Oracle® Database

Machine Learning for SQL

Release 21*c*

F39315-02

October 2021

Oracle Database Machine Learning for SQL, Release 21*c*

F39315-02

Primary Author: Sarika Surampudi

Contents

3 Use cases

4 Reference

1 Concepts

Machine Learning Overview

Machine learning is a subset of Artificial Intelligence (AI) that focuses on building systems that learn or improve performance based on the data they consume.

What Is Machine Learning?

Machine learning is a technique that discovers previously unknown relationships in data.

Machine learning and AI are often discussed together. An important distinction is that although all machine learning is AI, not all AI is machine learning. Machine learning automatically searches potentially large stores of data to discover patterns and trends that go beyond simple statistical analysis. Machine learning uses sophisticated algorithms that identify patterns in data creating models. Those models can be used to make predictions and forecasts, and categorize data.

The key features of machine learning are:

- Automatic discovery of patterns
- Prediction of likely outcomes
- Creation of actionable information
- Ability to analyze potentially large volumes of data

Machine learning can answer questions that cannot be addressed through traditional deductive query and reporting techniques.

Benefits of Machine Learning

Machine learning is a powerful technology that can help you find patterns and relationships within your data.

Find trends and patterns - Machine learning discovers hidden information in your data. You might already be aware of important patterns as a result of working with your data over time. Machine learning can confirm or qualify such empirical observations in addition to finding new patterns that are not immediately distinguishable through simple observation. Machine learning can discover predictive relationships that are not causal relationships. For example, machine learning might determine that males with incomes between $50,000 and $65,000 who subscribe to certain magazines are likely to buy a given product. You can use this information to help you develop a marketing strategy. Machine learning can handle large volume of data and can be used in financial analysis. Some of the benefits include stock price predictions (algorithmic trading) and portfolio management.

Make data driven decisions - Many companies have big data and extracting meaningful information from that data is important in making data driven business decisions. By leveraging machine learning algorithms, organizations are able to transform data into knowledge and actionable intelligence. With the changing demands, companies are able to make better decisions faster by using machine learning techniques.

Recommend products - Machine learning results can also be used to influence customer decisions by promoting or recommending relevant and useful products based on behavior patterns of customers online or their response to a marketing campaign.

Detect fraud, anomalies, and security risks - The Financial Services sector has benefited from machine learning algorithms and techniques by discovering unusual patterns or fraud and responding to new fraud behaviors much more quickly. Today companies and governments are conducting business and sharing information online. In such cases, network security is a concern. Machine learning can help in detecting anomalous behavior and automatically take corrective actions.

Retail analysis - Machine learning helps to analyze customer purchase patterns to provide promotional offers for target customers. This service ensures superior customer experience and improves customer loyalty.

Healthcare - Machine learning in medical use is becoming common, helping patients and doctors. Advanced machine learning techniques are used in radiology to make an intelligent decision by reviewing images such as radiographs, CT, MRI, PET images, and radiology reports. It is reported that machine learning-based automatic detection and diagnosis are at par or better than the diagnosis of an actual radiologist. Some of the machine learning applications are trained to detect breast cancer. Another common use of machine learning in the medical field is that of automated billing. Some applications using machine learning can also point out patient's risk for various conditions such as stroke, diabetes, coronary artery diseases, and kidney failures and recommend medication or procedure that may be necessary.

To summarize, machine learning can:

- easily identify trends and patterns
- simplify product marketing and sales forecast
- facilitate early anomaly detection
- minimize manual intervention by "learning"
- handle multidimensional data

Define Your Business Problem

Enterprises face problems such as classifying documents, predicting the financial outcomes, detecting hidden patterns and anomalies, and so on. Machine learning can help solve such problems provided that you have clear understanding of the business problem with enough data and learn to ask the right questions to obtain meaningful results.

You require skills in preparing data, applying ML techniques, and evaluating results. The patterns you find through machine learning may be very different depending on how you formulate the problem. For example, rather than trying to learn how to "improve the response to a direct mail campaign," you might try to find the characteristics of people who have responded to your campaigns in the past. You can then classify if a given profile of a prospect would respond to a direct email campaign.

Many forms of machine learning are predictive. For example, a model can predict income level based on education and other demographic factors. Predictions have an associated probability (How likely is this prediction to be true?). Prediction probabilities are also known as confidence (How confident can I be of this prediction?). Some forms of predictive machine learning generate rules, which are conditions that imply a given outcome. For example, a rule can specify that a person who has a bachelor's degree and lives in a certain neighborhood is likely to have an income greater than the regional average. Rules have an associated support (What percentage of the population satisfies the rule?).

Other forms of machine learning identify groupings in the data. For example, a model might identify the segment of the population that has an income within a specified range, that has a good driving record, and that leases a new car on a yearly basis.

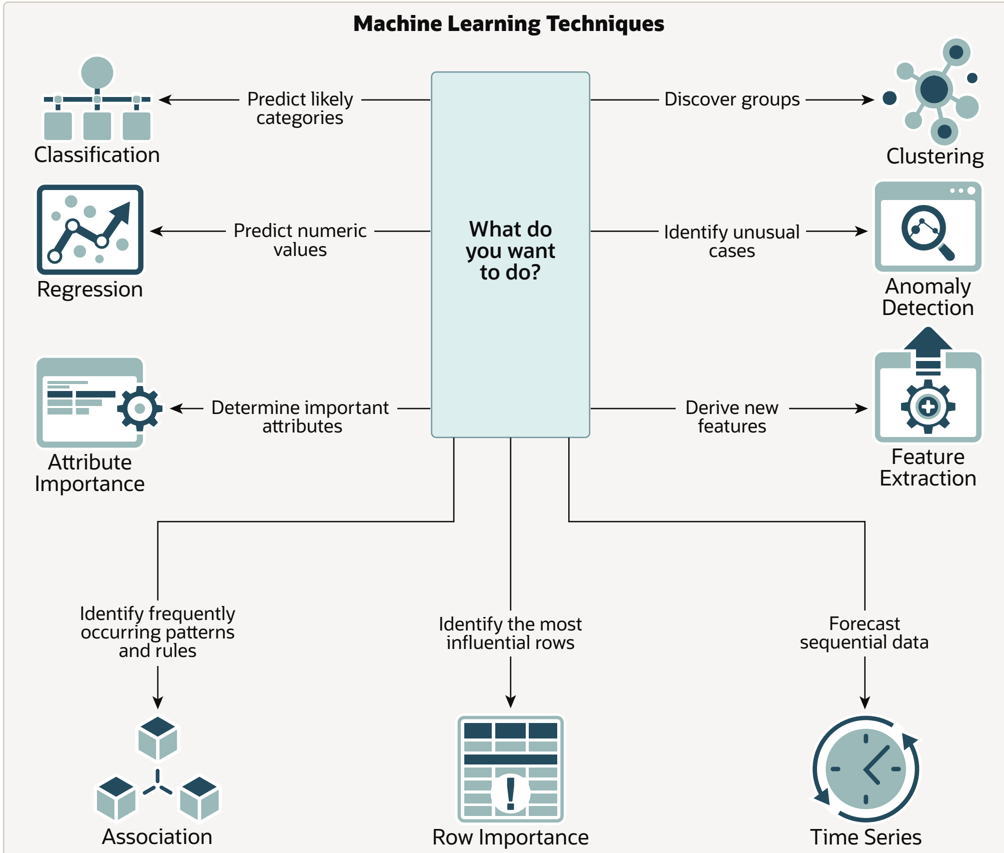
What Do You Want to Do?

Multiple machine learning techniques, also referred to as "mining function", are available through Oracle Database and Oracle Autonomous Database. Depending on your business problem, you can identify the appropriate mining function, or combination of mining functions, and select the algorithm or algorithms that may best support the solution.

For some mining functions, you can choose from among multiple algorithms. For specific problems, one technique or algorithm may be a better fit than the other or more than one algorithm can be used to solve the problem.

The following diagram provides a basic idea on how to select machine learning techniques that are available across Oracle Database and Oracle Autonomous Database.

Figure 1-1 Machine Learning Techniques



OML provides machine learning capabilities within Oracle Database by offering a broad set of in-database algorithms to perform a variety of machine learning techniques such as Classification, Regression, Clustering, Feature Extraction, Anomaly Detection, Association (Market Basket Analysis), and Time Series. Others include Attribute Importance, Row Importance, and Ranking. OML uses built-in features of Oracle Database to maximize scalability, improved memory, and performance. OML is also integrated with open source languages such as Python and R. Through the use of open source packages from R and Python, users can extend this set of techniques and algorithms in combination with embedded execution from OML4Py and OML4R.

Discover More Through Interfaces

Oracle supports programming language interfaces for SQL, R, and Python, and no-code user interfaces such as OML AutoML UI and Oracle Data Miner, and REST model management and deployment through OML Services.

Oracle Machine Learning Notebooks (OML Notebooks) is based on Apache Zeppelin technology enabling you to perform machine learning in Oracle Autonomous Database (Autonomous Data Warehouse (ADW), Autonomous Transactional Database (ATP), and Autonomous JSON Database (AJD)). OML Notebooks helps users explore, visualize, and prepare data, and develop and document analytical methodologies.

AutoML User Interface (AutoML UI) is an Oracle Machine Learning interface that provides you no-code automated machine learning. When you create and run an experiment in AutoML UI, it automatically performs algorithm and feature selection, as well as model tuning and selection, thereby enhancing productivity as well as model accuracy and performance. Business users without extensive data science background can use AutoML UI to create and deploy machine learning models.

Oracle Machine Learning Services (OML Services) extends OML functionality to support model deployment and model lifecycle management for both in-database OML models and third-party Open Neural Networks Exchange (ONNX) format machine learning models through REST APIs. The REST API for Oracle Machine Learning Services provides REST API endpoints hosted on Oracle Autonomous Database. These endpoints enable you to store machine learning models along with its metadata, and create scoring endpoints for the model.

Oracle Machine Learning for Python (OML4Py) enables you to run Python commands and scripts for data transformations and for statistical, machine learning, and graphical analysis on data stored in or accessible through Oracle Autonomous Database service using a Python API. OML4Py is a Python module that enables Python users to manipulate data in database tables and views using Python syntax. OML4Py functions and methods transparently translate a select set of Python functions into SQL for in-database execution. OML4Py users can use Automated Machine Learning (AutoML) to enhance user productivity and machine learning results through automated algorithm and feature selection, as well as model tuning and selection. Users can use Embedded Python Execution to run user-defined Python functions in Python engines spawned by the Autonomous Database environment.

Oracle Machine Learning for R (OML4R) provides a database-centric environment for end-to-end analytical processes in R, with immediate deployment of user-defined R functions to production environments. OML4R is a set of R packages and Oracle Database features that enable an R user to operate on database-resident data without using SQL and to run user-defined R functions, also referred to as "scripts",in one or more database-controlled R engines. OML4R is included with Oracle Database and Oracle Database Cloud Service.

Oracle Machine Learning for SQL (OML4SQL) provides SQL access to powerful, in-database machine learning algorithms. You can use OML4SQL to build and deploy predictive and descriptive machine learning models that can add intelligent capabilities to applications and dashboards. OML4SQL is included with Oracle Database, Oracle Database Cloud Service, and Oracle Autonomous Database.

Oracle Data Miner (ODMr) is an extension to Oracle SQL Developer. Oracle Data Miner is a graphical user interface to discover hidden patterns, relationships, and insights in data. ODMr provides a drag-and-drop workflow editor to define and capture the steps that users take to explore and prepare data and apply machine learning technology.

Oracle Machine Learning for Spark (OML4Spark) provides scalable machine learning algorithms through an R API for Spark and Hadoop environments to explore and prepare data and build and deploy machine learning models. OML4Spark is a component of the Oracle Big Data Connectors and included with Oracle Big Data Service.

Related Topics

- https://www.oracle.com/database/technologies/datawarehouse-bigdata/machine-learning.html

Process Overview

The lifecycle of a machine learning project is divided into six phases. The process begins by defining a business problem and restating the business problem in terms of a machine learning objective. The end goal of a machine learning process is to produce accurate results for solving your business problem.

Workflow

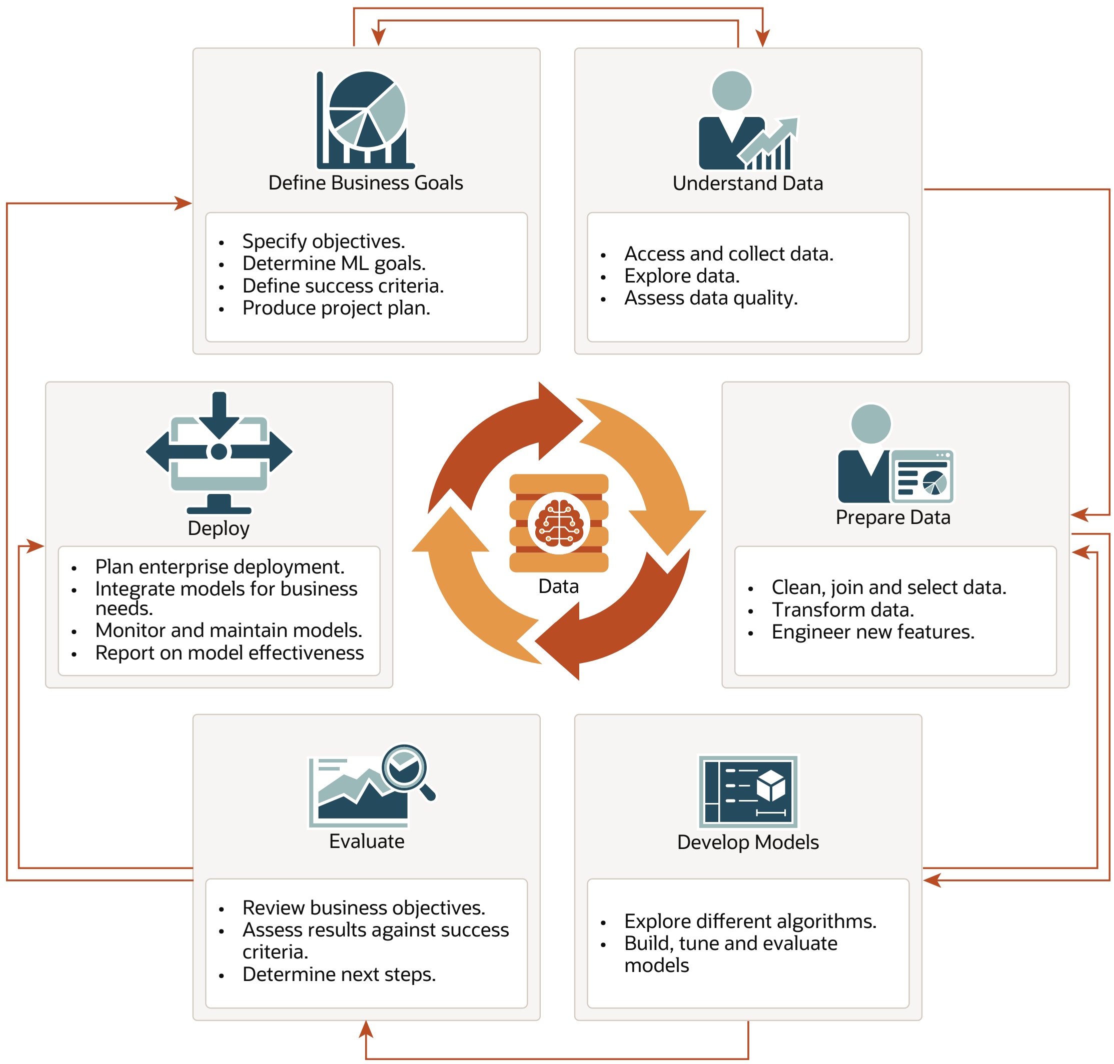
The machine learning process workflow illustration is based on the CRISP-DM methodology. Each stage in the workflow is illustrated with points that summarize the key tasks. The CRISP-DM methodology is the most commonly used methodology for machine learning.

The following are the phases of the machine learning process:

- Define business goals
- Understand data
- Prepare data
- Develop models
- Evaluate
- Deploy

Each of these phases are described separately. The following figure illustrates machine learning process workflow.

Figure 1-2 Machine Learning Process Workflow



Related Topics

- https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome
- https://www.sv-europe.com/crisp-dm-methodology/

Define Business Goals

The first phase of machine learning process is to define business objectives. This initial phase of a project focuses on understanding the project objectives and requirements.

Once you have specified the problem from a business perspective, you can formulate it as a machine learning problem and develop a preliminary implementation plan. Identify success criteria to determine if the machine learning results meet the business goals defined. For example, your business problem might be: "How can I sell more of my product to customers?" You might translate this into a machine learning problem such as: "Which customers are most likely to purchase the product?" A model that predicts who is most likely to purchase the product is typically built on data that describes the customers who have purchased the product in the past.

To summarize, in this phase, you will:

- Specify objectives
- Determine machine learning goals
- Define success criteria
- Produce project plan

Understand Data

The data understanding phase involves data collection and exploration which includes loading the data and analyzing the data for your business problem.

Assess the various data sources and formats. Load data into appropriate data management tools, such as Oracle Database. Explore relationships in data so it can be properly integrated. Query and visualize the data to address specific data mining questions such as distribution of attributes, relationship between pairs or small number of attributes, and perform simple statistical analysis. As you take a closer look at the data, you can determine how well it can be used to addresses the business problem. You can then decide to remove some of the data or add additional data. This is also the time to identify data quality problems such as:

- Is the data complete?
- Are there missing values in the data?
- What types of errors exist in the data and how can they be corrected?

To summarize, in this phase, you will:

- Access and collect data
- Explore data
- Assess data quality

Prepare Data

The preparation phase involves finalizing the data and covers all the tasks involved in making the data in a format that you can use to build the model.

Data preparation tasks are likely to be performed multiple times, iteratively, and not in any prescribed order. Tasks can include column (attributes) selection as well as selection of rows in a table. You may create views to join data or materialize data as required, especially if data is collected from various sources. To cleanse the data, look for invalid values, foreign key values that don't exist in other tables, and missing and outlier values. To refine the data, you can apply transformations such as aggregations, normalization, generalization, and attribute constructions needed to address the machine learning problem. For example, you can transform a `DATE_OF_BIRTH` column to `AGE`; you can insert the median income in cases where the `INCOME` column is null; you can filter out rows representing outliers in the data or filter columns that have too many missing or identical values.

Additionally you can add new computed attributes in an effort to tease information closer to the surface of the data. This process is referred as *Feature Engineering*. For example, rather than using the purchase amount, you can create a new attribute:

"Number of Times Purchase Amount Exceeds $500 in a 12 month time period." Customers who frequently make large purchases can also be related to customers who respond or don't respond to an offer.

Thoughtful data preparation and feature engineering that capture domain knowledge can significantly improve the patterns discovered through machine learning. Enabling the data professional to perform data assembly, data preparation, data transformations, and feature engineering inside the Oracle Database is a significant distinction for Oracle.

> **Note:**
>
> Oracle Machine Learning supports Automatic Data Preparation (ADP), which greatly simplifies the process of data preparation.

To summarize, in this phase, you will:

- Clean, join, and select data
- Transform data
- Engineer new features

Related Topics

- *Oracle Machine Learning for SQL User's Guide*

Develop Models

In this phase, you select and apply various modeling techniques and tune the algorithm parameters, called *hyperparameters*, to desired values.

If the algorithm requires specific data transformations, then you need to step back to the previous phase to apply them to the data. For example, some algorithms allow only numeric columns such that string categorical data must be "exploded" using one-hot encoding prior to modeling. In preliminary model building, it often makes sense to start with a sample of the data since the full data set might contain millions or billions of rows. Getting a feel for how a given algorithm performs on a subset of data can help identify data quality issues and algorithm setting issues sooner in the process reducing time-to-initial-results and compute costs. For supervised learning problem, data is typically split into train (build) and test data sets using an 80-20% or 60-40% distribution. After splitting the data, build the model with the desired model settings. Use default settings or customize by changing the model setting values. Settings can be specified through OML's PL/SQL, R and Python APIs. Evaluate model quality through metrics appropriate for the technique. For example, use a confusion matrix, precision, and recall for classification models; RMSE for regression models; cluster similarity metrics for clustering models and so on.

Automated Machine Learning (AutoML) features may also be employed to streamline the iterative modeling process, including algorithm selection, attribute (feature) selection, and model tuning and selection.

To summarize, in this phase, you will:

- Explore different algorithms
- Build, evaluate, and tune models

Related Topics

- *Oracle Machine Learning for SQL User's Guide*

Evaluate

At this stage of the project, it is time to evaluate how well the model satisfies the originally-stated business goal.

During this stage, you will determine how well the model meets your business objectives and success criteria. If the model is supposed to predict customers who are likely to purchase a product, then does it sufficiently differentiate between the two classes? Is there sufficient lift? Are the trade-offs shown in the confusion matrix acceptable? Can the model be improved by adding text data? Should transactional data such as purchases (market-basket data) be included? Should costs associated with false positives or false negatives be incorporated into the model?

It is useful to perform a thorough review of the process and determine if important tasks and steps are not overlooked. This step acts as a quality check based on which you can determine the next steps such as deploying the project or initiate further iterations, or test the project in a pre-production environment if the constraints permit.

To summarize, in this phase, you will:

- Review business objectives
- Assess results against success criteria
- Determine next steps

Deploy

Deployment is the use of machine learning within a target environment. In the deployment phase, one can derive data driven insights and actionable information.

Deployment can involve scoring (applying a model to new data), extracting model details (for example the rules of a decision tree), or integrating machine learning models within applications, data warehouse infrastructure, or query and reporting tools.

Because Oracle Machine Learning builds and applies machine learning models inside Oracle Database, the results are immediately available. Reporting tools and dashboards can easily display the results of machine learning. Additionally, machine learning supports scoring single cases or records at a time with dynamic, batch, or real-time scoring. Data can be scored and the results returned within a single database transaction. For example, a sales representative can run a model that predicts the likelihood of fraud within the context of an online sales transaction.

To summarize, in this phase, you will:

- Plan enterprise deployment
- Integrate models with application for business needs
- Monitor, refresh, retire, and archive models
- Report on model effectiveness

Related Topics

- *Oracle Machine Learning for SQL User's Guide*

Machine Learning Functions

Machine learning problems are categorized into mining functions. Each machine learning function specifies a class of problems that can be modeled and solved. Machine learning functions fall generally into two categories - supervised and unsupervised. Notions of supervised and unsupervised learning are derived from the science of machine learning, which is a sub-area of data science.

Algorithms

An algorithm is a mathematical procedure for solving a specific kind of problem. For some machine learning functions, you can choose among several algorithms.

Each algorithm produces a specific type of model, with different characteristics. Some machine learning problems can best be solved by using more than one algorithm in combination. For example, you might first use a feature extraction model to create an optimized set of predictors, then a classification model to make a prediction on the results.

Supervised Learning

Supervised learning is also known as directed learning. The learning process is directed by a previously known dependent attribute or target.

Supervised machine learning attempts to explain the behavior of the target as a function of a set of independent attributes or predictors. Supervised learning generally results in predictive models.

The building of a supervised model involves training, a process whereby the software analyzes many cases where the target value is already known. In the training process, the model "learns" the patterns in the data that enable making predictions. For example, a model that seeks to identify the customers who are likely to respond to a promotion must be trained by analyzing the characteristics of many customers who are known to have responded or not responded to a promotion in the past.

Oracle Machine Learning supports the following supervised machine learning functions:

Table 1-1 Supervised Machine Learning Functions

Function	Description	Sample Problem	Supported Algorithms
Attribute Importance	Identifies the attributes that are most important in predicting a target attribute	Given customer response to an affinity card program, find the most significant predictors	• CUR Decomposition • Expectation Maximization • Minimum Description Length

Table 1-1 (Cont.) Supervised Machine Learning Functions

Function	Description	Sample Problem	Supported Algorithms
Classification	Assigns items to discrete classes and predicts the class to which an item belongs	Given demographic data about a set of customers, predict customer response to an affinity card program	• Decision Tree • Explicit Semantic Analysis • Extreme Gradient Boosting • Generalized Linear Model • Naive Bayes • Neural Network • Random Forest • Support Vector Machine
Regression	Approximates and forecasts continuous values	Given demographic and purchasing data about a set of customers, predict customers' age	• Extreme Gradient Boosting • Generalized Linear Model • Neural Network • Support Vector Machine
Ranking	Predicts the probability of one item over other items	Recommend products to online customers based on their browsing history	Extreme Gradient Boosting
Time Series	Forecasts target value based on known history of target values taken at equally spaced points in time	Predict the length of the ocean waves, address tactical issues such as projecting costs, inventory requirements and customer satisfaction, and so on.	Exponential Smoothing

Splitting the Data

Separate data sets are required for building (training) and testing some predictive models. Typically, one large table or view is split into two data sets: one for building the model, and the other for testing the model.

The build data (training data) and test data must have the same column structure. The process of applying the model to test data helps to determine whether the model, built on one chosen sample, is generalizable to other data.

Unsupervised Learning

Unsupervised learning is non-directed. There is no distinction between dependent and independent attributes. There is no previously-known result to guide the algorithm in building the model.

Unsupervised learning can be used for descriptive purposes. In unsupervised learning, the goal is pattern detection. It can also be used to make predictions.

Oracle Machine Learning supports the following unsupervised machine learning functions:

Table 1-2 Unsupervised Machine Learning Functions

Function	Description	Sample Problem	Supported Algorithms
Anomaly Detection	Identifies rows (cases, examples) that do not satisfy the characteristics of "normal" data	Given demographic data about a set of customers, identify which customer purchasing behaviors are unusual in the dataset, which may be indicative of fraud.	• One-Class SVM • MSET-SPRT
Association Rules	Finds items that tend to co-occur in the data and specifies the rules that govern their co-occurrence	Find the items that tend to be purchased together and specify their relationship	Apriori
Clustering	Finds natural groupings in the data	Segment demographic data into clusters and rank the probability that an individual belongs to a given cluster	• Expectation Maximization • K-Means • Orthogonal Partitioning
Feature Extraction	Creates new attributes (features) using linear combinations of the original attributes	Given demographic data about a set of customers, transform the original attributes into fewer new attributes.	• Explicit Semantic Analysis • Non-negative Matrix Factorization • Principle Component Analysis • Singular Value Decomposition
Row Importance	Row importance technique is used in dimensionality reduction of large data sets. Row importance identifies the most influential rows of the data set.	Given a data set, select rows that meet a minimum importance value prior to model building.	CUR Decomposition

2 Get Started

- Install Database On-premises
- Install SQL Developer
- Access Autonomous Database
- Access OML Notebooks

Install Database On-premises

You can download the latest database version on your system and use clients like Oracle SQL Developer to connect to the Oracle database.

About Installation

Oracle Machine Learning for SQL is a component of the Oracle Database Enterprise Edition.

To install Oracle Database, follow the installation instructions for your platform. Choose a Data Warehousing configuration during the installation.

Oracle Data Miner, the graphical user interface to Oracle Machine Learning for SQL, is an extension to Oracle SQL Developer. Instructions for downloading SQL Developer and installing the Data Miner repository are available on https://www.oracle.com/database/technologies/odmrinstallation.html.

To perform machine learning activities, you must be able to log on to the Oracle Database, and your user ID must have the database privileges described in Grant Privileges for Oracle Machine Learning for SQL.

Related Topics

- `Oracle Data Miner`

> **See Also:**
>
> **Install and Upgrade** page of the Oracle Database online documentation library for your platform-specific installation instructions: `Oracle Database 21c Release`

Install SQL Developer

Oracle SQL Developer is a free, integrated development environment that simplifies the development and management of Oracle Database in both traditional and Cloud deployments.

About SQL Developer

Oracle SQL Developer is a graphical version of SQL*Plus that gives database developers a convenient way to perform basic tasks. You can browse, create, edit, and delete (drop); run SQL statements and scripts; edit and debug PL/SQL code; manipulate and export (unload) data; and view and create reports.

You can connect to any target Oracle Database schema using standard Oracle Database authentication. Once connected, you can perform operations on objects in the database.

You can connect to schemas for MySQL and selected third-party (non-Oracle) databases, such as Microsoft SQL Server, Sybase Adaptive Server, and IBM DB2, and view metadata and data in these databases; and you can migrate these databases to Oracle Database.

Install and Get Started with SQL Developer

To install and start SQL Developer, download a ZIP file and unzip it into the desired parent directory on your system or folder and then type a command or double-click a file name.

If Oracle Database (Release 11 or later) is also installed, a version of SQL Developer is also included and is accessible through the menu system under Oracle. This version of SQL Developer is separate from any SQL Developer kit that you download and unzip on your own, so do not confuse the two, and do not unzip a kit over the SQL Developer files that are included with Oracle Database.

> **Tip:**
>
> Create a shortcut for the SQL Developer executable file that you install, and use it to start SQL Developer.

1. Unzip the SQL Developer kit into a folder (directory) of your choice, which will be referred to as `<sqldeveloper_install>`. Unzipping the SQL Developer kit causes a folder named `sqldeveloper` to be created under the `<sqldeveloper_install>` folder.
   For example, if you unzip the kit into `C:\`, the folder `C:\sqldeveloper` is created, along with several sub-folders under it.
2. To start SQL Developer, go to the `sqldeveloper` directory under the `<sqldeveloper_install>` directory, and do one of the following:
   On Linux and Mac OS X systems, run `sh sqldeveloper.sh`.

   On Windows systems, double-click `sqldeveloper.exe`.

   If you are asked to enter the full pathname for the JDK, click **Browse** and find it. For example, on a Windows system the path might have a name similar to `C:\Program Files\Java\jdk1.7.0_51`. (If you cannot start SQL Developer, it could be due to an error in specifying or configuring the JDK.)
3. Create at least one database connection (or import some previously exported connections), so that you can view and work with database objects, use the SQL Worksheet, and use other features.

To create a new database connection:

a. Right-click the **Connections** node in the **Connections** navigator
b. Select **New Connection**, and complete the required entries in the Create/Edit/Select Database Connection dialog box. (You may also be able to generate connections automatically by right-clicking the Connections node and selecting Create Local Connections.)

Related Topics

- Database Connections

Access Autonomous Database

Oracle Autonomous Database is a family of self-driving, self-securing, and self-repairing cloud services. You can sign up for an Oracle Cloud Free Tier account and create a database instance.

Provision an Autonomous Database

This is a set of tutorials (as known as a workshop) that teaches you to access Oracle public cloud services. The workshop aims to provision an Autonomous Database instance on Shared infrastructure.

Provision an Autonomous Database

Create and Update User Accounts for Oracle Machine Learning Notebooks

An administrator can add an existing database user account to Oracle Machine Learning Notebooks or create a new user account and user credentials with the Oracle Machine Learning User Management interface.

Topics

- Create User
- Add Existing Database User Account to Oracle Machine Learning Notebooks

Create User

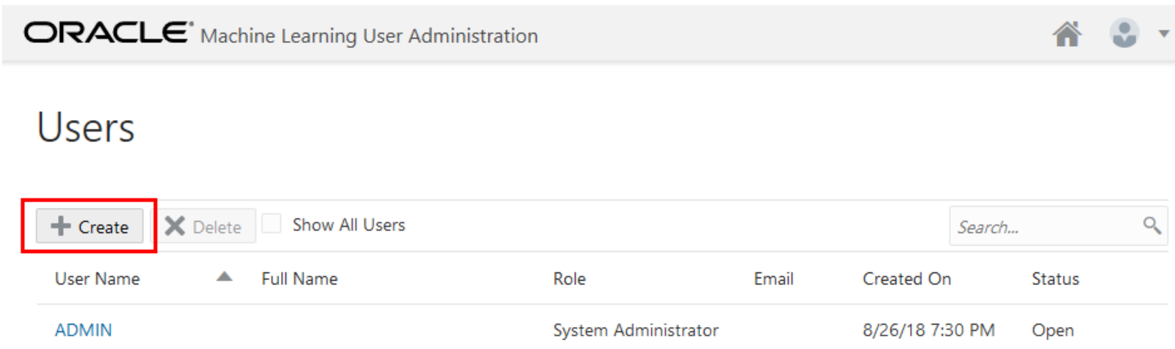
An administrator creates a new user account and user credentials for Oracle Machine Learning in the User Management interface.

> **Note:**
>
> You must have the administrator role to access the Oracle Machine Learning User Management interface.

To create a user account:

1. On the Autonomous Databases page, under the **Display Name** column, select an Autonomous Database.

2. On the Autonomous Database Details page, click **Service Console**.
3. On the Service Console click **Administration**.
4. Click **Manage OML Users** to open the Oracle Machine Learning User Administration page.
5. Click **Create** on the Oracle Machine Learning User Administration page.



6. In the **Username** field, enter a username for the account. Using the username, the user will log in to an Oracle Machine Learning instance.
7. Enter a name in the **First Name** field.
8. Enter a name in the **Last Name** field.
9. In the **Email Address** field, enter the email ID of the user.
10. Select the option **Generate password and email account details to user. User will be required to reset the password on first sign in.** to auto generate a temporary password and send an email with the account credentials to the user.

    If you select this option, you need not enter values in the **Password** and **Confirm Password** fields; the fields are grayed out.
11. In the **Password** field, enter a password for the user, if you choose to create a password for the user.

    This option is disabled if you select the **Generate password...** option to auto generate a temporary password for the user.
12. In the **Confirm Password** field, enter a password to confirm the value that you entered in the **Password** field.

    By doing so, you create the password for the user. The user can change the password when first logging in.
13. Click **Create**.

This creates a new database user and grants the required privileges to use Oracle Machine Learning.

> **Note:**
>
> With a new database user, an administrator needs to issue grant commands on the database to grant table access to the new user for the tables associated with the user's Oracle Machine Learning notebooks.

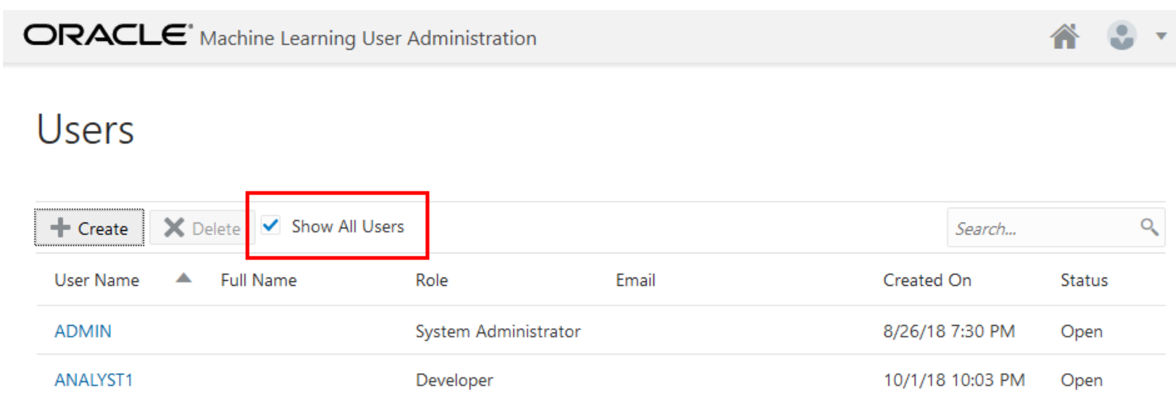
Add Existing Database User Account to Oracle Machine Learning Notebooks

An administrator adds an existing database user account for Oracle Machine Learning Notebooks in the User Management interface.

> **Note:**
>
> You must have the administrator role to access the Oracle Machine Learning User Management interface.

To add an existing database user account:

1. On the Autonomous Databases page, under the **Display Name** column, select an Autonomous Database.
2. On the Autonomous Database Details page, click **Service Console**.
3. On the Service Console, click **Administration**.
4. Click **Manage OML Users** to add Oracle Machine Learning Notebooks users.
5. Click **Show All Users** to display the existing database users.



> **Note:**
>
> Initially, the **Role** field shows the role **None** for existing database users. After adding a user the role **Developer** is assigned to the user.

6. Select a user. To select a user select a name in the **User Name** column. For example, select **ANALYST1**.

   Selecting the user shows the Oracle Machine Learning **Edit User** page.
7. Enter a name in the **First Name** field. (Optional)
8. Enter the last name of the user in the **Last Name** field. (Optional)
9. In the **Email Address** field, enter the email ID of the user.

Making any change on this page adds the existing database user with the required privileges as an Oracle Machine Learning Notebooks user.

10. Click **Save**.

This grants the required privileges to use the Oracle Machine Learning application. In Oracle Machine Learning this user can then access any tables the user has privileges to access in the database.

Access OML Notebooks

To perform Oracle Machine Learning tasks, you can access Oracle Machine Learning Notebooks from Autonomous Database

Access Oracle Machine Learning Notebooks

You can access Oracle Machine Learning **Notebooks** from Autonomous Database.

To access Oracle Machine Learning Notebooks from the Autonomous Database Service Console:

1. Select an Autonomous Database instance and on the Autonomous Database details page click **Service Console**.
2. Click **Development**.
3. On the Development page click **Oracle Machine Learning Notebooks**.
4. Enter your username and password.
5. Click **Sign In**.

This shows the Oracle Machine Learning user application.

Create a Notebook

A notebook is a web-based interface for data analysis, data discovery, data visualization and collaboration.

Whenever you create a notebook, it has an interpreter settings specification. The notebook contains an internal list of bindings that determines the order of the interpreter bindings. A notebook comprises paragraphs which is a notebook component where you can write SQL statements, run PL/SQL scripts, and run Python commands. A paragraph has an input section and an output section. In the input section, specify the interpreter to run along with the text. This information is sent to the interpreter to be executed. In the output section, the results of the interpreter are provided.

To create a notebook:

1. In the Oracle Machine Learning home page, click **Notebooks.** The Notebooks page opens.
2. In the Notebooks page, click **Create.**

   The Create Notebook window appears.
3. In the **Name** field, provide a name for the notebook.
4. In the **Comments** field, enter comments, if any.

5. In the **Connections** field, select a connection in the drop-down list. By default, the Global Connection Group is assigned.
6. Click **OK.**

Your notebook is created and it opens in the notebook editor. You can now use it to run SQL statements, run PL/SQL scripts, and run Python commands. To do so, specify any one of the following directives in the input section of the paragraph:

- `%sql` - To call the SQL interpreter and run SQL statements
- `%script` - To call PL/SQL interpreter and run PL/SQL scripts
- `%md` - To call the Markdown interpreter and generate static html from Markdown plain text
- `%python` - To call the Python interpreter run Python scripts

Edit Your Notebook

Upon creating a notebook, it opens automatically, presenting you with a single paragraph using the default `%sql` interpreter. You can change the interpreter by explicitly specifying one of `%script, %python,` or `%sql.`

Set the context with a project with which your notebook is associated.

You can edit an existing notebook in your project. To edit an existing notebook:

1. In Oracle Machine Learning home page, select the project in which your notebook is available.
2. Go to the Oracle Machine Learning navigator, and select **Notebooks.** Alternatively, you can click the **Notebooks** quick link in the home page.

   In the right pane, all notebooks that are available in the project are listed.
3. Click the notebook that you want to open and edit.

   The selected notebook opens in edit mode.
4. In edit mode, you can perform the following tasks:
   - Write code to fetch data
   - Run paragraphs. Click the run icon ▷ to run one or all paragraphs in the notebook.
   - Export notebooks. Click the export icon to export the notebook.
   - Set order for interpreter bindings. Click the gear icon to set the order for interpreter bindings for the notebook.
   - View list of shortcut keys. Click the keyboard shortcut icon to view the list of keyboard shortcuts.
   - Add dynamic forms such as the Text Input form, Select form, Check box form for easy selection of inputs and easy filtering of data in your notebook. Oracle Machine Learning supports the following Apache Zeppelin dynamic forms:
     – Text Input form — Allows you to create a simple form for text input.
     – Select form — Allows you to create a form containing a range of values that the user can select.
     – Check Box form — Allows you to insert check boxes for multiple selection of inputs.

> **Note:**
>
> The Apache Zeppelin dynamic forms are supported only on SQL interpreter notebooks.

5. Once you have finished editing the notebook, click **Back.**

   This takes you back to the Notebook page.

3
Use cases

- Regression
- Classification
- Clustering
- Time Series
- Association Rules

Regression

A real estate agent has approached you to help evaluate house prices in Boston. The agent wants this data daily to offer targeted services to clients. You are building a predictive model to estimate the median value of owner-occupied homes in the Boston area using Regression. You are using the Generalized Linear Model algorithm for this use case.

Before you start your OML4SQL use case journey, ensure that you have the following:

- Data set
  Download the data set from https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/datasets/data/boston_house_prices.csv.

  > **Note:**
  >
  > This data set is used for illustrative purpose only.

- Database
  Select a database out of the following options:
  - Get your FREE cloud account. Go to https://cloud.oracle.com/database and select Oracle Database Cloud Service (DBCS), or Oracle Autonomous Database. Create an account and create an instance. See Autonomous Database Quick Start Workshop.
  - Download the latest version of Oracle Database (on premises).
- Machine Learning Tools
  Depending on your database selection,
  - Use OML Notebooks for Oracle Autonomous Database.
  - Install and use Oracle SQL Developer connected to an on-premises database or DBCS. See Installing and Getting Started with SQL Developer.
- Other Requirements
  Data Mining Privileges (this is automatically set for ADW). See System Privileges for Oracle Machine Learning for SQL.

Load Data

Examine the data set and its attributes. Load the data in your database.

In this use case, you will modify the data set to add a column and upload the data set to your database. If you are using the Oracle Autonomous Database, you will upload files to the Oracle Cloud Infrastructure (OCI) Object Storage, create a sample table, load data into the sample table from files on the OCI Object Storage, and explore the data. If you are using the on-premises database, you will use Oracle SQL developer to import the data set and explore the data.

Examine Data

The data set contains 14 attributes. The following table displays information about the data attributes:

Attribute Name	Information
CRIM	Per capita crime rate by town
ZN	The proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	The proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	Nitric oxides concentration (parts per 10 million)
RM	The average number of rooms per dwelling
AGE	The proportion of owner-occupied units built before 1940
DIS	Weighted distances to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	The pupil-teacher ratio by town
B	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT	% lower status of the population
MEDV	The median value of owner-occupied homes in $1000's

Related Topics

- How ADP Transforms the Data

Add a Column

In this data set, no row identifier uniquely identifies each record in the data set. Add a new `case_id` column. The `case_id` assists with reproducible results, joining scores for individual customers with other data in, example, scoring data table.

Add a column called House ID (HID). The HID value is added as a primary key to the table so that identifying and retrieving each record is simple. Each record in the database is called a case and each case is identified by a `case_id`. Here, *HID* is the `case_id`.

To add the HID column:

1. Open the .csv file in a spreadsheet.

2. Delete the first row with 506 and 13. Now, the row with the column names becomes the first row.
3. To the left of the data set, add a column.
4. Enter *HID* as the column name.
5. In the *HID* column enter 1 as the first value identifying the first row.
6. You will see a + icon in the spreadsheet cell. Drag the + icon right to the bottom till the end of the records.
7. Right-click and select **Fill Series**.

Import Data

There are various methods to import data into the database. Two methods are explained here. One using SQL Developer (for on-premises) and the other using Object Storage (for Cloud).

Import Data into the Database (On premises)

To access the data set, import the modified data set into the database using SQL Developer.

The following steps help you to import the data set into an on premises database.

(Optional) Enter task prerequisites here.

1. Launch SQL Developer on your system.
2. Import the modified .csv file. See Tables.
3. Set House ID (HID) as a primary key. This column identifies each record and helps in retrieving information about a specific record. The HID column helps when you join tables or views. See Primary Key Constraint.

You are now ready to query the table in SQL Developer.

Import Data to the Cloud

If you are using a cloud account, one of the methods of importing the data is through Object Storage. Upload the data set to an Object Storage. The Object Storage URI will be used in another procedure.

You can load data into your Oracle Autonomous Database (Autonomous Data Warehouse [ADW] or Autonomous Transaction Processing [ATP]) using Oracle Database tools, and Oracle and 3rd party data integration tools. You can load data:

- from local files in your client computer, or
- from files stored in a cloud-based object store

Follow the steps to upload your data file to the Object Storage bucket.

1. Login to your cloud account.
2. Click the left-side hamburger menu and select **Object Storage** from the menu.
3. Select **Object Storage** from the displayed options.
4. Select the compartment in which you want to upload the data.
5. Click **Create Bucket**.

6. Enter a name for your bucket. For example, Bucket1. Leave the rest of the fields as default.
7. Click **Create Bucket**.
8. Click on the bucket that you created. Scroll down and click **Upload** under Objects.
9. Leave the OBJECT NAME PREFIX field black. Click **select files** to navigate to the data file that you want to upload or drag and drop the data file.
10. Click **Upload**. The data file appears under Objects.
11. Click the ellipses on the right side of the data file to view the menu. Click **View Object Details**.
12. Copy the URL PATH (URI) to a text file. This URI is used in the `DBMS_CLOUD.COPY_DATA` procedure.

This procedure creates an object storage containing the data file in your cloud account.

Create Auth Token

The Auth Token is required in the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure. You can generate the Auth Token in your cloud account.

1. Login into your ADW Cloud account.
2. Hover your mouse cursor over the human figure icon at the top right of the console and click **User Settings** from the drop-down menu.
3. Click **Auth Tokens** under Resources on the left of the console.
4. Click **Generate Token**. A pop-up dialog appears.
5. Enter a description (optional).
6. Click **Generate Token**.
7. Copy the generated token to a text file. The token does not appear again.
8. Click **Close**.

Create Object Storage Credential

The object storage credential is used in the `DBMS_CLOUD.COPY_DATA` procedure.

1. Login to the OML Notebooks page.
2. Open the notebook that you just created.
3. Enter the following query to create credentials:

```
%script
begin
  DBMS_CLOUD.create_credential (
    credential_name => 'CRED',
    username => '<your cloud account username>',
    password => '<your Auth Token>'
  ) ;
end;
/
```

Examine the query:

- `credential_name`: The name of the credential to be stored. Provide any name.
- `username`: This is your cloud account username.
- `password`: Enter your Auth Token password that you copied after generating the Auth Token.

4. Run the query in your notebook. Your credentials are stored in the ADW user schema.
5. In another para, run the following query to check the user credentials:

```
SELECT* FROM USER_CREDENTIALS;
```

Create a Table

Create a table called `BOSTON_HOUSING`. This table is used in `DBMS_CLOUD.COPY_DATA` procedure to access the data set.

Enter the following code in the OML Notebooks and run the notebook.

```
%sql
CREATE table boston_housing
(
 HID NUMBER NOT NULL,
 CRIM NUMBER,
 ZN NUMBER,
 INDUS NUMBER,
 CHAS VARCHAR2(32),
 NOX NUMBER,
 RM NUMBER,
 AGE NUMBER,
 DIS NUMBER,
 RAD NUMBER,
 TAX NUMBER,
 PTRATIO NUMBER,
 B NUMBER,
 LSTAT NUMBER,
 MEDV NUMBER
);
```

Load Data in the Table

Load the data set stored in object storage to the `BOSTON_HOUSING` table.

Add a new para in the OML Notebooks and enter the following statement:

```
%script
BEGIN
 DBMS_CLOUD.COPY_DATA(
    table_name =>'BOSTON_HOUSING',
    credential_name =>'CRED',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/channels.txt',
    format => json_object('type' value 'CSV', 'skipheaders' value 1)
```

```
 );
END;
```

Examine the statement:

- `table_name`: is the target table's name.
- `credential_name`: is the name of the credential created earlier.
- `file_uri_list`: is a comma delimited list of the source files you want to load.
- `format`: defines the options you can specify to describe the format of the source file, including whether the file is of type text, ORC, Parquet, or Avro.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name.

Related Topics

- DBMS_CLOUD.COPY_DATA Procedure

Explore Data

Explore the data to understand and assess the quality of the data. At this stage assess the data to identify data types and noise in the data. Look for missing values and numeric outlier values.

The following steps help you with the exploratory analysis of the data:

1. View the data in the `BOSTON_HOUSING` table by running the following query:

   ```
   SELECT * FROM BOSTON_HOUSING
   ORDER BY HID;
   ```

2. Since you created the table specifying each column's datatype, you already know the datatype. However, to view the datatype of the columns, run the following script:

   ```
   %script
   DESCRIBE BOSTON_HOUSING;
   ```

   ```
   Name    Null?    Type
   ------- -------- ------------
   HID  NOT NULL NUMBER
   CRIM          NUMBER
   ZN            NUMBER
   INDUS         NUMBER
   CHAS          VARCHAR2(32)
   NOX           NUMBER
   RM            NUMBER
   AGE           NUMBER
   DIS           NUMBER
   RAD           NUMBER(38)
   TAX           NUMBER
   PTRATIO       NUMBER
   B             NUMBER
   LSTAT         NUMBER
   ```

```
MEDV           NUMBER

---------------------------
```

3. Find the `COUNT` of the dataset to know how many rows are present.

```
SELECT COUNT (*) from BOSTON_HOUSING;



COUNT(*)
      506
---------------------------
```

4. To check if there are any missing values (NULL values), run the following query:

```
SELECT COUNT(*) FROM BOSTON_HOUSING WHERE PTRATIO=NULL OR CHAS=NULL OR
 LSTAT=NULL OR TAX=NULL OR CRIM=NULL OR MEDV=NULL OR ZN=NULL OR NOX=NULL
 OR AGE=NULL OR INDUS=NULL OR DIS=NULL OR B=NULL OR RAD=NULL OR PTRATIO=NULL OR
RM=NULL;



COUNT(*)
      0
---------------------------
```

NULLs, if found, are automatically handled by the OML algorithms. Alternately, you can manually replace NULLs with `NVL` SQL function.

5. To list the distinct values for the categorical column CHAS and the number of records for each distinct value of CHAS, run the following query:

```
%sql
SELECT CHAS, COUNT(1)
FROM BOSTON_HOUSING
GROUP BY CHAS;



CHAS   COUNT(1)
0             471
1              35
---------------------------
```

6. To calculate mean, median, min, max, and interquartile range (IQR) create a view called `unpivoted`.

The IQR describes the middle 50% of values (also called the mid spread or the H spread) when ordered from lowest to highest. To find the IQR, first, find the median (middle value) of the lower and upper half of the data. These values are quartile 1 (Q1) and quartile 3 (Q3). The IQR is the difference between Q3 and Q1. Sometimes, this assessment is helpful to find outliers in the data.

```
%sql
create or replace view unpivoted as
select *
  from (
```

```
SELECT 'CRIM' COL, ROUND(MIN(CRIM),2) MIN_VAL, PERCENTILE_CONT(0.25) WITHIN
GROUP (ORDER BY CRIM) FIRST_QUANTILE, ROUND(AVG(CRIM),2) MEAN_VAL,
ROUND(MEDIAN(CRIM),2) MEDIAN_VAL, PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER
BY CRIM) THIRD_QUANTILE, ROUND(MAX(CRIM),2) MAX_VAL
FROM BOSTON_HOUSING
UNION
SELECT 'AGE' COL, ROUND(MIN(AGE),2) MIN_VAL,  PERCENTILE_CONT(0.25) WITHIN
GROUP (ORDER BY AGE) FIRST_QUANTILE, ROUND(AVG(AGE),2) MEAN_VAL,
ROUND(MEDIAN(AGE),2) MEDIAN_VAL, PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER
BY AGE) THIRD_QUANTILE, ROUND(MAX(AGE),2) MAX_VAL
FROM BOSTON_HOUSING
UNION
SELECT 'DIS' COL, ROUND(MIN(DIS),2) MIN_VAL,  PERCENTILE_CONT(0.25) WITHIN
GROUP (ORDER BY DIS) FIRST_QUANTILE, ROUND(AVG(DIS),2) MEAN_VAL,
ROUND(MEDIAN(DIS),2) MEDIAN_VAL, PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER
BY DIS) THIRD_QUANTILE, ROUND(MAX(DIS),2) MAX_VAL
FROM BOSTON_HOUSING
  ) a
unpivot
(
  VALUE
    for stat in ("MIN_VAL", "FIRST_QUANTILE", "MEAN_VAL","MEDIAN_VAL",
"THIRD_QUANTILE", "MAX_VAL")
);
```

7. To view the values, pivot the table by running the following query:

```
%sql
select *
  from unpivoted
pivot(
  SUM(VALUE)
    for COL in ('CRIM', 'AGE','DIS')
);
```

```
STAT                'CRIM'       'AGE'     'DIS'
MEAN_VAL                 3.61     68.57         3.8
THIRD_QUARTILE      3.6770825    94.075    5.188425
MAX_VAL                 88.98       100       12.13
FIRST_QUARTILE       0.082045    45.025    2.100175
MEDIAN_VAL               0.26      77.5        3.21
MIN_VAL                  0.01       2.9        1.13


6 rows selected.

---------------------------
```

This completes the data understanding and data preparation stage. OML supports Automatic Data Preparation (ADP). ADP is enabled through the model settings. When ADP is enabled, the transformations required by the algorithm are performed automatically and embedded in the model. This step is done during the Build Model stage. The commonly used methods of data preparation are binning, normalization, and missing value treatment.

Related Topics

- How ADP Transforms the Data

Build Model

Build your model using the training data set. Use the `DBMS_DATA_MINING.CREATE_MODEL2` procedure to build your model and specify model settings.

For a supervised learning, like Regression, before creating the model, split the data in to training and test data. Although you can use the entire data set to build a model, it is difficult to validate the model unless there are new data sets available. Therefore, to evaluate the model and to accurately assess the performance of the model on the same data, you generally split or separate the data into training and test data. You use the training data set to train the model and then use the test data set to test the accuracy of the model by running prediction queries. The testing data set already contains known values for the attribute that you want to predict. It is thus easy to determine whether the model's predictions are correct.

Algorithm Selection

Before you build a model, choose the suitable algorithm. You can choose one of the following algorithms to solve a regression problem:

- Extreme Gradient Boosting
- Generalized Linear Model
- Neural Network
- Support Vector Machine

When you want to understand the data set, you always start from a simple and easy baseline model. The Generalized Linear Model algorithm is the right choice because it is simple and easy to interpret since it fits a linear relationship between the feature and the target. You can get an initial understanding of a new data set from the result of the linear model.

The following steps guide you to split your data and build your model with the selected algorithm.

1. Split the data into 80/20 as training and test data. Run the following statement:

   ```
   BEGIN
       EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW TRAINING_DATA AS SELECT * FROM
   BOSTON_HOUSING SAMPLE (80) SEED (1)';
       DBMS_OUTPUT.PUT_LINE ('Created TRAINING_DATA');
       EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW TEST_DATA AS SELECT * FROM
   BOSTON_HOUSING MINUS SELECT * FROM TRAINING_DATA';
       DBMS_OUTPUT.PUT_LINE ('Created TEST_DATA');

   END;
   ```

   After splitting the data, view the count of rows in `TRAINING_DATA` and `TEST_DATA`. You can verify the ratio of the training and test data by checking the number of rows of the training and test set.

2. To find the count of rows in `TRAINING_DATA`, run the following statement:

```
select count(*) from TRAINING_DATA;

COUNT(*)
407
---------------------------
```

3. To find the count of rows from TEST_DATA, run the following statement:

```
select COUNT(*) from TEST_DATA;

COUNT(*)
99
---------------------------
```

4. To find if any rows are not sampled (left out) in both TRAINING_DATA and TEST_DATA, run the following query:

```
SELECT COUNT(1)
FROM TRAINING_DATA train
JOIN TEST_DATA test
ON train.HID = test.HID

COUNT(*)
0
---------------------------
```

5. Build your model using the CREATE_MODEL2 procedure. First, declare a variable to store model settings or hyperparameters. Run the following script:

```
%script
DECLARE
    v_setlst DBMS_DATA_MINING.SETTING_LIST;
    BEGIN
    v_setlst('PREP_AUTO') := 'ON';
    v_setlst('ALGO_NAME') := 'ALGO_GENERALIZED_LINEAR_MODEL';
    v_setlst('GLMS_DIAGNOSTICS_TABLE_NAME') := 'GLMR_DIAG';
    v_setlst('GLMS_FTR_SELECTION') := 'GLMS_FTR_SELECTION_ENABLE';
    v_setlst('GLMS_FTR_GENERATION') := 'GLMS_FTR_GENERATION_ENABLE';

    DBMS_DATA_MINING.CREATE_MODEL2(
      MODEL_NAME          =>  'GLMR_REGR',
      MINING_FUNCTION     => 'REGRESSION'
      DATA_QUERY          =>  'SELECT * FROM TRAINING_DATA',
      SET_LIST            =>  v_setlst,
      CASE_ID_COLUMN_NAME =>  'HID',
      TARGET_COLUMN_NAME  =>  'MEDV');
END;
```

Examine the script:

- v_setlst is a variable to store SETTING_LIST.
- SETTING_LIST defines model settings or hyperparameters for your model.
- DBMS_DATA_MINING is the PL/SQL package used for Oracle Machine Learning. These settings are described in DBMS_DATA_MINING - Model Settings.
- ALGO_NAME specifies the algorithm name. Since you are using the Generalized Linear Model as your algorithm, set ALGO_GENERALIZED_LINEAR_MODEL.

- PREP_AUTO is the setting used for Automatic Data Preparation. Here, enable Automatic Data Preparation. The value of the setting is ON.
- GLMS_DIAGNOSTICS_TABLE_NAME generates per-row statistics if you specify the name of a diagnostics table in the setting. The value of the setting is GLMR_DIAG.
- GLMS_FTR_SELECTION indicates feature selection. The value GLMS_FTR_SELECTION_ENABLE indicates that feature selection is enabled. Feature selection selects columns that are most important in predicting a target attribute. If feature selection is not selected, then all the columns are considered for analysis which may not give accurate results.
- GLMS_FTR_GENERATION indicates feature generation. The value GLMS_FTR_GENERATION_ENABLE indicates that the feature generation is enabled. Feature generation generates new features from existing features which might be useful in our analysis.

The CREATE_MODEL2 procedure has the following parameters:

- MODEL_NAME: A unique model name that you want to give to your model. The name of the model is in the form [schema_name.]model_name. If you do not specify a schema, then your own schema is used. Here, the model name is GLMR_REGR
- MINING_FUNCTION: Specifies the machine learning function. Since you are solving a linear regression problem, in this use case, select REGRESSION.
- DATA_QUERY: A query that provides training data for building the model. Here, the query is SELECT * FROM TRAINING_DATA.
- SET_LIST: Specifies SETTING_LIST.
- CASE_ID_COLUMN_NAME: A unique case identifier column in the training data. In this use case, case_id is HID. If there is a composite key, you must create a new attribute before creating the model. The CASE_ID assists with reproducible results, joining scores for individual customers with other data in, example, scoring data table.
- TARGET_COLUMN_NAME: Specifies the column that needs to be predicted. Also referred to as the target variable of the model. In this use case, you are predicting MEDV value.

> **Note:**
>
> Any parameters or settings not specified are either system-determined or default values are used.

Evaluate

Evaluate your model by viewing diagnostic metrics and performing quality checks.

You can inspect various rules to see if they reveal new insights for item dependencies (antecedent itemset implying consequent) or for unexpected relationships among items.

Dictionary and Model Views

To obtain information about the model and view model settings, you can query data dictionary views and model detail views. Specific views in model detail views display model statistics which can help you evaluate the model.

The data dictionary views for Oracle Machine Learning are listed in the following table. A database administrator (DBA) and USER versions of the views are also available.

View Name	Description
ALL_MINING_MODELS	Provides information about all accessible machine learning models
ALL_MINING_MODEL_ATTRIBUTES	Provides information about the attributes of all accessible machine learning models
ALL_MINING_MODEL_SETTINGS	Provides information about the configuration settings for all accessible machine learning models
ALL_MINING_MODEL_VIEWS	Provides information about the model views for all accessible machine learning models
ALL_MINING_MODEL_XFORMS	Provides the user-specified transformations embedded in all accessible machine learning models.

Model detail views are specific to the algorithm. You can obtain more insights about the model you created by viewing the model detail views. The names of model detail views begin with DM$xx where xx corresponds to the view prefix. See Model Detail Views.

The following steps help you to view different dictionary views and model detail views.

1. Run the following statement to view the settings in `USER_MINING_MODEL_SETTINGS`:

   ```
   SELECT * FROM USER_MINING_MODEL_SETTINGS WHERE MODEL_NAME='GLMR_REGR';
   ```

   In this statement, you are selecting all the columns available in the `USER_MINING_MODEL_SETTINGS` view where the model name is `GLMR_REGR`.

MODEL_NAME	SETTING_NAME	SETTING_VALUE	SETTING_TYPE
GLMR_REGR	ALGO_NAME	ALGO_GENERALIZED_LINEAR_MODEL	INPUT
GLMR_REGR	PREP_AUTO	ON	INPUT
GLMR_REGR	GLMS_PRUNE_MODEL	GLMS_PRUNE_MODEL_ENABLE	DEFAULT
GLMR_REGR	GLMS_MAX_FEATURES	1000	DEFAULT
GLMR_REGR	GLMS_FTR_GENERATION	GLMS_FTR_GENERATION_ENABLE	INPUT
GLMR_REGR	GLMS_SELECT_BLOCK	GLMS_SELECT_BLOCK_DISABLE	DEFAULT
GLMR_REGR	GLMS_FTR_SEL_CRIT	GLMS_FTR_SEL_ALPHA_INV	DEFAULT
GLMR_REGR	GLMS_CONF_LEVEL	0.95	DEFAULT
GLMR_REGR	ODMS_DETAILS	ODMS_ENABLE	DEFAULT
GLMR_REGR	GLMS_FTR_SELECTION	GLMS_FTR_SELECTION_ENABLE	INPUT
GLMR_REGR	ODMS_MISSING_VALUE_TREATMENT	ODMS_MISSING_VALUE_AUTO	DEFAULT
GLMR_REGR	GLMS_DIAGNOSTICS_TABLE_NAME	GLMR_DIAG	INPUT
GLMR_REGR	ODMS_SAMPLING	ODMS_SAMPLING_DISABLE	DEFAULT

2. Run the following statement to view only the `SETTING_NAME` and `SETTING_VALUE` column from the above table:

   ```
   SELECT SETTING_NAME, SETTING_VALUE FROM USER_MINING_MODEL_SETTINGS
   WHERE MODEL_NAME = 'GLMR_REGR' ORDER BY SETTING_NAME;
   ```

SETTING_NAME	SETTING_VALUE
ALGO_NAME	ALGO_GENERALIZED_LINEAR_MODEL
GLMS_CONF_LEVEL	0.95
GLMS_DIAGNOSTICS_TABLE_NAME	GLMR_DIAG
GLMS_FTR_GENERATION	GLMS_FTR_GENERATION_ENABLE
GLMS_FTR_SELECTION	GLMS_FTR_SELECTION_ENABLE
GLMS_FTR_SEL_CRIT	GLMS_FTR_SEL_ALPHA_INV
GLMS_MAX_FEATURES	1000
GLMS_PRUNE_MODEL	GLMS_PRUNE_MODEL_ENABLE
GLMS_SELECT_BLOCK	GLMS_SELECT_BLOCK_DISABLE
ODMS_DETAILS	ODMS_ENABLE
ODMS_MISSING_VALUE_TREATMENT	ODMS_MISSING_VALUE_AUTO
ODMS_SAMPLING	ODMS_SAMPLING_DISABLE
PREP_AUTO	ON

3. Run the following statement to see attribute information in `USER_MINING_MODEL_ATTRIBUTES` view:

```
SELECT ATTRIBUTE_NAME, ATTRIBUTE_TYPE FROM USER_MINING_MODEL_ATTRIBUTES
WHERE MODEL_NAME = 'GLMR_REGR' ORDER BY ATTRIBUTE_NAME;
```

ATTRIBUTE_NAME	ATTRIBUTE_TYPE
B	NUMERICAL
CHAS	CATEGORICAL
CRIM	NUMERICAL
DIS	NUMERICAL
LSTAT	NUMERICAL
MEDV	NUMERICAL
NOX	NUMERICAL
PTRATIO	NUMERICAL
RM	NUMERICAL

4. Run the following statement to see information on various views in `USER_MINING_MODEL_VIEWS`:

```
SELECT VIEW_NAME, VIEW_TYPE FROM USER_MINING_MODEL_VIEWS WHERE
MODEL_NAME='GLMR_REGR' ORDER BY VIEW_NAME;
```

VIEW_NAME	VIEW_TYPE
DM$VAGLMR_REGR	GLM Regression Row Diagnostics
DM$VDGLMR_REGR	GLM Regression Attribute Diagnostics
DM$VGGLMR_REGR	Global Name-Value Pairs
DM$VNGLMR_REGR	Normalization and Missing Value Handling
DM$VSGLMR_REGR	Computed Settings
DM$VWGLMR_REGR	Model Build Alerts

5. From the table above, query the Global details for linear regression. See Model Detail Views for Generalized Linear Model. Run the following query to see all the columns of the view:

```
SELECT * FROM DM$VGGLMR_REGR;
```

PARTITION_NAME	NAME	NUMERIC_VALUE	STRING_VALUE
	NUM_ROWS	407	
	NUM_PARAMS	27	
	CONVERGED		YES
	VALID_COVARIANCE_MATRIX		YES
	DEPENDENT_MEAN	22.530712530712513	
	ERROR_SUM_SQUARES	3708.5723401860159	
	CORRECTED_TOT_SS	35660.446093366249	
	MODEL_DF	26	

6. From the above table, you can ignore the first column `PARTITION_NAME` and refine the query to display the rest of the columns ordered by name. Run the following statement:

```
SELECT NAME, NUMERIC_VALUE, STRING_VALUE FROM DM$VGGLMR_REGR ORDER
BY NAME;
```

When comparing models, a model with a lower Root Mean Square Error (RMSE) value is better. RMSE, which squares the errors, gives more weight to large errors. When we have a low RMSE value, we can say that our model is good at predicting the target.

NAME	NUMERIC_VALUE	STRING_VALUE
ADJUSTED_R_SQUARE	0.88888762767892038	
AIC	953.30275642811387	
COEFF_VAR	13.86553567824482	
CONVERGED		YES
CORRECTED_TOTAL_DF	406	
CORRECTED_TOT_SS	35660.446093366249	
ERROR_DF	380	
ERROR_MEAN_SQUARE	9.7594008952263582	
ERROR_SUM_SQUARES	3708.5723401860159	
F_VALUE	125.92148170460615	
GMSEP	10.454535107435209	
HOCKING_SP	0.025750398140438939	
J_P	10.406830438644324	
MODEL_DF	26	
MODEL_F_P_VALUE	0	
MODEL_MEAN_SQUARE	1228.9182212761627	
MODEL_SUM_SQUARES	31951.873753180233	
NUM_PARAMS	27	
NUM_ROWS	407	
ROOT_MEAN_SQ	3.12400398450872	
R_SQ	0.89600319832017172	
SBIC	1061.5407124350638	
VALID_COVARIANCE_MATRIX		YES

7. Query the GLM Regression Attributes Diagnostics view.

```
SELECT FEATURE_EXPRESSION, round(COEFFICIENT,6) COEFFICIENT,
round(P_VALUE,4) P_VALUE,
CASE
    when p_value < 0.001 THEN '***'
    when p_value < 0.01 THEN '**'
    when p_value < 0.05 THEN '*'
    when p_value < 0.1 THEN '.'
    else ' '
END AS significance_statement
FROM DM$VDGLMR_REGR ORDER BY FEATURE_EXPRESSION;
```

The columns of the view are described in Model Detail Views for Generalized Linear Model.
Let us examine the statement:

- `round(COEFFICIENT,6) COEFFICIENT`: returns the coefficient rounded to six places to the right of the decimal point.
- `p_value`: provides information about the relationship between a dependent variable and independent variable such that you could decide to accept or reject the null hypothesis. Generally, p_value less than 0.05 means that you can reject the null hypothesis and accept that there is a correlation between the dependent and independent variables with a significant coefficient value.

FEATURE_EXPRESSION	COEFFICIENT	P_VALUE	SIGNIFICANCE_CODE
B	0.005676	0.026	*
CHAS_1	14.476363	0.0003	***
CRIM*CHAS_1	6.172319	0	***
CRIM*DIS*CHAS_1	-2.597874	0	***
DIS	-3.358578	0	***
LSTAT	-14.501822	0	***
LSTAT*LSTAT	0.227312	0	***
LSTAT*LSTAT*PTRATIO	-0.010557	0	***
LSTAT*NOX	-1.137008	0	***
LSTAT*PTRATIO	0.751989	0	***
LSTAT*RM	1.564722	0	***
LSTAT*RM*PTRATIO	-0.086992	0	***
PTRATIO	-2.269784	0	***
RM*NOX*CHAS_1	-3.701075	0.0009	***
RM*PTRATIO	0.116474	0.0477	*
RM*RM*DIS	0.054546	0.0002	***
	65.809511	0	***

8. Now, run the following statement to query Normalization and Missing Value Handling view. The columns of the view are described in Model Detail Views for Normalization and Missing Value Handling.

```
SELECT ATTRIBUTE_NAME, round(NUMERIC_MISSING_VALUE,2)
NUMERIC_MISSING_VALUE FROM DM$VNGLMR_REGR
ORDER BY ATTRIBUTE_NAME;
```

Examine the query:

- `ATTRIBUTE_NAME`: Provides the column names in the data set.
- `round(NUMERIC_MISSING_VALUE,2)NUMERIC_MISSING_VALUE`: Provides numeric replacements for the missing values (NULLs) in the data set. The `ROUND (n,integer)` returns results of `NUMERIC_MISSING_VALUE` rounded to integer places to the right.

ATTRIBUTE_NAME	NUMERIC_MISSING_VALUE
AGE	67.62
B	355.63
CRIM	3.28
DIS	3.79
INDUS	10.95
LSTAT	12.43
MEDV	22.74
NOX	0.55
PTRATIO	18.38
RAD	9.25
RM	6.27
TAX	401.99
ZN	11.34

Since there are no missing values (NULLs) in your data, you can ignore the result.

Test Your Model

In this use case, you are evaluating a regression model by computing Root Mean Square Error (RMSE) and Mean Absolute Error Mean (MAE) on the test data with known target values and comparing the predicted values with the known values.

Test metrics are used to assess how accurately the model predicts the known values. If the model performs well and meets your business requirements, it can then be applied to new data to predict the future. These matrices can help you to compare models to arrive at one model that satisfies your evaluation criteria.
For this use case, you compute Root Mean Square Error (RMSE) and Mean Absolute Error Mean (MAE) values. The RMSE and MAE are popular regression statistics. RMSE is an estimator for predictive models. The score averages the residuals for each case to yield a single indicator of model error. Mean absolute error is useful for understanding how close overall the predictions were to actual values. A smaller score means predictions were more accurate.

The following steps computes the error metrics for your model.

- To compute RMSE and MAE, run the following statement:

```
%sql
SELECT round(SQRT(AVG((A.PRED_MEDV - B.MEDV) * (A.PRED_MEDV - B.MEDV))),2)
RMSE,
       round(AVG(ABS(A.PRED_MEDV - B.MEDV)),2) MAE
  FROM (SELECT HID, PREDICTION(GLMR_REGR using *) PRED_MEDV
          FROM TEST_DATA) A,
       TEST_DATA B
  WHERE A.HID = B.HID;
```

This statement is using the prediction query to score the median value from the test data. The predicted value and the actual value from the test data is used to compute RMSE and MAE .

RMSE	MAE
3.52	2.6

RMSE and MAE convey average model prediction errors in units consistent with the target variable. When comparing models, a model with lower values is better. RMSE, which squares the errors, gives more weight to large errors, while MAE error scales linearly. Therefore, the predictions look fair and the model is a good fit for prediction.

Score

Scoring involves applying the model to the target data. Use `PREDICTION` query to predict the `MEDV` value on the test data.

The following step scores the test data comparing with the original data.

- Predict the median value of owner-occupied homes in the Boston area from the `TEST_DATA` and compare the predicted `MEDV` value with the actual `MEDV` value in your result.

```
SELECT HID, ROUND(PREDICTION(GLMR_REGR USING *), 1) AS
PREDICTED_MEDV, MEDV AS ACTUAL_MEDV FROM TEST_DATA ORDER BY HID;
```

  Examine the query:
  - `HID:` is the House ID.
  - `ROUND (n,integer):` in this case, is `ROUND (PREDICTION(GLMR_REGR USING *), 1)` returns results of `PREDICTION(GLMR_REGR USING *)` rounded to integer places to the right. Here, rounded to 1 place to the right.
  - `PREDICTED_MEDV:` is the predicted `MEDV` value.
  - `ACTUAL_MEDV:` is the `MEDV` value in the test data.

HID	PREDICTED_MEDV	ACTUAL_MEDV
10	16.4	18.9
14	19.2	20.4
15	17.9	18.2
22	16.7	19.6
27	15.2	16.6
28	16.2	14.8
29	18.2	18.4
42	30.7	26.6
45	22.2	21.2
54	22	23.4

To conclude, you have successfully predicted the median house prices in Boston using Generalized Linear Model algorithm.

Classification

You are working in a retail chain company that sells some products. To better target their marketing materials, they need to identify customers who are likely to purchase a home theater package. To resolve this, you are using the Random Forest algorithm to identify the customers.

Before you start your OML4SQL use case journey, ensure that you have the following:

- Data Set
  The data set used for this use case is from the SH schema. The SH schema can be readily accessed in Oracle Autonomous Database. For on-premises databases, the schema is installed during the installation or can be manually installed by downloading the scripts. See Installing the Sample Schemas.
- Database
  Select a database out of the following options:
  – Get your FREE cloud account. Go to https://cloud.oracle.com/database and select Oracle Database Cloud Service (DBCS), or Oracle Autonomous Database. Create an account and create an instance. See Autonomous Database Quick Start Workshop.
  – Download the latest version of Oracle Database (on premises).
- Machine Learning Tools
  Depending on your database selection,
  – Use OML Notebooks for Oracle Autonomous Database.
  – Install and use Oracle SQL Developer connected to an on-premises database or DBCS. See Installing and Getting Started with SQL Developer.
- Other Requirements
  Data Mining Privileges (this is automatically set for ADW). See System Privileges for Oracle Machine Learning for SQL.

Related Topics

- Create a Notebook
- Edit your Notebook
- Uninstalling HR Schema

Load Data

Access the data set from the SH Schema and explore the data to understand the attributes.

> **Remember:**
>
> The data set used for this use case is from the SH schema. The SH schema can be readily accessed in Oracle Autonomous Database. For on-premises databases, the schema is installed during the installation or can be manually installed by downloading the scripts. See Installing the Sample Schemas.

To understand the data, you will perform the following:

- Access the data.
- Examine the various attributes or columns of the data set.

- Assess data quality (by exploring the data).

Access Data

You will use CUSTOMERS and SUPPLEMENTARY_DEMOGRAPHICS table data from the SH schema.

Examine Data

The following table displays information about the attributes from SUPPLEMENTARY_DEMOGRAPHICS:

Attribute Name	Information
CUST_ID	The ID of the customer
EDUCATION	Educational information of the customer
OCCUPATION	Occupation of the customer
HOUSEHOLD_SIZE	People per house
YRS_RESIDENCE	Number of years of residence
AFFINITY_CARD	Whether the customer holds an affinity card
BULK_PACK_DISKETTES	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
FLAT_PANEL_MONITOR	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
HOME_THEATER_PACKAGE	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
BOOKKEEPING_APPLICATION	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
PRINTER_SUPPLIES	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
Y_BOX_GAMES	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
OS_DOC_SET_KANJI	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
COMMENTS	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No

Explore Data

Explore the data to understand and assess the quality of the data. At this stage assess the data to identify data types and noise in the data. Look for missing values and numeric outlier values.

Assess Data Quality

To assess the data, first, you must be able to view the data in your database. For this reason, you will use SQL statements to query the `SH.CUSTOMERS` and the `SH.SUPPLEMENTARY_DEMOGRAPHICS` table.

If you are working with Oracle Autonomous Database, you can use the Oracle Machine Learning (OML) Notebooks for your data science project, including assessing data quality. If you are using on-premise Oracle Database, you can use the Oracle SQL Developer to assess data quality. Query the `SH` schema as described.

> **Note:**
>
> Each record in the database is called a case and each case is identified by a `case_id`. In this use case, `CUST_ID` is the `case_id`.

1. View the data in the `SH.CUSTOMERS` table by running the following statement:

```
SELECT * FROM SH.CUSTOMERS;
```

2. To see distinct data from the table, run the following statement:

```
SELECT DISTINCT * FROM SH.CUSTOMERS;
```

CUST_ID	CUST_FIRST_NA...	CUST_LAST_NAM...	CUST_GENDER	CUST_YEAR_OF_BIRT...	CUST_MARITAL_STATU...	CUST_STREET_ADDRE...
49671	Abigail	Ruddy	M	1976	married	27 North Sagadahoc Boulevard
32561	Abner	Everett	M	1969	married	97 East Page Avenue
16581	Abner	Kenney	M	1986		17 East Page Court
49672	Abner	Kenney	M	1963	married	27 North Saguache Boulevard
13895	Abner	Kenney	M	1983	married	57 North 5th Drive
34359	Abner	Robbinette	M	1971	married	17 North Kaufman Court
15673	Abner	Robbinette	M	1958	single	57 South Saguache Drive

3. Find the `COUNT` of rows in the data set by running the following statement:

```
SELECT COUNT(*) from SH.CUSTOMERS;



COUNT(*)
     55500
---------------------------
```

4. To identify distinct or unique customers in the table, run the following statement:

```
%script
SELECT COUNT (DISTINCT CUST_ID) FROM SH.CUSTOMERS;
```

```
COUNT(DISTINCTCUST_ID)
                   55500
---------------------------
```

5. Similarly, query the `SH.SUPPLEMENTARY_DEMOGRAPHICS` table.

```
SELECT * FROM SH.SUPPLEMENTARY_DEMOGRAPHICS;
```

CUST_ID	EDUCATION	OCCUPATION	HOUSEHOLD_SIZ...	YRS_RESIDENCE ...	AFFINITY_CARD	BULK_PACK_DISKETTE...	FLAT_PANEL_MONITO...	HOME_THEATER_PACKA...	BOOKKEEPING_APPLICATIO...	PRINTER_SUPPLIE...	Y_BOX_GAMES	OS_DOC_SET_KAN...	COMMENTS
102547	10th	Other	1	0	0	1	1	0	0	1	1	0	
101050	10th	Other	1	0	0	1	1	0	0	1	1	0	
100040	11th	Sales	1	0	0	1	1	0	0	1	1	0	
102117	HS-grad	Farming	1	0	0	0	0	0	1	1	1	0	
101074	10th	Handler	1	1	0	1	1	0	0	1	1	0	
104179	10th	Handler	1	1	0	1	1	0	0	1	1	0	
100417	11th	Handler	1	1	0	0	0	0	0	1	1	0	
101146	< Bach.	?	1	1	0	1	1	0	1	1	1	0	

6. To view the count of `SH.SUPPLEMENTARY_DEMOGRAPHICS`, run the following statement:

```
SELECT COUNT(*) from SH.SUPPLEMENTARY_DEMOGRAPHICS;
```

```
COUNT(*)
      4500
---------------------------
```

7. Create a table called `CUSTOMERDATA` by selecting the required columns from the `SH.CUSTOMERS` and the `SH.SUPPLIMENTARY_DEMOGRAPHICS` tables.

```
%script
CREATE TABLE CUSTOMERDATA AS
   SELECT a.CUST_ID,
         a.CUST_INCOME_LEVEL, a.CUST_CREDIT_LIMIT,
          b.HOUSEHOLD_SIZE, b.OCCUPATION, b.HOME_THEATER_PACKAGE
   FROM SH.CUSTOMERS a, SH.SUPPLEMENTARY_DEMOGRAPHICS b
   WHERE a.CUST_ID = b.CUST_ID;
```

```
Table CUSTOMERDATA created.
```

8. View the `CUSTOMERDATA` table.

```
SELECT * FROM CUSTOMERDATA;
```

CUST_ID	CUST_GENDER	CUST_MARITAL_STATU...	CUST_YEAR_OF_BIRT...	CUST_INCOME_LEV...	CUST_CREDIT_LIMIT...	HOUSEHOLD_SIZ...	YRS_RESIDENCE	Y_BOX_GAMES
103791	M	Divorc.	1952	B: 30,000 - 49,999	3000	2	5	0
100804	F	Divorc.	1943	A: Below 30,000	1500	2	6	0
101610	M	NeverM	1985	I: 170,000 - 189,999	3000	1	0	1
102308	M	NeverM	1980	J: 190,000 - 249,999	11000	2	2	1
100593	M	Married	1963	G: 130,000 - 149,999	1500	3	4	0
100558	M	Married	1964	J: 190,000 - 249,999	11000	3	4	0
103401	M	Divorc.	1975	I: 170,000 - 189,999	10000	2	4	1
102740	F	Divorc.	1929	K: 250,000 - 299,999	15000	2	0	0

9. Find the count of rows in the new table `CUSTOMERDATA`:

```
SELECT COUNT(*) FROM CUSTOMERDATA;



COUNT(*)
      4500
---------------------------
```

10. To view the data type of the columns, run the following script:

```
%script
DESCRIBE CUSTOMERDATA;


Name                Null?    Type
------------------- -------- ------------
CUST_ID              NOT NULL  NUMBER
CUST_GENDER         NOT NULL  CHAR(1)
CUST_MARITAL_STATUS           VARCHAR2(20)
CUST_YEAR_OF_BIRTH NOT NULL  NUMBER(4)
CUST_INCOME_LEVEL             VARCHAR2(30)
CUST_CREDIT_LIMIT             NUMBER
HOUSEHOLD_SIZE                VARCHAR2(21)
YRS_RESIDENCE                    NUMBER
Y_BOX_GAMES                    NUMBER(10)

---------------------------
```

11. To check if there are any missing values (NULL values), run the following statement:

```
SELECT COUNT(*) FROM CUSTOMERDATA WHERE CUST_ID=NULL OR CUST_GENDER=NULL
 OR CUST_MARITAL_STATUS=NULL OR CUST_YEAR_OF_BIRTH=NULL OR
CUST_INCOME_LEVEL=NULL
 OR CUST_CREDIT_LIMIT=NULL OR HOUSEHOLD_SIZE=NULL OR YRS_RESIDENCE=NULL OR
Y_BOX_GAMES=NULL;



COUNT(*)
         0
---------------------------
```

NULLs, if found, are automatically handled by the OML algorithms. Alternately, you can manually replace NULLs with `NVL` SQL function.

12. To know the income level of customers who responded to `HOME_THEATER_PACKAGE`, run the following statement:

```
SELECT COUNT(CUST_ID) AS NUM_CUSTOMERS, CUST_INCOME_LEVEL, HOME_THEATER_PACKAGE
FROM   CUSTOMERDATA
GROUP BY CUST_INCOME_LEVEL, HOME_THEATER_PACKAGE;


NUM_CUSTOMERS   CUST_INCOME_LEVEL      HOME_THEATER_PACKAGE
            214 K: 250,000 - 299,999                      0
            315 L: 300,000 and above                      1
            114 E: 90,000 - 109,999                       0
             27 A: Below 30,000                           0
             61 A: Below 30,000                           1
            206 F: 110,000 - 129,999                      1
            446 J: 190,000 - 249,999                      0
            196 E: 90,000 - 109,999                       1
             90 B: 30,000 - 49,999                        0
             99 C: 50,000 - 69,999                        1
            319 I: 170,000 - 189,999                      1
            165 I: 170,000 - 189,999                      0
            179 K: 250,000 - 299,999                      1
            142 H: 150,000 - 169,999                      0

NUM_CUSTOMERS   CUST_INCOME_LEVEL      HOME_THEATER_PACKAGE
            163 F: 110,000 - 129,999                      0
             83 D: 70,000 - 89,999                        1
             50 D: 70,000 - 89,999                        0
            328 L: 300,000 and above                      0
            519 J: 190,000 - 249,999                      1
            189 G: 130,000 - 149,999                      1
            150 G: 130,000 - 149,999                      0
            132 B: 30,000 - 49,999                        1
             72 C: 50,000 - 69,999                        0
            241 H: 150,000 - 169,999                      1


24 rows selected.
---------------------------
```

This completes the data exploration stage. OML supports Automatic Data Preparation (ADP). ADP is enabled through the model settings. When ADP is enabled, the transformations required by the algorithm are performed automatically and embedded in the model. This step is done during the Build Model stage. The commonly used methods of data preparation are binning, normalization, and missing value treatment.

Related Topics

- How ADP Transforms the Data

Build Model

Build your model using the training data set. Use the `DBMS_DATA_MINING.CREATE_MODEL2` procedure to build your model and specify the model settings.

For a supervised learning, like Classification, before creating the model, split the data into training and test data. Although you can use the entire data set to build a model, it is difficult to validate the model unless there are new data sets available. Therefore, to evaluate the model and to accurately assess the performance of the model on the same data, you generally split or separate the data into training and test data. You use the training data set to train the model and then use the test data set to test the accuracy of the model by running prediction queries. The testing data set already contains known values for the attribute that you want to predict. It is thus easy to determine whether the predictions of the model are correct.

Algorithm Selection

Before you build a model, choose the suitable algorithm. You can choose one of the following algorithms to solve a classification problem:

- Decision Tree
- Explicit Semantic Analysis (ESM)
- Generalized Linear Model (GLM)
- Naive Bayes
- Random Forest
- Support Vector Machine (SVM)
- XGBoost

From the above algorithms, ESM is more about Natural Language Processing (NLP) and text mining. ESM does not apply to this use case and data. If you were to select a relatively simple linear model like GLM, the prediction accuracy can be further improved by the Random Forest algorithm. Random Forest is an ensemble method that builds multiple decision trees on subsets of the data re-sampled at each time (bagging). This avoids the overfitting for a single decision tree. The random forest model is a widely used ensemble method that is known to have higher accuracy than linear models. Thus, Random Forest is selected for this use case.

For this use case, split the data into 60/40 as training and test data. You build the model using the training data and once the model is built, score the test data using the model.

The following steps guide you to split your data and build your model with the selected algorithm.

1. To create the training and test data with 60/40 split, run the following statement:

```
CREATE OR REPLACE VIEW TRAINING_DATA AS SELECT * FROM CUSTOMERDATA SAMPLE
(60) SEED (1);
--DBMS_OUTPUT.PUT_LINE ('Created TRAINING_DATA');
CREATE OR REPLACE VIEW TEST_DATA AS SELECT * FROM CUSTOMERDATA MINUS SELECT
* FROM TRAINING_DATA;
```

```
--DBMS_OUTPUT.PUT_LINE ('Created TEST_DATA');



View TRAINING_DATA created.
---------------------------
View TEST_DATA created.
```

2. To view the data in the `training_data` view, run the following statement:

```
SELECT * FROM TRAINING_DATA;
```

CUST_ID	CUST_INCOME_LEVEL	CUST_CREDIT_LIMIT	HOUSEHOLD_SIZE	OCCUPATION	HOME_THEATER_PACK..
100200	L: 300,000 and above	9000	1	Other	0
100300	G: 130,000 - 149,999	10000	3	Prof.	1
100400	C: 50,000 - 69,999	9000	6-8	Transp.	1
100900	F: 110,000 - 129,999	1500	3	Exec.	1
101000	G: 130,000 - 149,999	7000	3	Crafts	1
101200	L: 300,000 and above	9000	1	?	0
101300	J: 190,000 - 249,999	15000	3	TechSup	1
101400	B: 30,000 - 49,999	1500	3	Machine	1

3. To view the data in the `test_data` view, run the following statement:

```
SELECT* FROM TEST_DATA;
```

CUST_ID	CUST_INCOME_LEVEL	CUST_CREDIT_LIMIT	HOUSEHOLD_SIZE	OCCUPATION	HOME_THEATER_PACK..
100005	B: 30,000 - 49,999	1500	3	Crafts	1
100006	G: 130,000 - 149,999	5000	9+	Prof.	0
100007	L: 300,000 and above	9000	2	Other	1
100008	J: 190,000 - 249,999	15000	2	Crafts	1
100009	G: 130,000 - 149,999	3000	3	Prof.	0
100010	L: 300,000 and above	9000	3	Crafts	0
100011	F: 110,000 - 129,999	10000	2	Farming	0
100014	B: 30,000 - 49,999	3000	2	Cleric.	1

4. To view the distribution of `HOME_THEATER_PACKAGE` (target) owners, run the following script:

```
%script
select HOME_THEATER_PACKAGE, count(1)
from training_data
group by HOME_THEATER_PACKAGE;



HOME_THEATER_PACKAGE   COUNT(1)
                     1       1506
                     0       1208

---------------------------
```

5. Build your model using the `CREATE_MODEL2` procedure. First, declare a variable to store model settings or hyperparameters. Run the following script:

```
%script

BEGIN DBMS_DATA_MINING.DROP_MODEL('MODEL_RF');
EXCEPTION WHEN OTHERS THEN NULL; END;
/
DECLARE
    v_setlist DBMS_DATA_MINING.SETTING_LIST;

BEGIN
    v_setlist('PREP_AUTO') := 'ON';
    v_setlist('ALGO_NAME') := 'ALGO_RANDOM_FOREST';
    v_setlist('RFOR_NUM_TREES') := '25';

    DBMS_DATA_MINING.CREATE_MODEL2(
      MODEL_NAME          =>  'MODEL_RF',
      MINING_FUNCTION     => 'CLASSIFICATION',
      DATA_QUERY          =>  'SELECT * FROM TRAINING_DATA',
      SET_LIST            =>  v_setlist,
      CASE_ID_COLUMN_NAME =>  'CUST_ID',
      TARGET_COLUMN_NAME  =>  'HOME_THEATER_PACKAGE');
END;
```

```
PL/SQL procedure successfully completed.

---------------------------

PL/SQL procedure successfully completed.
```

Examine the script:

- `v_setlist` is a variable to store `SETTING_LIST`.
- `SETTING_LIST` defines model settings or hyperparameters for your model.
- `DBMS_DATA_MINING` is the PL/SQL package used for machine learning. These settings are described in DBMS_DATA_MINING - Model Settings.
- `ALGO_NAME` specifies the algorithm name. Since you are using Random Forest as the algorithm, set `ALGO_RANDOM_FOREST`.
- `PREP_AUTO` is the setting used for Automatic Data Preparation. Here, enable Automatic Data Preparation. The value of the setting is `ON`.
- `RFOR_NUM_TREES` is the number of trees in the forest. The value here is `25`. Random forest resolves the overfitting problem by training multiple trees on distinct sampled subsets of the data instead of on the same, entire training set. The more trees you select, the more accuracy it can obtain. However, keep in mind that more trees mean more computation load and longer model building time. You need to do a trade-off between the time cost and model accuracy here. Choosing the number of trees equal to 25 allows you to build the model in a reasonably short time and obtain an accurate enough model.

The `CREATE_MODEL2` procedure takes the following parameters:

- `MODEL_NAME`: A unique model name that you will give to the model. The name of the model is in the form [schema_name.]model_name. If you do not specify a schema, then your own schema is used. Here, the model name is `MODEL_RF`

- `MINING_FUNCTION`: Specifies the machine learning function. Since it is a classification problem in this use case, select `CLASSIFICATION`.
- `DATA_QUERY`: A query that provides training data for building the model. Here, the query is `SELECT * FROM TRAINING_DATA`.
- `SET_LIST`: Specifies `SETTING_LIST`.
- `CASE_ID_COLUMN_NAME`: A unique case identifier column in the build data. In this use case, case_id is `CUST_ID`. If there is a composite key, you must create a new attribute before creating the model. The `CASE_ID` assists with reproducible results, joining scores for individual customers with other data in, example, scoring data table.

> **Note:**
>
> Any parameters or settings not specified are either system-determined or default values are used.

Evaluate

Evaluate your model by viewing diagnostic metrics and performing quality checks.

You can inspect various rules to see if they reveal new insights for item dependencies (antecedent itemset implying consequent) or for unexpected relationships among items.

Dictionary and Model Views

To obtain information about the model and view model settings, you can query data dictionary views and model detail views. Specific views in model detail views display model statistics which can help you evaluate the model.

The data dictionary views for Oracle Machine Learning are listed in the following table. A database administrator (DBA) and USER versions of the views are also available.

View Name	Description
ALL_MINING_MODELS	Provides information about all accessible machine learning models
ALL_MINING_MODEL_ATTRIBUTES	Provides information about the attributes of all accessible machine learning models
ALL_MINING_MODEL_SETTINGS	Provides information about the configuration settings for all accessible machine learning models
ALL_MINING_MODEL_VIEWS	Provides information about the model views for all accessible machine learning models
ALL_MINING_MODEL_XFORMS	Provides the user-specified transformations embedded in all accessible machine learning models.

Model detail views are specific to the algorithm. You can obtain more insights about the model you created by viewing the model detail views. The names of model detail views begin with DM$xx where xx corresponds to the view prefix. See Model Detail Views.

The following steps help you to view different dictionary views and model detail views.

1. Run the following statement to view the settings in `USER_MINING_MODEL_SETTINGS`:

```
%script

SELECT SETTING_NAME, SETTING_VALUE
  FROM USER_MINING_MODEL_SETTINGS
  WHERE MODEL_NAME='MODEL_RF'
  ORDER BY SETTING_NAME;



SETTING_NAME                   SETTING_VALUE
ALGO_NAME                      ALGO_RANDOM_FOREST
CLAS_MAX_SUP_BINS              32
CLAS_WEIGHTS_BALANCED          OFF
ODMS_DETAILS                   ODMS_ENABLE
ODMS_MISSING_VALUE_TREATMENT   ODMS_MISSING_VALUE_AUTO
ODMS_RANDOM_SEED               0
ODMS_SAMPLING                  ODMS_SAMPLING_DISABLE
PREP_AUTO                      ON
RFOR_NUM_TREES                 25
RFOR_SAMPLING_RATIO            .5
TREE_IMPURITY_METRIC           TREE_IMPURITY_GINI
TREE_TERM_MAX_DEPTH            16
TREE_TERM_MINPCT_NODE          .05
TREE_TERM_MINPCT_SPLIT         .1

SETTING_NAME             SETTING_VALUE
TREE_TERM_MINREC_NODE    10
TREE_TERM_MINREC_SPLIT   20


16 rows selected.
---------------------------
```

2. Run the following statement to see attribute information in `USER_MINING_MODEL_ATTRIBUTES` view:

```
%script
SELECT ATTRIBUTE_NAME, ATTRIBUTE_TYPE
FROM USER_MINING_MODEL_ATTRIBUTES
WHERE MODEL_NAME = 'MODEL_RF'
ORDER BY ATTRIBUTE_NAME;



ATTRIBUTE_NAME         ATTRIBUTE_TYPE
CUST_CREDIT_LIMIT      NUMERICAL
HOME_THEATER_PACKAGE   CATEGORICAL
HOUSEHOLD_SIZE         CATEGORICAL
OCCUPATION             CATEGORICAL

---------------------------
```

3. Run the following statement to view various model detail views from `USER_MINING_MODEL_VIEWS`:

```
%script
SELECT VIEW_NAME, VIEW_TYPE
  FROM USER_MINING_MODEL_VIEWS
  WHERE MODEL_NAME='MODEL_RF'
  ORDER BY VIEW_NAME;


VIEW_NAME        VIEW_TYPE
DM$VAMODEL_RF    Variable Importance
DM$VCMODEL_RF    Scoring Cost Matrix
DM$VGMODEL_RF    Global Name-Value Pairs
DM$VSMODEL_RF    Computed Settings
DM$VTMODEL_RF    Classification Targets
DM$VWMODEL_RF    Model Build Alerts


6 rows selected.
---------------------------
```

4. Now, view the Classification targets view. This view describes the target (`HOME_THEATER_PACKAGE`) distribution for classification models.

```
%script
SELECT* from DM$VTMODEL_RF;



PARTITION_NAME   TARGET_VALUE   TARGET_COUNT   TARGET_WEIGHT
                             0            1178
                             1            1549

---------------------------
```

The distribution value from this view validates the earlier target distribution that was obtained from the training data. The difference in the values is minimal.

Related Topics

- PREDICTION_SET

Test Your Model

In this use case, you are evaluating a classification model by computing Lift and Confusion Matrix on the test data with known target values and comparing the predicted values with the known values.

Test metrics are used to assess how accurately the model predicts the known values. If the model performs well and meets your business requirements, it can then be applied to new data to predict the future. These matrices can help you to compare models to arrive at one model that satisfies your evaluation criteria.
Lift measures the degree to which the predictions of a classification model are better than randomly-generated predictions. Lift can be understood as a ratio of two percentages: the

percentage of correct positive classifications made by the model to the percentage of actual positive classifications in the test data.

A confusion matrix displays the number of correct and incorrect predictions made by the model compared with the actual classifications in the test data. The matrix is n-by-n, where n is the number of classes.

1. Create a result table to store the predictions for each row with likely and unlikely probabilities. Run the following script:

```
%script

BEGIN EXECUTE IMMEDIATE 'DROP TABLE APPLY_RESULT PURGE';
EXCEPTION WHEN OTHERS THEN NULL; END;
/

CREATE TABLE APPLY_RESULT AS
    SELECT cust_id, t.prediction, t.probability
    FROM TEST_DATA, TABLE(PREDICTION_SET(MODEL_RF USING *)) t;
```

```
PL/SQL procedure successfully completed.
---------------------------
Table APPLY_RESULT created.
---------------------------
```

   Examine the script:

   `APPLY_RESULT`: is a table that stores the results of the prediction.

   `TABLE(PREDICTION_SET(MODEL_RF USING *))`: is a table that has results from the `PREDICTION_SET` query. The `PREDICTION_SET` query returns probabilities for each row.

2. Compute lift by using the `DBMS_DATA_MINING.APPLY` and the `DBMS_DATA_MINING.COMPUTE_LIFT` procedures:

```
%script

BEGIN EXECUTE IMMEDIATE 'DROP TABLE APPLY_RESULT PURGE';
EXCEPTION WHEN OTHERS THEN NULL; END;
/

BEGIN
  DBMS_DATA_MINING.APPLY('MODEL_RF','TEST_DATA','CUST_ID','APPLY_RESULT');


       DBMS_DATA_MINING.COMPUTE_LIFT (
          apply_result_table_name           => 'APPLY_RESULT',
          target_table_name                  => 'TEST_DATA',
          case_id_column_name               => 'CUST_ID',
          target_column_name                 => 'HOME_THEATER_PACKAGE',
          lift_table_name                       => 'LIFT_TABLE',
          positive_target_value           =>  to_char(1),
          score_column_name                  => 'PREDICTION',
          score_criterion_column_name    => 'PROBABILITY',
          num_quantiles                         =>  10,
          cost_matrix_table_name             =>  null,
```

```
        apply_result_schema_name         =>  null,
        target_schema_name                 =>  null,
        cost_matrix_schema_name           =>  null,
        score_criterion_type             =>  'PROBABILITY');

END;



PL/SQL procedure successfully completed.
---------------------------
PL/SQL procedure successfully completed.
```

Examine the script:

- `DBMS_DATA_MINING.APPLY`: This procedure creates a table in the user's schema to hold the results. The `APPLY` procedure generates predictions (scores) in a target column.
  The `APPLY` procedure has the following parameters:
  - `model_name`: Name of the model in the form [schema_name.]model_name. If you do not specify a schema, then your own schema is used. Here, the model name is `MODEL_RF`.
  - `data_table_name`: Name of table or view containing the data to be scored. Here, you are using `TEST_DATA`.
  - `case_id_column_name`: Name of the case identifier column. The case ID is `CUST_ID`.
  - `result_table_name`: Name of the table in which to store apply results. Here, the result table name is `APPLY_RESULT`.
- `DBMS_DATA_MINING.COMPUTE_LIFT`: This procedure computes lift and stores them in the user's schema. To compute lift, one of the target values must be designated as the positive class.
  The `COMPUTE_LIFT` procedure has the following parameters:
  - `apply_result_table_name`: Table containing the predictions. For this use case, it is `APPLY_RESULT`.
  - `target_table_name`: Table containing the known target values from the test data. In this use case, the target table name is `TEST_DATA`.
  - `case_id_column_name`: Case ID column in the apply results table. Must match the case identifier in the targets table. The case ID column is `CUST_ID`.
  - `target_column_name`: Target column in the targets table. Contains the known target values from the test data. In this use case, the target is `HOME_THEATER_PACKAGE`.
  - `lift_table_name`: Table containing the lift statistics. The table will be created by the procedure in the user's schema. Type `LIFT_TABLE`.
  - `positive_target_value`: The positive class. This should be the class of interest, for which you want to calculate lift. If the target column is a `NUMBER`, you can use the `TO_CHAR()` operator to provide the value as a string.

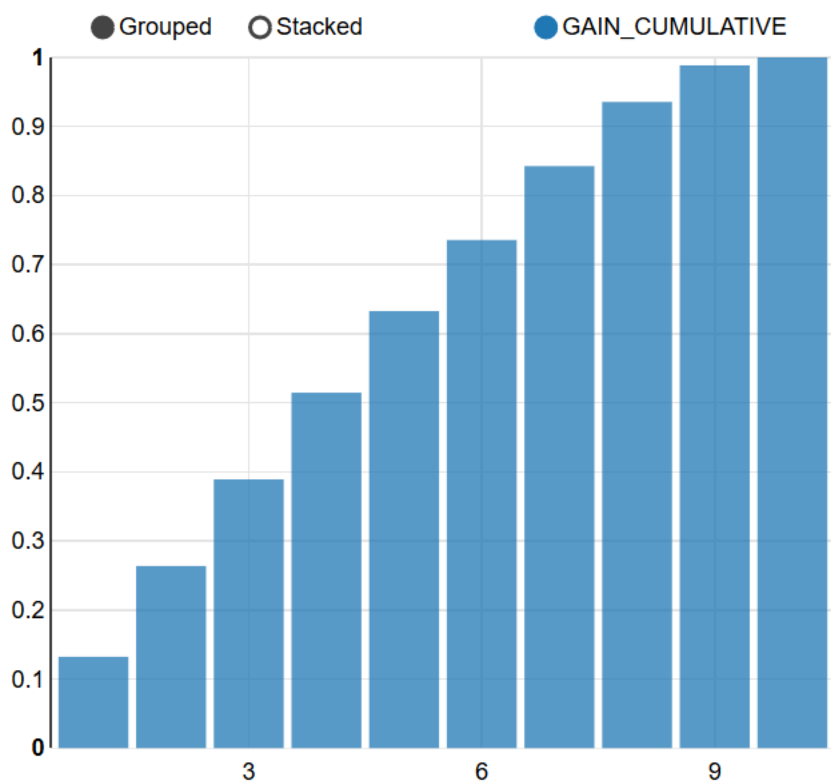
- `score_column_name`: Column containing the predictions in the apply results table. The default column name is `'PREDICTION'`, which is the default name created by the `APPLY` procedure.
- `score_criterion_column_name`: Column containing the scoring criterion in the apply results table. Contains either the probabilities or the costs that determine the predictions. By default, scoring is based on probability; the class with the highest probability is predicted for each case. If scoring is based on cost, the class with the lowest cost is predicted. The `score_criterion_type` parameter indicates whether probabilities or costs will be used for scoring. The default column name is `'PROBABILITY'`, which is the default name created by the `APPLY` procedure.
- `num_quantiles`: Number of quantiles to be used in calculating lift. The default is 10.
- `cost_matrix_table_name`: (Optional) Table that defines the costs associated with misclassifications. If a cost matrix table is provided and the score_criterion_type parameter is set to `'COST'`, the costs will be used as the scoring criteria.
- `apply_result_schema_name`: Schema of the apply results table. If null, the user's schema is assumed.
- `target_schema_name`: Schema of the table containing the known targets. If null, the user's schema is assumed.
- `cost_matrix_schema_name`: Schema of the cost matrix table, if one is provided. If null, the user's schema is assumed.
- `score_criterion_type`: Whether to use probabilities or costs as the scoring criterion. Probabilities or costs are passed in the column identified in the score_criterion_column_name parameter. The default value of `score_criterion_type` is `'PROBABILITY'`. To use costs as the scoring criterion, specify `'COST'`. If `score_criterion_type` is set to `'COST'` but no cost matrix is provided and if there is a scoring cost matrix associated with the model, then the associated costs are used for scoring.

3. To view the cumulative gains, run the following statement:

   Cumulative gain is the ratio of the cumulative number of positive targets (`HOME_THEATER_PACKAGE`) to the total number of positive targets of a quantile. Cumulative gains act as a visual aid for measuring performance of a model. The chart consists of a curve and a baseline. The greater the area between the curve and the baseline, the better the model.

```
%sql
SELECT QUANTILE_NUMBER, GAIN_CUMULATIVE FROM LIFT_TABLE;
```

QUANTILE_NUMBER	GAIN_CUMULATIVE
1	1.32572428385416666666666666666666666667E-01
2	2.64000007287779850746268656716417910448E-01
3	3.89386411448616298507462686567164179105E-01
4	5.14810590601679104477611940298507462687E-01
5	6.32985098919465174129353233830845771144E-01
6	7.35760589144123134328358208955223880597E-01
7	8.42706450656871890547263681592039800995E-01



4. To compute confusion matrix, run the following statement:

   A confusion matrix evaluates the prediction results. It makes it easy to understand and estimate the effects of wrong predictions. You can observe the number and percentages in each cell of this matrix and notice how often the model predicted accurately.

```
%script

DECLARE
   v_accuracy NUMBER;
   BEGIN
        DBMS_DATA_MINING.COMPUTE_CONFUSION_MATRIX (
                   accuracy => v_accuracy,
                   apply_result_table_name => 'apply_result',
                   target_table_name => 'test_data',
                   case_id_column_name => 'cust_id',
                   target_column_name => 'HOME_THEATER_PACKAGE',
                   confusion_matrix_table_name =>
'confusion_matrix',
                   score_column_name => 'PREDICTION',
                   score_criterion_column_name => 'PROBABILITY',
                   cost_matrix_table_name => null,
                   apply_result_schema_name => null,
                   target_schema_name => null,
                   cost_matrix_schema_name => null,
                   score_criterion_type => 'PROBABILITY');
        DBMS_OUTPUT.PUT_LINE('**** MODEL ACCURACY ****: ' ||
ROUND(v_accuracy,4));
      END;
      /


**** MODEL ACCURACY ****: .696
---------------------------
PL/SQL procedure successfully completed.
---------------------------
```

   Examine the script:

   `v_accuracy` is a variable declared for this procedure to store and output the model accuracy percentage.

   The `COMPUTE_CONFUSION_MATRIX` procedure has the following parameters:

   - `accuracy`: Output parameter containing the overall percentage accuracy of the predictions. Here, it is `v_accuracy`.
   - `apply_result_table_name`: Table containing the predictions. In this use case, it is `APPLY_RESULT`.
   - `target_table_name`: Table containing the known target values from the test data. In this use case, you are using `TEST_DATA`.
   - `case_id_column_name`: Case ID column in the apply results table. Must match the case identifier in the targets table. Here, it is `CUST_ID`.

- `target_column_name`: Target column in the targets table. Contains the known target values from the test data. In this use case, the target column is `HOME_THEATER_PACKAGE`.
- `confusion_matrix_table_name`: Table containing the confusion matrix. The table will be created by the procedure in the user's schema. Here set it as `confusion_matrix`.
- `score_column_name`: Column containing the predictions in the apply results table. The default column name is `PREDICTION`, which is the default name created by the `APPLY` procedure.
- `score_criterion_column_name`: Column containing the scoring criterion in the apply results table. Contains either the probabilities or the costs that determine the predictions. By default, scoring is based on probability; the class with the highest probability is predicted for each case. If scoring is based on cost, the class with the lowest cost is predicted. The `score_criterion_type` parameter indicates whether probabilities or costs will be used for scoring. The default column name is `'PROBABILITY'`, which is the default name created by the `APPLY` procedure.
- `cost_matrix_table_name`: (Optional) Table that defines the costs associated with misclassifications. If a cost matrix table is provided and the `score_criterion_type` parameter is set to `'COSTS'`, the costs in this table will be used as the scoring criteria. Otherwise, set it as `null`.
- `apply_result_schema_name`: Schema of the apply results table. If null, the user's schema is assumed.
- `target_schema_name`: Schema of the table containing the known targets. If null, the user's schema is assumed.
- `cost_matrix_schema_name`: Schema of the cost matrix table, if one is provided. If null, the user's schema is assumed.
- `score_criterion_type`: Whether to use probabilities or costs as the scoring criterion. Probabilities or costs are passed in the column identified in the `score_criterion_column_name` parameter. The default value of `score_criterion_type` is `'PROBABILITY'`. To use costs as the scoring criterion, specify `'COST'`. If `score_criterion_type` is set to `'COST'` but no cost matrix is provided and if there is a scoring cost matrix associated with the model, then the associated costs are used for scoring.

`DBMS_OUTPUT.PUT_LINE('**** MODEL ACCURACY ****: ' || ROUND(v_accuracy,4))`: Outputs the model accuracy percentage rounded to 4 digits after the decimal.

5. To check the confusion matrix with predicted values and actual values, run the following statement:

```
select * from confusion_matrix;



ACTUAL_TARGET_VALUE   PREDICTED_TARGET_VALUE   VALUE
                  0                        1     501
                  0                        0     282
                  1                        0      38
                  1                        1     952

---------------------------
```

The value column here indicates classification. From this confusion matrix, the model has predicted actual positive class (also called as True Positive (TP)) for this use case 952 times and incorrectly predicted (also called as False Negative (FN)) for this use case 38 times. The model correctly predicted the negative class (also called true negative (TN)) for this use case 282 times and incorrectly predicted (also called false positive (FP)) for this use case 501 times.

The accuracy percentage of 69% shows that the model is fairly good for this use case.

Related Topics

- PREDICTION_SET

Score

You are ready to predict the likely customers for the `HOME_THEATER_PACKAGE` responders. For classification problems, you can use `PREDICTION`, `PREDICTION_PROBABILITY`, or use analytic syntax to arrive at predictions.

1. To view customers who have more than 50% chance of buying a home theater package, run the following statement:

```
%sql
SELECT CUST_ID, PREDICTION PRED, ROUND(PROBABILITY,3) PROB, ROUND(COST,2)
COST
  FROM APPLY_RESULT WHERE PREDICTION = 1 AND PROBABILITY > 0.5
  ORDER BY PROBABILITY DESC;
```

CUST_ID	PRED	PROB	COST
104384	1	0.764	0.24
104136	1	0.764	0.24
101600	1	0.764	0.24
100009	1	0.764	0.24
100046	1	0.764	0.24
100178	1	0.764	0.24
100271	1	0.764	0.24
100282	1	0.764	0.24

2. You can score on multiple rows of test data. This is called batch scoring. This step shows how you can view and select customers who are likely or unlikely to respond to `HOME_THEATER_PACKAGE` with a probability of more than 50% and a cost matrix.

```
%sql

SELECT CUST_ID, PREDICTION, ROUND(PROBABILITY,2) PROB, ROUND(COST,2) COST
  FROM APPLY_RESULT WHERE PREDICTION = ${PREDICTION='1','1'|'0'}
  AND PROBABILITY > 0.5 ORDER BY PROBABILITY DESC;
```

CUST_ID	PREDICTION	PROB	COST
100129	0	0.92	0.08
104277	0	0.92	0.08
100188	0	0.92	0.08
100331	0	0.92	0.08
101172	0	0.92	0.08
101896	0	0.92	0.08
102038	0	0.92	0.08
102108	0	0.92	0.08

3. To interactively view probability of `HOME_THEATER_PACKAGE` respondents, run the following statement:

```
%sql
SELECT A.*, B.*
  FROM APPLY_RESULT A, TEST_DATA B
  WHERE PREDICTION = ${PREDICTION='1','1'|'0'} AND A.CUST_ID = B.CUST_ID;
```

CUST_ID	PREDICTION	PROBABILITY	COST	CUST_INCOME_LEVEL	CUST_CREDIT_LIMIT	HOUSEHOLD_SIZE	OCCUPATI
100001	0	0.3238174648999842	0.6761825351000158	G: 130,000 - 149,999	1500	2	Exec.
100002	0	0.3872678206049052	0.6127321793950948	L: 300,000 and above	7000	2	Prof.
100003	0	0.4632677673534442	0.5367322326465558	K: 250,000 - 299,999	7000	2	Sales
100004	0	0.49676712791833855	0.5032328720816615	K: 250,000 - 299,999	15000	2	Sales
100005	0	0.24724772293960184	0.7527522770603982	B: 30,000 - 49,999	1500	3	Crafts
100009	0	0.235521435198742	0.764478564801258	G: 130,000 - 149,999	3000	3	Prof.
100016	0	0.3872678206049052	0.6127321793950948	K: 250,000 - 299,999	7000	9+	Exec.

4. To dynamically score and select customers with more than 50% chance of purchasing a home theater package, run the following statement:

```
%sql

SELECT *
FROM (  SELECT CUST_ID, ROUND(PREDICTION_PROBABILITY(MODEL_RF, '1'  USING A.*),3)
PROBABILITY
    FROM TEST_DATA A)
WHERE PROBABILITY > 0.5;
```

You can use `PREDICTION_PROBABILITY` to score in real-time.

CUST_ID	PROBABILITY
100002	0.613
100003	0.537
100004	0.503
100005	0.753
100009	0.764
100016	0.613
100019	0.722
100021	0.752
100022	0.757
100025	0.653
100027	0.752

5. To apply the model to a single record (singleton scoring), run the following statement:

```
%script
SELECT ROUND(PREDICTION_PROBABILITY(MODEL_RF, '1' USING
                                    '3' AS HOUSEHOLD_SIZE,
                                     5 AS YRS_RESIDENCE,
                                     1 AS CUST_INCOME_LEVEL),3)
PROBABILITY_HOME_THEATER_PACKAGE_RESPONDER
  FROM DUAL;
```

This may be useful if you want to test the model manually and see how the model works.

```
PROBABILITY_HOME_TEATER_PACKAGE_RESPONDER
                                           0.65

------------------------------
```

To conclude, you have successfully identified customers who are likely to purchase `HOME_THEATER_PACKAGE`. This prediction helps to promote and offer home theater package to the target customers.

Clustering

You are a data scientist working in a gaming company. The marketing team in your company wants to promote a new game. They approach you to help them in identifying target customers. They want to target customers who already purchased a gaming product earlier with high credit limit. They want to segment customers based on their gaming product purchase history. You resolve this problem by segmenting the population using the *k*-Means algorithm.

Before you start your OML4SQL use case journey, ensure that you have the following:

- Data Set

The data set used for this use case is from the SH schema. The SH schema can be readily accessed in Oracle Autonomous Database. For on-premises databases, the schema is installed during the installation or can be manually installed by downloading the scripts. See Installing the Sample Schemas.

- Database
  Select a database out of the following options:
  - Get your FREE cloud account. Go to https://cloud.oracle.com/database and select Oracle Database Cloud Service (DBCS), or Oracle Autonomous Database. Create an account and create an instance. See Autonomous Database Quick Start Workshop.
  - Download the latest version of Oracle Database (on premises).
- Machine Learning Tools
  Depending on your database selection,
  - Use OML Notebooks for Oracle Autonomous Database.
  - Install and use Oracle SQL Developer connected to an on-premises database or DBCS. See Installing and Getting Started with SQL Developer.
- Other Requirements
  Data Mining Privileges (this is automatically set for ADW). See System Privileges for Oracle Machine Learning for SQL.

Related Topics

- Create a Notebook
- Edit your Notebook
- Installing Sample Schemas

Load Data

Access the data set from the SH Schema and explore the data to understand the attributes.

> **Remember:**
>
> The data set used for this use case is from the SH schema. The SH schema can be readily accessed in Oracle Autonomous Database. For on-premises databases, the schema is installed during the installation or can be manually installed by downloading the scripts. See Installing the Sample Schemas.

To understand the data, you will perform the following:

- Access the data.
- Examine the various attributes or columns of the data set.
- Assess data quality (by exploring the data).

Access Data

You will use `CUSTOMERS` and `SUPPLEMENTARY_DEMOGRAPHICS` table data from the SH schema.

Examine Data

The following table displays information about the attributes from SUPPLEMENTARY_DEMOGRAPHICS:

Attribute Name	Information
CUST_ID	The ID of the customer
EDUCATION	Educational information of the customer
OCCUPATION	Occupation of the customer
HOUSEHOLD_SIZE	People per house
YRS_RESIDENCE	Number of years of residence
AFFINITY_CARD	Whether the customer holds an affinity card
BULK_PACK_DISKETTES	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
FLAT_PANEL_MONITOR	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
HOME_THEATER_PACKAGE	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
BOOKKEEPING_APPLICATION	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
PRINTER_SUPPLIES	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
Y_BOX_GAMES	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
OS_DOC_SET_KANJI	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No
COMMENTS	Product. Indicates whether the customer already owns the product. 1 means Yes. 0 means No

Explore Data

Once the data is accessible, explore the data to understand and assess the quality of the data. At this stage assess the data to identify data types and noise in the data. Look for missing values and numeric outlier values.

Assess Data Quality

To assess the data, first, you must be able to view the data in your database. For this reason, you will use SQL statements to query the SH.CUSTOMERS and the SH.SUPPLEMENTARY_DEMOGRAPHICS table.

If you are working with Oracle Autonomous Database, you can use the Oracle Machine Learning (OML) Notebooks for your data science project, including assessing data quality. If you are using on-premise Oracle Database, you can use the Oracle SQL Developer to assess data quality. Query the `SH` schema as described.

> **Note:**
>
> Each record in the database is called a case and each case is identified by a `case_id`. In this use case, `CUST_ID` is the `case_id`.

The following steps help you with the exploratory analysis of the data:

1. View the data in the `SH.CUSTOMERS` table by running the following query:

```
SELECT * FROM SH.CUSTOMERS;
```

2. To see distinct data from the table, run the following query:

```
SELECT DISTINCT * FROM SH.CUSTOMERS;
```

CUST_ID	CUST_FIRST_NA...	CUST_LAST_NAM...	CUST_GENDER	CUST_YEAR_OF_BIRT...	CUST_MARITAL_STATU...	CUST_STREET_ADDRE...
49671	Abigail	Ruddy	M	1976	married	27 North Sagadahoc Boulevard
32561	Abner	Everett	M	1969	married	97 East Page Avenue
16581	Abner	Kenney	M	1986		17 East Page Court
49672	Abner	Kenney	M	1963	married	27 North Saguache Boulevard
13895	Abner	Kenney	M	1983	married	57 North 5th Drive
34359	Abner	Robbinette	M	1971	married	17 North Kaufman Court
15673	Abner	Robbinette	M	1958	single	57 South Saguache Drive

3. Find the `COUNT` rows in the data set, run the following statement:

```
SELECT DISTINCT COUNT(*) from SH.CUSTOMERS;


COUNT(*)
     55500
---------------------------
```

4. To find distinct or unique customers in the table, run the following statement:

```
%script
SELECT COUNT (DISTINCT CUST_ID) FROM SH.CUSTOMERS;


COUNT(DISTINCTCUST_ID)
                   55500
---------------------------
```

5. Similarly, query the `SH.SUPPLEMENTARY_DEMOGRAPHICS` table.

```
SELECT * FROM SH.SUPPLEMENTARY_DEMOGRAPHICS;
```

CUST_ID	EDUCATION	OCCUPATION	HOUSEHOLD_SIZ...	YRS_RESIDENCE ...	AFFINITY_CARD	BULK_PACK_DISKETTE...	FLAT_PANEL_MONITO...	HOME_THEATER_PACKA...	BOOKKEEPING_APPLICATIO...	PRINTER_SUPPLIE...	Y_BOX_GAMES	OS_DOC_SET_KAN...	COMMENTS
102547	10th	Other	1	0	0	1	1	0	0	1	1	0	
101050	10th	Other	1	0	0	1	1	0	0	1	1	0	
100040	11th	Sales	1	0	0	1	1	0	0	1	1	0	
102117	HS-grad	Farming	1	0	0	0	0	0	1	1	1	0	
101074	10th	Handler	1	1	0	1	1	0	0	1	1	0	
104179	10th	Handler	1	1	0	1	1	0	0	1	1	0	
100417	11th	Handler	1	1	0	0	0	0	0	1	1	0	
101146	< Bach.	?	1	1	0	1	1	0	1	1	1	0	

6. To view the count of rows in the `SH.SUPPLEMENTARY_DEMOGRAPHICS` table, run the following statement:

```
SELECT COUNT(*) from SH.SUPPLEMENTARY_DEMOGRAPHICS;


COUNT(*)
      4500
---------------------------
```

7. Create a table called `CUSTOMERDATA` by selecting the required columns from the `SH.CUSTOMERS` and the `SH.SUPPLIMENTARY_DEMOGRAPHICS` tables.

```
%script

CREATE OR REPLACE VIEW CUSTOMERDATA AS
   SELECT a.CUST_ID, a.CUST_GENDER, a.CUST_MARITAL_STATUS,
          a.CUST_YEAR_OF_BIRTH, a.CUST_INCOME_LEVEL, a.CUST_CREDIT_LIMIT,
          b.HOUSEHOLD_SIZE, b.YRS_RESIDENCE, b.Y_BOX_GAMES
   FROM SH.CUSTOMERS a, SH.SUPPLEMENTARY_DEMOGRAPHICS b
   WHERE a.CUST_ID = b.CUST_ID;



View CUSTOMERDATA created.
```

8. View the `CUSTOMERDATA` table.

```
SELECT * FROM CUSTOMERDATA;
```

CUST_ID	CUST_GENDER	CUST_MARITAL_STATU...	CUST_YEAR_OF_BIRT...	CUST_INCOME_LEV...	CUST_CREDIT_LIMIT...	HOUSEHOLD_SIZ...	YRS_RESIDENCE ...	Y_BOX_GAMES ...
103791	M	Divorc.	1952	B: 30,000 - 49,999	3000	2	5	0
100804	F	Divorc.	1943	A: Below 30,000	1500	2	6	0
101610	M	NeverM	1985	I: 170,000 - 189,999	3000	1	0	1
102308	M	NeverM	1980	J: 190,000 - 249,999	11000	2	2	1
100593	M	Married	1963	G: 130,000 - 149,999	1500	3	4	0
100558	M	Married	1964	J: 190,000 - 249,999	11000	3	4	0
103401	M	Divorc.	1975	I: 170,000 - 189,999	10000	2	4	1
102740	F	Divorc.	1929	K: 250,000 - 299,999	15000	2	0	0

9. Find the count of rows in the new `CUSTOMERDATA` table:

```
SELECT COUNT(*) FROM CUSTOMERDATA;


COUNT(*)
      4500
---------------------------
```

10. To view the data type of the columns, run the following statement:

```
%script
DESCRIBE CUSTOMERDATA;


Name                Null?    Type
------------------- -------- ------------
CUST_ID              NOT NULL  NUMBER
CUST_GENDER         NOT NULL  CHAR(1)
CUST_MARITAL_STATUS           VARCHAR2(20)
CUST_YEAR_OF_BIRTH NOT NULL  NUMBER(4)
CUST_INCOME_LEVEL             VARCHAR2(30)
CUST_CREDIT_LIMIT             NUMBER
HOUSEHOLD_SIZE                VARCHAR2(21)
YRS_RESIDENCE                 NUMBER
Y_BOX_GAMES                NUMBER(10)

---------------------------
```

11. To check if there are any missing values (NULL values), run the following statement:

```
SELECT COUNT(*) FROM CUSTOMERDATA WHERE CUST_ID=NULL OR CUST_GENDER=NULL
 OR CUST_MARITAL_STATUS=NULL OR CUST_YEAR_OF_BIRTH=NULL OR CUST_INCOME_LEVEL=NULL
 OR CUST_CREDIT_LIMIT=NULL OR HOUSEHOLD_SIZE=NULL OR YRS_RESIDENCE=NULL OR
Y_BOX_GAMES=NULL;


COUNT(*)
         0
---------------------------
```

NULLs, if found, are automatically handled by the OML algorithms. Alternately, you can manually replace NULLs with `NVL` SQL function.

This completes the data exploration stage. OML supports Automatic Data Preparation (ADP). ADP is enabled through the model settings. When ADP is enabled, the transformations required by the algorithm are performed automatically and embedded in the model. This step is done during the Build Model stage. The commonly used methods of data preparation are binning, normalization, and missing value treatment.

Related Topics

- How ADP Transforms the Data

Build Model

Build your model using your data set. Use the `DBMS_DATA_MINING.CREATE_MODEL2` procedure to build your model and specify the model settings.

To evaluate the model and to accurately assess the performance of the model on the same data, you generally split or separate the data into training and test data. For an unsupervised learning, like Clustering, you do not have labels or predictors to calculate the accuracy or assess the performance. Thus, you can create a model using your data set without splitting. For an unsupervised learning, you don't have a real way of knowing how good your model is. So, a training or a test split is not useful.

Algorithm Selection

Before you build a model, choose the suitable algorithm. You can choose one of the following algorithms to solve a clustering problem:

- *k*-Means
- Expectation Maximization (EM)
- Orthogonal Cluster (O-Cluster)

K-Means does not assume a particular distribution of the data. The *k*-Means algorithm is a distance-based clustering algorithm that partitions the data into a specified number of clusters. The EM algorithm is a probability density estimation technique. EM method is based on assumption that the data has several clusters and each cluster is distributed according to a certain Gaussian distribution. O-Cluster is a neighbor based method. It identifies areas of high density in the data and separates the dense areas into clusters. It is able to cluster data points that forms a certain shape, which sometimes can be a complex pattern like a circle, spiral, or even a tie shape.

K-Means tends to cluster points only close to each other and does not necessarily cluster the data based on the shapes. Therefore, *K*-Means method is the one with the simplest assumption. Thus, it is the clustering method to start with.

The following steps guide you to build your model with the selected algorithm.

- Build your model using the `CREATE_MODEL2` procedure. First, declare a variable to store model settings or hyperparameters. Run the following script:

```
%script

BEGIN DBMS_DATA_MINING.DROP_MODEL('KM_SH_CLUS1');
EXCEPTION WHEN OTHERS THEN NULL; END;
/
DECLARE
    v_setlist DBMS_DATA_MINING.SETTING_LIST;
BEGIN
    v_setlist('ALGO_NAME')        := 'ALGO_KMEANS';
    V_setlist('PREP_AUTO')        := 'ON';
    V_setlist('KMNS_DISTANCE')    := 'KMNS_EUCLIDEAN';
    V_setlist('KMNS_DETAILS')     := 'KMNS_DETAILS_ALL';
    V_setlist('KMNS_ITERATIONS')  := '10';
    V_setlist('KMNS_NUM_BINS')    := '10';
    v_setlist('CLUS_NUM_CLUSTERS'):= '1';

    DBMS_DATA_MINING.CREATE_MODEL2(
        MODEL_NAME          => 'KM_SH_CLUS1',
        MINING_FUNCTION     => 'CLUSTERING',
        DATA_QUERY          => 'select * from CUSTOMERDATA',
        SET_LIST            => v_setlist,
        CASE_ID_COLUMN_NAME => 'CUST_ID');
END;
```

```
PL/SQL procedure successfully completed.
---------------------------
PL/SQL procedure successfully completed.
```

Examine the script:

- `v_setlist` is a variable to store `SETTING_LIST`.
- `SETTING_LIST` specifies model settings or hyperparameters for our model.
- `DBMS_DATA_MINING` is the PL/SQL package used for machine learning. These settings are described in DBMS_DATA_MINING - Model Settings.
- `ALGO_NAME` specifies the algorithm name. Since you are using the *k*-Means as your algorithm, set `ALGO_KMEANS`.
- `PREP_AUTO` is the setting used for Automatic Data Preparation. Here, enable Automatic Data Preparation. The value of the setting is `ON`.
- `KMNS_DISTANCE` is the distance function that measures the similarity between the cases for *k*-Means. The value here is `KMNS_EUCLIDEAN`. This is the default value.
- `KMNS_DETAILS` determines the level of cluster details. `KMNS_DETAILS_ALL` computes cluster hierarchy, record counts, descriptive statistics (means, variances, modes, histograms, and rules).
- `KMNS_ITERATIONS` defines the maximum number of iterations for *k*-Means. The algorithm iterates until either the maximum number of iterations are reached or the minimum Convergence Tolerance, specified in `KMNS_CONV_TOLERANCE`, is satisfied. The default number of iterations is 20.
- `KMNS_NUM_BINS` provides a number of bins in the attribute histogram produced by *k*-Means.
- `CLUS_NUM_CLUSTERS` is the maximum number of leaf clusters generated by a clustering algorithm. The algorithm may return fewer clusters, depending on the data. Enhanced *k*-Means usually produces the exact number of clusters specified by `CLUS_NUM_CLUSTERS`, unless there are fewer distinct data points.

The `CREATE_MODEL2` procedure takes the following parameters:

- `MODEL_NAME`: A unique model name that you will give to your model. The name of the model is in the form [schema_name.]model_name. If you do not specify a schema, then your own schema is used. Here, the model name is `KM_SH_CLUS1`.
- `MINING_FUNCTION`: Specifies the machine learning function. Since you are solving a clustering problem in this use case, select `CLUSTERING`.
- `DATA_QUERY`: A query that provides training data for building the model. Here, the query is `SELECT * FROM CUSTOMERDATA`.
- `SET_LIST`: Specifies `SETTING_LIST`.
- `CASE_ID_COLUMN_NAME`: A unique case identifier column in the build data. In this use case, case_id is `CUST_ID`. If there is a composite key, you must create a new attribute before creating the model. This may involve concatenating values from the columns, or mapping a unique identifier to each distinct combination of values. The `CASE_ID` assists with reproducible results, joining scores for individual customers with other data in, example, scoring data table.

Note:

Any parameters or settings not specified are either system-determined or default values are used.

Evaluate

Evaluate your model by viewing diagnostic metrics and performing quality checks.

You can inspect various rules to see if they reveal new insights for item dependencies (antecedent itemset implying consequent) or for unexpected relationships among items.

Dictionary and Model Views

To obtain information about the model and view model settings, you can query data dictionary views and model detail views. Specific views in model detail views display model statistics which can help you evaluate the model.

The data dictionary views for Oracle Machine Learning are listed in the following table. A database administrator (DBA) and USER versions of the views are also available.

View Name	Description
ALL_MINING_MODELS	Provides information about all accessible machine learning models
ALL_MINING_MODEL_ATTRIBUTES	Provides information about the attributes of all accessible machine learning models
ALL_MINING_MODEL_SETTINGS	Provides information about the configuration settings for all accessible machine learning models
ALL_MINING_MODEL_VIEWS	Provides information about the model views for all accessible machine learning models
ALL_MINING_MODEL_XFORMS	Provides the user-specified transformations embedded in all accessible machine learning models.

Model detail views are specific to the algorithm. You can obtain more insights about the model you created by viewing the model detail views. The names of model detail views begin with DM$xx where xx corresponds to the view prefix. See Model Detail Views.

The following steps help you to view different dictionary views and model detail views.

1. Run the following statement to view the settings in `USER_MINING_MODEL_SETTINGS`:

```
%script
SELECT SETTING_NAME, SETTING_VALUE
FROM USER_MINING_MODEL_SETTINGS
WHERE MODEL_NAME = 'KM_SH_CLUS1'
ORDER BY SETTING_NAME;


SETTING_NAME                   SETTING_VALUE
ALGO_NAME                      ALGO_KMEANS
CLUS_NUM_CLUSTERS              1
KMNS_CONV_TOLERANCE            .001
KMNS_DETAILS                   KMNS_DETAILS_ALL
KMNS_DISTANCE                  KMNS_EUCLIDEAN
KMNS_ITERATIONS                3
KMNS_MIN_PCT_ATTR_SUPPORT      .1
```

```
KMNS_NUM_BINS                  10
KMNS_RANDOM_SEED               0
KMNS_SPLIT_CRITERION           KMNS_VARIANCE
ODMS_DETAILS                   ODMS_ENABLE
ODMS_MISSING_VALUE_TREATMENT   ODMS_MISSING_VALUE_AUTO
ODMS_SAMPLING                  ODMS_SAMPLING_DISABLE
PREP_AUTO                      ON


14 rows selected.


---------------------------
```

2. Run the following statement to see attribute information in `USER_MINING_MODEL_ATTRIBUTES` view:

```
%script
SELECT ATTRIBUTE_NAME, ATTRIBUTE_TYPE
FROM USER_MINING_MODEL_ATTRIBUTES
WHERE MODEL_NAME = 'KM_SH_CLUS1'
ORDER BY ATTRIBUTE_NAME;


ATTRIBUTE_NAME         ATTRIBUTE_TYPE
CUST_CREDIT_LIMIT      NUMERICAL
CUST_GENDER            CATEGORICAL
CUST_INCOME_LEVEL      CATEGORICAL
CUST_MARITAL_STATUS    CATEGORICAL
CUST_YEAR_OF_BIRTH     NUMERICAL
HOUSEHOLD_SIZE         CATEGORICAL
YRS_RESIDENCE          NUMERICAL
Y_BOX_GAMES            NUMERICAL


8 rows selected.
---------------------------
```

3. Run the following statement to see information on various views in `USER_MINING_MODEL_VIEWS`:

```
%script
SELECT VIEW_NAME, VIEW_TYPE FROM USER_MINING_MODEL_VIEWS
WHERE MODEL_NAME='KM_SH_CLUS1'
ORDER BY VIEW_NAME;


VIEW_NAME          VIEW_TYPE
DM$VAKM_SH_CLUS1   Clustering Attribute Statistics
DM$VCKM_SH_CLUS1   k-Means Scoring Centroids
DM$VDKM_SH_CLUS1   Clustering Description
DM$VGKM_SH_CLUS1   Global Name-Value Pairs
DM$VHKM_SH_CLUS1   Clustering Histograms
DM$VNKM_SH_CLUS1   Normalization and Missing Value Handling
DM$VRKM_SH_CLUS1   Clustering Rules
```

```
DM$VSKM_SH_CLUS1   Computed Settings
DM$VWKM_SH_CLUS1   Model Build Alerts


9 rows selected.


---------------------------
```

4. Now, view the Clustering Description model detail view:

```
SELECT CLUSTER_ID CLU_ID, RECORD_COUNT REC_CNT, PARENT,
       TREE_LEVEL, ROUND(TO_NUMBER(DISPERSION),3) DISPERSION
FROM   DM$VDKM_SH_CLUS1
ORDER BY CLUSTER_ID;


CLU_ID   REC_CNT   PARENT   TREE_LEVEL   DISPERSION
       1      4500                    1        6.731

---------------------------
```

5. To see the leaf cluster IDs, run the following query:

   Oracle supports hierarchical clustering. In hierarchical clustering, the data points having similar characteristics are grouped together. The cluster hierarchy is represented as a tree structure. The leaf clusters are the final clusters generated by the algorithm. Clusters higher up in the hierarchy are intermediate clusters.

```
SELECT CLUSTER_ID
FROM   DM$VDKM_SH_CLUS1
WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL
ORDER BY CLUSTER_ID;


CLUSTER_ID
           1
---------------------------
```

   Examine the query:

   `LEFT_CHILD_ID IS NULL`: Outputs the leaf nodes on the left of the hierarchical tree

   `RIGHT_CHILD_ID IS NULL`: Outputs the leaf nodes on the right of the hierarchical tree

6. View the dispersion details or the cluster description for the leaf cluster IDs:

   Dispersion is a measure of cluster quality and computationally it is the sum of squared error. This also indicates the quality of the cluster model.

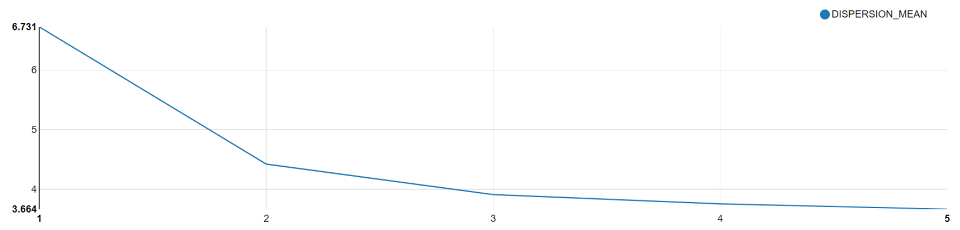
```
%script
SELECT CLUSTER_ID CLU_ID, RECORD_COUNT REC_CNT, PARENT,
       TREE_LEVEL, ROUND(TO_NUMBER(DISPERSION),3) DISPERSION
FROM   DM$VDKM_SH_CLUS1
WHERE CLUSTER_ID IN (SELECT CLUSTER_ID
                     FROM   DM$VDKM_SH_CLUS1
```

```
                    WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL)
ORDER BY CLUSTER_ID;


CLU_ID   REC_CNT   PARENT   TREE_LEVEL   DISPERSION
       1      4500                     1        6.731

--------------------------
```

7. To determine the optimal value of K (or the number of clusters) for the data, visualize the data with an Elbow method.

   The Elbow method is done with the leaf clusters. In cluster analysis, the elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the variance (or dispersion) as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use.

```
%sql
SELECT 1 ID, AVG(DISPERSION) DISPERSION_MEAN
FROM   DM$VDKM_SH_CLUS1
WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL
UNION
SELECT 2 ID, AVG(DISPERSION) DISPERSION_MEAN
FROM   DM$VDKM_SH_CLUS2
WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL
UNION
SELECT 3 ID, AVG(DISPERSION) DISPERSION_MEAN
FROM   DM$VDKM_SH_CLUS3
WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL
UNION
SELECT 4 ID, AVG(DISPERSION) DISPERSION_MEAN
FROM   DM$VDKM_SH_CLUS4
WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL
UNION
SELECT 5 ID, AVG(DISPERSION) DISPERSION_MEAN
FROM   DM$VDKM_SH_CLUS5
WHERE LEFT_CHILD_ID IS NULL AND RIGHT_CHILD_ID IS NULL;
```



ID	DISPERSION_MEAN
1	6.730705777777758
2	4.421941433706115
3	3.9079350267325625
4	3.752986215534802
5	3.663727003275104

From the resultant graph, the curve flattens after 3 or the dispersion value flattens after ID 3, which means that the optimal value of K (or the most suitable number of clusters that the data must be segmented into) is 3.

> **Note:**
>
> In Oracle SQL Developer, a visual aid to view the graph is not applicable. You can only compute the dispersion scores.

8. To view the Attribute details of the `KM_SH_CLUS3` model, run the following statement:

   The Attribute Details view displays statistics like mean, median, and mode of your model.

```
%script
SELECT CLUSTER_ID, ATTRIBUTE_NAME, ATTRIBUTE_SUBNAME, MEAN, VARIANCE,
MODE_VALUE
FROM  DM$VAKM_SH_CLUS3;


CLUSTER_ID   ATTRIBUTE_NAME        ATTRIBUTE_SUBNAME
MEAN                   VARIANCE                MODE_VALUE
           1 CUST_CREDIT_LIMIT
7924.222222222223      15914238.670321768
           1 CUST_YEAR_OF_BIRTH
1964.6244444444444     187.1267639722414
           1 YRS_RESIDENCE
4.021999999999995      3.617430984663253
           1 Y_BOX_GAMES
0.31244444444444447    0.2148706626163839
```

```
           1
CUST_GENDER
           M
           1
CUST_INCOME_LEVEL
           J: 190,000 - 249,999
           1
CUST_MARITAL_STATUS
           Married
           1
HOUSEHOLD_SIZE
           3
           2 CUST_CREDIT_LIMIT
7833.002645502645     15543554.858080933
           2 CUST_YEAR_OF_BIRTH
1957.631283068783     121.54941469457282
           2 YRS_RESIDENCE
4.8611111111111045    2.7838791487484835
           2 Y_BOX_GAMES
0.0                   0.0
           2
CUST_GENDER
           M
           2
CUST_INCOME_LEVEL
           J: 190,000 - 249,999

CLUSTER_ID   ATTRIBUTE_NAME        ATTRIBUTE_SUBNAME
MEAN                    VARIANCE               MODE_VALUE
           2
CUST_MARITAL_STATUS
           Married
           2
HOUSEHOLD_SIZE
           3
           3 CUST_CREDIT_LIMIT
8111.111111111114    16632730.696798513
           3 CUST_YEAR_OF_BIRTH
1978.9518970189702   15.976667585319932
           3 YRS_RESIDENCE
2.3028455284552827   0.9272054568003305
           3 Y_BOX_GAMES
0.9525745257452575   0.04520692664553768
           3
CUST_GENDER
           M
           3
CUST_INCOME_LEVEL
           J: 190,000 - 249,999
           3
CUST_MARITAL_STATUS
           NeverM
           3
HOUSEHOLD_SIZE
           1
```

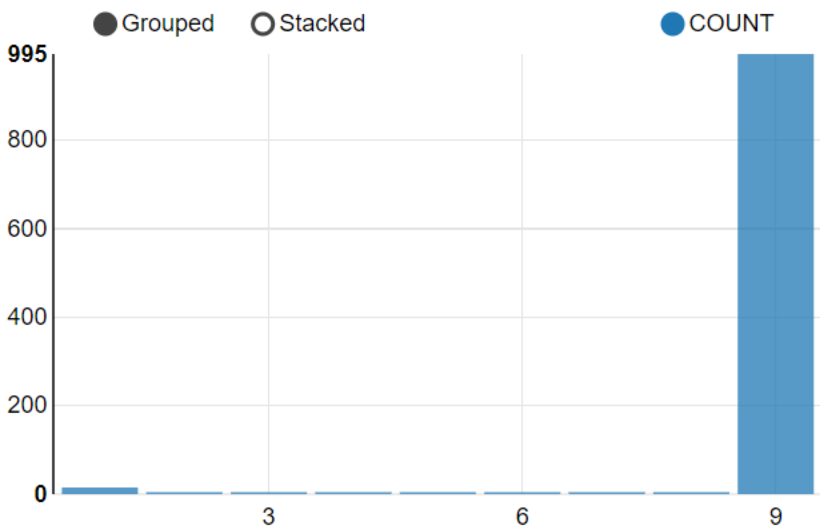
```
           4 CUST_CREDIT_LIMIT
3126.6094420600857   2978559.2320826976
           4 CUST_YEAR_OF_BIRTH
1978.4978540772531   22.143006137800537
           4 YRS_RESIDENCE
2.270386266094421    0.8944759795099003
           4 Y_BOX_GAMES
0.8819742489270386   0.10431953481932726

CLUSTER_ID   ATTRIBUTE_NAME        ATTRIBUTE_SUBNAME
MEAN                  VARIANCE                 MODE_VALUE
           4
CUST_GENDER
                 F
           4
CUST_INCOME_LEVEL
                 B: 30,000 - 49,999
           4
CUST_MARITAL_STATUS
                 NeverM
           4
HOUSEHOLD_SIZE
                 1
           5 CUST_CREDIT_LIMIT
10410.891089108914   6172923.883072166
           5 CUST_YEAR_OF_BIRTH
1979.1613861386138   13.01158975164117
           5 YRS_RESIDENCE
2.3178217821782146   0.9424967372852242
           5 Y_BOX_GAMES
0.9851485148514851   0.01464541895220246
           5
CUST_GENDER
                 M
           5
CUST_INCOME_LEVEL
                 J: 190,000 - 249,999
           5
CUST_MARITAL_STATUS
                 NeverM
           5
HOUSEHOLD_SIZE
                 1


40 rows selected.
```

Notice that Cluster ID 5 has the highest mean for `Y_BOX_GAMES` users and has the highest `CUST_CREDIT_LIMIT`.

9. Now, for the model `KM_SH_CLUS3`, view the histogram details with specific attributes for each leaf cluster. For this use-case, view the histogram details for `Y_BOX_GAMES` and `CUST_INCOME_LEVEL` attributes. In this step, leaf cluster ID 5 and the attribute `Y_BOX_GAMES` are picked.

```
%sql

SELECT CLUSTER_ID, ATTRIBUTE_NAME, ATTRIBUTE_SUBNAME,
       BIN_ID, LOWER_BIN_BOUNDARY, UPPER_BIN_BOUNDARY, ATTRIBUTE_VALUE, COUNT
FROM DM$VHKM_SH_CLUS3
WHERE CLUSTER_ID = 5 AND ATTRIBUTE_NAME = 'Y_BOX_GAMES'
ORDER BY BIN_ID;
```

In OML Notebooks, click the bar plot icon and expand settings. Drag `BIN_ID` to **keys** and **COUNT** to values.

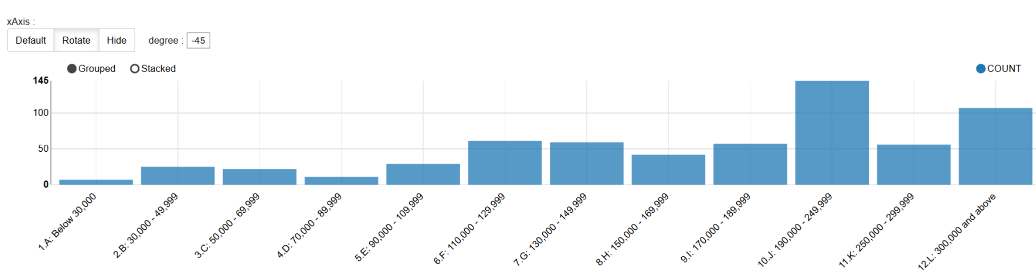


From this histogram, you can see that Cluster ID 5 is grouped into bins showing the count of `Y_BOX_GAMES` users. Bin 9 has the highest count of `Y_BOX_GAMES` users.

10. Similarly, for Cluster ID 5, view the histogram details for the `CUST_INCOME_LEVEL` attribute.

```
%sql

SELECT CLUSTER_ID, ATTRIBUTE_NAME, ATTRIBUTE_SUBNAME,
       BIN_ID, LOWER_BIN_BOUNDARY, UPPER_BIN_BOUNDARY, ATTRIBUTE_VALUE, COUNT
FROM DM$VHKM_SH_CLUS3
WHERE CLUSTER_ID = 5 AND ATTRIBUTE_NAME = 'CUST_INCOME_LEVEL'
ORDER BY BIN_ID;
```

In OML Notebooks, click the bar plot icon and expand settings. Drag `BIN_ID` and `ATTRIBUTE_VALUE` to **keys** and **COUNT** to values. In the xAxis options, click **Rotate**.



In this histogram, Cluster ID 5 is grouped into bins showing the count of customers with `CUST_INCOME_LEVEL` and indicates that the highest number of customers draw a salary package between 190,000 - 249,999 yearly.

11. Now, view the Rule details of leaf clusters (2, 4, and 5) to check the support and confidence level.

    Support and confidence are metrics that describe the relationships between clustering rules and cases. Support is the percentage of cases for which the rule holds. Confidence is the probability that a case described by this rule is actually assigned to the cluster.

```
%script

SELECT CLUSTER_ID, ATTRIBUTE_NAME, ATTRIBUTE_SUBNAME, OPERATOR,
       NUMERIC_VALUE, ATTRIBUTE_VALUE, SUPPORT, ROUND(CONFIDENCE,3)
CONFIDENCE
FROM DM$VRKM_SH_CLUS3
WHERE cluster_id IN (SELECT cluster_id
                     FROM DM$VDKM_SH_CLUS3
                     WHERE LEFT_CHILD_ID is NULL and RIGHT_CHILD_ID is NULL)
ORDER BY CLUSTER_ID, ATTRIBUTE_NAME, ATTRIBUTE_SUBNAME, OPERATOR,
NUMERIC_VALUE, ATTRIBUTE_VALUE;



CLUSTER_ID   ATTRIBUTE_NAME        ATTRIBUTE_SUBNAME   OPERATOR
NUMERIC_VALUE   ATTRIBUTE_VALUE        SUPPORT   CONFIDENCE
           2 CUST_CREDIT_LIMIT                         <=
15000.0                                    3024          0
           2 CUST_CREDIT_LIMIT                         >=
1500.0                                     3024          0
           2 CUST_GENDER
IN                         F                           3024
0.002
           2 CUST_GENDER
IN                         M                           3024
0.002
           2 CUST_INCOME_LEVEL
IN                         B: 30,000 - 49,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         E: 90,000 - 109,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         F: 110,000 - 129,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         G: 130,000 - 149,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         H: 150,000 - 169,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         I: 170,000 - 189,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         J: 190,000 - 249,999
2750            0
           2 CUST_INCOME_LEVEL
IN                         K: 250,000 - 299,999
2750            0
```

```
           2 CUST_INCOME_LEVEL
IN                          L: 300,000 and above        2750            0
           2 CUST_MARITAL_STATUS
IN                          Divorc.                     2720        0.014

CLUSTER_ID   ATTRIBUTE_NAME          ATTRIBUTE_SUBNAME    OPERATOR
NUMERIC_VALUE        ATTRIBUTE_VALUE    SUPPORT   CONFIDENCE
           2 CUST_MARITAL_STATUS
IN                               Married                 2720        0.014
           2 CUST_MARITAL_STATUS
IN                               NeverM                  2720        0.014
           2 CUST_YEAR_OF_BIRTH                          <=
1977.888888888889                             2854        0.041
           2 CUST_YEAR_OF_BIRTH                          >
1937.3333333333333                            2854        0.041
           2 HOUSEHOLD_SIZE
IN                               2                       2699        0.016
           2 HOUSEHOLD_SIZE
IN                               3                       2699        0.016
           2 HOUSEHOLD_SIZE
IN                               9+                      2699        0.016
           2 YRS_RESIDENCE                               <=
7.777777777777778                             2804        0.019
           2 YRS_RESIDENCE                               >
1.5555555555555556                            2804        0.019
           2 Y_BOX_GAMES                                 <=
0.1111111111111111                            3024        0.056
           2 Y_BOX_GAMES                                 >=
0.0                                           3024        0.056
           4 CUST_CREDIT_LIMIT                           <=
7500.0                                         466        0.128
           4 CUST_CREDIT_LIMIT                           >=
1500.0                                         466        0.128
           4 CUST_GENDER
IN                               F                        466        0.023

CLUSTER_ID   ATTRIBUTE_NAME          ATTRIBUTE_SUBNAME    OPERATOR
NUMERIC_VALUE        ATTRIBUTE_VALUE          SUPPORT   CONFIDENCE
           4 CUST_GENDER
IN                               M                             466
0.023
           4 CUST_INCOME_LEVEL
IN                               A: Below 30,000               466
0.079
           4 CUST_INCOME_LEVEL
IN                               B: 30,000 - 49,999            466
0.079
           4 CUST_INCOME_LEVEL
IN                               C: 50,000 - 69,999            466
0.079
           4 CUST_INCOME_LEVEL
IN                               D: 70,000 - 89,999            466
0.079
           4 CUST_INCOME_LEVEL
IN                               E: 90,000 - 109,999           466
```

```
0.079
           4 CUST_INCOME_LEVEL
IN                              F: 110,000 - 129,999
466        0.079
           4 CUST_INCOME_LEVEL
IN                              G: 130,000 - 149,999
466        0.079
           4 CUST_INCOME_LEVEL
IN                              H: 150,000 - 169,999
466        0.079
           4 CUST_INCOME_LEVEL
IN                              I: 170,000 - 189,999
466        0.079
           4 CUST_MARITAL_STATUS
IN                              Married
413        0.043
           4 CUST_MARITAL_STATUS
IN                              NeverM
413        0.043
           4 CUST_YEAR_OF_BIRTH                       <=
1986.0                                           451        0.103
           4 CUST_YEAR_OF_BIRTH                       >
1969.7777777777778                               451        0.103

CLUSTER_ID   ATTRIBUTE_NAME      ATTRIBUTE_SUBNAME   OPERATOR
NUMERIC_VALUE       ATTRIBUTE_VALUE         SUPPORT   CONFIDENCE
           4 HOUSEHOLD_SIZE
IN                             1
418        0.043
           4 HOUSEHOLD_SIZE
IN                             2
418        0.043
           4 HOUSEHOLD_SIZE
IN                             3
418        0.043
           4 HOUSEHOLD_SIZE
IN                             9+
418        0.043
           4 YRS_RESIDENCE                          <=
4.666666666666667                               464        0.086
           4 YRS_RESIDENCE                          >=
0.0                                             464        0.086
           4 Y_BOX_GAMES                            <=
1.0                                             466        0.083
           4 Y_BOX_GAMES                            >=
0.0                                             466        0.083
           5 CUST_CREDIT_LIMIT                      <=
15000.0                                        1010        0.056
           5 CUST_CREDIT_LIMIT                      >
6000.0                                         1010        0.056
           5 CUST_GENDER
IN                             F
1010        0.002
           5 CUST_GENDER
IN                             M
```

```
1010         0.002
           5 CUST_INCOME_LEVEL
IN                           F: 110,000 - 129,999        906
0.024
           5 CUST_INCOME_LEVEL
IN                           G: 130,000 - 149,999        906
0.024

CLUSTER_ID   ATTRIBUTE_NAME        ATTRIBUTE_SUBNAME   OPERATOR
NUMERIC_VALUE        ATTRIBUTE_VALUE        SUPPORT   CONFIDENCE
           5 CUST_INCOME_LEVEL
IN                            I: 170,000 - 189,999        906
0.024
           5 CUST_INCOME_LEVEL
IN                            J: 190,000 - 249,999        906
0.024
           5 CUST_INCOME_LEVEL
IN                            K: 250,000 - 299,999        906
0.024
           5 CUST_INCOME_LEVEL
IN                            L: 300,000 and above        906
0.024
           5 CUST_MARITAL_STATUS
IN                            Married                     944
0.046
           5 CUST_MARITAL_STATUS
IN                            NeverM                      944
0.046
           5 CUST_YEAR_OF_BIRTH                       <=
1986.0                                    1003         0.12
           5 CUST_YEAR_OF_BIRTH                       >
1969.7777777777778                        1003         0.12
           5 HOUSEHOLD_SIZE
IN                            1                           859
0.036
           5 HOUSEHOLD_SIZE
IN                            2                           859
0.036
           5 HOUSEHOLD_SIZE
IN                            3                           859
0.036
           5 YRS_RESIDENCE                            <=
4.666666666666667                          993        0.079
           5 YRS_RESIDENCE                            >=
0.0                                        993        0.079
           5 Y_BOX_GAMES                              <=
1.0                                        995        0.136

CLUSTER_ID   ATTRIBUTE_NAME   ATTRIBUTE_SUBNAME   OPERATOR
NUMERIC_VALUE        ATTRIBUTE_VALUE   SUPPORT   CONFIDENCE
           5 Y_BOX_GAMES                         >
0.8888888888888888                    995        0.136


71 rows selected.
```

```
---------------------------
```

12. To view the size of each cluster, run the following statement:

    In OML Notebooks, you can also click the bar icon or the pie chart icon to view the bar graph or the pie chart.

    ```
    %sql
    SELECT CLUSTER_ID(KM_SH_CLUS3 USING *) AS CLUS, COUNT(*) AS CNT
    FROM CUSTOMERDATA
    GROUP BY CLUSTER_ID(KM_SH_CLUS3 USING *)
    ORDER BY CNT DESC;



    CLUS   CNT
         2   3024
         5   1010
         4    466
    ---------------------------
    ```

Score

Scoring involves applying the model to the target data. Use `CLUSTER_PROBABILITY` function to predict the clusters. For Clustering, "scoring" involves assigning each record to a cluster, with a certain probability. However, one can also obtain the probability of a record belonging to each cluster.

1. In the following step, you are scoring the probability of the top 10 customers that belong to cluster 5.

    ```
    %script

    SELECT CUST_ID,
           ROUND(CLUSTER_PROBABILITY(KM_SH_CLUS3, 5 USING *),3)
           PROB
    FROM CUSTOMERDATA
    WHERE rownum < 10
    ORDER BY PROB DESC;



    CUST_ID   PROB
       102308   0.539
       101232   0.502
       101610   0.374
       102828   0.303
       100134   0.302
       103948   0.297
       100696    0.25
       103791   0.141
       100804   0.104


    9 rows selected.

    ---------------------------
    ```

2. To score the cluster ID of a given CUST_ID (customer), for this use case, you must target customers who have already purchased Y_BOX_GAMES and with high credit limit, to sell the new game product. In the previous stage, you have identified that cluster 5 has highest customers who have already purchased Y_BOX_GAMES with mean CUST_CREDIT_LIMIT of 10410. So, the target group is cluster ID 5. To score for a given CUST_ID (102308) and display the probability score, run the following query :

   ```
   %sql
   SELECT CLUSTER_ID(KM_SH_CLUS3 USING *) AS CLUSTER_ID, round (CLUSTER_PROBABILITY
   (KM_SH_CLUS3 USING *),3) AS PROB
     FROM CUSTOMERDATA
   where cust_id = 102308;
   ```

   ```
   CLUSTER_ID   PROB
              5   0.539
   ---------------------------
   ```

   Examine the query:

   - CLUSTER_ID(KM_SH_CLUS3 USING *) AS CLUSTER_ID: Provides CLUSTER_ID from the KM_SH_CLUS3 model.
   - round(CLUSTER_PROBABILITY(KM_SH_CLUS3 USING *),2) AS PROB: Provides cluster probability using KM_SH_CLUS3 model. ROUND (n,integer) returns results of CLUSTER_PROBABILITY rounded to n integer places to the right. Here, it is four places.

3. Additionally, you can obtain the probability of a record belonging to each cluster (such as 5, 3, 2) by running the following query:

   ```
   %script
   select CLUSTER_PROBABILITY(KM_SH_CLUS3,
         5 USING *) from CUSTOMERDATA;
   ```

   ```
   CLUSTER_PROBABILITY(KM_SH_CLUS3,5USING*)
   0.30701266050607
   0.3064062868515786
   0.2862730847381108
   0.2868527181838429
   0.3721982825972361
   0.2816026555211009
   0.30936576857241027
   0.3051489029060863
   0.1915573544647028
   0.25158448263351973
   0.37204422449011026
   0.3064062868515786
   0.35693390244389295
   0.1902596096427133
   ...
   ```

To conclude, you have successfully segmented the population into different clusters and determined that cluster 5 has the target population for the use case. You can safely target customers in cluster 5 to sell a new game product. You can select the customer IDs from Step 1. You can also display a full list of target customers by removing the `WHERE` clause.

Time Series

You are working in a electronic shop and the sale of laptops and tablets have gone up for the last two quarters. You want to forecast the sale of your products for the next four quarters on the basis of the historical timestamped data you have collected. You forecast the sales using the Exponential Smoothing algorithm as you are predicting the changes over evenly spaced intervals of time using the historical data.

Before you start your OML4SQL use case journey, ensure that you have the following:

- Data Set
  The data set used for this use case is from the SH schema. The SH schema can be readily accessed in Oracle Autonomous Database. For on-premises databases, the schema is installed during the installation or can be manually installed by downloading the scripts. See Installing the Sample Schemas.

  You will use the `SALES` table from the `SH` schema. You can access the table by running the `SELECT` statements in OML Notebooks.
- Database
  Select a database out of the following options:
  – Get your FREE cloud account. Go to https://cloud.oracle.com/database and select Oracle Database Cloud Service (DBCS), or Oracle Autonomous Database. Create an account and create an instance. See Autonomous Database Quick Start Workshop.
  – Download the latest version of Oracle Database (on premises).
- Machine Learning Tools
  Depending on your database selection,
  – Use OML Notebooks for Oracle Autonomous Database.
  – Install and use Oracle SQL Developer connected to an on-premises database or DBCS. See Installing and Getting Started with SQL Developer.
- Other Requirements
  Data Mining Privileges (this is automatically set for ADW). See System Privileges for Oracle Machine Learning for SQL.

Related Topics

- Create a Notebook
- Edit your Notebook
- Uninstalling HR Schema

Load Data

Access the data set from the SH schema and explore the data to understand the attributes.

> **Remember:**
>
> The data set used for this use case is from the SH schema. The SH schema can be readily accessed in Oracle Autonomous Database. For on-premises databases, the schema is installed during the installation or can be manually installed by downloading the scripts. See Installing the Sample Schemas.

To understand the data, you will perform the following:

- Access the data.
- Examine the various attributes or columns of the data set.
- Assess data quality (by exploring the data).

Access Data

You will use SALES table data from the SH schema.

Examine Data

The following table displays information about the attributes from SALES:

Attribute Name	Information
PROD_ID	The ID of the product
CUST_ID	The ID of the customer
TIME_ID	The timestamp of the purchase of the product in yyy-mm-dd hh:mm:ss format
CHANNEL_ID	The channel ID of the channel sales data
PROMO_ID	The product promotion ID
QUANTITY_SOLD	The number of items sold
AMOUNT_SOLD	The amount or sales data

Identify Target Variable

In this use case, the task is to train a model that predicts the amount sold. Therefore, the target variable is the attribute AMOUNT_SOLD.

Explore Data

Explore the data to understand and assess the quality of the data. At this stage assess the data to identify data types and noise in the data. Look for missing values and numeric outlier values.

If you are working with Oracle Autonomous Database, you can use the Oracle Machine Learning (OML) Notebooks for your data science project, including assessing data quality. If you are using an on-premise Oracle Database, you can use the Oracle SQL Developer to assess data quality. Query the `SH` schema as described.

> **Note:**
>
> Each record in the database is called a case and each case is identified by a `case_id`. In this use case `TIME_ID` is the `case_id` as it is an independent variable and you are forecasting the sales for evenly spaced time.

The following steps help you with exploratory analysis of the data.

1. View the data in the `SH.SALES` table by running the following statement:

```
SELECT * FROM SH.SALES;
```

2. To find the number of rows in `SH.SALES` table, run the following statement:

```
%script
SELECT COUNT(*) from SH.SALES;



COUNT(*)
    918843
---------------------------
```

3. Find the distinct users in the table, run the following query:

```
%sql SELECT COUNT (DISTINCT CUST_ID) FROM SH.SALES;



COUNT(DISTINCTCUST_ID)
                    7059
---------------------------
```

4. To view the datatype of the sales table, run the following query:

```
%script
DESCRIBE SH.SALES;



Name              Null?         Type
-------------     --------      ------------
PROD_ID      NOT NULL      NUMBER
CUST_ID      NOT NULL      NUMBER
TIME_ID       NOT NULL      DATE
CHANNEL_ID      NOT NULL      NUMBER
PROMO_ID      NOT NULL      NUMBER
QUANTITY_SOLD      NOT NULL      NUMBER(10,2)
AMOUNT_SOLD      NOT NULL      NUMBER(10,2)
```

```
---------------------------
```

5. To view all the NULLs and missing values, run the following query:

```
%sql SELECT COUNT(*) FROM SH.SALES WHERE PROD_ID=NULL OR CUST_ID=NULL OR
    TIME_ID=NULL OR CHANNEL_ID=NULL OR PROMO_ID=NULL OR
QUANTITY_SOLD=NULL OR
    AMOUNT_SOLD=NULL;



COUNT(*)
         0
---------------------------
```

   NULLs, if found, are automatically handled by the OML algorithms.

6. Now, prepare a view called `ESM_SH_DATA` by selecting the necessary columns from `SH.SALES` table. For this use case, select `TIME_ID` and `AMOUNT_SOLD`.

```
%script
CREATE OR REPLACE VIEW ESM_SH_DATA AS
  SELECT TIME_ID, AMOUNT_SOLD FROM SH.SALES;



View ESM_SH_DATA created.
---------------------------
```

7. Count the number of rows to ensure that we have the same amount of data. Run the following query:

```
%script
SELECT count(*) from ESM_SH_DATA;



COUNT(*)
    918843
---------------------------
```

This completes the data understanding and data exploration stage. Time series data can contain missing values. The setting `EXSM_SETMISSING` can be used to specify how to handle missing values. The special value `EXSM_MISS_AUTO` indicates that, if the series contains missing values it is to be treated as an irregular time series. The Automatic Data Preparation (ADP) setting does not impact this data for time series. See How ADP Transforms the Data to understand how ADP prepares the data for some algorithms.

Build Model

To build a model using the time series data, you will use Exponential Smoothing algorithm on the ESM_SH_DATA view that is generated during the exploratory stage.

Oracle offers the Exponential Smoothing algorithm for time series.
Exponential smoothing is a forecasting method for time series data. It is a moving average method where exponentially decreasing weights are assigned to past observations. Components of Exponential Smoothing Model (ESM) such as trend and seasonality extensions, can have an additive or multiplicative form. For additive forms, the amplitude of the variation is independent of the level, whereas for multiplicative forms, the variation is connected to the level. The simpler additive models assume that error or noise, trend, and seasonality are linear effects within the recursive formulation.

To build a model using a supervised learning algorithm you may use a subset of the data into training and test data. Time series models usually use historical data to predict the future. This is different from model validation for classification and regression, which normally involves splitting data randomly into training and test sets. In this use case, there is no need to split the data set because the model is always predicting the current value based on information from the past. This means that although it seems that you train and test on the same data set, but when the model is applied, the forecast is always based on the previous date. In this use case, you will use the ESM_SH_DATA view.

1. To see the data in the ESM_SH_DATA view, run the following statement:

   ```
   %sql
   SELECT * from ESM_SH_DATA;


   TIME_ID      AMOUNT_SOLD
   20-JAN-98          1205.99
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-APR-98          1250.25
   05-JUL-98          1210.21
   05-JUL-98          1210.21
   05-JUL-98          1210.21
   05-JUL-98          1210.21
   05-JUL-98          1210.21 ...
   ```

2. Build a model with the ESM_SH_DATA table, run the following script:

   ```
    %script

       BEGIN DBMS_DATA_MINING.DROP_MODEL('ESM_SALES_FORECAST_2');
       EXCEPTION WHEN OTHERS THEN NULL; END;
       /
       DECLARE
   ```

```
    v_setlist DBMS_DATA_MINING.SETTING_LIST;
BEGIN
    v_setlist('ALGO_NAME')            := 'ALGO_EXPONENTIAL_SMOOTHING';
    V_setlist('EXSM_INTERVAL')        := 'EXSM_INTERVAL_QTR';
    V_setlist('EXSM_PREDICTION_STEP') := '4';
    V_setlist('EXSM_MODEL')           := 'EXSM_WINTERS';
    V_setlist('EXSM_SEASONALITY')     := '4';
 V_setlist('EXSM_SETMISSING')     := 'EXSM_MISS_AUTO');

    DBMS_DATA_MINING.CREATE_MODEL2(
        MODEL_NAME          => 'ESM_SALES_FORECAST_1',
        MINING_FUNCTION     => 'TIME_SERIES',
        DATA_QUERY          => 'select * from ESM_SH_DATA',
        SET_LIST            => v_setlst,
        CASE_ID_COLUMN_NAME => 'TIME_ID',
        TARGET_COLUMN_NAME  =>'AMOUNT_SOLD');
END;


PL/SQL procedure successfully completed.
    ---------------------------
    PL/SQL procedure successfully completed.
```

Examine the script:

- `v_setlist` is a variable to store `SETTING_LIST`.
- `SETTING_LIST` specifies model settings or hyperparameters for the model.
- `DBMS_DATA_MINING` is the PL/SQL package used for machine learning. These settings are described in DBMS_DATA_MINING - Model Settings.
- `ALGO_NAME` specifies the algorithm name. Since you are using Exponential Smoothing as the algorithm, the value of the setting is `ALGO_EXPONENTIAL_SMOOTHING`.
- `EXSM_INTERVAL` indicates the interval of the data set or a unit of interval size. For example, day, week, month, and so on. You want to predict for quarterly sales. Hence, the setting is `EXSM_INTERVAL_QTR`. This setting applies only to the time column with datetime type.
- `EXSM_PREDICTION_STEP` specifies how many predictions to make. You want to display each value representing a quarter. Hence, a value of 4 gives four values ahead prediction.
- `EXSM_MODEL` specifies the type of exponential smoothing model to be used. Here the value is `EXSM_HW`. The Holt-Winters triple exponential smoothing model with additive trend and multiplicative seasonality is applied. This type of model considers various combinations of additive and multiplicative trend, seasonality and error, with and without trend damping. Other options are `EXSM_SIMPLE`, `EXSM_SIMPLE_MULT`, `EXSM_HOLT`, `EXSM_HOLT_DMP`, `EXSM_MUL_TRND`, `EXSM_MULTRD_DMP`, `EXSM_SEAS_ADD`, `EXSM_SEAS_MUL`, `EXSM_HW`, `EXSM_HW_DMP`, `EXSM_HW_ADDSEA`, `EXSM_DHW_ADDSEA`, `EXSM_HWMT`, `EXSM_HWMT_DMP`.
- `EXSM_SEASONALITY` indicates how long a season lasts. The parameter specifies a positive integer value as the length of seasonal cycle. The value it takes must be larger than `1`. For example, 4 means that every group of four values forms a seasonal cycle.

- EXSM_SETMISSING specifies how to handle missing values. In time series, the special value EXSM_MISS_AUTO indicates that, if the series contains missing values it is to be treated as an irregular time series.

The CREATE_MODEL2 procedure has the following settings:

- MODEL_NAME: A unique name that you will give to the model. Name of the model in the form [schema_name.]model_name. If you do not specify a schema, then your own schema is used. Here, the model name is ESM_SALES_FORECAST_1.
- MINING_FUNCTION: Specifies the machine learning function. Since it is a time series problem, select TIME_SERIES.
- DATA_QUERY: A query that provides training data for building the model. Here, the query is SELECT * FROM ESM_SH_DATA.
- SET_LIST: Specifies SETTING_LIST.
- CASE_ID_COLUMN_NAME: A unique case identifier column in the training data. In this use case, case_id is TIME_ID. If there is a composite key, you must create a new attribute before creating the model.
- TARGET_COLUMN_NAME: Specifies the column that is to be predicted. Also referred to as the target variable of the model. In other words, the value the model predicts. In this use case, you are predicting the sale of products in terms of their dollar price. Therefore, in this use case, the TARGET_COLUMN_NAME is AMOUNT_SOLD.

> **Note:**
>
> Any parameters or settings not specified are either system-determined or default values are used.

Evaluate

Evaluate your model by viewing diagnostic metrics and performing quality checks.

You can inspect various rules to see if they reveal new insights for item dependencies (antecedent itemset implying consequent) or for unexpected relationships among items.

Dictionary and Model Views

To obtain information about the model and view model settings, you can query data dictionary views and model detail views. Specific views in model detail views display model statistics which can help you evaluate the model.

By examining various statistics in the model detail views, you can compare models to arrive at one model that satisfies your evaluation criteria.

The data dictionary views for Oracle Machine Learning are listed in the following table. A database administrator (DBA) and USER versions of the views are also available.

View Name	Description
ALL_MINING_MODELS	Provides information about all accessible machine learning models
ALL_MINING_MODEL_ATTRIBUTES	Provides information about the attributes of all accessible machine learning models
ALL_MINING_MODEL_SETTINGS	Provides information about the configuration settings for all accessible machine learning models
ALL_MINING_MODEL_VIEWS	Provides information about the model views for all accessible machine learning models
ALL_MINING_MODEL_XFORMS	Provides the user-specified transformations embedded in all accessible machine learning models.

Model detail views are specific to the algorithm. You can obtain more insights about the model you created by viewing the model detail views. The names of model detail views begin with DM$xx where xx corresponds to the view prefix. See Model Detail Views.

1. You can review the model settings by running the following query:

```
 %sql

SELECT SETTING_NAME, SETTING_VALUE
  FROM USER_MINING_MODEL_SETTINGS
  WHERE MODEL_NAME = UPPER('ESM_SALES_FORECAST_1')
  ORDER BY SETTING_NAME;



SETTING_NAME                   SETTING_VALUE
ALGO_NAME                      ALGO_EXPONENTIAL_SMOOTHING
EXSM_ACCUMULATE                EXSM_ACCU_TOTAL
EXSM_CONFIDENCE_LEVEL          .95
EXSM_INTERVAL                  EXSM_INTERVAL_QTR
EXSM_MODEL                     EXSM_WINTERS
EXSM_NMSE                      3
EXSM_OPTIMIZATION_CRIT         EXSM_OPT_CRIT_LIK
EXSM_PREDICTION_STEP           4
EXSM_SEASONALITY               4
EXSM_SETMISSING                EXSM_MISS_AUTO
ODMS_DETAILS                   ODMS_ENABLE
ODMS_MISSING_VALUE_TREATMENT   ODMS_MISSING_VALUE_AUTO
ODMS_SAMPLING                  ODMS_SAMPLING_DISABLE
PREP_AUTO                      ON

14 rows selected.
---------------------------
```

2. To view the DM$VP model view, run the following statement:

The DM$VP view for time series contains the result of an ESM model. The output has a set of records such as partition, CASE_ID, value, prediction, lower, upper, and so on and ordered by partition and CASE_ID (time).

```
%script
SELECT CASE_ID, VALUE, PREDICTION, LOWER, UPPER FROM
DM$VPESM_SALES_FORECAST_1
ORDER BY CASE_ID;



CASE_ID      VALUE                PREDICTION           LOWER
UPPER
01-JAN-98      6480684.0000011446
6452375.7547333492
01-APR-98      5593994.1400007578
5848724.7899219571
01-JUL-98      6071823.1000010688
6214546.3092128271
01-OCT-98      5937413.7100012964
5869219.4189072186
01-JAN-99       6093747.209999715
6132016.410793812
01-APR-99      4925471.6299999086
5385954.0785653945
01-JUL-99      5827050.1500000218
5350240.2540956484
01-OCT-99      5373678.6700002998
5304626.0456054937
01-JAN-00      5984889.4899995513
5541123.2442497462
01-APR-00      5371730.9200002486
5236126.09628068
01-JUL-00      6121239.2899996703
5955258.7436284116
01-OCT-00      6287646.9199997969
6089446.4024073323
01-JAN-01      6547097.4400001625
6837567.1739504253
01-APR-01      6922468.3900004178
6188944.0536819538                 ...

-------------------------------------------------------------------
-----------------
```

Examine the statement:

- CASE_ID: Specifies the timestamp.
- VALUE: Specifies the AMOUNT_SOLD.
- PREDICTION: Indicates the predicted value for the model.
- LOWER and UPPER: Indicate the confidence bounds.

3. To view the model diagonistic view, DM$VG, and evaluate the model, run the following query:

The `DM$VG` view for time series contains the global information of the model along with the estimated smoothing constants, the estimated initial state, and global diagnostic measures.

```
%sql
SELECT NAME, round(NUMERIC_VALUE,4), STRING_VALUE
  FROM DM$VGESM_SALES_FORECAST_1
  ORDER BY NAME;
```

```
NAME                    ROUND(NUMERIC_VALUE,4)   STRING_VALUE
-2 LOG-LIKELIHOOD                     450.7508
AIC                                   466.7508
AICC                                  487.3223
ALPHA                                   0.4525
AMSE                         157764777942.4555
BETA                                    0.4195
BIC                                   472.9315
CONVERGED                                          YES
GAMMA                                   0.0001
INITIAL LEVEL                     6110212.8741
INITIAL SEASON 1                        0.9939
INITIAL SEASON 2                        1.0231
INITIAL SEASON 3                        0.9366
INITIAL SEASON 4                        1.0465

NAME                ROUND(NUMERIC_VALUE,4)   STRING_VALUE
INITIAL TREND                   55478.0794
MAE                                 0.0424
MSE                     104400146583.6485
NUM_ROWS                            918843
SIGMA                                0.054
STD                            323110.1153


20 rows selected.

---------------------------
```

- `NAME`: Indicates the diagnostic attribute name.
- `NUMERIC_VALUE`: Indicates the calculated statistical value for the model.
- `STRING_VALUE`: Indicates alphanumeric values for the diagnostic parameter.
  A few parameters to note for an exponential smoothing algorithm are:
  - `ALPHA`: Indicates the smoothing constant.
  - `BETA`: Indicates the trend smoothing constant.
  - `GAMMA`: Indicates the seasonal smoothing constant.
  - `MAE`: Indicates Mean Absolute Error.
  - `MSE`: Indicates Mean Square Error.

Exponential smoothing assumes that a series extends infinitely into the past, but that influence of past on future, decays smoothly and exponentially fast. The smooth rate of decay is expressed by one or more smoothing constants. The *smoothing constants* are parameters that the model estimates. These smoothing constants are represented as α, β, and γ. Values of a smoothing constant near one put almost all weight on the most recent observations. Values of a smoothing constant near zero allow the distant past observations to have a large influence.

Note that α is associated with the error or noise of the series, β is associated with the trend, and γ is associated with the seasonality factors. The γ value is closest to zero which means seasonality has an influence on the data set.

The MAE and MSE values are low which means that the model is good. The MSE magnitude depends on the actual scale of your original data. In this case, the STD is around 10^5. The square of it is roughly in the scale of 10^{10}. The error percentage is low and hence, the model is good.

Score

You are ready to forecast sales for the next four quarters.

For a time series model, you can use the `DM$VP` view to perform scoring or prediction.

1. Query the `DM$VP` model detail view to see the forecast (sales for four quarters). Run the following statement:

   ```
   %sql
   SELECT TO_CHAR(CASE_ID,'YYYY-MON') DATE_ID,
          round(VALUE,2) ACTUAL_SOLD,
          round(PREDICTION,2) FORECAST_SOLD,
          round(LOWER,2) LOWER_BOUND, round(UPPER,2) UPPER_BOUND
     FROM DM$VPESM_SALES_FORECAST_1
     ORDER BY CASE_ID;
   ```

   In this step, the prediction shows amount sold along with the `case_id`. The predictions display upper and lower confidence bounds showing that the estimates can vary between those values.

   Examine the statement:

   - `TO_CHAR(CASE_ID,'YYYY-MON') DATE_ID`: The `DATE_ID` column has timestamp or case_id extracted in year-month (yyyy-mon) format.
   - `round(VALUE,2) ACTUAL_SOLD`: Specifies the `AMOUNT_SOLD` value as `ACTUAL_SOLD` rounded to two numericals after the decimal.
   - `round(PREDICTION,2) FORECAST_SOLD`: Specifies the predicted value as `FORECAST_SOLD` rounded to two numericals after the decimal.
   - `round(LOWER,2) LOWER_BOUND, round(UPPER,2) UPPER_BOUND`: Specifies the lower and upper confidence levels rounded to two numericals after the decimal.

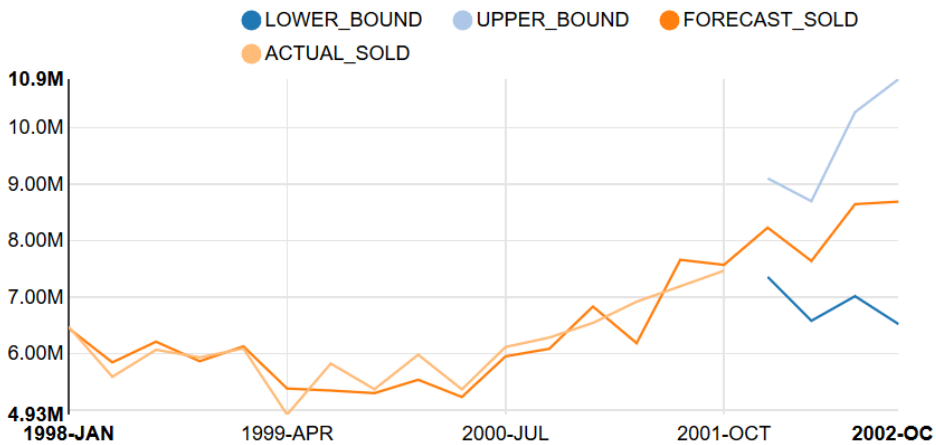
   ```
   DATE_ID     ACTUAL_SOLD    FORECAST_SOLD    LOWER_BOUND
   UPPER_BOUND
   1998-JAN         6480684
   6452375.75
   1998-APR      5593994.14
   5848724.79
   1998-JUL       6071823.1
   ```

```
6214546.31
1998-OCT      5937413.71      5869219.42
1999-JAN      6093747.21      6132016.41
1999-APR      4925471.63      5385954.08
1999-JUL      5827050.15      5350240.25
1999-OCT      5373678.67      5304626.05
2000-JAN      5984889.49      5541123.24
2000-APR      5371730.92       5236126.1
2000-JUL      6121239.29      5955258.74
2000-OCT      6287646.92       6089446.4
2001-JAN      6547097.44      6837567.17
2001-APR      6922468.39      6188944.05

DATE_ID     ACTUAL_SOLD   FORECAST_SOLD   LOWER_BOUND   UPPER_BOUND
2001-JUL      7195998.63      7663836.77
2001-OCT      7470897.52      7573926.96
2002-JAN                      8232820.51    7360847.49    9104793.54
2002-APR                      7642694.94    6584565.24    8700824.63
2002-JUL                      8648402.54    7019914.28   10276890.81
2002-OCT                      8692842.46    6523676.33    10862008.6


20 rows selected.


---------------------------
```

2. To see a visual representation of the predictions in OML Notebooks, run the above same query with the following settings:

   Click **settings** and drag `DATE_ID` to **keys** and `FORECASTED_SOLD (avg)`, `ACTUAL_SOLD (avge)`, `LOWER_BOUND (avg)`, and `UPPER_BOUND(avg)` to **values**.

```
%sql
SELECT TO_CHAR(CASE_ID,'YYYY-MON') DATE_ID, VALUE ACTUAL_SOLD,
       round(PREDICTION,2) FORECAST_SOLD,
       round(LOWER,2) LOWER_BOUND, round(UPPER,2) UPPER_BOUND
  FROM DM$VPESM_SALES_FORECAST_1
  ORDER BY CASE_ID;
```



This completes the prediction step. The model has successfully forecast sales for the next four quarters. This helps in tracking the sales and also gives us an idea on stocking our products.

Association Rules

A popular movie rental website is being revamped. The movie rental company wants to offer movie recommendation to their customers based on their frequently rented movies and purchase transaction history. They approach you, a data scientist, to help with the movie recommendations. You resolve this problem by analyzing popular movies frequently viewed together using the Apriori algorithm.

Before you start your OML4SQL use case journey, ensure that you have the following:

- Data set
  The data set used for this use case is called MovieStream data set.

  > **Note:**
  >
  > This data set is used for illustrative purpose only.

- Database
  Select a database out of the following options:
  - Get your FREE cloud account. Go to https://cloud.oracle.com/database and select Oracle Database Cloud Service (DBCS), or Oracle Autonomous Database. Create an account and create an instance. See Autonomous Database Quick Start Workshop.
  - Download the latest version of Oracle Database (on premises).
- Machine Learning Tools
  Depending on your database selection,
  - Use OML Notebooks for Oracle Autonomous Database.
  - Install and use Oracle SQL Developer connected to an on-premises database or DBCS. See Installing and Getting Started with SQL Developer.
- Other Requirements

Data Mining Privileges (this is automatically set for ADW). See System Privileges for Oracle Machine Learning for SQL.

Related Topics

- ADW: Data Loading and Management Using SQL on the MovieStream Dataset Workshop

Load Data

Examine the data set and its attributes. Load the data in your database.

In this use case, you will load the data set to your database. If you are using Oracle Autonomous Database, you will use an existing data file from the Oracle Cloud Infrastructure (OCI) Object Storage. You will create a sample table, load data into the sample table from files on the OCI Object Storage, and explore the data. If you are using the on-premises database, you will use Oracle SQL developer to import the data set and explore the data.

To understand the data, you will perform the following:

- Access the data.
- Examine the various attributes or columns of the data set.
- Assess data quality (by exploring the data).

Examine Data

The following table displays information about the attributes from `MOVIES_SALES_FACT`:

Attribute Name	Information
`ORDER_NUM`	Specifies the order number
`ACTUAL_PRICE`	Specifies the actual price of the movie
`AGE`	Specifies the age of the customer
`AGE_BAND`	Specifies the age band of the customer. The possible values are 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89 and so on.
`APP`	Specifies the application used for the movie
`CITY`	Specifies the name of the city
`CITY_ID`	Specifies the city ID
`COMMUTE_DISTANCE`	Specifies the commute distance
`COMMUTE_DISTANCE_BAND`	Specifies the commute distance band
`CONTINENT`	Specifies the continent name
`COUNTRY`	Specifies the country name
`COUNTRY_CODE`	Specifies the country code
`COUNTRY_ID`	Specifies the country ID
`CREDIT_BALANCE`	Specifies the credit balance of the customer
`CUSTOMER_ID`	Specifies the customer ID
`CUSTOMER_NAME`	Specifies the customer name
`DAY`	Specifies the day of the week in `YYYY-mm-dd hh:mm:ss` format
`DAY_NAME`	Specifies the day of the week
`DAY_NUM_OF_WEEK`	Specifies the day number of the week

Attribute Name	Information
DEVICE	Specifies the device information used by the customer
DISCOUNT_PERCENT	Specifies the discount percent
DISCOUNT_TYPE	Specifies the discount type availed by the customer. Possible values are referral, coupon, promotion, volume, none
EDUCATION	Specifies customer's education
EMAIL	Specifies email ID of the customer
FULL_TIME	Specifies customer's employment status such as full time, not employed, part time
GENDER	Specifies the gender of the customer
GENRE	Specifies the genre of the movie
HOUSEHOLD_SIZE	Specifies the household size of the customer
HOUSEHOLD_SIZE_BAND	Specifies the household size band
INCOME	Specifies the income of the customer
INCOME_BAND	Specifies the income band of the customer
INSUFF_FUNDS_INCIDENTS	Specifies the number of insufficient funds incidents that the customer had
JOB_TYPE	Specifies the cusotmer's job
LATE_MORT_RENT_PMTS	Specifies is the customer had any late mortgage or rent payment
LIST_PRICE	Specifies the list price of the movie
MARITAL_STATUS	Specifies the marital status of the customer
MONTH	Specifies the month in MON-YYYY format
MONTH_NAME	Specifies the month. For example, January.
MONTH_NUM_OF_YEAR	Specifies the month number of the year
MORTGAGE_AMT	Specifies the mortgage amount
MOVIE_ID	Specifies the movie ID
NUM_CARS	Specifies the number of the cars that the customer owns
NUM_MORTGAGES	Specifies the number of mortgages
OS	Specifies the OS information
PAYMENT_METHOD	Specifies the payment method
PET	Specifies if the customer owns a pet
POSTAL_CODE	Specifies the postal code of the address
PROMOTION_RESPONSE	Specifies the response of the customer to a promotional offer
QUANTITY_SOLD	Specifies the quantity sold
QUARTER_NAME	Specifies the quarter name in Qn-YYYY format. For example, Q1-2001.
QUARTER_NUM_OF_YEAR	Specifies the quarter number of the year
RENT_OWN	Specifies if the customer is living at a rented place or own place
SEARCH_GENRE	Specifies the genre of the movies searched
SEGMENT_DESCRIPTION	Describes the population segment

Attribute Name	Information
SEGMENT_NAME	Specifies the population segment name
SKU	Specifies the SKU ID
STATE_PROVINCE	Specifies the province
STATE_PROVINCE_ID	Specifies the province ID
STREET_ADDRESS	Specifies the customer's address
TITLE	Specifies the movie title
USERNAME	Specifies the username provided by the customer
WORK_EXPERIENCE	Specifies the work experience of the customer
WORK_EXPERIENCE_BAND	Specifies the work experience band of the customer
YEAR	Specifies the year
YEARS_CURRENT_EMPLOYER	Specifies the current employer of the customer
YEARS_CURRENT_EMPLOYER_BAND	Specifies the customer's employment band in years with the current employer
YEARS_RESIDENCE	Specifies the number of years the customer has been residing at a place
YEARS_RESIDENCE_BAND	Specifies the residence band

Create a Table

Create a table called `MOVIE_SALES_FACT`. This table is used in `DBMS_CLOUD.COPY_DATA` procedure to access the data set.

Enter the following code in the OML Notebooks and run the notebook.

```
%sql
CREATE TABLE MOVIE_SALES_FACT
( ORDER_NUM NUMBER(38,0),
 DAY DATE,
 DAY_NUM_OF_WEEK NUMBER(38,0),
 DAY_NAME VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 MONTH VARCHAR2(12 BYTE) COLLATE USING_NLS_COMP,
 MONTH_NUM_OF_YEAR NUMBER(38,0),
 MONTH_NAME VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 QUARTER_NAME VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 QUARTER_NUM_OF_YEAR NUMBER(38,0),
 YEAR NUMBER(38,0),
 CUSTOMER_ID NUMBER(38,0),
 USERNAME VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 CUSTOMER_NAME VARCHAR2(250 BYTE) COLLATE USING_NLS_COMP,
 STREET_ADDRESS VARCHAR2(250 BYTE) COLLATE USING_NLS_COMP,
 POSTAL_CODE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 CITY_ID NUMBER(38,0),
 CITY VARCHAR2(128 BYTE) COLLATE USING_NLS_COMP,
 STATE_PROVINCE_ID NUMBER(38,0),
 STATE_PROVINCE VARCHAR2(128 BYTE) COLLATE USING_NLS_COMP,
 COUNTRY_ID NUMBER(38,0),
 COUNTRY VARCHAR2(126 BYTE) COLLATE USING_NLS_COMP,
 COUNTRY_CODE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
```

```
 CONTINENT VARCHAR2(128 BYTE) COLLATE USING_NLS_COMP,
 SEGMENT_NAME VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 SEGMENT_DESCRIPTION VARCHAR2(128 BYTE) COLLATE USING_NLS_COMP,
 CREDIT_BALANCE NUMBER(38,0),
 EDUCATION VARCHAR2(128 BYTE) COLLATE USING_NLS_COMP,
 EMAIL VARCHAR2(128 BYTE) COLLATE USING_NLS_COMP,
 FULL_TIME VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 GENDER VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 HOUSEHOLD_SIZE NUMBER(38,0),
 HOUSEHOLD_SIZE_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 WORK_EXPERIENCE NUMBER(38,0),
 WORK_EXPERIENCE_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 INSUFF_FUNDS_INCIDENTS NUMBER(38,0),
 JOB_TYPE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 LATE_MORT_RENT_PMTS NUMBER(38,0),
 MARITAL_STATUS VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 MORTGAGE_AMT NUMBER(38,0),
 NUM_CARS NUMBER(38,0),
 NUM_MORTGAGES NUMBER(38,0),
 PET VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 PROMOTION_RESPONSE NUMBER(38,0),
 RENT_OWN VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 YEARS_CURRENT_EMPLOYER NUMBER(38,0),
 YEARS_CURRENT_EMPLOYER_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 YEARS_CUSTOMER NUMBER(38,0),
 YEARS_CUSTOMER_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 YEARS_RESIDENCE NUMBER(38,0),
 YEARS_RESIDENCE_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 AGE NUMBER(38,0),
 AGE_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 COMMUTE_DISTANCE NUMBER(38,0),
 COMMUTE_DISTANCE_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 INCOME NUMBER(38,0),
 INCOME_BAND VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 MOVIE_ID NUMBER(38,0),
 SEARCH_GENRE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 TITLE VARCHAR2(4000 BYTE) COLLATE USING_NLS_COMP,
 GENRE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 SKU NUMBER(38,0),
 LIST_PRICE NUMBER(38,2),
 APP VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 DEVICE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 OS VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 PAYMENT_METHOD VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 DISCOUNT_TYPE VARCHAR2(26 BYTE) COLLATE USING_NLS_COMP,
 DISCOUNT_PERCENT NUMBER(38,1),
 ACTUAL_PRICE NUMBER(38,2),
 QUANTITY_SOLD NUMBER(38,0)
 )
;
```

Load Data in the Table

Load the data set stored in object storage to the `MOVIE_SALES_FACT` table.

Before you load this data ensure that you set Compute Resources to Medium or High. If you are select High, then, set **Memory** field to `16` for High Resource Service. You must have Administrator privilege to configure the memory settings. See Compute Resource.

Add a new paragraph in your OML notebook and enter the following statement:

```
%script
BEGIN
 DBMS_CLOUD.COPY_DATA (table_name => 'MOVIE_SALES_FACT',file_uri_list =>
'https://objectstorage.uk-london-1.oraclecloud.com/n/adwc4pm/b/
moviestream_kl/o/d801_movie_sales_fact_m-*.csv', format =>
'{"delimiter":",", "recorddelimiter":"newline", "skipheaders":"1", "quote":"\
\\"", "rejectlimit":"1000", "trimspaces":"rtrim",
"ignoreblanklines":"false", "ignoremissingcolumns":"true", "dateformat":"DD-
MON-YYYY HH24:MI:SS"}');
END;
/

PL/SQL procedure successfully completed.


---------------------------
```

Examine the statement:

- `table_name`: is the target table's name.
- `credential_name`: is the name of the credential created earlier.
- `file_uri_list`: is a comma delimited list of the source files you want to load. The special character `*` in the file `d801_movie_sales_fact_m-*.csv` means you are bulk loading the MovieStream data set containing sales data for 2018-2020.
- `format`: defines the options you can specify to describe the format of the source file, including whether the file is of type text, ORC, Parquet, or Avro.
  - `delimiter`: Specifies the field delimiter (special character). Here, it is specified as "," (comma)
  - `recorddelimiter`: Specifies the record delimiter. The default value is `newline`. By default, `DBMS_CLOUD` tries to automatically find the correct newline character as the delimiter.
  - `skipheaders`: Specifies how many rows should be skipped from the start of the file. In this use case, it is `1`.
  - `quote`: Specifies the quote character for the fields.
  - `rejectlimit`: The operation will error out after specified number of rows are rejected. Here, the value is `1000`.
  - `trimspaces`: Specifies how the leading and trailing spaces of the fields are trimmed. Here it is `rtrim`. The `rtrim` value indicates that you want trailing spaces trimmed.

  - `ignoreblanklines`: Blank lines are ignored when set to true. The default value is `false`.
  - `ignoremissingcolumns`: If there are more columns in the `field_list` than there are in the source files, the extra columns are stored as null. The default value is `false`. In this use case, it is set to `true`.
  - `dateformat`: Specifies the date format in the source file.

In this example, *adwc4pm* is the Oracle Cloud Infrastructure object storage namespace and *moviestream_kl* is the bucket name.

Related Topics

- DBMS_CLOUD.COPY_DATA Procedure

Explore Data

Once the data is loaded into the table, explore the data to understand and assess the quality of the data. At this stage assess the data to identify data types and noise in the data. Look for missing values and numeric outlier values.

Assess Data Quality

To assess the data, first, you must be able to view the data in your database. For this reason, you will use SQL statements to query the table.

If you are working with Oracle Autonomous Database, you can use the Oracle Machine Learning (OML) Notebooks for your data science project, including assessing data quality. If you are using the on-premises Oracle Database, you can use the Oracle SQL Developer to assess data quality. Query the data as described.

> **Note:**
>
> Each record in the database is called a case and each case is identified by a `case_id`. In this use case, `CUSTOMER_ID` is the `case_id`.

The following steps help you with the exploratory analysis of the data:

1. View the data in the `MOVIE_SALES_FACT` table by running the following query:

   ```
   SELECT * FROM MOVIE_SALES_FACT;
   ```

2. Find the `COUNT` rows in the data set, run the following statement:

   ```
   SELECT DISTINCT COUNT(*) from MOVIE_SALES_FACT;



   COUNT(*)
    97890562
   ---------------------------
   ```

3. To find distinct or unique customers in the table, run the following statement:

```
%script SELECT COUNT (DISTINCT CUST_ID) FROM MOVIE_SALES_FACT;


COUNT(DISTINCTCUST_ID)
4845
---------------------------
```

4. To view the data type of the columns, run the following statement:

```
%script
DESCRIBE MOVIE_SALES_FACT;



Name                     Null?      Type
--------------------------- ----- --------------
ORDER_NUM                          NUMBER(38)
DAY DATE
DAY_NUM_OF_WEEK                                  NUMBER(38)
DAY_NAME                              VARCHAR2(26)
MONTH                                  VARCHAR2(12)
MONTH_NUM_OF_YEAR                            NUMBER(38)
MONTH_NAME                          VARCHAR2(26)
QUARTER_NAME                               VARCHAR2(26)
QUARTER_NUM_OF_YEAR                           NUMBER(38)
YEAR                              NUMBER(38)
CUSTOMER_ID                          NUMBER(38)
USERNAME                          VARCHAR2(26)
CUSTOMER_NAME                              VARCHAR2(250)
STREET_ADDRESS                              VARCHAR2(250)
POSTAL_CODE                          VARCHAR2(26)
CITY_ID                          NUMBER(38)
CITY                              VARCHAR2(128)
STATE_PROVINCE_ID                      NUMBER(38)
STATE_PROVINCE                          VARCHAR2(128)
COUNTRY_ID                          NUMBER(38)
COUNTRY                          VARCHAR2(126)
COUNTRY_CODE                          VARCHAR2(26)
CONTINENT                          VARCHAR2(128)
SEGMENT_NAME                          VARCHAR2(26)
SEGMENT_DESCRIPTION                      VARCHAR2(128)
CREDIT_BALANCE                          NUMBER(38)
EDUCATION                          VARCHAR2(128)
EMAIL                              VARCHAR2(128)
FULL_TIME                          VARCHAR2(26)
GENDER                              VARCHAR2(26)
HOUSEHOLD_SIZE                          NUMBER(38)
HOUSEHOLD_SIZE_BAND                      VARCHAR2(26)
WORK_EXPERIENCE                      NUMBER(38)
WORK_EXPERIENCE_BAND                      VARCHAR2(26)
INSUFF_FUNDS_INCIDENTS                      NUMBER(38)
JOB_TYPE                          VARCHAR2(26)
LATE_MORT_RENT_PMTS                      NUMBER(38)
MARITAL_STATUS                          VARCHAR2(26)
```

```
MORTGAGE_AMT                    NUMBER(38)
NUM_CARS                    NUMBER(38)
NUM_MORTGAGES                    NUMBER(38)
PET                     VARCHAR2(26)
PROMOTION_RESPONSE                  NUMBER(38)
RENT_OWN                    VARCHAR2(26)
YEARS_CURRENT_EMPLOYER                  NUMBER(38)
YEARS_CURRENT_EMPLOYER_BAND                  VARCHAR2(26)
YEARS_CUSTOMER                    NUMBER(38)
YEARS_CUSTOMER_BAND                  VARCHAR2(26)
YEARS_RESIDENCE                 NUMBER(38)
YEARS_RESIDENCE_BAND                  VARCHAR2(26)
AGE                     NUMBER(38)
AGE_BAND                    VARCHAR2(26)
COMMUTE_DISTANCE                  NUMBER(38)
COMMUTE_DISTANCE_BAND                  VARCHAR2(26)
INCOME                    NUMBER(38)
INCOME_BAND                   VARCHAR2(26)
MOVIE_ID                    NUMBER(38)
SEARCH_GENRE                    VARCHAR2(26)
TITLE                     VARCHAR2(4000)
GENRE                     VARCHAR2(26)
SKU                     NUMBER(38)
LIST_PRICE                    NUMBER(38,2)
APP                     VARCHAR2(26)
DEVICE                      VARCHAR2(26)
OS                      VARCHAR2(26)
PAYMENT_METHOD                    VARCHAR2(26)
DISCOUNT_TYPE                   VARCHAR2(26)
DISCOUNT_PERCENT                  NUMBER(38,1)
ACTUAL_PRICE                    NUMBER(38,2)
QUANTITY_SOLD                   NUMBER(38)

---------------------------
```

5. Select the required columns from `MOVIE_SALES_FACT` table.

```
%sql
 SELECT ORDER_NUM, MONTH, CUSTOMER_ID, MOVIE_ID, TITLE, GENRE, ACTUAL_PRICE, 
QUANTITY_SOLD FROM MOVIE_SALES_FACT
 ORDER BY CUSTOMER_ID;
```

ORDER_NUM	MONTH	CUSTOMER_ID	MOVIE_ID	TITLE	GENRE	ACTUAL_PRICE	QUANTITY_SOLD
40398397	OCT-2018	1000050	431	Batman v Superman: Dawn of Justice	Adventure	3.99	1
64170360	OCT-2018	1000050	3407	The Matrix	Sci-Fi	0	1
82398523	OCT-2018	1000050	1075	Election	Comedy	0.49	1
71313229	OCT-2018	1000050	3748	Tusk	Comedy	0	1
96433181	JUL-2018	1000050	503	Bill & Ted's Excellent Adventure	Adventure	0.49	1
45314161	JUL-2018	1000050	219	All the President's Men	Drama	0.99	1

6. Select customers who watched, for example, the movie "Titanic" and check other popular movies watched among those customers.

```
%sql
select title, count(1) cnt
from movie_sales_fact a
```

```
join (
select distinct customer_id
from movie_sales_fact
where title = 'Titanic' ) b
on a.customer_id = b.customer_id
group by title
having count(1) > 800000
```

TITLE	CNT
Aladdin	917211
Avengers: Endgame	2528542
Captain Marvel	1203588
Black Panther	1446928
Avengers: Infinity War	2099647
Venom	846548
Spider-Man: Far From Home	922436
Star Wars: The Rise of Skywalker	899424
The Lion King	1134846
Aquaman	822025
Deadpool 2	804730

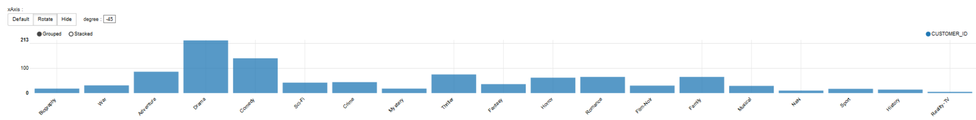
7. The data set is huge with millions of records. Create a view called `MOVIES` to select a smaller data set by providing a customer ID range.

```
%script
CREATE OR REPLACE VIEW MOVIES AS
SELECT DISTINCT CUSTOMER_ID, MOVIE_ID, TITLE, GENRE
FROM MOVIE_SALES_FACT
WHERE CUSTOMER_ID BETWEEN 1000000 AND 1000120;


View MOVIES created.
 ---------------------------
```

8. You can check the distribution of genre from the new view `MOVIES`:

```
%sql
SELECT * FROM MOVIES;
```

In OML Notebooks, click the bar icon and expand settings. Drag GENRE to **keys** and CUSTOMER_ID to **values** and select **COUNT**.



9. Now, check the count of rows by running the following statement:

```
%script
SELECT DISTINCT COUNT (*) FROM MOVIES;



COUNT(*)
 10194
---------------------------
```

10. To check if there are any missing values (NULL values), run the following statement:

```
SELECT COUNT(*) FROM MOVIES WHERE CUSTOMER_ID=NULL OR MOVIE_ID=NULL OR
TITLE=NULL OR GENRE=NULL;

COUNT(*)
0
---------------------------
```

    NULLs, if found, are automatically handled by the OML algorithms. Alternately, you can manually replace NULLs with `NVL` SQL function.

This completes the data exploration stage. OML supports Automatic Data Preparation (ADP). ADP is enabled through the model settings. When ADP is enabled, the transformations required by the algorithm are performed automatically and embedded in the model. This step is done during the Build Model stage. The commonly used methods of data preparation are binning, normalization, and missing value treatment.

Related Topics

- How ADP Transforms the Data

Build Model

Build your model using your data set. Use the `DBMS_DATA_MINING.CREATE_MODEL2` procedure to build your model and specify the model settings.

For unsupervised learning, like Association Rules, you do not have labels or predictors to calculate the accuracy or assess the performance. So you don't need to train your model on a separate training data set and then evaluate it on a test set. The entire data set can be used to build the model. For an unsupervised learning, you don't have an objective way to assess your model. So, a training or a test split is not useful.

Algorithm Selection

Oracle supports the Apriori algorithm to build an Association Rules model.

Apriori calculates the probability of an item being present in a frequent itemset, given that another item or group of items is present. An itemset is any combination of two or more items in a transaction. Frequent itemsets are those that occur with a minimum frequency that the user specifies. An association rule states that an item or group of items implies the presence of another item with some probability and support.

The following steps guide you to build your model with the Apriori algorithm.

- Build your model using the `CREATE_MODEL2` procedure. First, declare a variable to store model settings or hyperparameters. Run the following script:

  ```
  %script
   BEGIN DBMS_DATA_MINING.DROP_MODEL('AR_MOVIES');
  EXCEPTION WHEN OTHERS THEN NULL; END;
  /
  DECLARE
   v_setlist DBMS_DATA_MINING.SETTING_LIST;
  BEGIN
   v_setlist('ALGO_NAME') := 'ALGO_APRIORI_ASSOCIATION_RULES';
   V_setlist('PREP_AUTO') := 'ON';
   V_setlist('ASSO_MIN_SUPPORT') := '0.02';
   V_setlist('ASSO_MIN_CONFIDENCE') := '0.1';
   V_setlist('ASSO_MAX_RULE_LENGTH'):= '2';
   V_setlist('ODMS_ITEM_ID_COLUMN_NAME'):= 'TITLE';

   DBMS_DATA_MINING.CREATE_MODEL2(MODEL_NAME         => 'AR_MOVIES',
                    MINING_FUNCTION     => 'ASSOCIATION',
                    DATA_QUERY         => 'select * from MOVIES',
                            SET_LIST         => v_setlist,
                    CASE_ID_COLUMN_NAME => 'CUSTOMER_ID');
  END;



  PL/SQL procedure successfully completed.
  ---------------------------
  PL/SQL procedure successfully completed.
  ```

  Examine the script:

  - `v_setlist` is a variable to store `SETTING_LIST`.
  - `DBMS_DATA_MINING` is the PL/SQL package used for machine learning. These settings are described in DBMS_DATA_MINING - Model Settings.
  - `SETTING_LIST` specifies model settings or hyperparameters for our model.
  - `ALGO_NAME` specifies the algorithm name. Since you are using Apriori as your algorithm, set `ALGO_APRIORI_ASSOCIATION_RULES`.
  - `PREP_AUTO` is the setting used for Automatic Data Preparation. Here, enable Automatic Data Preparation. The value of the setting is `ON`.
  - `ASSO_MIN_SUPPORT` is minimum support for association rules (in percentage) that limits the number of itemsets used for association rules. An itemset must appear in at least this percentage of all the transactions if it is to be used as a basis for rules. Apriori discovers patterns with frequencies above the minimum support threshold. This is the minimum threshold that each rule must satisfy. Here, the algorithms finds patterns with frequenqies above 0.02. Increase the minimum support if you want to decrease the build time for the model and generate fewer rules.

- `ASSO_MIN_CONFIDENCE` determines minimum confidence for association rules. It is a conditional probability that the consequent occurs given the occurrence of an antecedent. In other words, the confidence of a rule indicates the probability of both the antecedent and the consequent appearing in the same transaction. The default value is 0.1.
- `ASSO_MAX_RULE_LENGTH` specifies the maximum number of items in an itemset. If the maximum is two, all the item pairs are counted. In this use case, if you want to increase the value to 3, consider working with a smaller data set since each customer would watch lot of movies. If the maximum is greater than two, all the item pairs, all the item triples, and all the item combinations up to the specified maximum are counted. Increasing this value increases the run time and complexity significantly. Hence, for demonstration purposes on this data set, it is recommended to set the value to 2.

> **Tip:**
>
> One way to limit the number of rules produced is to raise the support and confidence. Support is the joint probability of two items that are purchased together. For instance, item beer and diaper happens together with probability of 0.1, vodka and ice cream are purchased together with the probability of 0.05. If you raise the support threshold to 0.1. You will not see vodka and ice cream in the rules. Similarly, the confidence is the probability of people purchasing item A given they have purchased B. The probability of people who purchase beer given that they have already purchased a diaper is 0.2; The probability of people who purchase ice cream given that they have purchased vodka is 0.6. Using the threshold 0.6, you can remove the rule of people purchasing beer given that they already purchased diaper.

- `ODMS_ITEM_ID_COLUMN_NAME` name of a column that contains the items in a transaction. In this use case, it is `TITLE`. When this setting is specified, the algorithm expects the data to be presented in a native transactional format, consisting of two columns:
  - Case ID, either categorical or numeric
  - Item ID, either categorical or numeric

The `CREATE_MODEL2` procedure takes the following parameters:

- `MODEL_NAME`: Specify a unique name for your model. The name of the model is in the form [schema_name.]model_name. If you do not specify a schema, then your own schema is used. Here, the model name is `AR_MOVIES`.
- `MINING_FUNCTION`: Specifies the machine learning function. Since you are solving an association problem in this use case, select `ASSOCIATION`.
- `DATA_QUERY`: A query that provides training data for building the model. Here, the query is `SELECT * FROM MOVIES`.
- `SET_LIST`: Specifies `SETTING_LIST` variable. Here, it is `v_setlist`.
- `CASE_ID_COLUMN_NAME`: A unique case identifier column in the build data. In this use case, case_id is `CUSTOMER_ID`. If there is a composite key, you must create a new attribute before creating the model. This may involve concatenating values from the columns, or mapping a unique identifier to each

distinct combination of values. The `CASE_ID` assists with reproducible results, joining scores for individual customers with other data in, example, scoring data table.

> **Note:**
>
> Any parameters or settings not specified are either system-determined or default values are used.

Evaluate

Evaluate your model by viewing diagnostic metrics and performing quality checks.

You can inspect various rules to see if they reveal new insights for item dependencies (antecedent itemset implying consequent) or for unexpected relationships among items.

Dictionary and Model Views

To obtain information about the model and view model settings, you can query data dictionary views and model detail views. Specific views in model detail views display model statistics which can help you evaluate the model.

By examining various statistics in the model detail views, you can compare models to arrive at one model that satisfies your evaluation criteria.
The results of an association model are the rules that identify patterns of association within the data. Oracle Machine Learning for SQL does not support a scoring operation for association modeling. Instead, support and confidence are the primary metrics for evaluating the quality of the rules that the model generates. These statistical measures can be used to rank the rules and hence the usefulness of the predictions.

Association rules can be applied as follows:

- Support: How often do these items occur together in the data when you apply Association Rules?
- Confidence: How frequently the consequent occurs in transactions that contain the antecedent.
- Value: How much business value is connected to item associations

Additionally, Oracle Machine Learning for SQL supports lift for association rules. Lift indicates the strength of a rule over the random co-occurrence of the antecedent and the consequent, given their individual support. Lift provides information about the improvement, the increase in probability of the consequent given the antecedent. Lift is defined as confidence of the combination of items divided by the support of the consequent. Any rule with an improvement of less than 1 does not indicate a real cross-selling opportunity, no matter how high its support and confidence, because it actually offers less ability to predict a purchase than does random chance.

The data dictionary views for Oracle Machine Learning are listed in the following table. A database administrator (DBA) and USER versions of the views are also available.

View Name	Description
ALL_MINING_MODELS	Provides information about all accessible machine learning models

View Name	Description
ALL_MINING_MODEL_ATTRIBUTES	Provides information about the attributes of all accessible machine learning models
ALL_MINING_MODEL_SETTINGS	Provides information about the configuration settings for all accessible machine learning models
ALL_MINING_MODEL_VIEWS	Provides information about the model views for all accessible machine learning models
ALL_MINING_MODEL_XFORMS	Provides the user-specified transformations embedded in all accessible machine learning models.

Model detail views are specific to the algorithm. You can obtain more insights about the model you created by viewing the model detail views. The names of model detail views begin with DM$xx where xx corresponds to the view prefix. See Model Detail Views.

1. You can review the model settings in `USER_MINING_MODEL_SETTINGS` by running the following query:

```
SELECT SETTING_NAME, SETTING_VALUE
 FROM USER_MINING_MODEL_SETTINGS
 WHERE MODEL_NAME = 'AR_MOVIES'
 ORDER BY SETTING_NAME;



SETTING_NAME                 SETTING_VALUE
ALGO_NAME                    ALGO_APRIORI_ASSOCIATION_RULES
ASSO_MAX_RULE_LENGTH         2
ASSO_MIN_CONFIDENCE          0.1
ASSO_MIN_REV_CONFIDENCE      0
ASSO_MIN_SUPPORT             0.02
ASSO_MIN_SUPPORT_INT         1
ODMS_DETAILS                 ODMS_ENABLE
ODMS_ITEM_ID_COLUMN_NAME     TITLE
ODMS_MISSING_VALUE_TREATMENT ODMS_MISSING_VALUE_AUTO
ODMS_SAMPLING                ODMS_SAMPLING_DISABLE
PREP_AUTO                    ON


11 rows selected.


---------------------------
```

2. Run the following statement to see information on various views in `USER_MINING_MODEL_VIEWS`:

```
SELECT view_name, view_type FROM USER_MINING_MODEL_VIEWS
 WHERE MODEL_NAME = 'AR_MOVIES'
 ORDER BY VIEW_NAME;



VIEW_NAME         VIEW_TYPE
DM$VAAR_MOVIES    Association Rules For Transactional Data
DM$VGAR_MOVIES    Global Name-Value Pairs
DM$VIAR_MOVIES    Association Rule Itemsets
DM$VRAR_MOVIES    Association Rules
DM$VSAR_MOVIES    Computed Settings
DM$VTAR_MOVIES    Association Rule Itemsets For Transactional Data
DM$VWAR_MOVIES    Model Build Alerts


7 rows selected.


---------------------------
```

3. To view the Association Rules Itemsets For Transactional Data (DM$VTxx) model detail view, run the following script:

```
%script
SELECT ITEM_NAME, SUPPORT, NUMBER_OF_ITEMS
FROM DM$VTAR_MOVIES;




ITEM_NAME                SUPPORT                NUMBER_OF_ITEMS
Dallas Buyers Club                         1                  2
Dallas Buyers Club         0.666666666666666663               2
Dallas Buyers Club         0.333333333333333331               2
Elvira's Haunted Hills                     1                  2
Elvira's Haunted Hills     0.666666666666666663               2
Elvira's Haunted Hills     0.333333333333333331               2
Elvira's Haunted Hills                     1                  2
Elvira's Haunted Hills                     1                  2
Ghostbusters {{nbsp II                     1                  2
Ghostbusters {{nbsp II     0.666666666666666663               2
Ghostbusters {{nbsp II     0.333333333333333331               2
Ghostbusters {{nbsp II                     1                  2
Ghostbusters {{nbsp II                     1                  2
Hits                       0.333333333333333331               2
...
```

This view provides the itemsets information in transactional format. In the first transaction, *Dallas Buyers Club* and another movie are purchased or rented together with 100% support (support 1).

4. Now, view the Association Rules for Transactional Data (DM$VAxx) model detail view:

```
%sql SELECT * FROM DM$VAAR_MOVIES;
```

PARTITION_NAME	RULE_ID	ANTECEDENT_PREDICAT...	CONSEQUENT_PREDICA...	RULE_SUPPORT	RULE_CONFIDEN...	RULE_LIFT
	3798278	Your Sister's Sister	Zootopia	1	1	1
	3798284	Yours, Mine and Ours	Zootopia	1	1	1
	3788230	Unicorn Store	Zootopia	1	1	1
	3788434	Unknown Soldier	Zootopia	1	1	1
	3788818	Up	Zootopia	1	1	1
	3789016	Valley Girl	Zootopia	1	1	1
	3789212	Valley of the Dolls	Zootopia	1	1	1
	3789406	Velvet Buzzsaw	Zootopia	1	1	1

From this view, you can see that both antecedent and consequent are purchased together frequently (Support =1). You can expect the consequent to be present whenever the listed antecedent is present (Confidence=1). You can say that the probability of purchasing the consequent increases with the presence of the listed antecedent (Lift=1).

5. To see top 10 association rules, run the following query:

The IF component of an association rule is known as the **antecedent**. The THEN component is known as the **consequent**. The antecedent and the consequent are disjoint; they have no items in common. Oracle Machine Learning for SQL supports association rules that have one or more items in the antecedent and a single item in the consequent.

```
%script
SELECT * FROM
 (SELECT RULE_ID, ANTECEDENT_PREDICATE ANTECEDENT,
CONSEQUENT_PREDICATE CONSEQUENT,
ROUND(RULE_SUPPORT,3) SUPP, ROUND(RULE_CONFIDENCE,3) CONF,
NUMBER_OF_ITEMS NUM_ITEMS
FROM DM$VAAR_MOVIES
ORDER BY RULE_CONFIDENCE DESC, RULE_SUPPORT DESC)
WHERE ROWNUM <= 10
ORDER BY RULE_ID;



RULE_ID   ANTECEDENT                      CONSEQUENT    SUPP    CONF
NUM_ITEMS
    10759 101 Dalmatians                  10                  1
1           2
    10761 12 Years a Slave                10                  1
1           2
    10763 127 Hours                       10                  1
1           2
    10771 1984                            10                  1
1           2
    10773 2-Headed Shark Attack           10                  1
1           2
    10777 20,000 Leagues Under the Sea    10                  1
1           2
    10779 2001: A Space Odyssey           10                  1
```

```
1           2
    10781 2012                          10               1
1           2
    10785 3 Ninjas                      10               1
1           2
    10787 3 from Hell                   10               1
1           2


10 rows selected.


---------------------------
```

Examine the statement:

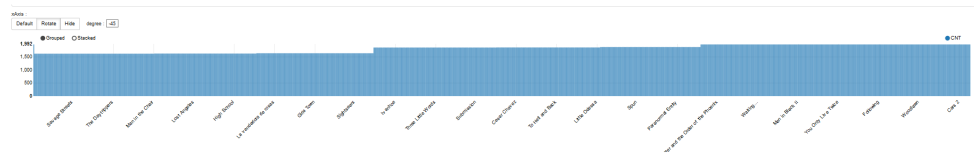
- `RULE_ID` is the rule identifier.
- `ANTECEDENT_PREDICATE`: provides the name of the antecedent.
- `CONSEQUENT_PREDICATE`: provides name of the consequent item.
- `ROUND (RULE_SUPPORT, 3) SUPP`: provides support of the rule rounded to 3 digits after the decimal.
- `ROUND(RULE_CONFIDENCE, 3) CONF`: the likelihood a transaction satisfying the rule when it contains the antecedent, rounded to 3 digits after the decimal.
- `NUM_OF_ITEMS`: specifies number of items in a rule.

6. You can also view which consequent items occur most frequently or which consequent items are included in most rules. To do so, run the following query:

```
%sql
SELECT CONSEQUENT, COUNT(1) CNT FROM
(SELECT ANTECEDENT_PREDICATE ANTECEDENT,
CONSEQUENT_PREDICATE CONSEQUENT,
RULE_SUPPORT SUPP, RULE_CONFIDENCE CONF, NUMBER_OF_ITEMS NUM
FROM DM$VAAR_MOVIES
ORDER BY RULE_CONFIDENCE DESC)
GROUP BY  CONSEQUENT
ORDER BY CNT;
```

In OML Notebooks, click **settings** and click the **Bar Chart** icon to visualize the result. Click **Rotate** to rotate the bar graph to 45 degrees.

CONSEQUENT	CNT
Naked in New York	1627
The Harry Hill Movie	1627
Hits	1627
Fan Girl	1627
Clay Pigeons	1627
Amy's Orgasm	1627
Wish You Were Here	1627
Then She Found Me	1627

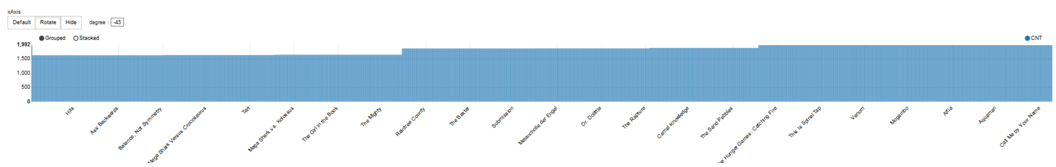


7. To view which antecedent items occur most frequently or which antecedent items are included in most rules, run the following script:

```
SELECT ANTECEDENT, COUNT(1) CNT
FROM
(SELECT ANTECEDENT_PREDICATE ANTECEDENT,
CONSEQUENT_PREDICATE CONSEQUENT,
RULE_SUPPORT SUPP, RULE_CONFIDENCE CONF, NUMBER_OF_ITEMS NUM
FROM DM$VAAR_MOVIES
ORDER BY RULE_CONFIDENCE DESC)
GROUP BY  ANTECEDENT
ORDER BY CNT
```

In OML Notebooks, click **settings** and click the **Bar Chart** icon to visualize the result. Click **Rotate** to rotate the bar graph to 45 degrees.

ANTECEDENT	CNT
In the Cut	1627
Just a Little Harmless Sex	1627
Let's Kill Ward's Wife	1627
Mega Python vs. Gatoroid	1627
Roadie	1627
Rottweiler	1627
Santa Claus Conquers the Martians	1627
School-Live!	1627



8. To check how many rules show up in each band of support, run the following query:

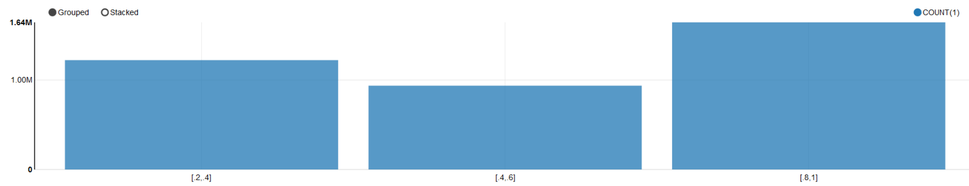
```
%sql
SELECT '['|| (SUPP_BIN -1)*0.2 ||','||SUPP_BIN*0.2||']' BUCKET, COUNT(1)
FROM (
SELECT ANTECEDENT_PREDICATE ANTECEDENT,
CONSEQUENT_PREDICATE CONSEQUENT,
RULE_SUPPORT SUPP, RULE_CONFIDENCE CONF, NUMBER_OF_ITEMS NUM,
WIDTH_BUCKET(RULE_SUPPORT, 0, 1, 4) SUPP_BIN
 FROM DM$VAAR_MOVIES ) a
GROUP BY SUPP_BIN
ORDER BY SUPP_BIN;
```

Examine the query:

- `SELECT '['|| (SUPP_BIN -1)*0.2 ||','||SUPP_BIN*0.2||']' BUCKET, COUNT(1)` creates the intervals for the buckets.
- The function `WIDTH_BUCKET` lets you construct equiwidth histograms, in which the histogram range is divided into intervals that have identical size. Here it produces buckets ranging from 0 to 1 and assigns number 1, …, 5, with identical size of 0.2. For instance the first bucket has the value = 1, for the range [0, 0.2].

In OML Notebooks, click **settings** and click the **Bar Chart** icon to visualize the result.

BUCKET	COUNT(1)
[.2,.4]	1220630
[.4,.6]	935418
[.8,1]	1642242

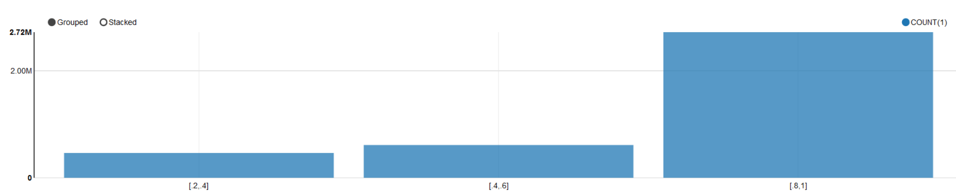


9. To check how many rules show up in each band of confidence, run the following query:

```
%sql
SELECT '['|| (CONF_BIN -1)*0.2 ||','||CONF_BIN*0.2||']' BUCKET,
COUNT(1)
FROM (
SELECT ANTECEDENT_PREDICATE ANTECEDENT,
CONSEQUENT_PREDICATE CONSEQUENT,
RULE_SUPPORT SUPP, RULE_CONFIDENCE CONF, NUMBER_OF_ITEMS NUM,
WIDTH_BUCKET(RULE_CONFIDENCE, 0, 1, 4) CONF_BIN
 FROM DM$VAAR_MOVIES ) a
GROUP BY CONF_BIN
ORDER BY CONF_BIN;
```

BUCKET	COUNT(1)
[.2,.4]	464084
[.4,.6]	612320
[.8,1]	2721886

In OML Notebooks, click **settings** and click the **Bar Char**t icon to visualize the result.



10. To recommend top five movies based on customer's selection, use the `NUMBER_OF_ITEMS` and `EXTRACT` as predicate and query the Association Rules model detail view (`DM$VRxx`).

    Association Rules support only a single consequent item.

```
%sql
```

```
SELECT ROWNUM RANK,
 CONSEQUENT_NAME RECOMMENDATION,
 NUMBER_OF_ITEMS NUM,
 ROUND(RULE_SUPPORT, 3) SUPPORT,
 ROUND(RULE_CONFIDENCE, 3) CONFIDENCE,
 ROUND(RULE_LIFT, 3) LIFT,
 ROUND(RULE_REVCONFIDENCE, 3) REVERSE_CONFIDENCE
 FROM (SELECT * FROM DM$VRAR_MOVIES
 WHERE NUMBER_OF_ITEMS = 2
 AND EXTRACT(antecedent, '//item[item_name="101 Dalmatians"]') IS NOT NULL
 ORDER BY NUMBER_OF_ITEMS
 )
 WHERE ROWNUM <= 5;
```

Examine the query:

- `ROUND(RULE_LIFT, 3) LIFT`: The degree of improvement in the prediction over random chance when the rule is satisfied.
- `ROUND(RULE_REVCONFIDENCE, 3) REVERSE_CONFIDENCE`: The number of transactions in which the rule occurs divided by the number of transactions in which the consequent occurs rounded to 3 digits after the decimal.
- `NUMBER_OF_ITEMS`: Here, this parameter controls the size of the rule.

  > **Note:**
  >
  > In this use case, since you are looking for `ASSO_MAX_RULE_LENGTH` =2, you can skip this parameter.

- `EXTRACT`: Filters on the antecedent. If the antecedent must include “101 Dalmatians”, then use `extract(antecedent, '//item[item_name=“101 Dalmatians”]') IS NOT NULL`

RANK	RECOMMENDATION...	NUM	SUPPORT	CONFIDENCE	LIFT	REVERSE_CONFIDE...
1	'Graduation Day'	2	0.667	0.667	1	1
2	'How to Be'	2	0.667	0.667	1	1
3	1 Day	2	0.333	0.333	1	1
4	10	2	1	1	1	1
5	10 Minutes Gone	2	0.667	0.667	1	1

In this step, if the customer's cart has 101 Dalmatians movie, the customer is 66.7% likely to rent or buy *Graduation Day*, *How to Be*, and *10 Minutes Gone* and there are 100% chances that they will buy *10*.

To conclude, you have successfully examined association rules and provided top movie recommendations to customers based on their frequently purchased and/or rented movies.

4
Reference

Specify Model Settings

Understand how to configure machine learning models at build time.

Numerous configuration settings are available for configuring machine learning models at build time. To specify settings, create a settings table with the columns shown in the following table and pass the table to `CREATE_MODEL`.

You can use `CREATE_MODEL2` procedure where you can directly pass the model settings to a variable that can be used in the procedure. The variable can be declared with `DBMS_DATA_MINING.SETTING_LIST` procedure.

Table 4-1 Settings Table Required Columns

Column Name	Data Type
`setting_name`	`VARCHAR2(30)`
`setting_value`	`VARCHAR2(4000)`

Example 4-1 creates a settings table for a Support Vector Machine (SVM) classification model. Since SVM is not the default classifier, the `ALGO_NAME` setting is used to specify the algorithm. Setting the `SVMS_KERNEL_FUNCTION` to `SVMS_LINEAR` causes the model to be built with a linear kernel. If you do not specify the kernel function, the algorithm chooses the kernel based on the number of attributes in the data.

Example 4-2 creates a model with the model settings that are stored in a variable from `SETTING_LIST`.

Some settings apply generally to the model, others are specific to an algorithm. Model settings are referenced in Table 4-2 and Table 4-3.

Table 4-2 General Model Settings

Settings	Description
Machine learning function settings	Machine Learning Function Settings
Algorithm names	Algorithm Names
Global model characteristics	Global Settings
Automatic Data Preparation	Automatic Data Preparation

Table 4-3 Algorithm-Specific Model Settings

Algorithm	Description
CUR Matrix Decomposition	DBMS_DATA_MINING —Algorithm Settings: CUR Matrix Decomposition
Decision Tree	DBMS_DATA_MINING —Algorithm Settings: Decision Tree
Expectation Maximization	DBMS_DATA_MINING —Algorithm Settings: Expectation Maximization
Explicit Semantic Analysis	DBMS_DATA_MINING —Algorithm Settings: Explicit Semantic Analysis
Exponential Smoothing	DBMS_DATA_MINING —Algorithm Settings: Exponential Smoothing Models
Generalized Linear Model	DBMS_DATA_MINING —Algorithm Settings: Generalized Linear Models
k-Means	DBMS_DATA_MINING —Algorithm Settings: *k*-Means
Multivariate State Estimation Technique - Sequential Probability Ratio Test	DBMS_DATA_MINING - Algorithm Settings: Multivariate State Estimation Technique - Sequential Probability Ratio Test
Naive Bayes	Algorithm Settings: Naive Bayes
Neural Network	DBMS_DATA_MINING —Algorithm Settings: Neural Network
Non-Negative Matrix Factorization	DBMS_DATA_MINING —Algorithm Settings: Non-Negative Matrix Factorization
O-Cluster	Algorithm Settings: O-Cluster
Random Forest	DBMS_DATA_MINING — Algorithm Settings: Random Forest
Singular Value Decomposition	DBMS_DATA_MINING —Algorithm Settings: Singular Value Decomposition
Support Vector Machine	DBMS_DATA_MINING —Algorithm Settings: Support Vector Machine
XGBoost	DBMS_DATA_MINING — Algorithm Settings: XGBoost

Note:

Some XGBoost objectives apply only to classification function models and other objectives apply only to regression function models. If you specify an incompatible `objective` value, an error is raised. In the `DBMS_DATA_MINING.CREATE_MODEL` procedure, if you specify `DBMS_DATA_MINING.CLASSIFICATION` as the function, then the only objective values that you can use are the `binary` and `multi` values. The one exception is `binary: logitraw`, which produces a continuous value and applies only to a regression model. If you specify `DBMS_DATA_MINING.REGRESSION` as the function, then you can specify `binary: logitraw` or any of the `count`, `rank`, `reg`, and `survival` values as the objective.

The values for the XGBoost objective setting are listed in the Settings for Learning Tasks table in DBMS_DATA_MINING — Algorithm Settings: XGBoost.

Example 4-1 Creating a Settings Table and Creating an SVM Classification Model Using CREATE.MODEL procedure

```
CREATE TABLE svmc_sh_sample_settings (
  setting_name VARCHAR2(30),
  setting_value VARCHAR2(4000));
```

```
BEGIN
  INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
    (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);
  INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
    (dbms_data_mining.svms_kernel_function, dbms_data_mining.svms_linear);
  COMMIT;
END;
/
-- Create the model using the specified settings
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'svm_model',
    mining_function     => dbms_data_mining.classification,
    data_table_name     => 'mining_data_build_v',
    case_id_column_name => 'cust_id',
    target_column_name  => 'affinity_card',
    settings_table_name => 'svmc_sh_sample_settings');
END;
```

Example 4-2 Specify Model Settings for a GLM Regression Model Using CREATE_MODEL2 procedure

```
DECLARE
    v_setlist DBMS_DATA_MINING.SETTING_LIST;
BEGIN
    v_setlist('PREP_AUTO') := 'ON';
    v_setlist('ALGO_NAME') := 'ALGO_GENERALIZED_LINEAR_MODEL';
    v_setlist('GLMS_DIAGNOSTICS_TABLE_NAME') := 'GLMR_DIAG';
    v_setlist('GLMS_FTR_SELECTION') := 'GLMS_FTR_SELECTION_ENABLE';
    v_setlist('GLMS_FTR_GENERATION') := 'GLMS_FTR_GENERATION_ENABLE';

    DBMS_DATA_MINING.CREATE_MODEL2(
        MODEL_NAME          => 'GLM_REGR',
        MINING_FUNCTION     => 'REGRESSION',
        DATA_QUERY          => 'select * from TRAINING_DATA',
        SET_LIST            => v_setlist,
        CASE_ID_COLUMN_NAME => 'HID',
    TARGET_COLUMN_NAME  => 'MEDV');
END;
```

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

Model Settings

Oracle Machine Learning uses settings to specify the algorithm and model settings or hyperparameters. Some settings are general, some are specific to a machine learning function, and some are specific to an algorithm.

OML4SQL

- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Solver Settings
- Summary of DBMS_DATA_MINING Subprograms
- DBMS_DATA_MINING_TRANSFORM
- DBMS_PREDICTIVE_ANALYTICS

- Oracle Machine Learning Data Dictionary Views
- Model Detail Views
- Oracle Machine Learning SQL Statistical Functions
- Oracle Machine Learning for SQL Scoring Functions
- OML4SQL Examples on GitHub