictive analytics.

23.8.2 Syntax

Below is the syntax for the UnifyML Matrix Factorization collaborative filtering function.

```
SELECT * FROM table(Unifyml.Nltk.Matrix\_Factorization(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.8.3 Input Arguments

Below are the input parameters for the NLTK Matrix Factorization collaborative filtering function.

USERCOLUMN	Name of the user identifier feature for collaborative filtering. **DataType**: INTEGER
ITEMCOLUMN	Name of the item identifier feature for recommendation modeling. **DataType**: INTEGER
RATING	Name of the interaction score feature for preference modeling. **DataType**: DOUBLE
PARALLELISM	[optional] Parallel processing level for matrix decomposition operations. **DataType**: INTEGER, **Default**: 2
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Name of the input interaction dataset. **DataType**: STRING
INPUTCACHEID	[optional] Identifier for cached interaction matrix. **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
CACHETYPE	[optional] Caching strategy for latent factor matrices. **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Chunk size for matrix factorization optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist latent factor embeddings to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output embedding dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export latent factors to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
OUTPUTCACHEID	[optional] Identifier for cached latent factor embeddings. **DataType**: STRING

23.8.4 Output

Below is the expected output schema for the UnifyML Matrix Factorization collaborative filtering function.

InputTableName	Source interaction dataset name for factorization. **DataType**: STRING
DataCached	Latent factor embedding caching status. **DataType**: STRING
OutputCacheId	Identifier of cached recommendation model. **DataType**: STRING
UserColumn	Name of the user feature column processed. **DataType**: STRING

23.8.5 Examples

Below are examples demonstrating the Matrix Factorization collaborative filtering SQL function.

```
select * from table(Unifyml.Nltk.Matrix\_Factorization(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("Matrix_factor_cache")));

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | UserColumn    |
+-------------------+------------+----------------------+-----------------+
| user_item_ratings | true       | Matrix_factor_cache  | user_id     |
+-------------------+------------+----------------------+-----------------+
```

23.9 CosineSimilarity

23.9.1 Introduction

Cosine Similarity is a fundamental distance metric and similarity measure in machine learning that quantifies the angular similarity between high-dimensional feature vectors, regardless of their magnitude. This metric is particularly effective for text analytics, collaborative filtering, and content-based recommendation systems, providing robust similarity calculations for sparse feature spaces and normalized vector comparisons.

23.9.2 Syntax

Below is the syntax for the UnifyML NLTK Cosine Similarity distance computation function.

```
SELECT * FROM table(Unifyml.Nltk.Cosine\_Similarity(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.9.3 Input Arguments

Below are the input parameters for the NLTK Cosine Similarity distance computation function.

USERCOLUMN	Name of the user identifier feature for similarity computation. **DataType**: INTEGER
ITEMCOLUMN	Name of the item identifier feature for vector comparison. **DataType**: INTEGER
RATING	Name of the interaction score feature for similarity weighting. **DataType**: DOUBLE
PARALLELISM	[optional] Parallel processing level for similarity matrix computation. **DataType**: INTEGER, **Default**: 2
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Name of the input feature dataset. **DataType**: STRING
INPUTCACHEID	[optional] Identifier for cached feature vectors. **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
CACHETYPE	[optional] Caching strategy for similarity matrices. **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Chunk size for similarity computation optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist similarity scores to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output similarity dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export similarity matrix to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
OUTPUTCACHEID	[optional] Identifier for cached similarity matrix. **DataType**: STRING

23.9.4 Output

Below is the expected output schema for the UnifyML NLTK Cosine Similarity distance computation function.

InputTableName	Source feature dataset name for similarity analysis. **DataType**: STRING
DataCached	Similarity matrix caching status. **DataType**: STRING
OutputCacheId	Identifier of cached similarity computations. **DataType**: STRING
UserColumn	Name of the user feature column processed. **DataType**: STRING

23.9.5 Examples

Below are examples demonstrating the Cosine Similarity distance computation SQL function.

```
select * from table(Unifyml.Nltk.Cosine\_Similarity(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("Cosine_Similarity_cache")));

SQL Results:

+-------------------+------------+--------------------------+--------------
|   InputTableName  | DataCached | OutputCacheId         | UserColumn   |
+-------------------+------------+----------------------+---------------
| user_item_ratings | true       | Cosine_Similarity_cache  | user_id    |
+-------------------+------------+--------------------------+--------------
```

23.10 WeightedHybrid

23.10.1 Introduction

Weighted Hybrid Recommendation Systems represent an advanced ensemble learning approach that combines multiple recommendation algorithms through weighted aggregation strategies. This sophisticated methodology leverages the strengths of diverse collaborative filtering, content-based, and matrix factorization techniques to deliver superior prediction accuracy and robust personalization for complex recommendation scenarios and cold-start problems.

23.10.2 Syntax

Below is the syntax for the UnifyML NLTK Weighted Hybrid ensemble recommendation function.

```
SELECT * FROM table(Unifyml.Nltk.Weighted\_Hybrid(
   USERCOLUMN => ( userColumn )  [,...]
   ITEMCOLUMN => ( itemColumn )  [,...]
   RATING => (rating )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.10.3 Input Arguments

Below are the input parameters for the NLTK Weighted Hybrid ensemble recommendation function.

USERCOLUMN	Name of the user identifier feature for ensemble modeling. **DataType**: INTEGER
ITEMCOLUMN	Name of the item identifier feature for hybrid recommendations. **DataType**: INTEGER
RATING	Name of the interaction score feature for ensemble weighting. **DataType**: DOUBLE
PARALLELISM	[optional] Parallel processing level for ensemble model training. **DataType**: INTEGER, **Default**: 2
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Name of the input interaction dataset. **DataType**: STRING
INPUTCACHEID	[optional] Identifier for cached interaction features. **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
CACHETYPE	[optional] Caching strategy for ensemble model components. **DataType**: STRING, **Default**: Cachetype
BYTESPERCHUNK	[optional] Chunk size for ensemble training optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist ensemble predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output recommendation dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export ensemble recommendations to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
OUTPUTCACHEID	[optional] Identifier for cached ensemble model. **DataType**: STRING

23.10.4 Output

Below is the expected output schema for the UnifyML NLTK Weighted Hybrid ensemble recommendation function.

InputTableName	Source interaction dataset name for ensemble training. **DataType**: STRING
DataCached	Ensemble model caching status. **DataType**: STRING
OutputCacheId	Identifier of cached hybrid recommendation system. **DataType**: STRING
UserColumn	Name of the user feature column processed. **DataType**: STRING

23.10.5 Examples

Below are examples demonstrating the Weighted Hybrid ensemble recommendation SQL function.

```
select * from table(Unifyml.Nltk.Weighted\_Hybrid(
INPUTTABLE => ("user_item_ratings"),
USERCOLUMN => ("user_id"),
ITEMCOLUMN => ("movie_id"),
RATING => ("rating"),
OUTPUTCACHEID => ("Weighted_Hybrid_cache")));

SQL Results:

+------------------+-----------+-------------------------+--------------
|   InputTableName  | DataCached | OutputCacheId         | UserColumn   |
+------------------+-----------+-------------------------+--------------
| user_item_ratings | true       | Weighted_Hybrid_cache     | user_id    |
+------------------+-----------+-------------------------+--------------
```

23.11 Tfidf

23.11.1 Introduction

Term Frequency - Inverse Document Frequency (TF-IDF) is a fundamental text mining and feature engineering technique used to quantify the statistical importance of terms within documents relative to a corpus. This numerical statistic transforms raw text into meaningful feature vectors for machine learning applications, enabling document classification, information retrieval, and semantic analysis workflows.

23.11.2 Syntax

Below is the syntax for the NLTK TF-IDF vectorization function.

```
SELECT * FROM table(Unifyml.Nltk.Tf\_Idf(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  INPUTTEXTCOLUMNNAME => ( inputTextcolumn )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.11.3 Input Arguments

Below are the input parameters for the NLTK TF-IDF vectorization function.

INPUTTABLE	Specifies the input dataset containing text features for vectorization. **DataType**: STRING
INPUTTEXTCOLUMNNAME	Name of the text feature column for TF-IDF transformation. **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARALLELISM	[optional] Parallel processing level for vectorization operations. **DataType**: INTEGER, **Default**: 2
INPUTCACHEID	[optional] Identifier for cached text dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for text processing optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist vectorized features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output feature dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export vectorized features to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
CACHETYPE	[optional] Caching strategy for feature vectors. **DataType**: STRING, **Default**: Cachetype
OUTPUTCACHEID	[optional] Identifier for cached TF-IDF feature vectors. **DataType**: STRING

23.11.4 Output

Below is the expected output schema for the UnifyML NLTK TF-IDF vectorization function.

InputTableName	Source text dataset name for feature extraction. **DataType**: STRING
DataCached	Feature vector caching status. **DataType**: STRING
OutputCacheId	Identifier of cached TF-IDF feature vectors. **DataType**: STRING
InputColumns	Name of the processed text feature column. **DataType**: STRING

23.11.5 Examples

Below are examples demonstrating the UnifyML NLTK TF-IDF vectorization SQL function.

```
select * from table(Unifyml.Nltk.Tf\_Idf(
INPUTTABLE => ("name_entity_data"),
INPUTTEXTCOLUMNNAME => ("text"),
OUTPUTCACHEID => ("tf_idf_cache")));

SQL Results:

+------------------+-----------+---------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | InputColumns   |
+------------------+-----------+---------------------+-----------------+
| name_entity_data  | true       | tf_idf_cache         | text        |
+------------------+-----------+---------------------+-----------------+
```

23.12 NameEntity

23.12.1 Introduction

Named Entity Recognition (NER) is a fundamental information extraction technique in Natural Language Processing that identifies and classifies structured entities within unstructured text data. This supervised learning task is essential for building intelligent applications including conversational AI systems, semantic search engines, knowledge graph construction, and automated document classification pipelines.
NER implements sequence labeling algorithms to detect and categorize named entities in textual data into predefined semantic classes. This process involves locating spans of text that represent specific entity types such as persons, organizations, locations, and temporal expressions, then applying appropriate entity labels for downstream machine learning applications.

23.12.2 Syntax

Below is the syntax for the NLTK NameEntity recognition function.

```
SELECT * FROM table(Unifyml.Nltk.NameEntity(
  INPUTTABLE => ( { table | view | (query) })  [,...]
  INPUTTEXTCOLUMNNAME => ( inputTextcolumn )  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ CACHETYPE => (cacheType) ] [,...]
  [ OUTPUTCACHEID => (outputCacheId) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => (outputtablename) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

23.12.3 Input Arguments

Below are the input parameters for the NLTK NameEntity recognition function.

INPUTTABLE	Specifies the input dataset containing text features for entity extraction.
INPUTTEXTCOLUMNNAME	Name of the text feature column for entity recognition processing. **DataType**: STRING
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARALLELISM	[optional] Parallel processing level for NER model inference. **DataType**: INTEGER, **Default**: 2
INPUTCACHEID	[optional] Identifier for cached text dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for entity extraction optimization. **DataType**: INTEGER, **Default**: 64000000
SAVETODB	[optional] Persist extracted entity features to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output entity dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export entity annotations to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
CACHETYPE	[optional] Caching strategy for entity annotations. **DataType**: STRING, **Default**: Cachetype
OUTPUTCACHEID	[optional] Identifier for cached entity extractions. **DataType**: STRING

23.12.4 Output

Below is the expected output schema for the UnifyML NLTK NameEntity recognition function.

InputTableName	Source text dataset name for entity extraction. **DataType**: STRING
DataCached	Entity annotation caching status. **DataType**: STRING
OutputCacheId	Identifier of cached entity recognition results. **DataType**: STRING
InputColumns	Name of the processed text feature column. **DataType**: STRING

23.12.5 Examples

Below are examples demonstrating the UnifyML NLTK NameEntity recognition SQL function.

```
select * from table(Unifyml.Nltk.NameEntity(
INPUTTABLE => ("name_entity_data"),
INPUTTEXTCOLUMNNAME => ("text"),
OUTPUTCACHEID => ("nameentity_cache")));

SQL Results:

+-------------------+------------+----------------------+-----------------+
|   InputTableName  | DataCached | OutputCacheId        | InputColumns    |
+-------------------+------------+----------------------+-----------------+
| name_entity_data  | true       | nameentity_cache     | text            |
+-------------------+------------+----------------------+-----------------+
```

X Unsupervised Learning and Clustering Algorithms

24. Unifyml

This section provides details about the supported UnifyML unsupervised learning and clustering algorithms for pattern discovery and data segmentation.

24.1 BisectingKmeans

24.1.1 Introduction

Bisecting K-Means is an advanced hierarchical clustering algorithm that combines the efficiency of centroid-based clustering with divisive hierarchical approaches. This top-down clustering methodology iteratively partitions datasets by applying K-Means recursively, creating a dendrogram structure that enables multi-resolution cluster analysis and improved scalability for large-scale unsupervised learning tasks.

For further references please see BisectingKmeans.

24.1.2 Syntax

Below is the syntax for the UnifyML BisectingKmeans hierarchical clustering function.

```
SELECT * FROM table(Unifyml.BisectingKmeans(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ SEED => seed ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERPARAMSPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

24.1.3 Input Arguments

Below are the input parameters for the UnifyML BisectingKmeans hierarchical clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for clustering a **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for cluster analysis.
INPUTDATATYPES	[optional] Data types specification for feature columns in the dataset. **DataType**: ARRAY
SEED	[optional] Random seed parameter for reproducible clustering results. **DataType**: LONG, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITER	[optional] Maximum iterations parameter for convergence control. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Target number of clusters for hierarchical partitioning. **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for model validation and performance evaluat **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal cluste **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation or TrainValidation for **DataType**: STRING, **Default**: CrossValidation
SEEDPARAMS	[optional] Parameter grid for random seed hyperparameter tuning. **DataType**: ARRAY, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITERPARAMSPARAMS	[optional] Parameter grid for maximum iterations hyperparameter tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
PARALLELISM	[optional] Parallel processing level for distributed clustering operations. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for model evaluation and selecti **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reductio **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Name of the pre-trained PCA model for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Persist the trained clustering model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output clustering model name for persistence. **DataType**: STRING

DATACLEANING	[optional] Enable data preprocessing and quality improvement operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeoutliers, fil **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill cleaning metho **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization for improved clustering performa **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Normalizer, B **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed data processing optimization. **DataType**: INTEGER, **Default**: 64000000

24.1.4 Output

Below is the expected output schema for the UnifyML BisectingKmeans hierarchical clustering function.

Silhouette	Silhouette coefficient for cluster quality evaluation using squared Euclidean distance. **DataType**: DOUBLE
ClusterSummary	Cluster analysis summary statistics and metadata. **DataType**: ARRAY
TrainingCost	Within-cluster sum of squares (WCSS) training cost metric. **DataType**: ARRAY
NumberOfIter	Number of iterations required for algorithm convergence. **DataType**: ARRAY

24.1.5 Examples

Below are examples demonstrating the BisectingKmeans hierarchical clustering SQL function.

```
select * from table(Unifyml.BisectingKmeans(
INPUTTABLE => ("select * from eshoppinglimit limit 100"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "pageclothing",
"color", "location", "modelphoto", "price", "pricetwo", "page"],
NUMPARTITION => 5,
PARTITIONCOLUMN => ("page"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+-----------------+-------------------+
| CoefficientName |       Value       |
+-----------------+-------------------+
| ClusterSummary  | 2                 |
| TrainingCost    | 21066.32051283121 |
| NumberOfIter    | 32                |
| Silhouette      | 0.767463340881147 |
+-----------------+-------------------+
```

24.2 BisectingKmeansPredict

24.2.1 Introduction

Once UnifyML Bisecting K-means clustering model is trained, the model enables prediction and cluster assignment for new unseen data points. This inference capability supports real-time cluster assignment, anomaly detection, and pattern recognition applications in production machine learning pipelines.

For further references please see BisectingKmeans.

24.2.2 Syntax

Below is the syntax for the UnifyML BisectingKmeansPredict cluster inference function.

```
SELECT * FROM table(Unifyml.BisectingKmeansPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

24.2.3 Input Arguments

Below are the input parameters for the UnifyML BisectingKmeansPredict cluster inference function.

INPUTMODELNAME	Specifies the pre-trained clustering model name for cluster assignment infere **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for cluster predictio
INPUTDATATYPES	[optional] Data types specification for feature columns in the prediction datas **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for prediction dataset quality improvem **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeou **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in prediction dataset. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for prediction dataset consistency with train **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Norm **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export cluster predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

24.2.4 Output

Below is the expected output schema for the UnifyML BisectingKmeansPredict cluster inference function.

INPUTCOLUMNNAMES	Feature column names processed by the clustering model.
PREDICTCOLUMN	Predicted cluster assignment labels for input data points.

24.2.5 Examples

Below are examples demonstrating the BisectingKmeansPredict cluster inference SQL function.

```
select * from table(Unifyml.BisectingKmeansPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelBKM")))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

24.3 Gmm

24.3.1 Introduction

Gaussian Mixture Model (GMM) is a sophisticated probabilistic clustering algorithm that models data as a weighted combination of multiple Gaussian distributions. Unlike hard clustering approaches, GMM implements soft clustering with probabilistic membership assignments, enabling flexible cluster boundaries and handling overlapping clusters through expectation-maximization optimization for complex data distributions and uncertainty quantification.

For further references please see Gmm.

24.3.2 Syntax

Below is the syntax for the UnifyML GMM probabilistic clustering function.

```
SELECT * FROM table(Unifyml.Gmm(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ SEED => seed ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERPARAMSPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

24.3.3 Input Arguments

Below are the input parameters for the UnifyML GMM probabilistic clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for probabilisti **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for mixture model e
INPUTDATATYPES	[optional] Data types specification for feature columns in the dataset. **DataType**: ARRAY
SEED	[optional] Random seed parameter for reproducible probabilistic clustering r **DataType**: LONG, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITER	[optional] Maximum iterations parameter for expectation-maximization conv **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Number of Gaussian components for mixture model estimation. **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for probabilistic model validation and evaluat **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal mixtu **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation or TrainValidation for **DataType**: STRING, **Default**: CrossValidation
SEEDPARAMS	[optional] Parameter grid for random seed hyperparameter tuning. **DataType**: ARRAY, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITERPARAMSPARAMS	[optional] Parameter grid for maximum iterations hyperparameter tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
PARALLELISM	[optional] Parallel processing level for distributed expectation-maximization **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for probabilistic model evaluatio **DataType**: INTEGER, **Default**: 2
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reductio **DataType**: BOOLEAN, **Default**: false
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
INPUTPCAMODELNAME	[optional] Name of the pre-trained PCA model for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Persist the trained mixture model for future probabilistic inference **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output mixture model name for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing and quality improvement operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeou **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill cleani **DataType**: ARRAY

DATASCALING	[optional] Enable feature scaling and normalization for improved mixture model perfor **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Normalizer, Bin **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed data processing optimization. **DataType**: INTEGER, **Default**: 64000000

24.3.4 Output

Below is the expected output schema for the UnifyML GMM probabilistic clustering function.

Silhouette	Silhouette coefficient for probabilistic cluster quality evaluation using squared Euclidean dis **DataType**: DOUBLE
Weights	Mixture component weights representing cluster prevalence in the dataset. **DataType**: DOUBLE
Mu	Gaussian component mean vectors for each mixture component. **DataType**: DOUBLE
Sigma	Gaussian component covariance matrices for each mixture component. **DataType**: DOUBLE
ClusterSummary	Mixture model summary statistics and component analysis. **DataType**: ARRAY
LogLikeliHood	Log-likelihood score for mixture model goodness-of-fit evaluation. **DataType**: ARRAY
NumberOfIter	Number of expectation-maximization iterations required for convergence. **DataType**: ARRAY

24.3.5 Examples

Below are examples demonstrating the GMM probabilistic clustering SQL function.

```
select * from table(Unifyml.Gmm(
INPUTTABLE => ("select * from eshoppinglimit limit 100"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "pageclothing",
"color", "location", "modelphoto", "price", "pricetwo", "page"],
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+-------------------------------------------------------------------------+
| CoefficientName |     Value                                             |
+-------------------------------------------------------------------------+
| Weights         | 0.26711479419420114,0.7328852058057987                |
| Mu              | [2008.0,3.9999999999999996][2008.0000000000002,4.0]|
| Sigma           | -1.7164802454545531E-9  1.676250239701712E-12         |
| ClusterSummary  | 2                                                     |
| LogLikeliHood   | -96.75781744733415                                    |
| NumberOfIter    | 32                                                    |
| Silhouette      | 0.0                                                   |
+-----------------+-------------------------------------------------------+
```

24.4 GmmPredict

24.4.1 Introduction

Once UnifyML GMM probabilistic clustering model is trained, the model enables probabilistic cluster assignment and membership prediction for new unseen data points. This inference capability supports soft clustering predictions with uncertainty quantification, anomaly detection through likelihood estimation, and probabilistic pattern recognition in production machine learning pipelines.

For further references please see Gmm.

24.4.2 Syntax

Below is the syntax for the UnifyML GmmPredict probabilistic inference function.

```
SELECT * FROM table(Unifyml.GmmPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

24.4.3 Input Arguments

Below are the input parameters for the UnifyML GmmPredict probabilistic inference function.

INPUTMODELNAME	Specifies the pre-trained GMM model name for probabilistic cluster assignm **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for probabilistic clu
INPUTDATATYPES	[optional] Data types specification for feature columns in the prediction datas **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for prediction dataset quality improvem **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeou **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in prediction dataset. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for prediction dataset consistency with train **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Norm **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist probabilistic cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export probabilistic cluster assignments to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

24.4.4 Output

Below is the expected output schema for the UnifyML GmmPredict probabilistic inference function.

INPUTCOLUMNNAMES	Feature column names processed by the probabilistic clustering model.
PREDICTCOLUMN	Predicted probabilistic cluster assignment labels with membership probabilities.

24.4.5 Examples

Below are examples demonstrating the GmmPredict probabilistic inference SQL function.

```
select * from table(Unifyml.GmmPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelGMM")))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

24.5 Kmeans

24.5.1 Introduction

K-Means is a foundational centroid-based clustering algorithm in unsupervised machine learning that partitions data points into K distinct clusters based on feature similarity. This iterative optimization algorithm minimizes the within-cluster sum of squared distances (WCSS) between data points and their assigned cluster centroids, enabling efficient pattern discovery and data segmentation for exploratory data analysis and customer analytics applications.

For further references please see Kmeans.

24.5.2 Syntax

Below is the syntax for the UnifyML K-Means clustering function.

```
SELECT * FROM table(Unifyml.Kmeans(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ SEED => seed ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERPARAMSPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

24.5.3 Input Arguments

Below are the input parameters for the UnifyML K-Means clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for centroid-ba **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for cluster analysis.
INPUTDATATYPES	[optional] Data types specification for feature columns in the dataset. **DataType**: ARRAY
SEED	[optional] Random seed parameter for reproducible clustering results. **DataType**: LONG, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITER	[optional] Maximum iterations parameter for centroid convergence control. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Target number of clusters (K value) for partitioning. **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for model validation and performance evaluat **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal cluste **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation or TrainValidation for **DataType**: STRING, **Default**: CrossValidation
SEEDPARAMS	[optional] Parameter grid for random seed hyperparameter tuning. **DataType**: ARRAY, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITERPARAMSPARAMS	[optional] Parameter grid for maximum iterations hyperparameter tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
PARALLELISM	[optional] Parallel processing level for distributed clustering operations. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for model evaluation and selecti **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reductio **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Name of the pre-trained PCA model for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Persist the trained clustering model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output clustering model name for persistence. **DataType**: STRING

DATACLEANING	[optional] Enable data preprocessing and quality improvement operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeoutliers, fil **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill cleaning metho **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization for improved clustering performa **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Normalizer, B **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed data processing optimization. **DataType**: INTEGER, **Default**: 64000000

24.5.4 Output

Below is the expected output schema for the UnifyML K-Means clustering function.

Silhouette	Silhouette coefficient for cluster quality evaluation using squared Euclidean distance. **DataType**: DOUBLE
Centers	Cluster centroid coordinates for each identified cluster. **DataType**: DOUBLE

24.5.5 Examples

Below are examples demonstrating the K-Means clustering SQL function.

```
select * from table(Unifyml.Kmeans(
INPUTCACHEID => ("cacheid"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "pageclothing",
 "color", "location", "modelphoto", "price", "pricetwo", "page"],
SAVEMODEL => true,
OUTPUTMODEL => ("modelKM")))

SQL Results:
+-----------------+-------------------+
| CoefficientName |       Value       |
+-----------------+-------------------+
| ClusterSummary  | 2                 |
| TrainingCost    | 21066.32051283121 |
| NumberOfIter    | 32                |
| Silhouette      | 0.767463340881147 |
| OutputModelName | modelKM           |
+-----------------+-------------------+
```

24.6 KmeansPredict

24.6.1 Introduction

Once UnifyML K-Means clustering model is trained, the model enables cluster assignment prediction for new unseen data points. This inference capability supports real-time cluster classification, customer segmentation, and pattern recognition applications in production machine learning pipelines.

For further references please see Kmeans.

24.6.2 Syntax

Below is the syntax for the UnifyML KmeansPredict cluster inference function.

```
SELECT * FROM table(Unifyml.KmeansPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

24.6.3 Input Arguments

Below are the input parameters for the UnifyML KmeansPredict cluster inference function.

INPUTMODELNAME	Specifies the pre-trained K-Means model name for cluster assignment infere **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for cluster predictio
INPUTDATATYPES	[optional] Data types specification for feature columns in the prediction data **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for prediction dataset quality improvem **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeou **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in prediction dataset. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for prediction dataset consistency with train **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Norm **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export cluster predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

24.6.4 Output

Below is the expected output schema for the UnifyML KmeansPredict cluster inference function.

INPUTCOLUMNNAMES	Feature column names processed by the clustering model.
PREDICTCOLUMN	Predicted cluster assignment labels for input data points.

24.6.5 Examples

Below are examples demonstrating the KmeansPredict cluster inference SQL function.

```
select * from table(Unifyml.KmeansPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelKM3")))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

25. Spark

This section provides details about the supported Spark distributed clustering algorithms for big data analytics and scalable machine learning operations.

25.1 BisectingKmeans

25.1.1 Introduction

Bisecting K-Means is an advanced hierarchical clustering algorithm that combines the efficiency of centroid-based clustering with divisive hierarchical approaches. This top-down clustering methodology iteratively partitions datasets by applying K-Means recursively, creating a dendrogram structure that enables multi-resolution cluster analysis and improved scalability for large-scale distributed data processing tasks.

For further references please see BisectingKmeans.

25.1.2 Syntax

Below is the syntax for the Spark BisectingKmeans distributed clustering function.

```
SELECT * FROM table(Unifyml.Spark.BisectingKmeans(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ SEED => seed ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERPARAMSPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

25.1.3 Input Arguments

Below are the input parameters for the Spark BisectingKmeans distributed clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for distributed **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for distributed clust
INPUTDATATYPES	[optional] Data types specification for feature columns in the distributed data **DataType**: ARRAY
SEED	[optional] Random seed parameter for reproducible distributed clustering res **DataType**: LONG, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITER	[optional] Maximum iterations parameter for distributed convergence contro **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Target number of clusters for distributed hierarchical partitioning. **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed model validation and evaluatio **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal distrib **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Distributed model validation strategy: CrossValidation or TrainVa **DataType**: STRING, **Default**: CrossValidation
SEEDPARAMS	[optional] Parameter grid for random seed hyperparameter tuning in distribu **DataType**: ARRAY, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITERPARAMSPARAMS	[optional] Parameter grid for maximum iterations hyperparameter tuning in **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
PARALLELISM	[optional] Distributed parallel processing level for Spark clustering operation **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for distributed model evaluation **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Number of Spark data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for Spark data partitioning strategy. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for distributed dimensionali **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Name of the pre-trained PCA model for distributed feature transfo **DataType**: STRING
SAVEMODEL	[optional] Persist the trained distributed clustering model for future inferenc **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output distributed clustering model name for persistence. **DataType**: STRING

DATACLEANING	[optional] Enable distributed data preprocessing and quality improvement operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data cleaning strategies: removerow, removeduplicates, remove **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in distributed processing. **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling and normalization for improved clusterin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature scaling techniques: MinMaxScaler, StandardScaler, No **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for distributed MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for distributed MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached distributed dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed big data processing optimization. **DataType**: INTEGER, **Default**: 64000000

25.1.4 Output

Below is the expected output schema for the Spark BisectingKmeans distributed clustering function.

Silhouette	Silhouette coefficient for distributed cluster quality evaluation using squared Euclidean dista **DataType**: DOUBLE
ClusterSummary	Distributed cluster analysis summary statistics and metadata. **DataType**: ARRAY
TrainingCost	Within-cluster sum of squares (WCSS) training cost metric for distributed clustering. **DataType**: ARRAY
NumberOfIter	Number of iterations required for distributed algorithm convergence. **DataType**: ARRAY

25.1.5 Examples

Below are examples demonstrating the Spark BisectingKmeans distributed clustering SQL function.

```
select * from table(Unifyml.Spark.BisectingKmeans(
INPUTTABLE => ("select * from eshoppinglimit limit 100"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "pageclothing",
"color", "location", "modelphoto", "price", "pricetwo", "page"],
NUMPARTITION => 5,
PARTITIONCOLUMN => ("page"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+-----------------+-------------------+
| CoefficientName |       Value       |
+-----------------+-------------------+
| ClusterSummary  | 2                 |
| TrainingCost    | 21066.32051283121 |
| NumberOfIter    | 32                |
| Silhouette      | 0.767463340881147 |
+-----------------+-------------------+
```

25.2 BisectingKmeansPredict

25.2.1 Introduction

Once Spark Bisecting K-Means distributed clustering model is trained, the model enables scalable cluster assignment prediction for new unseen data points across distributed computing environments. This inference capability supports real-time big data cluster classification, large-scale customer segmentation, and distributed pattern recognition applications in production machine learning pipelines.

For further references please see BisectingKmeans.

25.2.2 Syntax

Below is the syntax for the Spark BisectingKmeansPredict distributed inference function.

```
SELECT * FROM table(Unifyml.Spark.BisectingKmeansPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

25.2.3 Input Arguments

Below are the input parameters for the Spark BisectingKmeansPredict distributed inference function.

INPUTMODELNAME	Specifies the pre-trained distributed clustering model name for scalable clust **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for distributed clust
INPUTDATATYPES	[optional] Data types specification for feature columns in the distributed prec **DataType**: ARRAY
DATACLEANING	[optional] Enable distributed data preprocessing for prediction dataset quality **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data cleaning strategies: removerow, removeduplicates **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in distributed predicti **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling for prediction dataset consistenc **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature scaling techniques: MinMaxScaler, StandardSc **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for distributed MinMaxScaler feature transformat **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for distributed MinMaxScaler feature transformat **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached distributed prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist distributed cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output distributed prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed cluster predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

25.2.4 Output

Below is the expected output schema for the Spark BisectingKmeansPredict distributed inference function.

INPUTCOLUMNNAMES	Feature column names processed by the distributed clustering model.
PREDICTCOLUMN	Predicted cluster assignment labels for distributed input data points.

25.2.5 Examples

Below are examples demonstrating the Spark BisectingKmeansPredict distributed inference SQL function.

```
select * from table(Unifyml.Spark.BisectingKmeansPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelBKM")))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

25.3 Gmm

25.3.1 Introduction

Gaussian Mixture Model (GMM) is a sophisticated probabilistic clustering algorithm optimized for distributed computing that models large-scale datasets as weighted combinations of multiple Gaussian distributions. Unlike hard clustering approaches, this Spark-based implementation enables soft clustering with probabilistic membership assignments across distributed computing clusters, providing flexible cluster boundaries and handling overlapping clusters through distributed expectation-maximization optimization for complex big data distributions and uncertainty quantification.

For further references please see Gmm.

25.3.2 Syntax

Below is the syntax for the Spark GMM distributed probabilistic clustering function.

```
SELECT * FROM table(Unifyml.Spark.Gmm(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ SEED => seed ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERPARAMSPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

25.3.3 Input Arguments

Below are the input parameters for the Spark GMM distributed probabilistic clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for distributed **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for distributed mixt
INPUTDATATYPES	[optional] Data types specification for feature columns in the distributed data **DataType**: ARRAY
SEED	[optional] Random seed parameter for reproducible distributed probabilistic **DataType**: LONG, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITER	[optional] Maximum iterations parameter for distributed expectation-maxim **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Number of Gaussian components for distributed mixture model es **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed probabilistic model validation **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal distrib **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Distributed model validation strategy: CrossValidation or TrainVa **DataType**: STRING, **Default**: CrossValidation
SEEDPARAMS	[optional] Parameter grid for random seed hyperparameter tuning in distribu **DataType**: ARRAY, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITERPARAMSPARAMS	[optional] Parameter grid for maximum iterations hyperparameter tuning in **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
PARALLELISM	[optional] Distributed parallel processing level for Spark expectation-maxim **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for distributed probabilistic mod **DataType**: INTEGER, **Default**: 2
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for distributed dimensionali **DataType**: BOOLEAN, **Default**: false
NUMPARTITION	[optional] Number of Spark data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for Spark data partitioning strategy. **DataType**: STRING
INPUTPCAMODELNAME	[optional] Name of the pre-trained PCA model for distributed feature transfo **DataType**: STRING
SAVEMODEL	[optional] Persist the trained distributed mixture model for future probabilist **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output distributed mixture model name for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable distributed data preprocessing and quality improvement op **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data cleaning strategies: removerow, removeduplicates **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in distributed process **DataType**: ARRAY

DATASCALING	[optional] Enable distributed feature scaling and normalization for improved mixture m **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature scaling techniques: MinMaxScaler, StandardScaler, Norm **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for distributed MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for distributed MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached distributed dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed big data processing optimization. **DataType**: INTEGER, **Default**: 64000000

25.3.4 Output

Below is the expected output schema for the Spark GMM distributed probabilistic clustering function.

Silhouette	Silhouette coefficient for distributed probabilistic cluster quality evaluation using squared E **DataType**: DOUBLE
Weights	Distributed mixture component weights representing cluster prevalence across the dataset. **DataType**: DOUBLE
Mu	Distributed Gaussian component mean vectors for each mixture component. **DataType**: DOUBLE
Sigma	Distributed Gaussian component covariance matrices for each mixture component. **DataType**: DOUBLE
ClusterSummary	Distributed mixture model summary statistics and component analysis. **DataType**: ARRAY
LogLikeliHood	Log-likelihood score for distributed mixture model goodness-of-fit evaluation. **DataType**: ARRAY
NumberOfIter	Number of distributed expectation-maximization iterations required for convergence. **DataType**: ARRAY

25.3.5 Examples

Below are examples demonstrating the Spark GMM distributed probabilistic clustering SQL function.

```
select * from table(Unifyml.Spark.Gmm(
INPUTTABLE => ("select * from eshoppinglimit limit 100"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "pageclothing",
"color", "location", "modelphoto", "price", "pricetwo", "page"],
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"]))

SQL Results:
+-------------------------------------------------------------------------+
| CoefficientName |     Value                                             |
+-------------------------------------------------------------------------+
| Weights         | 0.26711479419420114,0.7328852058057987                |
| Mu              | [2008.0,3.9999999999999996][2008.0000000000002,4.0]|
| Sigma           | -1.7164802454545531E-9  1.676250239701712E-12         |
| ClusterSummary  | 2                                                     |
| LogLikeliHood   | -96.75781744733415                                    |
| NumberOfIter    | 32                                                    |
| Silhouette      | 0.0                                                   |
+-----------------+-------------------------------------------------------+
```

25.4 GmmPredict

25.4.1 Introduction

Once Spark GMM distributed probabilistic clustering model is trained, the model enables scalable probabilistic cluster assignment and membership prediction for new unseen data points across distributed computing environments. This inference capability supports distributed soft clustering predictions with uncertainty quantification, large-scale anomaly detection through likelihood estimation, and distributed probabilistic pattern recognition in production big data machine learning pipelines.

For further references please see Gmm.

25.4.2 Syntax

Below is the syntax for the Spark GmmPredict distributed probabilistic inference function.

```
SELECT * FROM table(Unifyml.Spark.GmmPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

25.4.3 Input Arguments

Below are the input parameters for the Spark GmmPredict distributed probabilistic inference function.

INPUTMODELNAME	Specifies the pre-trained distributed GMM model name for scalable probabil **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for distributed proba
INPUTDATATYPES	[optional] Data types specification for feature columns in the distributed prec **DataType**: ARRAY
DATACLEANING	[optional] Enable distributed data preprocessing for prediction dataset quality **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data cleaning strategies: removerow, removeduplicates **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in distributed predicti **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling for prediction dataset consistenc **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature scaling techniques: MinMaxScaler, StandardSc **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for distributed MinMaxScaler feature transformat **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for distributed MinMaxScaler feature transformat **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached distributed prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist distributed probabilistic cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output distributed prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed probabilistic cluster assignments to CSV forma **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

25.4.4 Output

Below is the expected output schema for the Spark GmmPredict distributed probabilistic inference function.

INPUTCOLUMNNAMES	Feature column names processed by the distributed probabilistic clustering model
PREDICTCOLUMN	Predicted probabilistic cluster assignment labels with membership probabilities fo

25.4.5 Examples

Below are examples demonstrating the Spark GmmPredict distributed probabilistic inference SQL function.

```
select * from table(Unifyml.Spark.GmmPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelGMM")))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

25.5 Kmeans

25.5.1 Introduction

K-Means is a fundamental centroid-based clustering algorithm in unsupervised machine learning that partitions data points into K distinct clusters based on feature similarity. This iterative optimization algorithm minimizes the within-cluster sum of squared distances (WCSS) between data points and their assigned cluster centroids, enabling effective pattern discovery and data segmentation for exploratory data analysis and customer segmentation applications.

For further references please see Kmeans.

K-means clustering with support for k-means|| initialization proposed by Bahmani et al. implements scalable parallel initialization strategies for improved convergence and performance in distributed computing environments.

25.5.2 Syntax

Below is the syntax for the Spark K-Means distributed clustering function.

```
SELECT * FROM table(Unifyml.Spark.Kmeans(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ SEED => seed ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERPARAMSPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

25.5.3 Input Arguments

Below are the input parameters for the Spark K-Means distributed clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for distributed **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for large-scale clust
INPUTDATATYPES	[optional] Data types specification for feature columns in the distributed data **DataType**: ARRAY
SEED	[optional] Random seed parameter for reproducible distributed clustering res **DataType**: LONG, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITER	[optional] Maximum iterations parameter for distributed convergence contro **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Target number of clusters (K) for distributed centroid-based partit **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for distributed model validation and evaluatio **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for optimal distrib **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation or TrainValidation for **DataType**: STRING, **Default**: CrossValidation
SEEDPARAMS	[optional] Parameter grid for random seed distributed hyperparameter tuning **DataType**: ARRAY, **Default**: 926680331, **Min**: 0, **Max**: 926680331
MAXITERPARAMSPARAMS	[optional] Parameter grid for maximum iterations distributed hyperparamete **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
PARALLELISM	[optional] Parallel processing level for distributed clustering operations acro **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for distributed model evaluation **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Number of data partitions for Spark distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for Spark data partitioning strategy. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for distributed dimensionali **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Name of the pre-trained PCA model for distributed feature transfo **DataType**: STRING
SAVEMODEL	[optional] Persist the trained distributed clustering model for future inferenc **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output distributed clustering model name for persistence. **DataType**: STRING

DATACLEANING	[optional] Enable distributed data preprocessing and quality improvement operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data cleaning strategies: removerow, removeduplicates, remove **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for distributed missing data handling when using fill clea **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling and normalization for improved clusterin **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature scaling techniques: MinMaxScaler, StandardScaler, No **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for distributed MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for distributed MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached distributed dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for Spark distributed data processing optimization. **DataType**: INTEGER, **Default**: 64000000

25.5.4 Output

Below is the expected output schema for the Spark K-Means distributed clustering function.

Silhouette	Silhouette coefficient for distributed cluster quality evaluation using squared Euclidean distance. **DataType**: DOUBLE
Centers	Distributed K-Means cluster centroid coordinates for each identified cluster. **DataType**: DOUBLE

25.5.5 Examples

Below are examples demonstrating the Spark K-Means distributed clustering SQL function.

```
select * from table(Unifyml.Spark.Kmeans(
INPUTCACHEID => ("cacheid"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "pageclothing",
 "color", "location", "modelphoto", "price", "pricetwo", "page"],
SAVEMODEL => true,
OUTPUTMODEL => ("modelKM")))

SQL Results:
+-----------------+-------------------+
| CoefficientName |       Value       |
+-----------------+-------------------+
| ClusterSummary  | 2                 |
| TrainingCost    | 21066.32051283121 |
| NumberOfIter    | 32                |
| Silhouette      | 0.767463340881147 |
| OutputModelName | modelKM           |
+-----------------+-------------------+
```

25.6 KmeansPredict

25.6.1 Introduction

Once Spark K-Means distributed clustering model is trained, the model enables scalable cluster assignment prediction for new unseen data points across distributed computing environments. This inference capability supports real-time cluster assignment, pattern recognition, and large-scale data segmentation applications in production machine learning pipelines for customer analytics and distributed anomaly detection.

For further references please see Kmeans.

25.6.2 Syntax

Below is the syntax for the Spark KmeansPredict distributed inference function.

```
SELECT * FROM table(Unifyml.Spark.KmeansPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

25.6.3 Input Arguments

Below are the input parameters for the Spark KmeansPredict distributed inference function.

INPUTMODELNAME	Specifies the pre-trained Spark K-Means model name for distributed cluster a **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for distributed clust
INPUTDATATYPES	[optional] Data types specification for feature columns in the distributed prec **DataType**: ARRAY
DATACLEANING	[optional] Enable distributed data preprocessing for prediction dataset quality **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data cleaning strategies: removerow, removeduplicates **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for distributed missing data handling in predicti **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling for prediction dataset consistenc **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature scaling techniques: MinMaxScaler, StandardSc **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for distributed MinMaxScaler feature transformat **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for distributed MinMaxScaler feature transformat **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached distributed prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist distributed cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output distributed prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed cluster predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for distributed prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

25.6.4 Output

Below is the expected output schema for the Spark KmeansPredict distributed inference function.

INPUTCOLUMNNAMES	Feature column names processed by the distributed clustering model.
PREDICTCOLUMN	Predicted cluster assignment labels for distributed input data points.

25.6.5 Examples

Below are examples demonstrating the Spark KmeansPredict distributed inference SQL function.

```
select * from table(Unifyml.Spark.KmeansPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelKM3")))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

26. ScikitLearn

This section provides details about the supported Scikit-Learn machine learning algorithms for single-node clustering and regression analysis.

26.1 Kmeans

26.1.1 Introduction

K-Means is a fundamental centroid-based clustering algorithm in unsupervised machine learning that partitions data points into K distinct clusters based on feature similarity. This iterative optimization algorithm minimizes the within-cluster sum of squared distances (WCSS) between data points and their assigned cluster centroids, enabling effective pattern discovery and data segmentation for exploratory data analysis and customer segmentation applications.

For further references please see Kmeans.

26.1.2 Syntax

Below is the syntax for the Scikit-Learn K-Means clustering function.

```
SELECT * FROM table(Unifyml.Scikitlearn.Kmeans(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

26.1.3 Input Arguments

Below are the input parameters for the Scikit-Learn K-Means clustering function.

INPUTCOLUMNS	Specifies the feature columns containing numerical variables for clustering analys **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the input dataset containing features for cluster analysis.
TARGETCOLUMN	Specifies the target variable column for supervised learning evaluation metrics. **DataType**: STRING
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Data types specification for feature columns in the dataset. **DataType**: ARRAY
MAXITER	[optional] Maximum iterations parameter for convergence control. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Target number of clusters (K) for centroid-based partitioning. **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Train-test split ratio for model validation and performance evaluation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Parallel processing level for clustering operations. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
SAVEMODEL	[optional] Persist the trained clustering model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Output clustering model name for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing and quality improvement operations. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeoutliers **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill cleaning m **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization for improved clustering perfo **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Normalize **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached dataset. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for data processing optimization. **DataType**: INTEGER, **Default**: 64000000

26.1.4 Output

Below is the expected output schema for the Scikit-Learn K-Means clustering function.

Rmse	Root Mean Square Error measuring standard deviation of prediction residuals. **DataType**: DOUBLE
R2	R-squared coefficient of determination measuring goodness-of-fit for regression evaluation. **DataType**: DOUBLE
Mse	Mean Squared Error measuring average squared prediction errors. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction errors. **DataType**: DOUBLE
Accuracy	Model accuracy score for clustering performance evaluation. **DataType**: DOUBLE

26.1.5 Examples

Below are examples demonstrating the Scikit-Learn K-Means clustering SQL function.

```
select * from table(Unifyml.Scikitlearn.Kmeans(
INPUTTABLE => ("eshoppinglimit"),
INPUTCOLUMNS => ARRAY["year", "month", "day", "purchaseorder",
"country", "sessionid", "pagemaincat", "color",
"location", "modelphoto", "price"],
TARGETCOLUMN =>  ("pricetwo"),
PARALLELISM => 2,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler")))

SQL Results:
+-------------------------------+--------------------+
|        CoefficientName        |       Value        |
+-------------------------------+--------------------+
| Mean Squared Error(MSE)       | 7.281684313725489  |
| Mean Absolute Error(MAE)      | 2.24843137254902   |
| Root Mean Squared Error(RMSE) | 2.6984596186946153 |
| Regression Score(R2)          | 0.1593268057930256 |
| Accuracy                      | 97.75156862745098  |
+-------------------------------+--------------------+
```

26.2 KmeansPredict

26.2.1 Introduction

Once Scikit-Learn K-Means clustering model is trained, the model enables cluster assignment prediction for new unseen data points. This inference capability supports real-time cluster assignment, pattern recognition, and data segmentation applications in single-node machine learning pipelines for customer analytics and anomaly detection.

For further references please see Kmeans.

26.2.2 Syntax

Below is the syntax for the Scikit-Learn KmeansPredict inference function.

```
SELECT * FROM table(Unifyml.Scikitlearn.KmeansPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

26.2.3 Input Arguments

Below are the input parameters for the Scikit-Learn KmeansPredict inference function.

INPUTMODELNAME	Specifies the pre-trained Scikit-Learn K-Means model name for cluster assig **DataType**: STRING
INPUTTABLE	[optional] Specifies the input dataset containing features for cluster predictio
PARTITIONCOLUMN	[optional] Feature column for data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Data types specification for feature columns in the prediction datas **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for prediction dataset quality improvem **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies: removerow, removeduplicates, removeou **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in prediction dataset. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for prediction dataset consistency with train **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling techniques: MinMaxScaler, StandardScaler, Norm **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Identifier for cached prediction dataset. **DataType**: STRING
SAVETODB	[optional] Persist cluster predictions to database. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Name of the output prediction dataset. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output dataset if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export cluster predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Name of the output CSV file. **DataType**: STRING
BYTESPERCHUNK	[optional] Chunk size for prediction processing optimization. **DataType**: INTEGER, **Default**: 64000000

26.2.4 Output

Below is the expected output schema for the Scikit-Learn KmeansPredict inference function.

INPUTCOLUMNNAMES	Feature column names processed by the clustering model.
PREDICTCOLUMN	Predicted cluster assignment labels for input data points.

26.2.5 Examples

Below are some examples demonstrating the KmeansPredict inference function for making predictions on new data points using a pre-trained clustering model.

```
select * from table(Unifyml.Scikitlearn.KmeansPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelKM"),
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

27. Dask

27.1 Kmeans

27.1.1 Introduction

K-Means is an unsupervised clustering algorithm that partitions data into K distinct clusters by minimizing the within-cluster sum of squared distances (WCSS). The algorithm iteratively assigns data points to the nearest centroid and updates centroids based on cluster membership, making it effective for exploratory data analysis and pattern discovery in unlabeled datasets.

For further references please see Kmeans.

27.1.2 Syntax

Below is the syntax for the distributed K-means clustering function.

```
SELECT * FROM table(Unifyml.Dask.Kmeans(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITER => maxiter ] [,...]
  [ INPUTCLUSTERSIZE => inputClusterSize ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

27.1.3 Input Arguments

Below are the input parameters for the K-means clustering function.

INPUTCOLUMNS	Specifies the feature columns for clustering analysis. **DataType**: ARRAY
INPUTTABLE	[optional] Specifies the dataset table containing the feature vectors.
TARGETCOLUMN	Specifies the target variable column for supervised validation. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used for data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Data types specification for feature engineering. **DataType**: ARRAY
MAXITER	[optional] Maximum iterations for algorithm convergence. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
INPUTCLUSTERSIZE	[optional] Number of clusters (K) for partitioning the feature space. **DataType**: INTEGER, **Default**: 2
TRAININGSAMPLESIZE	[optional] Training set proportion for model validation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelization for distributed computing. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column used for distributed data partitioning. **DataType**: STRING
SAVEMODEL	[optional] Model persistence flag for reusability. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Model artifact identifier for storage. **DataType**: STRING
DATACLEANING	[optional] Enable data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data preprocessing techniques (removerow, removeduplicates, removeoutliers, fill) for data quality. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for data normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for data pipeline optimization. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing. **DataType**: INTEGER, **Default**: 64000000

27.1.4 Output

Below are the model evaluation metrics returned by the K-means clustering function.

Rmse	Root Mean Square Error measuring prediction accuracy and model performance. **DataType**: DOUBLE
R2	Coefficient of determination indicating variance explained by the clustering model. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average squared prediction errors. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute deviations. **DataType**: DOUBLE
Accuracy	Classification accuracy for cluster assignment validation. **DataType**: DOUBLE

27.2 KmeansPredict

27.2.1 Introduction

Once the Dask K-means clustering model is trained, it can be deployed for inference to predict cluster assignments for new, unseen data points using the learned cluster centroids.

For further references please see Kmeans.

27.2.2 Syntax

Below is the syntax for the KmeansPredict inference function.

```
SELECT * FROM table(Unifyml.Dask.KmeansPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

27.2.3 Input Arguments

Below are the input parameters for the KmeansPredict inference function.

INPUTMODELNAME	Specifies the trained model artifact for cluster prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing new feature vectors for inference.
PARTITIONCOLUMN	[optional] Column for distributed data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Schema specification for input feature types. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing pipeline for inference. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling for consistent normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized data retrieval. **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk. **DataType**: INTEGER, **Default**: 64000000

27.2.4 Output

Below is the expected output schema for the KmeansPredict inference function.

INPUTCOLUMNNAMES	Original feature columns from the trained model schema.
PREDICTCOLUMN	Cluster assignment predictions for input data points.

27.2.5 Examples

Below are practical examples demonstrating the KmeansPredict function for cluster inference tasks.

```
select * from table(Unifyml.Scikitlearn.KmeansPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelKM"),
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1))

SQL Results:
+------+-------+-----+---------------+------------+
| year | month | day | purchaseorder | prediction |
+------+-------+-----+---------------+------------+
| 2008 | 4     | 1   | 1             |    0       |
| 2008 | 4     | 1   | 2             |    0       |
| 2008 | 4     | 1   | 3             |    0       |
+------+-------+-----+---------------+------------+
```

XI Classification Algorithms

28. Unifyml

This section provides comprehensive details about the supported Spark-based supervised classification algorithms for predictive modeling tasks.

28.1 DecisionTreeClassifierModel

28.1.1 Introduction

UnifyML Decision Tree Classifier is a supervised learning algorithm for classification tasks that constructs a tree-like decision model. The algorithm recursively partitions the feature space using optimal splits based on information gain or Gini impurity, creating interpretable decision rules. Each internal node represents a feature-based decision boundary, branches represent decision outcomes, and leaf nodes contain class predictions, making it highly suitable for both binary and multiclass classification problems.

For further references please see DecisionTreeClassifier.

28.1.2 Syntax

Below is the syntax for the Unifyml DecisionTreeClassifier training function.

```
SELECT * FROM table(Unifyml.DecisionTreeClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.1.3 Input Arguments

Below are the input parameters for the Unifyml DecisionTreeClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for predictive modeling. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset table containing training data.
INPUTDATATYPES	[optional] Schema specification for feature data types. **DataType**: ARRAY
MAXBINS	[optional] Maximum bins for feature discretization and optimal split point selection. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Maximum tree depth for complexity control and overfitting prevention. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Minimum information gain threshold for node splitting decisions. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Random seed for reproducible model training. **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation(cv) or TrainValidation for hyperparameter tuning. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Hyperparameter grid for maximum bins in feature discretization tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Hyperparameter search space for tree depth optimization. **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Parameter grid for information gain threshold hyperparameter tuning. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Random seed array for ensemble reproducibility. **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Degree of parallelization for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] K-fold cross-validation splits for model performance estimation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Data partitioning strategy for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable computation. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reductio **DataType**: BOOLEAN, **Default**: false

INPUTPCAMODELNAME	[optional] Pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enable model persistence for deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Model artifact name for storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for numerical stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized data pipeline performance. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

28.1.4 Output

Below are the model performance metrics returned by the Unifyml DecisionTreeClassifier function.

Accuracy	Classification accuracy measuring correct prediction ratio. **DataType**: DOUBLE
TestError	Test set error rate indicating model generalization performance. **DataType**: DOUBLE

28.1.5 Examples

Below are practical examples demonstrating the DecisionTreeClassifier function for supervised classification tasks.

```
select * from table(Unifyml.DecisionTreeClassifier(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelDTC")))

SQL Results:
+-----------------+----------------------+
| CoefficientName |        Value         |
+-----------------+----------------------+
| Accuracy        | 0.9886363636363636   |
| TestError       | 0.011363636363636354 |
| OutputModelName | modelDTC1            |
+-----------------+----------------------+
```

28.2 DecisionTreeClassifierPredict

28.2.1 Introduction

Decision Tree Classifier inference leverages the trained decision tree structure to assign class labels to new data points by traversing the learned decision paths, where each leaf node contains the final classification prediction based on the feature-based decision rules established during training.

For further references please see DecisionTreeClassifier.

28.2.2 Syntax

Below is the syntax for the Unifyml DecisionTreeClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.DecisionTreeClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.2.3 Input Arguments

Below are the input parameters for the Unifyml DecisionTreeClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained classifier model for prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing new instances for classification.
INPUTDATATYPES	[optional] Feature schema specification for inference pipeline. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in inference. **DataType**: ARRAY
DATASCALING	[optional] Apply consistent feature scaling for inference. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for inference pipeline optimization. **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for optimization. **DataType**: INTEGER, **Default**: 64000000

28.2.4 Output

Below is the expected output schema for the Unifyml DecisionTreeClassifierPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained model schema.
PREDICTCOLUMN	Classification predictions with assigned class labels.

28.2.5 Examples

Below are practical examples demonstrating the DecisionTreeClassifierPredict function for classification inference tasks.

```
select * from table(Unifyml.DecisionTreeClassifierPredict(
INPUTCACHEID => ("cacheid4"),
INPUTMODELNAME => ("modelDTC1")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

28.3 GradientBoostedClassifierModel

28.3.1 Introduction

UnifyML Gradient Boosted Classifier implements an ensemble learning approach that sequentially builds weak decision tree learners, where each subsequent tree corrects the prediction errors of the previous ensemble. This boosting technique iteratively minimizes classification loss through gradient descent optimization, resulting in a robust classifier with superior predictive performance compared to individual decision trees.

For further references please see GradientBoostedTreeClassifier.

28.3.2 Input Arguments

Below are the input parameters for the Unifyml GradientBoostedClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for ensemble classification modeling. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised boosting classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset table containing training instances.
INPUTDATATYPES	[optional] Schema specification for feature data types in ensemble training. **DataType**: ARRAY
MAXBINS	[optional] Maximum bins for feature discretization and optimal split selection in weak learners. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Maximum depth constraint for individual trees in the boosting ensemble. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERS	[optional] Maximum boosting iterations for ensemble convergence. **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAIN	[optional] Minimum information gain threshold for tree node splitting in weak learners. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Random seed for reproducible ensemble training. **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for ensemble evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for ensemble performance. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation(cv) or TrainValidation for hyperparameter search. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Hyperparameter grid for maximum bins in ensemble feature discretization tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Hyperparameter search space for tree depth in ensemble optimizat **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERNUMPARAMS	[optional] Iteration parameter grid for boosting convergence optimization. **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAINPARAMS	[optional] Information gain threshold array for ensemble hyperparameter tuning. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Random seed array for ensemble reproducibility across runs. **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Degree of parallelization for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] K-fold cross-validation splits for ensemble performance estimation. **DataType**: INTEGER, **Default**: 2

NUMPARTITION	[optional] Data partitioning strategy for distributed ensemble training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable ensemble computation. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enable ensemble model persistence for deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Ensemble model artifact name for storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for ensemble numerical stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized ensemble training pipeline. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for ensemble processing. **DataType**: INTEGER, **Default**: 64000000

28.3.3 Output

Below are the ensemble model performance metrics returned by the Unifyml GradientBoostedClassifier function.

Accuracy	Classification accuracy measuring ensemble prediction performance. **DataType**: DOUBLE
TestError	Test set error rate indicating ensemble generalization capability. **DataType**: DOUBLE

28.3.4 Examples

Below are practical examples demonstrating the GradientBoostedClassifier function for ensemble classification tasks.

```
select * from table(Unifyml.GradientBoostedClassifier(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
INPUTDATATYPES => ARRAY["variance:double","skewness:double",
"curtosis:double","entropy:double","class:int"],
PARTITIONCOLUMN => ("variance"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelGBC")))

SQL Results:
+-----------------+----------------------+
| CoefficientName |        Value         |
+-----------------+----------------------+
| Accuracy        | 0.9886363636363636   |
| TestError       | 0.011363636363636354 |
| OutputModelName | modelGBC             |
+-----------------+----------------------+
```

28.4 GradientBoostedClassifierPredict

28.4.1 Introduction

UnifyML Gradient Boosted Classifier inference leverages the trained ensemble of decision trees to generate predictions by aggregating weighted outputs from multiple weak learners, where each tree contributes to the final classification decision based on its learned gradient corrections.

For further references please see GradientBoostedTreeClassifier.

28.4.2 Syntax

Below is the syntax for the Unifyml GradientBoostedClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.GradientBoostedClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.4.3 Input Arguments

Below are the input parameters for the Unifyml GradientBoostedClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained ensemble model for boosted classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing new instances for ensemble predic
INPUTDATATYPES	[optional] Feature schema specification for ensemble inference pipeline. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in ensemble inference **DataType**: ARRAY
DATASCALING	[optional] Apply consistent feature scaling for ensemble inference. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for ensemble inference pipeline optimization. **DataType**: STRING
SAVETODB	[optional] Persist ensemble prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing ensemble predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export ensemble prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for ensemble prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for optimization. **DataType**: INTEGER, **Default**: 64000000

28.4.4 Examples

Below are practical examples demonstrating the GradientBoostedClassifierPredict function for ensemble classification inference.

```
select * from table(Unifyml.GradientBoostedClassifierPredict(
INPUTTABLE => ("banknotes"),
INPUTMODELNAME => ("modelGBTC"),
INPUTDATATYPES => ARRAY["variance:double","skewness:double",
"curtosis:double","entropy:double"]))

SQL Results:
+----------+----------+----------+-------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+-------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                           |
| 5.3063   | 5.2684   | -2.8904  | 0.0                           |
| 3.5594   | 1.3078   | 1.291    | 0.0                           |
| 3.966    | 3.9213   | 0.70574  | 0.0                           |
| 4.1665   | -0.4449  | 0.23448  | 0.0                           |
+----------+----------+----------+-------------------------------+
```

28.5 NaiveBayesClassifierModel

28.5.1 Introduction

Naïve Bayes Classifier is a probabilistic supervised learning algorithm based on Bayes' Theorem that applies the conditional independence assumption between features given the class label. This generative model estimates class-conditional probability distributions and prior class probabilities from training data, making it particularly effective for text classification, spam detection, and scenarios with categorical features.

For further references please see NaiveBayesClassifier.

28.5.2 Syntax

Below is the syntax for the Unifyml NaiveBayesClassifier training function.

```
SELECT * FROM table(Unifyml.NaiveBayesClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ]  [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.5.3 Input Arguments

Below are the input parameters for the Unifyml NaiveBayesClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for probabilistic classification modeling. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised Bayesian classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset table containing training instances.
INPUTDATATYPES	[optional] Schema specification for feature data types in probabilistic modeling. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for Bayesian model evaluation. **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Data partitioning strategy for distributed probabilistic training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable Bayesian computation. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enable probabilistic model persistence for deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Bayesian model artifact name for storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for probabilistic model stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized probabilistic training pipeline. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for Bayesian processing. **DataType**: INTEGER, **Default**: 64000000

28.5.4 Output

Below are the probabilistic model performance metrics returned by the Unifyml NaiveBayesClassifier function.

Accuracy	Classification accuracy measuring Bayesian prediction performance. **DataType**: DOUBLE
TestError	Test set error rate indicating probabilistic model generalization. **DataType**: DOUBLE

28.5.5 Examples

Below are practical examples demonstrating the NaiveBayesClassifier function for probabilistic classification tasks.

```
select * from table(Unifyml.NaiveBayesClassifier(
INPUTCACHEID => ("cacheid"),
INPUTCOLUMNS => ARRAY["accountstatus", "duration", "credithistory", "purpose",
"creditamount", "bonds", "employementsince", "installmentrate", "personalstatus",
"guarantors", "age", "residencesince", "telexist", "property", "installmentplans",
"extcredit", "housing", "noofexistingcredit", "noofpeople", "telephone"],
TARGETCOLUMN =>  ("class"),
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelNB1")))

SQL Results:
+-----------------+----------------------+
| CoefficientName |        Value         |
+-----------------+----------------------+
| Accuracy        | 0.9886363636363636   |
| TestError       | 0.011363636363636354 |
| OutputModelName | modelNB1             |
+-----------------+----------------------+
```

28.6 NaiveBayesClassifierPredict

28.6.1 Introduction

UnifyML Naïve Bayes Classifier inference leverages learned class-conditional probability distributions and prior probabilities to compute posterior probabilities for new instances, applying Bayes' theorem to assign the most likely class based on feature evidence and the conditional independence assumption.

For further references please see NaiveBayesClassifier.

28.6.2 Syntax

Below is the syntax for the Unifyml NaiveBayesClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.NaiveBayesClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.6.3 Input Arguments

Below are the input parameters for the Unifyml NaiveBayesClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained probabilistic model for Bayesian classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing new instances for probabilistic inf
INPUTDATATYPES	[optional] Feature schema specification for Bayesian inference pipeline. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for probabilistic inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in Bayesian inference **DataType**: ARRAY
DATASCALING	[optional] Apply consistent feature scaling for probabilistic inference. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for probabilistic inference pipeline optimization. **DataType**: STRING
SAVETODB	[optional] Persist Bayesian prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing probabilistic predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export Bayesian prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for probabilistic prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for optimization. **DataType**: INTEGER, **Default**: 64000000

28.6.4 Output

Below is the expected output schema for the Unifyml NaiveBayesClassifierPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained probabilistic model schema.
PREDICTCOLUMN	Bayesian classification predictions with maximum likelihood class assignments.

28.7 RandomForestClassifierModel

28.7.1 Introduction

UnifyML Random Forest Classifier is an ensemble learning algorithm that constructs multiple decision trees through bootstrap aggregating (bagging) and combines their predictions via majority voting. This ensemble method leverages random feature subsampling at each node split to reduce overfitting and improve generalization, making it highly effective for complex classification tasks with high-dimensional feature spaces.

For further references please see RandomForestClassifier.

28.7.2 Syntax

Below is the syntax for the Unifyml RandomForestClassifier training function.

```
SELECT * FROM table(Unifyml.RandomForestClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.7.3 Input Arguments

Below are the input parameters for the Unifyml RandomForestClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for ensemble classification modeling. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised ensemble classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset table containing training instances.
INPUTDATATYPES	[optional] Schema specification for feature data types in ensemble training. **DataType**: ARRAY
MAXBINS	[optional] Maximum bins for feature discretization and split candidate selection in ensemble trees. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Maximum tree depth constraint for individual trees in the random forest ensemble. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Minimum information gain threshold for node splitting in ensemble trees. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Random seed for reproducible ensemble training and bootstrap sa **DataType**: Long, **Default**: 235498149, **Min**: 0, **Max**: 235498149
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for ensemble evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for ensemble performance. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation(cv) or TrainValidation for hyperparameter search. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Hyperparameter grid for maximum bins in ensemble feature discretization tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Hyperparameter search space for tree depth in ensemble optimizat **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Information gain threshold array for ensemble hyperparameter tuning. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Random seed array for ensemble reproducibility across bootstrap **DataType**: ARRAY, **Default**: 235498149, **Min**: 0, **Max**: 235498149
PARALLELISM	[optional] Degree of parallelization for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] K-fold cross-validation splits for ensemble performance estimation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Data partitioning strategy for distributed ensemble training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable ensemble computation. **DataType**: STRING

APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enable ensemble model persistence for deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Random forest model artifact name for storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for ensemble numerical stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized ensemble training pipeline. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for ensemble processing. **DataType**: INTEGER, **Default**: 64000000

28.7.4 Output

Below are the comprehensive classification metrics returned by the Unifyml RandomForestClassifier function.

Accuracy	Overall classification accuracy measuring ensemble prediction performance. **DataType**: DOUBLE
TestError	Test set error rate indicating ensemble generalization capability. **DataType**: DOUBLE
FalsePositiveRateByLabel	Class-specific false positive rate for multi-class evaluation. **DataType**: DOUBLE
FMeasureByLabel	Per-class F1-score measuring precision-recall harmonic mean. **DataType**: DOUBLE
PrecisionByLabel	Class-specific precision indicating positive prediction accuracy. **DataType**: DOUBLE
WeightedPrecision	Weighted average precision across all classes. **DataType**: DOUBLE
WeightedFMeasure	Weighted average F1-score for overall model performance. **DataType**: DOUBLE
WeightedFalsePositiveRate	Weighted average false positive rate across classes. **DataType**: DOUBLE
WeightedTruePositiveRate	Weighted average recall (sensitivity) across all classes. **DataType**: DOUBLE
TotalIterations	Number of ensemble training iterations completed. **DataType**: DOUBLE

28.8 RandomForestClassifierPredict

28.8.1 Introduction

UnifyML Random Forest Classifier inference leverages the trained ensemble of decision trees to generate class predictions through majority voting aggregation, where each tree in the forest contributes its individual prediction and the final classification is determined by the most frequently predicted class across all ensemble members.

For further references please see RandomForestClassifier.

28.8.2 Syntax

Below is the syntax for the Unifyml RandomForestClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.RandomForestClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.8.3 Input Arguments

Below are the input parameters for the Unifyml RandomForestClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained ensemble model for random forest classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing new instances for ensemble predic
INPUTDATATYPES	[optional] Feature schema specification for ensemble inference pipeline. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in ensemble inference **DataType**: ARRAY
DATASCALING	[optional] Apply consistent feature scaling for ensemble inference. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for ensemble inference pipeline optimization. **DataType**: STRING
SAVETODB	[optional] Persist ensemble prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing ensemble predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export ensemble prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for ensemble prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for optimization. **DataType**: INTEGER, **Default**: 64000000

28.8.4 Output

Below is the expected output schema for the Unifyml RandomForestClassifierPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained ensemble model schema.
PREDICTCOLUMN	Ensemble classification predictions via majority voting aggregation.

28.8.5 Examples

Below are practical examples demonstrating the RandomForestClassifierPredict function for ensemble classification inference.

```
select * from table(Unifyml.RandomForestClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelRFC2")
DATASCALING => true,
SCALINGMETHOD => ("Normalizer")))

SQL Results:
+----------+----------+----------+-------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+-------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                           |
| 5.3063   | 5.2684   | -2.8904  | 0.0                           |
| 3.5594   | 1.3078   | 1.291    | 0.0                           |
| 3.966    | 3.9213   | 0.70574  | 0.0                           |
| 4.1665   | -0.4449  | 0.23448  | 0.0                           |
+----------+----------+----------+-------------------------------+
```

28.9 Svm

28.9.1 Introduction

Support Vector Machine (SVM) is a discriminative supervised learning algorithm that constructs optimal hyperplanes to separate classes in high-dimensional feature space. SVM maximizes the margin between decision boundaries while handling non-linearly separable data through kernel transformations, making it highly effective for binary and multi-class classification tasks with robust generalization capabilities.

For further references please see Svm.

28.9.2 Syntax

Below is the syntax for the Unifyml SVM training function.

```
SELECT * FROM table(Unifyml.Svm(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.9.3 Input Arguments

Below are the input parameters for the Unifyml SVM training function.

INPUTCOLUMNS	Specifies the feature columns for margin-based classification modeling. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised SVM classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset table containing training instances.
INPUTDATATYPES	[optional] Schema specification for feature data types in SVM training. **DataType**: ARRAY
MAXITERNUM	[optional] Maximum iterations for SVM optimization convergence. **DataType**: INTEGER, **Default**: 25, **Min**: 2, **Max**: 25
REGPARAM	[optional] Regularization parameter for margin-loss trade-off control. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for SVM evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for SVM performance. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation(cv) or TrainValidation for hyperparameter search. **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[optional] Iteration parameter grid for SVM convergence optimization. **DataType**: INTEGER, **Default**: 25, **Min**: 2, **Max**: 25
REGPARAMPARAMS	[optional] Regularization parameter array for margin optimization tuning. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
PARALLELISM	[optional] Degree of parallelization for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] K-fold cross-validation splits for SVM performance estimation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Data partitioning strategy for distributed SVM training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable SVM computation. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply Principal Component Analysis for dimensionality reductio **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained PCA model identifier for feature transformation. **DataType**: STRING
SAVEMODEL	[optional] Enable SVM model persistence for deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] SVM model artifact name for storage and versioning. **DataType**: STRING

DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling for SVM numerical stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized SVM training pipeline. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for SVM processing. **DataType**: INTEGER, **Default**: 64000000

28.9.4 Output

Below are the comprehensive classification metrics returned by the Unifyml SVM function.

Accuracy	Overall classification accuracy measuring SVM prediction performance. **DataType**: ARRAY
Intercept	Decision boundary intercept parameter in hyperplane equation. **DataType**: DOUBLE
TestError	Test set error rate indicating SVM generalization capability. **DataType**: DOUBLE
AreaUnderROC	Area Under ROC Curve measuring binary classification performance. **DataType**: DOUBLE
FalsePositiveRateByLabel	Class-specific false positive rate for multi-class evaluation. **DataType**: DOUBLE
FMeasureByLabel	Per-class F1-score measuring precision-recall harmonic mean. **DataType**: DOUBLE
PrecisionByLabel	Class-specific precision indicating positive prediction accuracy. **DataType**: DOUBLE
WeightedPrecision	Weighted average precision across all classes. **DataType**: DOUBLE
WeightedFMeasure	Weighted average F1-score for overall model performance. **DataType**: DOUBLE
WeightedFalsePositiveRate	Weighted average false positive rate across classes. **DataType**: DOUBLE
WeightedTruePositiveRate	Weighted average recall (sensitivity) across all classes. **DataType**: DOUBLE
TotalIterations	Number of optimization iterations completed for convergence. **DataType**: DOUBLE

28.10 SvmPredict

28.10.1 Introduction

UnifyML SVM inference leverages the trained support vector machine model to classify new data points by evaluating their position relative to the learned decision hyperplane, utilizing the optimal margin boundary and support vectors to assign class labels based on the discriminative decision function.

For further references please see Svm.

28.10.2 Syntax

Below is the syntax for the Unifyml SvmPredict inference function.

```
SELECT * FROM table(Unifyml.SvmPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

28.10.3 Input Arguments

Below are the input parameters for the Unifyml SvmPredict inference function.

INPUTMODELNAME	Specifies the trained SVM model for margin-based classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing new instances for SVM prediction
INPUTDATATYPES	[optional] Feature schema specification for SVM inference pipeline. **DataType**: ARRAY
DATACLEANING	[optional] Enable data preprocessing for inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling in SVM inference. **DataType**: ARRAY
DATASCALING	[optional] Apply consistent feature scaling for SVM inference. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for SVM inference pipeline optimization. **DataType**: STRING
SAVETODB	[optional] Persist SVM prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing SVM predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export SVM prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for SVM prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per processing chunk for optimization. **DataType**: INTEGER, **Default**: 64000000

28.10.4 Output

Below is the expected output schema for the Unifyml SvmPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained SVM model schema.
PREDICTCOLUMN	SVM classification predictions based on hyperplane decision function.

28.10.5 Examples

Below are practical examples demonstrating the SvmPredict function for margin-based classification inference.

```
select * from table(Unifyml.SvmPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modesvm7")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

29. Spark

This section provides comprehensive details about the supported Spark-based distributed machine learning classification algorithms for scalable predictive modeling.

29.1 DecisionTreeClassifierModel

29.1.1 Introduction

Spark Decision Tree Classifier is a distributed supervised learning algorithm that constructs interpretable tree-based decision models for classification tasks. The algorithm recursively partitions the feature space using optimal splits based on information gain or Gini impurity, leveraging Spark's distributed computing framework to handle large-scale datasets efficiently while maintaining the interpretability of traditional decision trees.

For further references please see DecisionTreeClassifier.

29.1.2 Syntax

Below is the syntax for the Spark DecisionTreeClassifier training function.

```
SELECT * FROM table(Unifyml.Spark.DecisionTreeClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.1.3 Input Arguments

Below are the input parameters for the Spark DecisionTreeClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for distributed classification modeling. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised tree-based classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the distributed dataset table containing training instances
INPUTDATATYPES	[optional] Schema specification for feature data types in distributed training. **DataType**: ARRAY
MAXBINS	[optional] Maximum bins for distributed feature discretization and optimal split point selection. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Maximum tree depth for complexity control and distributed overfitting prevention. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Minimum information gain threshold for distributed node splitting decisions. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Random seed for reproducible distributed model training. **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for distributed model evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for distributed tree performance. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Distributed validation strategy: CrossValidation(cv) or TrainValidation for hyperparameter search. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Hyperparameter grid for maximum bins in distributed feature discretization tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Hyperparameter search space for distributed tree depth optimizati **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Information gain threshold array for distributed hyperparameter tuning. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Random seed array for distributed ensemble reproducibility. **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Degree of Spark parallelization for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] K-fold cross-validation splits for distributed performance estimation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Spark RDD partitioning strategy for distributed computation. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable Spark computation. **DataType**: STRING
APPLYPCAMODEL	[optional] Apply distributed Principal Component Analysis for dimensionali **DataType**: BOOLEAN, **Default**: false

INPUTPCAMODELNAME	[optional] Pre-trained distributed PCA model identifier for feature transformatio **DataType**: STRING
SAVEMODEL	[optional] Enable distributed model persistence for scalable deployment. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Spark model artifact name for distributed storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated distributed data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Distributed imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling for numerical stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for distributed MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for distributed MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized distributed training pipeline. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per Spark partition for distributed processing. **DataType**: INTEGER, **Default**: 64000000

29.1.4 Output

Below are the distributed model performance metrics returned by the Spark DecisionTreeClassifier function.

Accuracy	Distributed classification accuracy measuring tree prediction performance. **DataType**: DOUBLE
TestError	Distributed test set error rate indicating model generalization capability. **DataType**: DOUBLE

29.2 DecisionTreeClassifierPredict

29.2.1 Introduction

Spark Decision Tree Classifier inference leverages the distributed trained tree structure to assign class labels to new data points by traversing the learned decision paths across Spark partitions, where each leaf node contains the final classification prediction based on the feature-based decision rules established during distributed training.

For further references please see DecisionTreeClassifier.

29.2.2 Syntax

Below is the syntax for the Spark DecisionTreeClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.Spark.DecisionTreeClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.2.3 Input Arguments

Below are the input parameters for the Spark DecisionTreeClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained distributed classifier model for prediction. **DataType**: STRING
INPUTTABLE	[optional] Specifies the distributed dataset containing new instances for class
INPUTDATATYPES	[optional] Feature schema specification for distributed inference pipeline. **DataType**: ARRAY
DATACLEANING	[optional] Enable distributed data preprocessing for inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Distributed imputation values for missing data handling in inferen **DataType**: ARRAY
DATASCALING	[optional] Apply consistent distributed feature scaling for inference. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for distributed MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for distributed MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for distributed inference pipeline optimization. **DataType**: STRING
SAVETODB	[optional] Persist distributed prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing distributed predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for distributed prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per Spark partition for processing optimization **DataType**: INTEGER, **Default**: 64000000

29.2.4 Output

Below is the expected output schema for the Spark DecisionTreeClassifierPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained distributed model schema.
PREDICTCOLUMN	Distributed classification predictions with assigned class labels.

29.2.5 Examples

Below are practical examples demonstrating the Spark DecisionTreeClassifierPredict function for distributed classification inference tasks.

```
select * from table(Unifyml.Spark.DecisionTreeClassifierPredict(
INPUTCACHEID => ("cacheid4"),
INPUTMODELNAME => ("modelDTC1")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

29.3 GradientBoostedClassifierModel

29.3.1 Introduction

Spark Gradient Boosted Classifier implements a distributed ensemble learning approach that sequentially builds weak decision tree learners across Spark partitions, where each subsequent tree corrects the prediction errors of the previous ensemble through gradient descent optimization. This distributed boosting technique leverages Spark's computational framework to handle large-scale datasets while iteratively minimizing classification loss for superior predictive performance.

For further references please see GradientBoostedTreeClassifier.

29.3.2 Syntax

Below is the syntax for the Spark GradientBoostedClassifier training function.

```
SELECT * FROM table(Unifyml.Spark.GradientBoostedClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MAXITERS => maxiters ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.3.3 Input Arguments

Below are the input parameters for the Spark GradientBoostedClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for distributed ensemble classification modelir **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised distributed boosting classification **DataType**: STRING
INPUTTABLE	[optional] Specifies the distributed dataset table containing training instances
INPUTDATATYPES	[optional] Schema specification for feature data types in distributed ensembl **DataType**: ARRAY
MAXBINS	[optional] Maximum bins for distributed feature discretization and optimal split selection in weak learners. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Maximum depth constraint for individual trees in the distributed boosting ensemble. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERS	[optional] Maximum distributed boosting iterations for ensemble convergenc **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAIN	[optional] Minimum information gain threshold for tree node splitting in distributed weak learners. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Random seed for reproducible distributed ensemble training. **DataType**: Long, **Default**: 159147643, **Min**: 0, **Max**: 159147643
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for distributed ensemble evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization for distributed ensemble performance. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Distributed validation strategy: CrossValidation(cv) or TrainValidation for hyperparameter search. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Hyperparameter grid for maximum bins in distributed ensemble feature discretization tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Hyperparameter search space for tree depth in distributed ensembl **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MAXITERNUMPARAMS	[optional] Distributed iteration parameter grid for boosting convergence opti **DataType**: INTEGER, **Default**: 10, **Min**: 0, **Max**: 10
MININFOGAINPARAMS	[optional] Information gain threshold array for distributed ensemble hyperparameter tuning. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Random seed array for distributed ensemble reproducibility acros **DataType**: ARRAY, **Default**: 159147643, **Min**: 0, **Max**: 159147643
PARALLELISM	[optional] Degree of Spark parallelization for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] K-fold cross-validation splits for distributed ensemble performance estimation. **DataType**: INTEGER, **Default**: 2

NUMPARTITION	[optional] Spark RDD partitioning strategy for distributed ensemble training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable distributed ensemble computat **DataType**: STRING
APPLYPCAMODEL	[optional] Apply distributed Principal Component Analysis for dimensionality re **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained distributed PCA model identifier for feature transformatio **DataType**: STRING
SAVEMODEL	[optional] Enable distributed ensemble model persistence for scalable deployme **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Distributed ensemble model artifact name for storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated distributed data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Distributed imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling for ensemble numerical stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for distributed MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for distributed MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized distributed ensemble training pipeline. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per Spark partition for distributed ensemble proces **DataType**: INTEGER, **Default**: 64000000

29.3.4 Output

Below are the distributed ensemble model performance metrics returned by the Spark Gradient-BoostedClassifier function.

Accuracy	Distributed classification accuracy measuring ensemble prediction performance. **DataType**: DOUBLE
TestError	Distributed test set error rate indicating ensemble generalization capability. **DataType**: DOUBLE

29.3.5 Examples

Below are practical examples demonstrating the Spark GradientBoostedClassifier function for distributed ensemble classification tasks.

```
select * from table(Unifyml.Spark.GradientBoostedClassifier(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
INPUTDATATYPES => ARRAY["variance:double","skewness:double",
"curtosis:double","entropy:double","class:int"],
PARTITIONCOLUMN => ("variance"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelGBC")))

SQL Results:
+-----------------+----------------------+
| CoefficientName |        Value         |
+-----------------+----------------------+
| Accuracy        | 0.9886363636363636   |
| TestError       | 0.011363636363636354 |
| OutputModelName | modelGBC             |
+-----------------+----------------------+
```

29.4 GradientBoostedClassifierPredict

29.4.1 Introduction

Spark Gradient Boosted Classifier inference leverages the trained distributed ensemble of decision trees to generate predictions by aggregating weighted outputs from multiple weak learners across Spark partitions, where each tree contributes to the final classification decision based on its learned gradient corrections through the distributed boosting framework.

For further references please see GradientBoostedTreeClassifier.

Gradient-Boosted Trees (GBTs) distributed learning algorithm supports binary classification as well as both continuous and categorical features through Spark's scalable computing infrastructure.

29.4.2 Syntax

Below is the syntax for the Spark GradientBoostedClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.Spark.GradientBoostedClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.4.3 Input Arguments

Below are the input parameters for the Spark GradientBoostedClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained distributed ensemble model for boosted classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the distributed dataset containing new instances for ense
INPUTDATATYPES	[optional] Feature schema specification for distributed ensemble inference pi **DataType**: ARRAY
DATACLEANING	[optional] Enable distributed data preprocessing for inference consistency. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Distributed imputation values for missing data handling in ensemb **DataType**: ARRAY
DATASCALING	[optional] Apply consistent distributed feature scaling for ensemble inferenc **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for distributed MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for distributed MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for distributed ensemble inference pipeline optimi **DataType**: STRING
SAVETODB	[optional] Persist distributed ensemble prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing distributed ensemble predictions. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed ensemble prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for distributed ensemble prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per Spark partition for processing optimizatio **DataType**: INTEGER, **Default**: 64000000

29.4.4 Output

Below is the expected output schema for the Spark GradientBoostedClassifierPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained distributed ensemble model schema.
PREDICTCOLUMN	Distributed ensemble classification predictions with aggregated class labels.

29.5 NaiveBayesClassifierModel

29.5.1 Introduction

Spark Naïve Bayes Classifier is a distributed probabilistic supervised learning algorithm based on Bayes' Theorem that applies the conditional independence assumption between features given the class label. This scalable generative model leverages Spark's distributed computing framework to estimate class-conditional probability distributions and prior class probabilities from large-scale

training data, making it particularly effective for distributed text classification, spam detection, and scenarios with categorical features across massive datasets.

For further references please see NaiveBayesClassifier.

29.5.2 Syntax

Below is the syntax for the Spark NaiveBayesClassifier training function.

```
SELECT * FROM table(Unifyml.Spark.NaiveBayesClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ]  [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.5.3 Input Arguments

Below are the input parameters for the Spark NaiveBayesClassifier training function.

INPUTCOLUMNS	Specifies the feature columns for distributed probabilistic classification modeling **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable for supervised distributed Bayesian classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the distributed dataset table containing training instances.
INPUTDATATYPES	[optional] Schema specification for feature data types in distributed probabilistic **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for distributed Bayesian model evaluatio **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Spark RDD partitioning strategy for distributed probabilistic training. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based partitioning for scalable distributed Bayesian computat **DataType**: STRING
APPLYPCAMODEL	[optional] Apply distributed Principal Component Analysis for dimensionality re **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained distributed PCA model identifier for feature transformatio **DataType**: STRING
SAVEMODEL	[optional] Enable distributed probabilistic model persistence for scalable deploy **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Distributed Bayesian model artifact name for storage and versioning. **DataType**: STRING
DATACLEANING	[optional] Enable automated distributed data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Distributed imputation strategy values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable distributed feature scaling for probabilistic model stability. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature normalization technique (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for preprocessing. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for distributed MinMaxScaler normalization. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for distributed MinMaxScaler normalization. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for optimized distributed probabilistic training pipelin **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per Spark partition for distributed Bayesian proces **DataType**: INTEGER, **Default**: 64000000

29.5.4 Output

Below are the distributed probabilistic model performance metrics returned by the Spark NaiveBayesClassifier function.

Accuracy	Distributed classification accuracy measuring Bayesian prediction performance. **DataType**: DOUBLE
TestError	Distributed test set error rate indicating probabilistic model generalization. **DataType**: DOUBLE

29.5.5 Examples

Below are practical examples demonstrating the Spark NaiveBayesClassifier function for distributed probabilistic classification tasks.

```
select * from table(Unifyml.Spark.NaiveBayesClassifier(
INPUTCACHEID => ("cacheid"),
INPUTCOLUMNS => ARRAY["accountstatus", "duration", "credithistory", "purpose",
"creditamount", "bonds", "employementsince", "installmentrate", "personalstatus",
"guarantors", "age", "residencesince", "telexist", "property", "installmentplans",
"extcredit", "housing", "noofexistingcredit", "noofpeople", "telephone"],
TARGETCOLUMN =>  ("class"),
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelNB1")))

SQL Results:
+-----------------+----------------------+
| CoefficientName |        Value         |
+-----------------+----------------------+
| Accuracy        | 0.9886363636363636   |
| TestError       | 0.011363636363636354 |
| OutputModelName | modelNB1             |
+-----------------+----------------------+
```

29.6 NaiveBayesClassifierPredict

29.6.1 Introduction

Spark Naïve Bayes Classifier inference leverages the trained distributed probabilistic model to classify new data points by computing posterior probabilities across Spark partitions, utilizing learned class-conditional probability distributions and prior probabilities to assign the most likely class based on feature evidence and the conditional independence assumption in a scalable distributed manner.

For further references please see NaiveBayesClassifier.

29.6.2 Syntax

Below is the syntax for the Spark NaiveBayesClassifierPredict inference function.

```
SELECT * FROM table(Unifyml.Spark.NaiveBayesClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.6.3 Input Arguments

Below are the input parameters for the Spark NaiveBayesClassifierPredict inference function.

INPUTMODELNAME	Specifies the trained distributed probabilistic model for Bayesian classificatio **DataType**: STRING
INPUTTABLE	[optional] Specifies the distributed dataset containing new instances for probs
INPUTDATATYPES	[optional] Feature schema specification for distributed Bayesian inference pi **DataType**: ARRAY
DATACLEANING	[optional] Enable distributed data preprocessing for probabilistic inference co **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Distributed data quality techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Distributed imputation values for missing data handling in Bayesia **DataType**: ARRAY
DATASCALING	[optional] Apply consistent distributed feature scaling for probabilistic infere **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Distributed feature transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for scaling. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower boundary for distributed MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper boundary for distributed MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cache identifier for distributed probabilistic inference pipeline opt **DataType**: STRING
SAVETODB	[optional] Persist distributed Bayesian prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for storing distributed probabilistic prediction **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing prediction table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export distributed Bayesian prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV filename for distributed probabilistic prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per Spark partition for processing optimization **DataType**: INTEGER, **Default**: 64000000

29.6.4 Output

Below is the expected output schema for the Spark NaiveBayesClassifierPredict function.

INPUTCOLUMNNAMES	Original feature columns from the trained distributed probabilistic model schema.
PREDICTCOLUMN	Distributed Bayesian classification predictions with maximum likelihood class ass

29.6.5 Examples

Below are practical examples demonstrating the Spark NaiveBayesClassifierPredict function for distributed probabilistic classification inference.

```
select * from table(Unifyml.Spark.NaiveBayesClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelNB2")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

29.7 RandomForestClassifierModel

29.7.1 Introduction

The Random Forest Classifier is an ensemble learning algorithm that leverages multiple decision trees to enhance classification accuracy and robustness. This algorithm employs bootstrap aggregating (bagging) to create diverse tree learners, reducing overfitting and improving generalization performance on unseen data.

For further references please see RandomForestClassifier.

29.7.2 Syntax

Below is the syntax for the Spark RandomForest function.

```
SELECT * FROM table(Unifyml.Spark.RandomForestClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXBINS => maxBins ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ MININFOGAIN => minInfoGain ] [,...]
  [ SEED => seed ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXBINSPARAMS => ARRAY[val [,] ]] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ]] [,...]
  [ MININFOGAINPARAMS => ARRAY[val [,] ]] [,...]
  [ SEEDPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.7.3 Input Arguments

Below are the input arguments for the Spark RandomForest function.

INPUTCOLUMNS	Specifies the feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable column for classification. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data types specification for feature columns. **DataType**: ARRAY
MAXBINS	[optional] Maximum number of bins for discretizing continuous features and determining optimal split points. **DataType**: INTEGER, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTH	[optional] Maximum depth constraint for individual decision trees. **DataType**: INTEGER, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAIN	[optional] Minimum information gain threshold required for node splitting. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEED	[optional] Random seed for reproducible model training. **DataType**: Long, **Default**: 235498149, **Min**: 0, **Max**: 235498149
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation(cv) or TrainValidationSplit. **DataType**: STRING, **Default**: CrossValidation
MAXBINSPARAMS	[optional] Parameter grid for maximum bins hyperparameter tuning. **DataType**: ARRAY, **Default**: 32, **Min**: 2, **Max**: 32
MAXDEPTHPARAMS	[optional] Parameter grid for maximum depth hyperparameter tuning. **DataType**: ARRAY, **Default**: 5, **Min**: 0, **Max**: 5
MININFOGAINPARAMS	[optional] Parameter grid for minimum information gain hyperparameter tuning. **DataType**: ARRAY, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
SEEDPARAMS	[optional] Parameter grid for random seed hyperparameter tuning. **DataType**: ARRAY, **Default**: 235498149, **Min**: 0, **Max**: 235498149
PARALLELISM	[optional] Degree of parallelism for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for model evaluation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING

APPLYPCAMODEL	[optional] Enable Principal Component Analysis dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained PCA transformation model identifier. **DataType**: STRING
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

29.7.4 Output

Below is the expected output for the Spark RandomForest function.

Accuracy	Overall classification accuracy on test set. **DataType**: DOUBLE
TestError	Classification error rate on holdout test data. **DataType**: DOUBLE
FalsePositiveRateByLabel	Class-specific false positive rate metrics. **DataType**: DOUBLE
FMeasureByLabel	Harmonic mean of precision and recall per class. **DataType**: DOUBLE
PrecisionByLabel	Class-wise precision scores. **DataType**: DOUBLE
WeightedPrecision	Sample-weighted average precision across all classes. **DataType**: DOUBLE
WeightedFMeasure	Sample-weighted F1-score across all classes. **DataType**: DOUBLE
WeightedFalsePositiveRate	Sample-weighted false positive rate. **DataType**: DOUBLE
WeightedTruePositiveRate	Sample-weighted true positive rate (sensitivity). **DataType**: DOUBLE
TotalIterations	Number of training iterations completed. **DataType**: DOUBLE

29.7.5 Examples

Below are the some of the examples for running RandomForestClassifier SQL function.

```
select * from table(Unifyml.Spark.RandomForestClassifier(
INPUTTABLE => ("select * from banknotes limit 100"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow","removeduplicates","removeoutliers"]
))

SQL Results:
+--------------------------+-------+
|      CoefficientName     | Value |
+--------------------------+-------+
| Accuracy                 | 1.0   |
| TestError                | 0.0   |
| FalsePositiveRateByLabel | NaN   |
| FMeasureByLabel          | 1.0   |
| PrecisionByLabel         | 1.0   |
| WeightedPrecision        | 1.0   |
| WeightedFMeasure         | 1.0   |
| WeightedFalsePositiveRate| NaN   |
| WeightedTruePositiveRate | 1.0   |
| TotalIterations          | 0.0   |
+--------------------------+-------+
```

29.8 RandomForestClassifierPredict

29.8.1 Introduction

Once the Random Forest Classifier model is trained, it can be deployed for inference to generate class predictions on new, unseen data instances using the learned ensemble of decision trees.

For further references please see RandomForestClassifier.

29.8.2 Syntax

Below is the syntax for the Spark RandomForest Classifier predict function.

```
SELECT * FROM table(Unifyml.Spark.RandomForestClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view  }) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.8.3 Input Arguments

Below are the input arguments for the Spark RandomForest Classifier predict function.

INPUTMODELNAME	Pre-trained model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for prediction.
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

29.8.4 Output

Below is the expected output for the Spark RandomForest Classifier predict function.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

29.8.5 Examples

Below are the some of the examples for running RandomForestClassifierPredict SQL function.

```
select * from table(Unifyml.Spark.RandomForestClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelRFC2")
DATASCALING => true,
SCALINGMETHOD => ("Normalizer")))

SQL Results:
+----------+----------+----------+-------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+-------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                           |
| 5.3063   | 5.2684   | -2.8904  | 0.0                           |
| 3.5594   | 1.3078   | 1.291    | 0.0                           |
| 3.966    | 3.9213   | 0.70574  | 0.0                           |
| 4.1665   | -0.4449  | 0.23448  | 0.0                           |
+----------+----------+----------+-------------------------------+
```

29.9 Svm

29.9.1 Introduction

Support Vector Machine (SVM) is a powerful supervised learning algorithm that constructs optimal hyperplanes to separate classes in high-dimensional feature space. It excels in binary and multi-class classification tasks by maximizing the margin between decision boundaries and support vectors.

For further references please see Svm.

29.9.2 Syntax

Below is the syntax for the Spark SVM function.

```
SELECT * FROM table(Unifyml.Spark.Svm(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXITERNUM => maxiternum ] [,...]
  [ REGPARAM => regparam ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => ( tuningType ) ] [,...]
  [ MAXITERNUMPARAMS => ARRAY[val [,] ]] [,...]
  [ REGPARAMPARAMS => ARRAY[val [,] ]] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMFOLDS => numFolds ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ APPLYPCAMODEL => ARRAYpplyPcaModel ] [,...]
  [ INPUTPCAMODELNAME => ( inputPcaModelName ) ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.9.3 Input Arguments

Below are the input arguments for the Spark SVM function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for binary/multi-class classification. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
MAXITERNUM	[optional] Maximum optimization iterations for convergence. **DataType**: INTEGER, **Default**: 25, **Min**: 2, **Max**: 25
REGPARAM	[optional] Regularization parameter controlling model complexity. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Model validation strategy: CrossValidation(cv) or TrainValidationSplit. **DataType**: STRING, **Default**: CrossValidation
MAXITERNUMPARAMS	[optional] Parameter grid for maximum iterations tuning. **DataType**: INTEGER, **Default**: 25, **Min**: 2, **Max**: 25
REGPARAMPARAMS	[optional] Parameter grid for regularization hyperparameter tuning. **DataType**: DOUBLE, **Default**: 0.0, **Min**: 0.0, **Max**: 0.1
PARALLELISM	[optional] Degree of parallelism for hyperparameter evaluation. **DataType**: INTEGER, **Default**: 2
NUMFOLDS	[optional] Number of cross-validation folds for model evaluation. **DataType**: INTEGER, **Default**: 2
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
APPLYPCAMODEL	[optional] Enable Principal Component Analysis dimensionality reduction. **DataType**: BOOLEAN, **Default**: false
INPUTPCAMODELNAME	[optional] Pre-trained PCA transformation model identifier. **DataType**: STRING
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING

DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

29.9.4 Output

Below is the expected output for the Spark SVM function.

Accuracy	Overall classification accuracy on test set. **DataType**: ARRAY
Intercept	Hyperplane intercept term (bias parameter). **DataType**: DOUBLE
TestError	Misclassification rate on holdout test data. **DataType**: DOUBLE
AreaUnderROC	Area Under Receiver Operating Characteristic curve. **DataType**: DOUBLE
FalsePositiveRateByLabel	Class-specific false positive rate metrics. **DataType**: DOUBLE
FMeasureByLabel	Harmonic mean of precision and recall per class. **DataType**: DOUBLE
PrecisionByLabel	Class-wise precision scores. **DataType**: DOUBLE
WeightedPrecision	Sample-weighted average precision across all classes. **DataType**: DOUBLE
WeightedFMeasure	Sample-weighted F1-score across all classes. **DataType**: DOUBLE
WeightedFalsePositiveRate	Sample-weighted false positive rate. **DataType**: DOUBLE
WeightedTruePositiveRate	Sample-weighted true positive rate (sensitivity). **DataType**: DOUBLE
TotalIterations	Number of optimization iterations completed. **DataType**: DOUBLE

29.9.5 Examples

Below are the some of the examples for running SVM SQL function.

```
select * from table(Unifyml.Spark.Svm(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis", "entropy"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelSVM")))

SQL Results:
```

29.10 SvmPredict

29.10.1 Introduction

Once the SVM model is trained, it can be deployed for inference to generate class predictions on new data instances using the learned hyperplane decision boundaries.

For further references please see Svm.

29.10.2 Syntax

Below is the syntax for the Spark SvmPredict function.

```
SELECT * FROM table(Unifyml.Spark.SvmPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

29.10.3 Input Arguments

Below are the input arguments for the Spark SvmPredict function.

INPUTMODELNAME	Pre-trained SVM model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for classification prediction.
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

29.10.4 Output

Below is the expected output for the Spark SvmPredict function.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

29.10.5 Examples

Below are the some of the examples for running SvmPredict SQL function.

```
select * from table(Unifyml.Spark.SvmPredict(
INPUTCACHEID => ("cacheid1"),
INPUTMODELNAME => ("modesvm7")))

SQL Results:
+--------------+-----------------+------------+-------------------+
| fixedacidity | volatileacidity | citricacid |  quality_Predict  |
+--------------+-----------------+------------+-------------------+
| 7.0          | 0.27            | 0.36       | 5.888687696874322 |
| 6.3          | 0.3             | 0.34       | 5.922737242385676 |
| 8.1          | 0.28            | 0.4        | 5.740148757253767 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
| 7.2          | 0.23            | 0.32       | 5.935959167819812 |
+--------------+-----------------+------------+-------------------+
```

30. ScikitLearn

This section provides details about the supported scikit-learn classification algorithms for machine learning workflows.

30.1 AdaBoostClassifier

30.1.1 Introduction

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that sequentially combines multiple weak learners to create a robust classifier. It adaptively adjusts sample weights to focus on previously misclassified instances, thereby improving overall predictive performance through iterative learning.

For further references please see AdaBoostClassifier.

30.1.2 Syntax

Below is the syntax for the Scikitlearn AdaBoost Classifier function.

```
SELECT * FROM table(Unifyml.Scikitlearn.AdaBoostClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.1.3 Input Arguments

Below are the input arguments for the Scikitlearn AdaBoost Classifier function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for classification tasks. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelism for model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.1.4 Output

Below is the expected output for the Scikitlearn AdaBoost Classifier function.

Rmse	Root Mean Square Error measuring prediction deviation. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying prediction accuracy. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average prediction error. **DataType**: DOUBLE
Accuracy	Overall classification accuracy on test set. **DataType**: DOUBLE

30.1.5 Examples

Below are the some of the examples for running AdaBoost Classification SQL function.

```
select * from table(Unifyml.Scikitlearn.AdaboostClassifier(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelAbC")))

SQL Results:
+-------------------------------+--------------------+
|        CoefficientName        |       Value        |
+-------------------------------+--------------------+
| Mean Squared Error(MSE)       | 7.281684313725489  |
| Mean Absolute Error(MAE)      | 2.24843137254902   |
| Root Mean Squared Error(RMSE) | 2.6984596186946153 |
| Regression Score(R2)          | 0.1593268057930256 |
| Accuracy                      | 97.75156862745098  |
+-------------------------------+--------------------+
```

30.2 AdaBoostClassifierPredict

30.2.1 Introduction

Once the AdaBoost Classifier model is trained, it can be deployed for inference to generate class predictions on new data instances using the learned ensemble of weak learners.

For further references please see AdaBoostClassifier.

30.2.2 Syntax

Below is the syntax for the Scikitlearn AdaBoostClassifierPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.AdaBoostClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.2.3 Input Arguments

Below are the input arguments for the Scikitlearn AdaBoostClassifierPredict function.

INPUTMODELNAME	Pre-trained model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for prediction.
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.2.4 Output

Below is the expected output for the Scikitlearn AdaBoostClassifierPredict function.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

30.2.5 Examples

Below are the some of the examples for running AdaBoostClassifierPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.AdaBoostClassifierPredict(
INPUTTABLE => ("indiandiabetes"),
PARTITIONCOLUMN => ("age"),
INPUTMODELNAME => ("modelABC1")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

30.3 DecisionTreeClassifier

30.3.1 Introduction

The Decision Tree Classifier is a supervised learning algorithm that creates a tree-like model for classification by recursively partitioning the feature space. It makes predictions by traversing decision nodes that test feature values, ultimately reaching leaf nodes that represent class predictions. This interpretable algorithm excels at capturing non-linear relationships and feature interactions.

For further references please see DecisionTreeClassifier.

30.3.2 Syntax

Below is the syntax for the Scikitlearn DecisionTree Classifier function.

```
SELECT * FROM table(Unifyml.Scikitlearn.DecisionTreeClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.3.3 Input Arguments

Below are the input arguments for the Scikitlearn DecisionTree Classifier function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for classification tasks. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelism for model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.3.4 Output

Below is the expected output for the Scikitlearn DecisionTree Classifier function.

Rmse	Root Mean Square Error measuring prediction deviation. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying prediction accuracy. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average prediction error. **DataType**: DOUBLE
Accuracy	Overall classification accuracy on test set. **DataType**: DOUBLE

30.3.5 Examples

Below are the some of the examples for running DecisionTree Classification SQL function.

```
select * from table(Unifyml.Scikitlearn.DecisionTreeClassifier(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelDTC")))

SQL Results:
+-------------------------------+--------------------+
|        CoefficientName        |       Value        |
+-------------------------------+--------------------+
| Mean Squared Error(MSE)       | 7.281684313725489  |
| Mean Absolute Error(MAE)      | 2.24843137254902   |
| Root Mean Squared Error(RMSE) | 2.6984596186946153 |
| Regression Score(R2)          | 0.1593268057930256 |
| Accuracy                      | 97.75156862745098  |
+-------------------------------+--------------------+
```

30.4 DecisionTreeClassifierPredict

30.4.1 Introduction

The Decision Tree Classifier generates predictions by navigating through the learned tree structure, where each decision path leads to a leaf node containing the predicted class for the given feature combination.

For further references please see DecisionTreeClassifier.

30.4.2 Syntax

Below is the syntax for the Scikitlearn DecisionTreeClassifierPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.DecisionTreeClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.4.3 Input Arguments

Below are the input arguments for the Scikitlearn DecisionTreeClassifierPredict. function.

INPUTMODELNAME	Pre-trained model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for prediction.
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.4.4 Output

Below is the expected output for the Scikitlearn DecisionTreeClassifierPredict.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

30.4.5 Examples

Below are the some of the examples for running DecisionTreeClassifierPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.DecisionTreeClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelDTC2"),
DATASCALING => true,
SCALINGMETHOD => ("Normalizer")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

30.5 KnnClassifier

30.5.1 Introduction

The K-Nearest Neighbors (KNN) classifier is a non-parametric, instance-based learning algorithm that performs classification by identifying the K nearest training examples in the feature space and assigning the majority class among these neighbors to new data points.

For further references please see KnnClassifier.

30.5.2 Syntax

Below is the syntax for the Scikitlearn Knn Classifier function.

```
SELECT * FROM table(Unifyml.Scikitlearn.KnnClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.5.3 Input Arguments

Below are the input arguments for the Scikitlearn Knn Classifier function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for classification tasks. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelism for model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.5.4 Output

Below is the expected output for the Scikitlearn Knn Classifier function.

Rmse	Root Mean Square Error measuring prediction deviation. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying prediction accuracy. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average prediction error. **DataType**: DOUBLE

30.5.5 Examples

Below are the some of the examples for running Knn Classification SQL function.

```
select * from table(Unifyml.Scikitlearn.KnnClassifier(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
DATASCALING => true,
SCALINGMETHOD => ("binarizer")))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

30.6 KnnClassifierPredict

30.6.1 Introduction

Once the K-Nearest Neighbors classifier is trained, it can be deployed for inference to classify new data points based on the majority vote of their K nearest neighbors in the feature space.

For further references please see KnnClassifier.

30.6.2 Syntax

Below is the syntax for the Scikitlearn KnnClassifierPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.KnnClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.6.3 Input Arguments

Below are the input arguments for the Scikitlearn KnnClassifierPredict function.

INPUTMODELNAME	Pre-trained model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for prediction.
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.6.4 Output

Below is the expected output for the Scikitlearn KnnClassifierPredict function.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

30.6.5 Examples

Below are the some of the examples for running KnnClassifierPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.KnnClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelKNN2")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

30.7 RandomForestClassifer

30.7.1 Introduction

The Random Forest Classifier is an ensemble learning algorithm that leverages multiple decision trees with bootstrap sampling and random feature selection to create a robust classification model with improved generalization and reduced overfitting compared to individual decision trees.

For further references please see RandomForestClassifier.

30.7.2 Syntax

Below is the syntax for the Scikitlearn RandomForest Classifier function.

```
SELECT * FROM table(Unifyml.Scikitlearn.RandomForestClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ MAXDEPTH => maxDepth ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ HYPERPARAMETERTUNING => hyperParameterTuning  ] [,...]
  [ TUNINGTYPE => tuningType ] [,...]
  [ MAXDEPTHPARAMS => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.7.3 Input Arguments

Below are the input arguments for the Scikitlearn RandomForest Classifier function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for classification tasks. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
MAXDEPTH	[optional] Maximum depth constraint for individual decision trees. **DataType**: INTEGER, **Default**: 2, **Min**: 0, **Max**: 2
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
HYPERPARAMETERTUNING	[optional] Enable automated hyperparameter optimization. **DataType**: Boolean, **Default**: false
TUNINGTYPE	[optional] Hyperparameter search strategy: GridSearchCV or RandomSearcl **DataType**: STRING, **Default**: Gridsearchcv
MAXDEPTHPARAMS	[optional] Parameter grid for maximum depth hyperparameter tuning. **DataType**: ARRAY, **Default**: 2, **Min**: 0, **Max**: 2
PARALLELISM	[optional] Degree of parallelism for model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.7.4 Output

Below is the expected output for the Scikitlearn RandomForest Classifier function.

Rmse	Root Mean Square Error measuring prediction deviation. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying prediction accuracy. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average prediction error. **DataType**: DOUBLE
Accuracy	Overall classification accuracy on test set. **DataType**: DOUBLE

30.7.5 Examples

Below are the some of the examples for running RandomForest Classifier SQL function.

```
select * from table(Unifyml.Scikitlearn.RandomForestClassifier(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
PARTITIONCOLUMN => ("entropy"),
INPUTDATATYPES => ARRAY["variance:double",
"skewness:double","curtosis:double","class:int"],
PARALLELISM => 2,
NUMPARTITION => 4,
DATACLEANING => true,
CLEANINGMETHOD => ARRAY["removerow"],
DATASCALING => true,
SCALINGMETHOD => ("MinMaxScaler"),
MINSCALER => 0,
MAXSCALER => 1))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

30.8 RandomForestClassifierPredict

30.8.1 Introduction

Once the Random Forest Classifier model is trained, it can be deployed for inference to generate class predictions on new data instances using the ensemble voting mechanism of multiple decision trees.

For further references please see RandomForestClassifier.

30.8.2 Syntax

Below is the syntax for the Scikitlearn RandomForestClassifierPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.RandomForestClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.8.3 Input Arguments

Below are the input arguments for the Scikitlearn RandomForestClassifierPredict function.

INPUTMODELNAME	Pre-trained model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for prediction.
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.8.4 Output

Below is the expected output for the Scikitlearn RandomForestClassifierPredict function.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

30.8.5 Examples

Below are the some of the examples for running RandomForestClassifierPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.RandomForestClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelRFC2")
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler")))

SQL Results:
+----------+----------+----------+-------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+-------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                           |
| 5.3063   | 5.2684   | -2.8904  | 0.0                           |
| 3.5594   | 1.3078   | 1.291    | 0.0                           |
| 3.966    | 3.9213   | 0.70574  | 0.0                           |
| 4.1665   | -0.4449  | 0.23448  | 0.0                           |
+----------+----------+----------+-------------------------------+
```

30.9 SgdClassifier

30.9.1 Introduction

The Stochastic Gradient Descent (SGD) Classifier is a linear classification algorithm that uses iterative optimization to learn model parameters efficiently. It is particularly well-suited for large-scale datasets due to its incremental learning approach and computational efficiency.

For further references please see SgdClassifier.

30.9.2 Syntax

Below is the syntax for the Scikitlearn Sgd Classifier function.

```
SELECT * FROM table(Unifyml.Scikitlearn.SgdClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.9.3 Input Arguments

Below are the input arguments for the Scikitlearn Sgd Classifier function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for classification tasks. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelism for model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.9.4 Output

Below is the expected output for the Scikitlearn Sgd Classifier function.

Rmse	Root Mean Square Error measuring prediction deviation. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying prediction accuracy. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average prediction error. **DataType**: DOUBLE
Accuracy	Overall classification accuracy on test set. **DataType**: DOUBLE

30.9.5 Examples

Below are the some of the examples for running Sgd Classification SQL function.

```
select * from table(Unifyml.Scikitlearn.SgdClassifier(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelSGDC")))

SQL Results:
+-------------------------------+--------------------+
|        CoefficientName        |       Value        |
+-------------------------------+--------------------+
| Mean Squared Error(MSE)       | 7.281684313725489  |
| Mean Absolute Error(MAE)      | 2.24843137254902   |
| Root Mean Squared Error(RMSE) | 2.6984596186946153 |
| Regression Score(R2)          | 0.1593268057930256 |
| Accuracy                      | 97.75156862745098  |
+-------------------------------+--------------------+
```

30.10 SgdClassifierPredict

30.10.1 Introduction

Once the Stochastic Gradient Descent Classifier is trained, it can be deployed for inference to generate class predictions on new data instances using the learned linear decision boundary.

For further references please see SgdClassifier.

30.10.2 Syntax

Below is the syntax for the Scikitlearn SgdClassifierPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.SgdClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.10.3 Input Arguments

Below are the input arguments for the Scikitlearn SgdClassifierPredict function.

INPUTMODELNAME	Pre-trained model identifier for inference deployment. **DataType**: STRING
INPUTTABLE	[optional] Dataset containing features for prediction.
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTDATATYPES	[optional] Feature column data type specifications. **DataType**: ARRAY
DATACLEANING	[optional] Apply data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Apply feature scaling transformation. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
SAVETODB	[optional] Persist predictions to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Database table name for prediction storage. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Overwrite existing output table if present. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export predictions to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] CSV file name for prediction export. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.10.4 Output

Below is the expected output for the Scikitlearn SgdClassifierPredict function.

INPUTCOLUMNNAMES	Feature columns used in trained model.
PREDICTCOLUMN	Generated class predictions.

30.10.5 Examples

Below are the some of the examples for running SgdClassifierPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.SgdClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelSGD2")
DATASCALING => true,
SCALINGMETHOD => ("StandardScaler")))

SQL Results:
+----------+----------+----------+-------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+-------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                           |
| 5.3063   | 5.2684   | -2.8904  | 0.0                           |
| 3.5594   | 1.3078   | 1.291    | 0.0                           |
| 3.966    | 3.9213   | 0.70574  | 0.0                           |
| 4.1665   | -0.4449  | 0.23448  | 0.0                           |
+----------+----------+----------+-------------------------------+
```

30.11 Naivebayes

30.11.1 Introduction

Naive Bayes classifiers are probabilistic machine learning algorithms based on Bayes' theorem with strong feature independence assumptions. They excel in text classification and scenarios with categorical features, providing fast training and prediction with robust performance despite their simplified assumptions.

For further references please see NaivebayesClassifier.

30.11.2 Syntax

Below is the syntax for the scikitlearn Naivebayes Classifier function.

```
SELECT * FROM table(Unifyml.Scikitlearn.Naivebayes(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.11.3 Input Arguments

Below are the input arguments for the scikitlearn Naivebayes Classifier function.

INPUTCOLUMNS	Feature columns containing the independent variables. **DataType**: ARRAY
TARGETCOLUMN	Target variable column for classification tasks. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column-based data partitioning strategy. **DataType**: STRING
INPUTTABLE	[optional] Dataset source containing the feature matrix.
INPUTDATATYPES	[optional] Data type specifications for feature columns. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for model evaluation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Degree of parallelism for model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist trained model for future inference. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Trained model identifier for persistence. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data cleaning strategies (removerow, removeduplicates, removeoutliers, fill). **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling. **DataType**: ARRAY
DATASCALING	[optional] Enable feature scaling and normalization. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling algorithm (MinMaxScaler, StandardScaler, Normalizer, Binarizer). **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Minimum value for MinMaxScaler transformation. **DataType**: INTEGER
MAXSCALER	[optional] Maximum value for MinMaxScaler transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Data cache identifier for optimized processing. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data processing chunk. **DataType**: INTEGER, **Default**: 64000000

30.11.4 Output

Below is the expected output for the scikitlearn Naivebayes Classifier function.

Rmse	Root Mean Square Error measuring prediction deviation. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying prediction accuracy. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average prediction error. **DataType**: DOUBLE
Accuracy	Overall classification accuracy on test set. **DataType**: DOUBLE

30.11.5 Examples

Below are the some of the examples for running Naivebayes Classification SQL function.

```
select * from table(Unifyml.Scikitlearn.Naivebayes(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance"," skewness", "curtosis"],
TARGETCOLUMN =>  ("class")))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

30.12 NaivebayesPredict

30.12.1 Introduction

Once the UnifyML Naïve Bayes classifier model has been trained, it can be deployed to generate predictions for new data instances using the learned probability distributions and feature independence assumptions.

For further references please see NaivebayesClassifier.

30.12.2 Syntax

Below is the syntax for the dask NaivebayesPredict function.

```
SELECT * FROM table(Unifyml.Scikitlearn.NaivebayesPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

30.12.3 Input Arguments

Below are the input arguments for the dask NaivebayesPredict function.

INPUTMODELNAME	Specifies the trained model identifier for prediction inference. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing the feature variables for prediction
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill metho **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for storing prediction results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if it already exists. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export of predictions. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

30.12.4 Output

Below is the expected output for the dask NaivebayesPredict function.

INPUTCOLUMNNAMES	Feature variables used in the trained model.
PREDICTCOLUMN	Model predictions for the target variable.

30.12.5 Examples

Below are the some of the examples for running NaivebayesPredict SQL function.

```
select * from table(Unifyml.Scikitlearn.NaivebayesPredict(
INPUTCACHEID => ("cacheid2"),
INPUTMODELNAME => ("modelINb")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

31. Dask

31.1 Naivebayes

31.1.1 Introduction

Naïve Bayes classifiers are a family of probabilistic machine learning algorithms based on Bayes' Theorem with strong feature independence assumptions. These supervised learning algorithms are particularly effective for classification tasks involving categorical and text data, making them popular choices for spam detection, sentiment analysis, and document classification.

For further references please see NaivebayesClassifier.

31.1.2 Syntax

Below is the syntax for the dask Naivebayes Classifier function.

```
SELECT * FROM table(Unifyml.Dask.Naivebayes(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

31.1.3 Input Arguments

Below are the input arguments for the dask Naivebayes Classifier function.

INPUTCOLUMNS	Specifies the feature variables (independent variables) for model training. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable (dependent variable) for classification. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing the training features and target variable
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-test split ratio for model validation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Number of parallel processes for distributed model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist the trained model for future predictions. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model artifact. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill method. **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

31.1.4 Output

Below is the expected output for the dask Naivebayes Classifier function.

Rmse	Root Mean Square Error measuring prediction accuracy against actual values. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power for variance. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average prediction error magnitude. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction deviations. **DataType**: DOUBLE
Accuracy	Classification accuracy representing correct prediction percentage. **DataType**: DOUBLE

31.1.5 Examples

Below are the some of the examples for running Naivebayes Classification SQL function.

```
select * from table(Unifyml.Dask.Naivebayes(
INPUTCACHEID => ("cacheid2"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis"],
TARGETCOLUMN =>  ("class")))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |       Value        |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 7.281684313725489  |
| Mean Absolute Error(MAE)     | 2.24843137254902   |
| Root Mean Squared Error(RMSE)| 2.6984596186946153 |
| Regression Score(R2)         | 0.1593268057930256 |
| Accuracy                     | 97.75156862745098  |
+------------------------------+--------------------+
```

31.2 NaivebayesPredict

31.2.1 Introduction

Once the UnifyML Naïve Bayes classifier model has been trained, it can be deployed to generate predictions for new data instances using the learned probability distributions and feature independence assumptions.

For further references please see NaivebayesClassifier.

31.2.2 Syntax

Below is the syntax for the dask NaivebayesPredict function.

```
SELECT * FROM table(Unifyml.Dask.NaivebayesPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

31.2.3 Input Arguments

Below are the input arguments for the dask NaivebayesPredict function.

INPUTMODELNAME	Specifies the trained model identifier for prediction inference. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing the feature variables for prediction
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill metho **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for storing prediction results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if it already exists. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export of predictions. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

31.2.4 Output

Below is the expected output for the dask NaivebayesPredict function.

INPUTCOLUMNNAMES	Feature variables used in the trained model.
PREDICTCOLUMN	Model predictions for the target variable.

31.2.5 Examples

Below are the some of the examples for running NaivebayesPredict SQL function.

```
select * from table(Unifyml.Dask.NaivebayesPredict(
INPUTCACHEID => ("cacheid2"),
INPUTMODELNAME => ("modelNb")))

SQL Results:
+----------+----------+----------+------------------------------+
| variance | skewness | curtosis | m_targetColumnNameFea_Predict |
+----------+----------+----------+------------------------------+
| 4.8272   | 3.0687   | 0.68604  | 0.0                          |
| 5.3063   | 5.2684   | -2.8904  | 0.0                          |
| 3.5594   | 1.3078   | 1.291    | 0.0                          |
| 3.966    | 3.9213   | 0.70574  | 0.0                          |
| 4.1665   | -0.4449  | 0.23448  | 0.0                          |
+----------+----------+----------+------------------------------+
```

31.3 XgbClassifier

31.3.1 Introduction

The XGBoost (Extreme Gradient Boosting) Classifier is a high-performance ensemble learning algorithm that uses gradient boosting decision trees. It excels in structured data prediction tasks and is widely adopted in machine learning competitions and production environments due to its superior accuracy, speed, and feature importance capabilities.

For further references please see XgbClassifier.

31.3.2 Syntax

Below is the syntax for the dask Xgb Classifier function.

```
SELECT * FROM table(Unifyml.Dask.XgbClassifier(
  INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ PARALLELISM => parallelism ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ SAVEMODEL => savemodel ] [,...]
  [ OUTPUTMODEL => ( outputModel ) ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

31.3.3 Input Arguments

Below are the input arguments for the dask Xgb Classifier function.

INPUTCOLUMNS	Specifies the feature variables (independent variables) for model training. **DataType**: ARRAY
TARGETCOLUMN	Specifies the target variable (dependent variable) for classification. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing the training features and target variable
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-test split ratio for model validation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Number of parallel processes for distributed model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist the trained model for future predictions. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model artifact. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill method. **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

31.3.4 Output

Below is the expected output for the dask Xgb Classifier function.

Rmse	Root Mean Square Error measuring prediction accuracy against actual values. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power for variance. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average prediction error magnitude. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction deviations. **DataType**: DOUBLE
Accuracy	Classification accuracy representing correct prediction percentage. **DataType**: DOUBLE

31.3.5 Examples

Below are the some of the examples for running Xgb Classification SQL function.

```
select * from table(Unifyml.Dask.XgbClassifier(
INPUTTABLE => ("banknotes"),
INPUTCOLUMNS => ARRAY["variance", "skewness", "curtosis"],
TARGETCOLUMN =>  ("class"),
SAVEMODEL => true,
OUTPUTMODEL => ("modelSGD")))

SQL Results:
+------------------------------+----------------------+
|        CoefficientName       |        Value         |
+------------------------------+----------------------+
| Mean Squared Error(MSE)      | 0.006146971558895783 |
| Mean Absolute Error(MAE)     | 0.01378212859615066  |
| Root Mean Squared Error(RMSE)| 0.07840262469392069  |
| Regression Score(R2)         | 0.9752032023792953   |
| Accuracy                     | 99.98621787140385    |
| OutputModelName              | modelSGD             |
+------------------------------+----------------------+
```

31.4 XgbClassifierPredict

31.4.1 Introduction

Once the Dask XGBoost (Extreme Gradient Boosting) Classifier model has been trained, it can be deployed to generate predictions for new data instances using the learned ensemble of decision trees and gradient boosting patterns.

For further references please see XgbClassifier.

31.4.2 Syntax

Below is the syntax for the dask XgbClassifierPredict function.

```
SELECT * FROM table(Unifyml.Dask.XgbClassifierPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [\hline
  DATACLEANING & \makecell[l]{[optional] Enable automated data preprocessing pipeline.\\
\hline
  CLEANINGMETHOD & \makecell[l]{[optional] Data quality improvement techniques \\ (remov
\hline
  FILLWITHVALUES & \makecell[l]{[optional] Imputation values for missing data handling w
\hline
  DATASCALING & \makecell[l]{[optional] Apply feature normalization to the dataset.\\{\b
\hline
  SCALINGMETHOD & \makecell[l]{[optional] Feature scaling transformation method \\ (MinM
\hline
  MINSCALER & \makecell[l]{[optional] Lower bound for MinMaxScaler feature transformatio
\hline
  MAXSCALER & \makecell[l]{[optional] Upper bound for MinMaxScaler feature transformatio
\hline
  INPUTCACHEID & \makecell[l]{[optional] Cached dataset identifier for efficient data re
\hline
  SAVETODB & \makecell[l]{[optional] Persist prediction results to database storage.\\{\
\hline
  OUTPUTTABLENAME & \makecell[l]{[optional] Target table name for storing prediction res
\hline
  OVERWRITEOUTPUTTABLE & \makecell[l]{[optional] Replace existing output table if it alr
\hline
  SAVETOCSV & \makecell[l]{[optional] Export prediction results to CSV format.\\{\bf{Dat
\hline
  OUTPUTCSVNAME & \makecell[l]{[optional] Filename for CSV export of predictions.\\{\bf{
\hline
  BYTESPERCHUNK & \makecell[l]{[optional] Memory allocation per data chunk for processin
\hline
\end{tabular}

\subsection{Output}
Below is the expected output for the dask XgbClassifierPredict function.

\begin{tabular}{|l|l|l|}
\hline
  INPUTCOLUMNNAMES & Feature variables used in the trained model.\\
\hline
  PREDICTCOLUMN & Model predictions for the target variable.\\
\hline
\end{tabular}

\subsection{Examples}
Below are the some of the examples for running XgbClassifierPredict SQL function.

\begin{tcolorbox} [sharp corners]
\begin{verbatim}
select * from table(Unifyml.Dask.XgbClassifierPredict(
INPUTCACHEID => ("cacheid"),
INPUTMODELNAME => ("modelXGB2")))
```

XII Time Series Forecasting Algorithms

32. Unifyml

32.1 ArimaTimeSeries

32.1.1 Introduction

ARIMA (AutoRegressive Integrated Moving Average) is a powerful statistical forecasting model for time series analysis that captures temporal dependencies, trends, and seasonality patterns. This univariate forecasting technique is particularly effective for predicting future values in datasets with historical patterns and is widely used in financial forecasting, demand planning, and econometric analysis.

For further references please see ArimaTimeSeries.

32.1.2 Syntax

Below is the syntax for the Unifyml ArimaTimeSeries function.

```
SELECT * FROM table(Unifyml.ArimaTimeSeries(
INPUTDATECOLUMN => ( inputDateColumn ) [,...]
TARGETCOLUMN => ( targetcolumn ) [,...]
[ INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
[ FORECASTCOUNT => (forecastCount ) ] [,...]
[ INPUTTABLE => ( { table | view | (query) })  [,...]
[ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
[ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
[ PARALLELISM => ( parallelism ) ] [,...]
[ SEASONAL => ( seasonal ) ] [,...]
[ MAXORDER => ( maxorder ) ] [,...]
[ M => numberOfPeriods] [,...]
[ INFORMATION_CRITERION => ( information_criterion ) ] [,...]
[ METHOD => ( method ) ] [,...]
[ STEPWISE => (stepwise ) ] [,...]
[ D => d] [,...]
[ START_P => p] [,...]
[ START_Q => q] [,...]
[ MAX_P => max_p] [,...]
[ MAX_D => max_d] [,...]
[ MAX_Q => max_q] [,...]
[ MAXITER => maxiter] [,...]
[ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
[ NUMPARTITION => numPartition ] [,...]
[ DATACLEANING => dataCleaning ] [,...]
[ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
[ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
[ DATASCALING => ARRAYScaling ] [,...]
[ SCALINGMETHOD => scalingMethod ] [,...]
[ MINSCALER => minScaler ] [,...]
[ MAXSCALER => maxScaler ] [,...]
[ INPUTTRAINCACHEID => (cacheid) ] [,...]
[ SUMMARY => (summary) ] [,...]
[ SAVETODB => (saveToDb) ] [,...]
[ OUTPUTTABLENAME => (outputtablename) ] [,...]
[ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
[ SAVETOCSV => (savetocsv) ] [,...]
[ OUTPUTCSVNAME => (outputcsvname) ] [,...]
[ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

32.1.3 Input Arguments

Below are the input arguments for the Unifyml ArimaTimeSeries function.

INPUTDATECOLUMN	Specifies the temporal dimension variable containing datetime observations. **DataType**: STRING
TARGETCOLUMN	Specifies the time series variable to be forecasted. **DataType**: STRING
INPUTCOLUMNS	[optional] Specifies additional exogenous variables for multivariate forecasting **DataType**: ARRAY
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTTTABLE	[optional] Specifies the time series dataset containing temporal observations.
FORECASTCOUNT	[optional] Number of future time periods to predict beyond the training data. **DataType**: INTEGER, **Default**: 10
INPUTDATATYPES	[optional] Schema specification for time series data types. **DataType**: ARRAY
PARALLELISM	[optional] Number of parallel processes for model optimization. **DataType**: INTEGER, **Default**: 1
SEASONAL	[optional] Enable seasonal decomposition for periodic patterns (yearly, monthl **DataType**: BOOLEAN, **Default**: false
MAXORDER	[optional] Maximum order for ARIMA model parameter optimization. **DataType**: INTEGER, **Default**: 5
M	[optional] Seasonal periodicity indicating the number of observations per seas **DataType**: INTEGER, **Default**: 1, **Min**: 1, **Max**: 12
INFORMATIONCRITERION	[optional] Model selection criterion for optimal parameter estimation. **DataType**: STRING, **Default**: aic
METHOD	[optional] Optimization algorithm for maximum likelihood estimation. **DataType**: STRING, **Default**: lbfgs
STEPWISE	[optional] Enable stepwise model selection for automated parameter tuning. **DataType**: BOOLEAN, **Default**: false
D	[optional] Degree of differencing to achieve stationarity in the time series. **DataType**: INTEGER, **Default**: 0, **Min**: 1, **Max**: 12
STARTP	[optional]Initial value for autoregressive order (p) representing the number of lag observations in the model. **DataType**: INTEGER, **Default**: 2, **Min**: 2, **Max**: 12
STARTQ	[optional]Initial value for moving average order (q) representing the size of the moving average window. **DataType**: INTEGER, **Default**: 2, **Min**: 2, **Max**: 12
MAXP	[optional] Maximum autoregressive order (p) for model search space. Must be greater than or equal to start_p. **DataType**: INTEGER, **Default**: 5, **Min**: 5
MAXD	[optional]Maximum differencing order (d) for achieving stationarity. Must be greater than or equal to the d parameter. **DataType**: INTEGER, **Default**: 2, **Min**: 2
MAXQ	[optional] Maximum moving average order (q) for model search space. Must be greater than `start_q` parameter. **DataType**: INTEGER, **Default**: 5, **Min**: 5
MAXITER	[optional]Maximum iterations for model optimization convergence. **DataType**: INTEGER, **Default**: 50, **Min**: 1
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for time series cross-validation. **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER

DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill metho **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the time series data. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTTRAINCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
SUMMARY	[optional] Display comprehensive forecasting performance metrics and valid **DataType**: BOOLEAN, **Default**: false
SAVETODB	[optional] Persist forecasting results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for storing forecasting results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if it already exists. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export forecasting results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export of forecasts. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

32.1.4 Output

Below is the expected output for the Unifyml ArimaTimeSeries function.

Rmse	Root Mean Square Error measuring forecasting accuracy against actual values. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power for temporal variance. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average forecasting error magnitude. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute forecasting deviations. **DataType**: DOUBLE
FuturePredict	Time series forecasts for future time periods beyond training data. **DataType**: TABLE

32.1.5 Examples

Below are the some of the examples for running ArimaTimeSeries SQL function.

```
select * from table(Unifyml.ArimaTimeSeries(
INPUTTABLE => ("timeseries"),
INPUTDATECOLUMNS => ("ds"),
TARGETCOLUMN =>  ("y"),
PARTITIONCOLUMN => ("id"),
INPUTDATATYPES => ARRAY["ds:datetime","y:float",
"add1:int"],
M => 3,
D => 2,
START_P => 3,
START_Q => 4,
MAX_P => 6,
MAX_D => 3,
MAX_Q => 6,
MAXITER => 20,
DATASCALING => true,
SUMMARY => true,
SCALINGMETHOD => ("Binarizer")))

SQL Results:
+-------------------------------+-----------+
|        CoefficientName        |   Value   |
+-------------------------------+-----------+
| Mean Squared Error(MSE)       | 1.0       |
| Mean Absolute Error(MAE)      | 1.0       |
| Root Mean Squared Error(RMSE) | 1.0       |
| Regression Score(R2)          | 0.0       |
+-------------------------------+-----------+
```

33. Scikitlearn

This section provides details about the scikit-learn Time Series forecasting algorithms.

33.1 ArimaTimeSeries

33.1.1 Introduction

ARIMA (AutoRegressive Integrated Moving Average) is a powerful statistical forecasting model for time series analysis that captures temporal dependencies, trends, and seasonality patterns. This univariate forecasting technique is particularly effective for predicting future values in datasets with historical patterns and is widely used in financial forecasting, demand planning, and econometric analysis.

For further references please see ArimaTimeSeries.

33.1.2 Syntax

Below is the syntax for the ArimaTimeSeries function.

```
SELECT * FROM table(Unifyml.Scikitlearn.ArimaTimeSeries(
INPUTDATECOLUMN => ( inputDateColumn ) [,...]
TARGETCOLUMN => ( targetcolumn ) [,...]
[ INPUTCOLUMNS => ARRAY[val [,] ] ] [,...]
[ FORECASTCOUNT => (forecastCount ) ] [,...]
[ INPUTTABLE => ( { table | view | (query) })  [,...]
[ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
[ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
[ PARALLELISM => ( parallelism ) ] [,...]
[ SEASONAL => ( seasonal ) ] [,...]
[ MAXORDER => ( maxorder ) ] [,...]
[ M => numberOfPeriods] [,...]
[ INFORMATION_CRITERION => ( information_criterion ) ] [,...]
[ METHOD => ( method ) ] [,...]
[ STEPWISE => (stepwise ) ] [,...]
[ D => d] [,...]
[ START_P => p] [,...]
[ START_Q => q] [,...]
[ MAX_P => max_p] [,...]
[ MAX_D => max_d] [,...]
[ MAX_Q => max_q] [,...]
[ MAXITER => maxiter] [,...]
[ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
[ NUMPARTITION => numPartition ] [,...]
[ DATACLEANING => dataCleaning ] [,...]
[ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
[ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
[ DATASCALING => ARRAYScaling ] [,...]
[ SCALINGMETHOD => scalingMethod ] [,...]
[ MINSCALER => minScaler ] [,...]
[ MAXSCALER => maxScaler ] [,...]
[ INPUTTRAINCACHEID => (cacheid) ] [,...]
[ SUMMARY => (summary) ] [,...]
[ SAVETODB => (saveToDb) ] [,...]
[ OUTPUTTABLENAME => (outputtablename) ] [,...]
[ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
[ SAVETOCSV => (savetocsv) ] [,...]
[ OUTPUTCSVNAME => (outputcsvname) ] [,...]
[ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

33.1.3 Input Arguments

Below are the input arguments for the Unifyml ArimaTimeSeries function.

INPUTDATECOLUMN	Specifies the temporal dimension variable containing datetime observations. **DataType**: STRING
TARGETCOLUMN	Specifies the time series variable to be forecasted. **DataType**: STRING
INPUTCOLUMNS	[optional] Specifies additional exogenous variables for multivariate forecasting **DataType**: ARRAY
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTTABLE	[optional] Specifies the time series dataset containing temporal observations.
FORECASTCOUNT	[optional] Number of future time periods to predict beyond the training data. **DataType**: INTEGER, **Default**: 10
INPUTDATATYPES	[optional] Schema specification for time series data types. **DataType**: ARRAY
PARALLELISM	[optional] Number of parallel processes for model optimization. **DataType**: INTEGER, **Default**: 1
SEASONAL	[optional] Enable seasonal decomposition for periodic patterns (yearly, monthl **DataType**: BOOLEAN, **Default**: false
MAXORDER	[optional] Maximum order for ARIMA model parameter optimization. **DataType**: INTEGER, **Default**: 5
M	[optional] Seasonal periodicity indicating the number of observations per seas **DataType**: INTEGER, **Default**: 1, **Min**: 1, **Max**: 12
INFORMATIONCRITERION	[optional] Model selection criterion for optimal parameter estimation. **DataType**: STRING, **Default**: aic
METHOD	[optional] Optimization algorithm for maximum likelihood estimation. **DataType**: STRING, **Default**: lbfgs
STEPWISE	[optional] Enable stepwise model selection for automated parameter tuning. **DataType**: BOOLEAN, **Default**: false
D	[optional] Degree of differencing to achieve stationarity in the time series. **DataType**: INTEGER, **Default**: 0, **Min**: 1, **Max**: 12
STARTP	[optional]Initial value for autoregressive order (p) representing the number of lag observations in the model. **DataType**: INTEGER, **Default**: 2, **Min**: 2, **Max**: 12
STARTQ	[optional]Initial value for moving average order (q) representing the size of the moving average window. **DataType**: INTEGER, **Default**: 2, **Min**: 2, **Max**: 12
MAXP	[optional] Maximum autoregressive order (p) for model search space. Must be greater than or equal to `start_p`. **DataType**: INTEGER, **Default**: 5, **Min**: 5
MAXD	[optional]Maximum differencing order (d) for achieving stationarity. Must be greater than or equal to the d parameter. **DataType**: INTEGER, **Default**: 2, **Min**: 2
MAXQ	[optional] Maximum moving average order (q) for model search space. Must be greater than `start_q` parameter. **DataType**: INTEGER, **Default**: 5, **Min**: 5
MAXITER	[optional]Maximum iterations for model optimization convergence. **DataType**: INTEGER, **Default**: 50, **Min**: 1
TRAININGSAMPLESIZE	[optional] Training-validation split ratio for time series cross-validation. **DataType**: Double, **Default**: 0.7
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER

DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill metho **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the time series data. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTTRAINCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
SUMMARY	[optional] Display comprehensive forecasting performance metrics and valid **DataType**: BOOLEAN, **Default**: false
SAVETODB	[optional] Persist forecasting results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for storing forecasting results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if it already exists. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export forecasting results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export of forecasts. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

33.1.4 Output

Below is the expected output for the Unifyml ArimaTimeSeries function.

Rmse	Root Mean Square Error measuring forecasting accuracy against actual values. **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power for temporal variance. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average forecasting error magnitude. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute forecasting deviations. **DataType**: DOUBLE
FuturePredict	Time series forecasts for future time periods beyond training data. **DataType**: TABLE

33.1.5 Examples

Below are the some of the examples for running ArimaTimeSeries SQL function.

```
select * from table(Unifyml.Scikitlearn.ArimaTimeSeries(
INPUTTABLE => ("timeseries"),INPUTDATECOLUMNS => ("ds"),
TARGETCOLUMN =>  ("y"),
PARTITIONCOLUMN => ("id"),
INPUTDATATYPES => ARRAY["ds:datetime","y:float",
"add1:int"],
M => 3,
D => 2,
START_P => 3,
START_Q => 4,
MAX_P => 6,
MAX_D => 3,
MAX_Q => 6,
MAXITER => 20,
DATASCALING => true,
SUMMARY => true,
SCALINGMETHOD => ("Binarizer")))

SQL Results:
+-------------------------------+-----------+
|        CoefficientName        |   Value   |
+-------------------------------+-----------+
| Mean Squared Error(MSE)       | 1.0       |
| Mean Absolute Error(MAE)      | 1.0       |
| Root Mean Squared Error(RMSE) | 1.0       |
| Regression Score(R2)          | 0.0       |
+-------------------------------+-----------+
```

XII Deep Learning Algorithms

This section provides details about the TensorFlow deep learning algorithms.

34. Tensorflow

34.1 Tensorflow SequentialModel

34.1.1 Introduction

TensorFlow is an open-source deep learning framework developed by Google that enables the construction and training of neural networks for complex machine learning tasks. The Sequential Model provides a linear stack of layers for building feedforward neural networks, making it ideal for deep learning applications including computer vision, natural language processing, predictive modeling, and pattern recognition.

For further references please see Tensorflow SequentialModel.

34.1.2 Syntax

Below is the syntax for the Tensorflow SequentialModel function.

```
SELECT * FROM table(Tensorflow.SequentialModel(
  INPUTCOLUMNS => ( inputColumns ) [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => ( parallelism ) ] [,...]
  [ EPOCHS => ( epochs ) ] [,...]
  [ LAYERS => ARRAY[val [,] ] ] [,...]
  [ BINARYVALUE => ( binary ) ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVEMODEL => (savemodel) ] [,...]
  [ OUTPUTMODEL => ( outputModelName ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

34.1.3 Input Arguments

Below are the input arguments for the Tensorflow SequentialModel function.

INPUTCOLUMNS	Specifies the feature variables (independent variables) for neural network training **DataType**: STRING
TARGETCOLUMN	Specifies the target variable (dependent variable) for supervised learning. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
EPOCHS	[optional] Number of complete passes through the training dataset during model t **DataType**: INTEGER,**Default**: 10
LAYERS	[optional] Architecture specification for dense neural network layers. **DataType**: ARRAY
BINARYVALUE	[optional] Configure loss function for binary classification tasks. **DataType**: BOOLEAN,**Default**: false
INPUTTABLE	[optional] Specifies the dataset containing the training features and target variable
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-test split ratio for model validation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Number of parallel processes for distributed model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist the trained neural network model for future predictions. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model artifact. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill method. **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

34.1.4 Output

Below is the expected output for the Tensorflow SequentialModel function.

Rmse	Root Mean Square Error measuring neural network prediction accuracy against actual val **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power for variance. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average prediction error magnitude. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction deviations. **DataType**: DOUBLE
OutputModelName	Identifier of the trained neural network model artifact. **DataType**: STRING

34.1.5 Examples

Below are the some of the examples for running Tensorflow SequentialModel SQL function.

```
select * from table(Tensorflow.SequentialModel(
INPUTTABLE => ('banknotes'),
INPUTCOLUMNS => ('variance, skewness, curtosis,entropy'),
TARGETCOLUMN => ('class'),
LAYERS => ARRAY[1],
EPOCHS => 100,
BINARYVALUE => true,
SAVEMODEL => true,
OUTPUTMODEL => ('modelSM1')))

SQL Results:
+-------------------------------+---------------------+
|        CoefficientName        |        Value        |
+-------------------------------+---------------------+
| Mean Squared Error(MSE)       | 14061.4716796875    |
| Mean Absolute Error(MAE)      | 111.27932739257812  |
| Root Mean Squared Error(RMSE) | 118.5810775756836   |
| Regression Score(R2)          | 0.0                 |
| Accuracy                      | -11.279327392578125 |
| OutputModelName               | modelSQM            |
+-------------------------------+---------------------+
```

34.2 Tensorflow SequentialModelPredict

34.2.1 Introduction

Once the TensorFlow Sequential Model has been trained, it can be deployed to generate predictions for new data instances using the learned neural network weights and architecture patterns for inference tasks.

For further references please see Tensorflow SequentialModelPredict.

This estimator implements TensorFlow SequentialModel prediction capabilities.

34.2.2 Syntax

Below is the syntax for the Tensorflow SequentialModelPredict function.

```
SELECT * FROM table(Tensorflow.SequentialModelPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

34.2.3 Input Arguments

Below are the input arguments for the Tensorflow SequentialModelPredict function.

INPUTMODELNAME	Specifies the trained neural network model identifier for prediction inference **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing the feature variables for prediction
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill metho **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for storing prediction results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if it already exists. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export of predictions. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

34.2.4 Output

Below is the expected output for the Tensorflow SequentialModelPredict function.

INPUTCOLUMNNAMES	Feature variables used in the trained neural network model.
PREDICTCOLUMN	Neural network predictions for the target variable.

34.2.5 Examples

Below are the some of the examples for running Tensorflow SequentialModelPredict SQL function.

```
select * from table(Tensorflow.SequentialModelPredict(
INPUTTABLE => ('select fixedacidity,volatileacidity,
quality,quality_Predict from  winequality'),
INPUTMODELNAME => ('modelSQM'))) limit 2

SQL Results:
+--------------+-----------------+---------+-----------------+
| fixedacidity | volatileacidity | quality | quality_Predict |
+--------------+-----------------+---------+-----------------+
| 7.2          | 0.27            | 6       | 5.33243656      |
| 6.3          | 0.3             | 6       | 5.52135419      |
+--------------+-----------------+------------+--------------+
```

35. UnifyML Tensorflow

35.1 Tensorflow SequentialModel

35.1.1 Introduction

TensorFlow is an open-source deep learning framework developed by Google that enables the construction and training of neural networks for complex machine learning tasks. The Sequential Model provides a linear stack of layers for building feedforward neural networks, making it ideal for deep learning applications including computer vision, natural language processing, predictive modeling, and pattern recognition.

For further references please see Tensorflow SequentialModel.

35.1.2 Syntax

Below is the syntax for the Tensorflow SequentialModel function.

```
SELECT * FROM table(Unifyml.Tensorflow.SequentialModel(
  INPUTCOLUMNS => ( inputColumns ) [,...]
  TARGETCOLUMN => ( targetcolumn ) [,...]
  [ INPUTTABLE => ( { table | view | (query) })  [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ PARALLELISM => ( parallelism ) ] [,...]
  [ EPOCHS => ( epochs ) ] [,...]
  [ LAYERS => ARRAY[val [,] ] ] [,...]
  [ BINARYVALUE => ( binary ) ] [,...]
  [ TRAININGSAMPLESIZE => trainsamplesize ] [,...]
  [ NUMPARTITION => numPartition ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVEMODEL => (savemodel) ] [,...]
  [ OUTPUTMODEL => ( outputModelName ) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

35.1.3 Input Arguments

Below are the input arguments for the Tensorflow SequentialModel function.

INPUTCOLUMNS	Specifies the feature variables (independent variables) for neural network training **DataType**: STRING
TARGETCOLUMN	Specifies the target variable (dependent variable) for supervised learning. **DataType**: STRING
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
EPOCHS	[optional] Number of complete passes through the training dataset during model t **DataType**: INTEGER,**Default**: 10
LAYERS	[optional] Architecture specification for dense neural network layers. **DataType**: ARRAY
BINARYVALUE	[optional] Configure loss function for binary classification tasks. **DataType**: BOOLEAN,**Default**: false
INPUTTABLE	[optional] Specifies the dataset containing the training features and target variable
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
TRAININGSAMPLESIZE	[optional] Training-test split ratio for model validation. **DataType**: Double, **Default**: 0.7
PARALLELISM	[optional] Number of parallel processes for distributed model training. **DataType**: INTEGER, **Default**: 1
NUMPARTITION	[optional] Number of data partitions for distributed processing. **DataType**: INTEGER
SAVEMODEL	[optional] Persist the trained neural network model for future predictions. **DataType**: BOOLEAN, **Default**: false
OUTPUTMODEL	[optional] Identifier for the saved model artifact. **DataType**: STRING
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill method. **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

35.1.4 Output

Below is the expected output for the Tensorflow SequentialModel function.

Rmse	Root Mean Square Error measuring neural network prediction accuracy against actual val **DataType**: DOUBLE
R2	Coefficient of determination indicating model's explanatory power for variance. **DataType**: DOUBLE
Mse	Mean Squared Error quantifying average prediction error magnitude. **DataType**: DOUBLE
Mae	Mean Absolute Error measuring average absolute prediction deviations. **DataType**: DOUBLE
OutputModelName	Identifier of the trained neural network model artifact. **DataType**: STRING

35.1.5 Examples

Below are the some of the examples for running Tensorflow SequentialModel SQL function.

```
select * from table(Unifyml.Tensorflow.SequentialModel(
INPUTTABLE => ('banknotes'),
INPUTCOLUMNS => ('variance, skewness, curtosis,entropy'),
TARGETCOLUMN => ('class'),
LAYERS => ARRAY[1],
EPOCHS => 100,
BINARYVALUE => true,
SAVEMODEL => true,
OUTPUTMODEL => ('modelSM1')))

SQL Results:
+------------------------------+--------------------+
|        CoefficientName       |        Value       |
+------------------------------+--------------------+
| Mean Squared Error(MSE)      | 14061.4716796875   |
| Mean Absolute Error(MAE)     | 111.27932739257812 |
| Root Mean Squared Error(RMSE)| 118.5810775756836  |
| Regression Score(R2)         | 0.0                |
| Accuracy                     | -11.279327392578125|
| OutputModelName              | modelSQM           |
+------------------------------+--------------------+
```

35.2 Tensorflow SequentialModelPredict

35.2.1 Introduction

Once the TensorFlow Sequential Model has been trained, it can be deployed to generate predictions for new data instances using the learned neural network weights and architecture patterns for inference tasks.

For further references please see Tensorflow SequentialModelPredict.

This estimator implements TensorFlow SequentialModel prediction capabilities.

35.2.2 Syntax

Below is the syntax for the Tensorflow SequentialModelPredict function.

```
SELECT * FROM table(Unifyml.Tensorflow.SequentialModelPredict(
  INPUTMODELNAME => ( inputModelName ) [,...]
  [ INPUTTABLE => ( { table | view | (query) }) ] [,...]
  [ PARTITIONCOLUMN => ( partitionColumn ) ] [,...]
  [ INPUTDATATYPES => ARRAY[val [,] ] ] [,...]
  [ DATACLEANING => dataCleaning ] [,...]
  [ CLEANINGMETHOD => ARRAY[val [,] ] ] [,...]
  [ FILLWITHVALUES => ARRAY[val [,] ] ] [,...]
  [ DATASCALING => ARRAYScaling ] [,...]
  [ SCALINGMETHOD => scalingMethod ] [,...]
  [ MINSCALER => minScaler ] [,...]
  [ MAXSCALER => maxScaler ] [,...]
  [ INPUTCACHEID => (cacheid) ] [,...]
  [ SAVETODB => (saveToDb) ] [,...]
  [ OUTPUTTABLENAME => ( outputTableName) ] [,...]
  [ OVERWRITEOUTPUTTABLE => (overwriteoutputtable) ] [,...]
  [ SAVETOCSV => (savetocsv) ] [,...]
  [ OUTPUTCSVNAME => (outputcsvname) ] [,...]
  [ BYTESPERCHUNK => (bytesPerChunk) ]
))
```

35.2.3 Input Arguments

Below are the input arguments for the Tensorflow SequentialModelPredict function.

INPUTMODELNAME	Specifies the trained neural network model identifier for prediction inference **DataType**: STRING
INPUTTABLE	[optional] Specifies the dataset containing the feature variables for prediction
PARTITIONCOLUMN	[optional] Column used for data partitioning in distributed processing. **DataType**: STRING
INPUTDATATYPES	[optional] Schema specification for input feature data types. **DataType**: ARRAY
DATACLEANING	[optional] Enable automated data preprocessing pipeline. **DataType**: BOOLEAN
CLEANINGMETHOD	[optional] Data quality improvement techniques (removerow, removeduplicates, removeoutliers, fill) for preprocessing. **DataType**: ARRAY, **Default**: removerow, removeduplicates, removeoutliers
FILLWITHVALUES	[optional] Imputation values for missing data handling when using fill metho **DataType**: ARRAY
DATASCALING	[optional] Apply feature normalization to the dataset. **DataType**: BOOLEAN
SCALINGMETHOD	[optional] Feature scaling transformation method (MinMaxScaler, StandardScaler, Normalizer, Binarizer) for normalization. **DataType**: STRING, **Default**: StandardScaler
MINSCALER	[optional] Lower bound for MinMaxScaler feature transformation. **DataType**: INTEGER
MAXSCALER	[optional] Upper bound for MinMaxScaler feature transformation. **DataType**: INTEGER
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
SAVETODB	[optional] Persist prediction results to database storage. **DataType**: BOOLEAN, **Default**: false
OUTPUTTABLENAME	[optional] Target table name for storing prediction results. **DataType**: STRING
OVERWRITEOUTPUTTABLE	[optional] Replace existing output table if it already exists. **DataType**: BOOLEAN, **Default**: false
SAVETOCSV	[optional] Export prediction results to CSV format. **DataType**: BOOLEAN, **Default**: false
OUTPUTCSVNAME	[optional] Filename for CSV export of predictions. **DataType**: STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimization. **DataType**: INTEGER, **Default**: 64000000

35.2.4 Output

Below is the expected output for the Tensorflow SequentialModelPredict function.

INPUTCOLUMNNAMES	Feature variables used in the trained neural network model.
PREDICTCOLUMN	Neural network predictions for the target variable.

35.2.5 Examples

Below are the some of the examples for running Tensorflow SequentialModelPredict SQLfunction.

```
select * from table(Unifyml.Tensorflow.SequentialModelPredict(
INPUTTABLE => ('select fixedacidity,volatileacidity,
quality,quality_Predict from  winequality'),
INPUTMODELNAME => ('modelSQM'))) limit 2

SQL Results:
+--------------+-----------------+---------+-----------------+
| fixedacidity | volatileacidity | quality | quality_Predict |
+--------------+-----------------+---------+-----------------+
| 7.2          | 0.27            | 6       | 5.33243656      |
| 6.3          | 0.3             | 6       | 5.52135419      |
+--------------+-----------------+------------+--------------+
```

XIV

System Administration Functions

35.3 AddUser

35.3.1 Introduction

The function executes user account creation functionality. Adds new user credentials to the authentication configuration system.

35.3.2 Syntax

Below is the syntax for the AddUser function.

```
SELECT * FROM table(Unifyml.AddUser(
  USERNAME => ( userName )[,...]
  PASSWORD => ( password )
))
```

35.3.3 Input Arguments

Below are the input arguments for the AddUser function.

USERNAME	User identifier for new account creation. **DataType**:STRING
PASSWORD	Authentication credential for user account access. **DataType**:STRING

35.3.4 Output

Below is the expected output for the AddUser function.

AddedUser	Status confirmation of user account creation.
AddedUserName	Identifier of the newly created user account.

35.3.5 Examples

Below are the some of the examples for running AddUser SQL function.

```
select * from table(Unifyml.AddUser(
USERNAME => ("user4"),
PASSWORD => ("user4")))

SQL Results:
+---------------+---------------+
| AddedUser     | AddedUserName |
+---------------+---------------+
| true          | user4         |
+---------------+---------------+
```

35.4 CancelQuery

35.4.1 Introduction

The function executes query termination functionality. Cancels actively executing SQL queries in the system.

35.4.2 Syntax

Below is the syntax for the CancelQuery function.

```
SELECT * FROM table(Unifyml.CancelQuery(
  SESSIONID => (sessionId)  [,..]
  SESSIONQUERYID => (sessionQueryId)
))
```

35.4.3 Input Arguments

Below are the input arguments for the CancelQuery function.

SESSIONID	Unique session identifier for the active database connection. **DataType**:STRING
SESSIONQUERYID	Unique query identifier within the specified session. **DataType**:STRING

35.4.4 Output

Below is the expected output for the CancelQuery function.

SESSIONID	Session identifier for the terminated query. **DataType**:STRING
QUERYID	Query identifier for the terminated operation. **DataType**:STRING
STATUS	Termination status confirmation. **DataType**:BOOLEAN

35.4.5 Examples

Below are the some of the examples for running Cancel query SQL function.

```
select * from table(Unifyml.CancelQuery(
SESSIONID => ("5c6036b3-012b-4151-8842-2ae02eb1cf46"),
SESSIONQUERYID => ("1")))

SQL Results:
+--------------------------------------+---------+--------+
|              SessionId               | QueryId | Status |
+--------------------------------------+---------+--------+
| 596abcd1-6c78-4dbc-996e-2d56f179ecb9 | 14      | 1      |
+--------------------------------------+---------+--------+
```

35.5 DeleteUser

35.5.1 Introduction

The function executes user account removal functionality. Removes user credentials from the authentication configuration system.

35.5.2 Syntax

Below is the syntax for the DeleteUser function.

```
SELECT * FROM table(Unifyml.DeleteUser(
  USERNAME => ( userName )
))
```

35.5.3 Input Arguments

Below are the input arguments for the DeleteUser function.

USERNAME	User identifier for account deletion. **DataType**:STRING

35.5.4 Output

Below is the expected output for the DeleteUser function.

DeletedUser	Status confirmation of user account deletion.
DeletedUserName	Identifier of the deleted user account.

35.5.5 Examples

Below are the some of the examples for running DeleteUser SQL function.

```
select * from table(Unifyml.DeleteUser(
USERNAME => ("user4")))

SQL Results:
+---------------+----------------+
| DeletedUser   | DeletedUserName|
+---------------+----------------+
| true          | user4          |
+---------------+----------------+
```

35.6 DataExport

35.6.1 Introduction

The function executes data extraction and export functionality. Exports datasets to external storage systems and file formats.

35.6.2 Syntax

Below is the syntax for the DataExport function.

```
SELECT * FROM table(Unifyml.DataExport(
  INPUTTABLE => ( ({ table | view | (query) }) ) [,...]
  FILETYPE => ( fileType ) [,...]
  FILENAME => ( fileName ) [,...]
  OUTPUTBUCKET => ( outputBucket ) [,...]
  [ SINGLEFILE => ( singlefile) ] [,...]
  [ S3_ACCESS_KEY => ( S3_Access_Key) ] [,...]
  [ S3_SECRET_KEY => ( S3_Secret_Key) ] [,...]
  [ AZURE_ACCOUNT_KEY => ( Azure_Account_Key) ] [,...]
  [ AZURE_STORAGE_ACCOUNT_NAME => ( Azure_Storage_Account_Name) ] [,...]
  [ BYTESPERCHUNK => ( bytesPerChunk) ] [,...]
  [ PARALLELISM => ( parallelism) ]
  ))
```

35.6.3 Input Arguments

Below are the input arguments for the DataExport function.

INPUTTABLE	Dataset identifier for export operation. **DataType**:STRING
FILETYPE	Output file format specification. **DataType**:STRING, **Default**: csv, json, parquet
FILENAME	Target filename for exported dataset. **DataType**:STRING
OUTPUTBUCKET	Cloud storage destination specification. **DataType**:STRING, **Default**: S3, azure(az)
SINGLEFILE	[optional] File consolidation preference for export output. **DataType**:STRING, **Default**: True
S3ACCESSKEY	[optional] AWS S3 authentication access key. **DataType**:STRING
S3SECRETKEY	[optional] AWS S3 authentication secret key. **DataType**:STRING
AZUREACCOUNTKEY	[optional] Azure storage authentication account key. **DataType**:STRING
AZURESTORAGEACCOUNTNAME	[optional] Azure storage account identifier. **DataType**:STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimizat **DataType**: INTEGER, **Default**: 64000000
PARALLELISM	[optional] Number of parallel processes for distributed export operatio **DataType**: INTEGER, **Default**: 1

35.6.4 Output

Below is the expected output for the DataExport function.

INPUTTABLE	Source dataset identifier for the export operation.
DATAEXPORT	Export operation completion status. **DataType**:BOOLEAN

35.6.5 Examples

Below are the some of the examples for running DataExport SQL function.

```
select * from table(Unifyml.DataExport(
INPUTTABLE => ("housing"),
FILETYPE => ("csv"),
SINGLEFILE => (false),
FILENAME => ("housingtests3"),
OUTPUTBUCKET => ("s3://awsunifyml"),
S3_ACCESS_KEY => ("AKIAW7GJVDMWXNM5HZOT"),
S3_SECRET_KEY => ("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")))

SQL Results:
+----------------+------------+
| InputTableName | DataExport |
+----------------+------------+
| housing        | true       |
+----------------+------------+
```

35.7 DataImport

35.7.1 Introduction

The function executes data ingestion and import functionality. Imports datasets from external storage systems into the analytics platform.

35.7.2 Syntax

Below is the syntax for the DataImport function.

```
SELECT * FROM table(Unifyml.DataImport(
  OUTPUTTABLE => ( ({ table | view | (query) }) ) [,...]
  FILETYPE => ( fileType ) [,...]
  FILENAME => ( fileName ) [,...]
  INPUTBUCKET => ( outputBucket ) [,...]
  [ S3_ACCESS_KEY => ( S3_Access_Key) ] [,...]
  [ S3_SECRET_KEY => ( S3_Secret_Key) ] [,...]
  [ AZURE_ACCOUNT_KEY => ( Azure_Account_Key) ] [,...]
  [ AZURE_STORAGE_ACCOUNT_NAME => ( Azure_Storage_Account_Name) ] [,...]
  [ PARALLELISM => ( parallelism) ]
  ))
```

35.7.3 Input Arguments

Below are the input arguments for the DataImport function.

OUTPUTTABLE	Target dataset identifier for import operation. **DataType**:STRING
FILETYPE	Source file format specification. **DataType**:STRING, **Default**: csv, json, parquet
FILENAME	Source filename for dataset import. **DataType**:STRING
INPUTBUCKET	Cloud storage source specification. **DataType**:STRING, **Default**: S3, azure(az)
S3ACCESSKEY	[optional] AWS S3 authentication access key. **DataType**:STRING
S3SECRETKEY	[optional] AWS S3 authentication secret key. **DataType**:STRING
AZUREACCOUNTKEY	[optional] Azure storage authentication account key. **DataType**:STRING
AZURESTORAGEACCOUNTNAME	[optional] Azure storage account identifier. **DataType**:STRING
BYTESPERCHUNK	[optional] Memory allocation per data chunk for processing optimizat **DataType**: INTEGER, **Default**: 64000000

35.7.4 Output

Below is the expected output for the DataImport function.

OUTPUTTABLE	Target dataset identifier for the import operation.
DATAIMPORT	Import operation completion status. **DataType**:BOOLEAN

35.7.5 Examples

Below are the some of the examples for running DataImport SQL function.

```
select * from table(Unifyml.DataImport(
OUTPUTTABLE => ("housingout"),
FILETYPE => ("csv"),
FILENAME => ("housingtests3/*"),
INPUTBUCKET => ("s3://awsunifyml"),
S3_ACCESS_KEY => ("AKIAW7GJVDMWXNM5HZOT"),
S3_SECRET_KEY => ("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")))

SQL Results:
+----------------+------------+
| InputTableName | DataImport |
+----------------+------------+
| housingout     | true       |
+----------------+------------+
```

35.8 ListAllModel

35.8.1 Introduction

The function executes model registry inspection functionality. Retrieves comprehensive metadata for all trained machine learning models in the system.

35.8.2 Syntax

Below is the syntax for the ListAllModel function.

```
SELECT * FROM table(Unifyml.ListAllModel())
```

35.8.3 Output

Below is the expected output for the ListAllModel function.

ModelFileName	Trained model artifact identifier.
FunctionName	Machine learning algorithm implementation (spark or scikit-learn).
InputTableName	Source dataset used for model training.
InputColumnsList	Feature variables used in model training.
InputCategoricalColumnsList	Categorical feature variables with string data types.
TargetColumnName	Target variable used for supervised learning.
Rsme	Root Mean Square Error model performance metric.
R2	Coefficient of determination model performance metric.
ModelParameters	Algorithm-specific hyperparameters and configuration.

35.8.4 Examples

Below are the some of the examples for running ListAllModel SQL function.

```
SELECT * FROM table(Unifyml.ListAllModel())

SQL Results:
```

35.9 ListAllCache

35.9.1 Introduction

The function executes cache registry inspection functionality. Retrieves metadata for all cached datasets in the system.

35.9.2 Syntax

Below is the syntax for the ListAllCache function.

```
SELECT * FROM table(Unifyml.ListAllCache())
```

35.9.3 Output

Below is the expected output for the ListAllCache function.

CacheID	Cached dataset unique identifier.
InputTableName	Source dataset name for the cached data.
InputColumnsList	Feature variables included in the cached dataset.

35.9.4 Examples

Below are the some of the examples for running ListAllCache SQL function.

```
SELECT * FROM table(Unifyml.ListAllCache())

SQL Results:
+--------------+------------------------+--------------------+
|   CacheId    |     InputTableName     | InputColumnsList   |
+--------------+------------------------+--------------------+
| datacacheid1 | select * from banknotes | variance, skewness |
| cacheid2     | banknotes               | variance, skewness |
| cacheid      | select * from banknotes | variance ,skewness |
+--------------+------------------------+--------------------+
```

35.10 ModifyPassword

35.10.1 Introduction

The function executes user authentication credential update functionality. Modifies user password in the authentication configuration system.

35.10.2 Syntax

Below is the syntax for the ModifyPassword function.

```
SELECT * FROM table(Unifyml.ModifyPassword(
  PASSWORD => ( password )
))
```

35.10.3 Input Arguments

Below the input arguments for the ModifyPassword function.

PASSWORD	New authentication credential for user account access. **DataType**:STRING

35.10.4 Output

Below is the expected output for the ModifyPassword function.

ModifiedPassword	Status confirmation of password update operation.
ModifiedPasswordName	Identifier associated with the password modification.

35.10.5 Examples

Below are the some of the examples for running ModifyPassword SQL function.

```
select * from table(Unifyml.ModifyPassword(
PASSWORD => ("user4")))

SQL Results:
+------------------+----------------------+
| ModifiedPassword | ModifiedPasswordName |
+------------------+----------------------+
| true             |      user4           |
+------------------+----------------------+
```

35.11 PurgeAllCache

35.11.1 Introduction

The function executes comprehensive cache cleanup functionality. Removes all cached datasets from the user's workspace environment.

35.11.2 Syntax

Below is the syntax for the PurgeAllCache function.

```
SELECT * FROM table(Unifyml.PurgeAllCache())
```

35.11.3 Output

Below is the expected output for the PurgeAllCache function.

DeletedModel	Confirmation of cache deletion operati

35.11.4 Examples

Below are the some of the examples for running PurgeAllCache SQL function.

```
SELECT * FROM table(Unifyml.PurgeAllCache())

SQL Results:
+---------------+---------------+
| DeletedCache  | DeletedCaches |
+---------------+---------------+
| Deleted cache | Deleted cache |
+---------------+---------------+
```

35.12 PurgeAllModel

35.12.1 Introduction

The function executes comprehensive model cleanup functionality. Removes all trained machine learning model artifacts from the system.

35.12.2 Syntax

Below is the syntax for the PurgeAllModel function.

```
SELECT * FROM table(Unifyml.PurgeAllModel())
```

35.12.3 Output

Below is the expected output for the PurgeAllModel function.

DeletedModel	Confirmation of model deletion operat

35.12.4 Examples

Below are the some of the examples for running PurgeAllModel SQL function.

```
SELECT * FROM table(Unifyml.PurgeAllModel())

SQL Results:
+---------------+---------------+
| DeletedModel  | DeletedModels |
+---------------+---------------+
| Deleted model | Deleted model |
+---------------+---------------+
```

35.13 PurgeModel

35.13.1 Introduction

The function executes selective model cleanup functionality. Removes a specific trained machine learning model artifact from the system.

35.13.2 Syntax

Below is the syntax for the PurgeModel function.

```
SELECT * FROM table(Unifyml.PurgeModel(
  INPUTMODELNAME => ( inputModelName )
))
```

35.13.3 Input Arguments

Below are the input arguments for the PurgeModel function.

INPUTMODELNAME	Trained model identifier for deletion operation. **DataType**:STRING

35.13.4 Output

Below is the expected output for the PurgeModel function.

DeletedModel	Confirmation of specific model deletion.

35.13.5 Examples

Below are the some of the examples for running PurgeModel SQL function.

```
SELECT * FROM table(Unifyml.PurgeModel(
  INPUTMODELNAME => ("modelABR")))

SQL Results:
+--------------+------------------+
| DeletedModel | SourceInputTable |
+--------------+------------------+
| modelXGB     | modelXGB         |
+--------------+------------------+
```

35.14 PurgeCache

35.14.1 Introduction

The function executes selective cache cleanup functionality. Removes a specific cached dataset from the system.

35.14.2 Syntax

Below is the syntax for the PurgeCache function.

```
SELECT * FROM table(Unifyml.PurgeCache(
  INPUTCACHENAME => ( inputCacheName )
))
```

35.14.3 Input Arguments

Below are the input arguments for the PurgeCache function.

INPUTCACHENAME	Cached dataset identifier for deletion operation. **DataType**:STRING

35.14.4 Output

Below is the expected output for the PurgeCache function.

DeletedCACHE	Confirmation of specific cache deletion.

35.14.5 Examples

Below are the some of the examples for running PurgeCache SQL function.

```
SELECT * FROM table(Unifyml.PurgeCache(
  INPUTCACHENAME => ("cacheid")))

SQL Results:
+--------------+------------------+
| DeletedCache | SourceInputTable |
+--------------+------------------+
| cacheid      | cacheid          |
+--------------+------------------+
```

35.15 ShowQueries

35.15.1 Introduction

The function executes active query monitoring functionality. Displays currently executing SQL queries and their session metadata.

35.15.2 Syntax

Below is the syntax for the ShowQueries function.

```
SELECT * FROM table(Unifyml.ShowQueries())
```

35.15.3 Output

Below is the expected output for the ShowQueries function.

SessionId	Unique session identifier for the database connection.
QueryId	Unique query identifier within the session.
ClientId	Client connection identifier for the user.
QueryText	Currently executing SQL statement text.

35.15.4 Examples

Below are the some of the examples for running ShowQueries SQL function.

```
SELECT * FROM table(Unifyml.ShowQueries())

SQL Results:
+-------------------------------------------------------------------------+
| UserName |              SessionId               | QueryId | ClientId  |
+----------+--------------------------------------+---------+--------+---+
| admin    | 596abcd1-6c78-4dbc-996e-2d56f179ecb9 | 14      | 127.0.1.1 |
+----------+--------------------------------------+---------+-----------+
+-------------------------------------------+
|    QueryText                              |
+-------------------------------------------+
| SELECT * FROM table(Unifyml.PurgeAllModel())|
+-------------------------------------------+
```

35.16 EncryptPassword

35.16.1 Introduction

The function executes data encryption functionality. Encrypts passwords and sensitive text using cryptographic algorithms.

35.16.2 Syntax

Below is the syntax for the EncryptPassword function.

```
SELECT * FROM table(Unifyml.Encrypt(
  INPUTTEXT => ( inputText )
))
```

35.16.3 Input Arguments

Below are the input arguments for the EncryptPassword function.

INPUTTEXT	Plain text string to be encrypted using cryptographic transformation. **DataType**:STRING

35.16.4 Output

Below is the expected output for the EncryptPassword function.

EncryptedText	Cryptographically transformed output text.
IsEncrypted	Confirmation status of encryption operation.

35.16.5 Examples

Below are the some of the examples for running EncryptPassword SQL function.

```
SELECT * FROM table(Unifyml.Encrypt(
INPUTTEXT => ("cloud456")))

SQL Results:
+--------------------------+------------+
|      EncryptrdText       | IsEncrpted |
+--------------------------+------------+
| ZcINm6wZxX3mR3xZYAtPlg== | true       |
+--------------------------+------------+
```

35.17 ExportTable

35.17.1 Introduction

The function executes dataset export functionality. Exports table data to specified file paths and formats.

35.17.2 Syntax

Below is the syntax for the ExportTable function.

```
SELECT * FROM table(Unifyml.ExportTable(
  INPUTQUERY => ( inputQuery )[,...]
  FILENAME => ( fileName )[,...]
  [ TYPE => ( type) ] [,...]
  [ FILEPATH => ( filePath ) ]
))
```

35.17.3 Input Arguments

Below are the input arguments for the ExportTable function.

INPUTQUERY	Source dataset identifier for export operation. **DataType**:STRING
FILENAME	Target filename for exported dataset. **DataType**: STRING
TYPE	[optional] Output file format specification. **DataType**: STRING,**Default**: CSV
FILEPATH	[optional] Target directory path for export operation. **DataType**: STRING,**Default**: client jar location

35.17.4 Output

Below is the expected output for the ExportTable function.

ExportFile	Export operation completion status.
ExportFilePath	Complete file path for the exported dataset.

35.17.5 Examples

Below are the some of the examples for running ExportTable SQL function.

```
select * from table(Unifyml.ExportTable(
INPUTQUERY => ("housing"),
FILENAME => ("exporthousing")))

SQL Results:
+-------------+-----------------------------------------------------+
| ExportFile  |                  ExportFilePath                     |
+-------------+-----------------------------------------------------+
| True        |/datanalyt/unifyml/calcite_client/exporthousing.csv|
+-------------+-----------------------------------------------------+
```

35.18 ExportModelFile

35.18.1 Introduction

The function executes model artifact export functionality. Exports trained machine learning models to specified file system locations.

35.18.2 Syntax

Below is the syntax for the ExportModelFile function.

```
SELECT * FROM table(Unifyml.ExportModelFile(
  INPUTMODELNAME => ( inputModelName )[,...]
  [ EXPORTDIRPATH => ( exportDirPath) ]
))
```

35.18.3 Input Arguments

Below are the input arguments for the ExportModelFile function.

INPUTMODELNAME	Trained model identifier for export operation. **DataType**:STRING
EXPORTDIRPATH	[optional] Target directory path for model artifact export. **DataType**: STRING

35.18.4 Output

Below is the expected output for the ExportModelFile function.

ExportModel	Model export operation completion status.
ExportModelFile	Complete file path for the exported model artifact.

35.18.5 Examples

Below are the some of the examples for running ExportModelFile SQL function.

```
select * from table(Unifyml.ExportModelFile(
 INPUTMODELNAME => ("modelDTC")))

SQL Results:
+-------------+-------------------------------------------------------+
| ExportModel |                  ExportModelFile                      |
+-------------+-------------------------------------------------------+
| True        | /home/venu/datanalyt/unifyml/calcite/modelDTC.zip |
+-------------+-------------------------------------------------------+
```

35.19 ImportModelFile

35.19.1 Introduction

The function executes model artifact import functionality. Imports trained machine learning models from external file system locations into the user workspace.

35.19.2 Syntax

Below is the syntax for the ImportModelFile function.

```
SELECT * FROM table(Unifyml.ImportModelFile(
  IMPORTFILEPATH => ( importFilePath )
))
```

35.19.3 Input Arguments

Below are the input arguments for the ImportModelFile function.

IMPORTFILEPATH	Source file path containing the model artifact to import. **DataType**:STRING

35.19.4 Output

Below is the expected output for the ImportModelFile function.

ImportModel	Model import operation completion status.
ModelFileName	Identifier of the imported model artifact.

35.19.5 Examples

Below are the some of the examples for running ImportModelFile SQL function.

```
select * from table(Unifyml.ImportModelFile(
 IMPORTFILEPATH => ("modelDTC.zip")))

SQL Results:
+-------------+---------------+
| ImportModel | ModelFileName |
+-------------+---------------+
| True        | modelDTC      |
+-------------+---------------+
```

35.20 ImportTable

35.20.1 Introduction

The function executes dataset import functionality. Imports external datasets into the connected storage system for analysis.

35.20.2 Syntax

Below is the syntax for the ImportTable function.

```
SELECT * FROM table(Unifyml.ImportTable(
  FILENAME => ( fileName )[,...]
  INPUTPATH => ( inputPath )[,...]
  [ TYPE => ( type) ] [,...]
  [ CHUNKSIZE => ( chunkSize ) [,...]
  [ OUTPUTTABLENAME => ( outputTableName ) ]
))
```

35.20.3 Input Arguments

Below are the input arguments for the ImportTable function.

FILENAME	Source filename for dataset import operation. **DataType**: STRING
INPUTPATH	Source directory path containing the dataset file. **DataType**:STRING
TYPE	[optional] Source file format specification. **DataType**: STRING,**Default**: CSV
CHUNKSIZE	[optional] Data partitioning size for distributed processing. **DataType**: INTEGER,**Default**: 16
OUTPUTTABLENAME	[optional] Target table identifier for imported dataset. **DataType**: STRING,**Default**: FILENAME

35.20.4 Output

Below is the expected output for the ImportTable function.

OutputTableName	Identifier of the imported dataset table.
ImportFilePath	Source file path for the imported dataset.

35.20.5 Examples

Below are the some of the examples for running ImportTable SQL function.

```
select * from table(Unifyml.ImportTable(
INPUTPATH => ("./csvs"),
FILENAME => ("housing.csv"),
OUTPUTTABLENAME => ("housingcsv")));

SQL Results:
+-----------------+-----------------------------------------------------+
| OutputTableName  |                  ImportFilePath                    |
+-----------------+-----------------------------------------------------+
| housingcsv       | ./csvs/housing.csv                                  |
+-----------------+-----------------------------------------------------+
```

35.21 ShowModelDetails

35.21.1 Introduction

The function executes model metadata inspection functionality. Displays comprehensive details and performance metrics for trained machine learning models.

35.21.2 Syntax

Below is the syntax for the ShowModelDetails function.

```
SELECT * FROM table(Unifyml.ShowModelDetails(
  INPUTMODELNAME => ( inputModelName )
))
```

35.21.3 Input Arguments

Below are the input arguments for the ShowModelDetails function.

INPUTMODELNAME	Trained model identifier for metadata inspection. **DataType**:STRING

35.21.4 Output

Below is the expected output for the ShowModelDetails function.

CoefficientName	Model performance metrics and configuration attributes.
Value	Corresponding values for the model attributes and metrics.

35.21.5 Examples

Below are the some of the examples for running ShowModelDetails SQL function.

```
SELECT * FROM table(Unifyml.ShowModelDetails(
  INPUTMODELNAME => ("modelXGB")))

SQL Results:
+------------------------------+-----------------------+
|        CoefficientName       |         Value         |
+------------------------------+-----------------------+
| Mean Squared Error(MSE)      | 0.0024154589371980675 |
| Mean Absolute Error(MAE)     | 0.0024154589371980675 |
| Root Mean Squared Error(RMSE)| 0.04914731871829904   |
| Regression Score(R2)         | 0.9902560723027678    |
| Accuracy                     | 99.9975845410628      |
| OutputModelName              | modelXGB              |
+------------------------------+-----------------------+
```

35.22 ShowSampleData

35.22.1 Introduction

The function executes dataset preview functionality. Displays a sample subset of rows from the specified dataset for data exploration.

35.22.2 Syntax

Below is the syntax for the ShowSampleData function.

```
SELECT * FROM table(Unifyml.ShowSampleData(
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTCACHEID => ( cacheId ) ][,...]
  [ ROWLIMIT => ( rowsLimit )]
))
```

35.22.3 Input Arguments

Below are the input arguments for the ShowSampleData function.

INPUTTABLE	[optional] Source dataset identifier for data preview.
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING
ROWLIMIT	[optional] Maximum number of rows to display in the sample. **DataType**: INTEGER, **Default**: 10

35.22.4 Output

Below is the expected output for the ShowSampleData function.

InputDf	Sample dataset preview with specified row limit.

35.22.5 Examples

Below are the some of the examples for running ShowSampleData SQL function.

```
SELECT * FROM table(Unifyml.ShowSampleData(
  INPUTTABLE => ("banknotes"),
  ROWLIMIT => 5))

SQL Results:
+----------+----------+----------+----------+-------+
| variance | skewness | curtosis | entropy  | class |
+----------+----------+----------+----------+-------+
| 3.6216   | 8.6661   | -2.8073  | -0.44699 | 0     |
| 4.5459   | 8.1674   | -2.4586  | -1.4621  | 0     |
| 3.866    | -2.6383  | 1.9242   | 0.10645  | 0     |
| 3.4566   | 9.5228   | -4.0112  | -3.5944  | 0     |
| 0.32924  | -4.4552  | 4.5718   | -0.9888  | 0     |
+----------+----------+----------+----------+-------+
```

35.23 ShowDataTypes

35.23.1 Introduction

The function executes schema inspection functionality. Displays the data type schema for all columns in the specified dataset.

35.23.2 Syntax

Below is the syntax for the ShowDataTypes function.

```
SELECT * FROM table(Unifyml.ShowDataTypes(
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTCACHEID => ( cacheId )]
))
```

35.23.3 Input Arguments

Below are the input arguments for the ShowDataTypes function.

INPUTTABLE	[optional] Source dataset identifier for schema inspection.
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING

35.23.4 Output

Below is the expected output for the ShowDataTypes function.

ColumnNames	Feature variable identifiers in the dataset.
ColumnDataTypes	Data type specifications for each feature variable.

35.23.5 Examples

Below are the some of the examples for running ShowDataTypes SQL function.

```
SELECT * FROM table(Unifyml.ShowDataTypes(
  INPUTTABLE => ("housing")))

SQL Results:
+------------+----------------+
| ColumnName | ColumnDataType |
+------------+----------------+
| crim       | DoubleType     |
| zn         | DoubleType     |
| indus      | DoubleType     |
| chas       | IntegerType    |
| nox        | DoubleType     |
| rm         | DoubleType     |
| age        | DoubleType     |
| dis        | DoubleType     |
| rad        | DoubleType     |
| tax        | DoubleType     |
| ptratio    | DoubleType     |
| b          | DoubleType     |
| lstat      | DoubleType     |
| medv       | DoubleType     |
+------------+----------------+
```

35.24 ShowRowCount

35.24.1 Introduction

The function executes dataset size inspection functionality. Displays the total number of observations (rows) in the specified dataset.

35.24.2 Syntax

Below is the syntax for the ShowRowCount function.

```
SELECT * FROM table(Unifyml.ShowRowCount(
  [ INPUTTABLE => ( { table | view | (query) })]  [,...]
  [ INPUTCACHEID => ( cacheId ) ]
))
```

35.24.3 Input Arguments

Below are the input arguments for the ShowRowCount function.

INPUTTABLE	[optional] Source dataset identifier for row count analysis. **DataType**: STRING
INPUTCACHEID	[optional] Cached dataset identifier for efficient data retrieval. **DataType**: STRING

35.24.4 Output

Below is the expected output for the ShowRowCount function.

InputSource	Dataset identifier or cache ID used for the operation.
Count	Total number of observations in the dataset.

35.24.5 Examples

Below are the some of the examples for running ShowRowCount SQL function.

```
select * from table(Unifyml.ShowRowCount(
INPUTTABLE => ("housing")))

SQL Results:
+-------------+-------+
| InputSource | Count |
+-------------+-------+
| housing     | 506   |
+-------------+-------+
```

35.25 ListAllTables

35.25.1 Introduction

The function executes database catalog inspection functionality. Retrieves all available datasets and tables in the connected data source.

35.25.2 Syntax

Below is the syntax for the ListAllTables function.

```
SELECT * FROM table(Unifyml.ListAllTables())
```

35.25.3 Output

Below is the expected output for the ListAllTables function.

Tableschema	Database schema identifier containing the tables.
InputTableNames	Available dataset and table identifiers in the database.

35.25.4 Examples

Below are the some of the examples for running ListAllTables SQL function.

```
SELECT * FROM table(Unifyml.ListAllTables())

SQL Results:
+-------------+------------------------------+
| TableSchema |          TablesList          |
+-------------+------------------------------+
| testdb      | Cement_concrete              |
| testdb      | FileIteratorcacheaudio       |
| testdb      | Health_Insurance_train       |
+-------------+------------------------------+
```