

Oracle® Machine Learning for R

Use Cases



Release 2.0
G29016-03
September 2025

ORACLE®

Oracle Machine Learning for R Use Cases, Release 2.0

G29016-03

Copyright © 2025, 2025, Oracle and/or its affiliates.

Primary Author: Sadhana Ashokkumar

Contributors: Mark Hornick, Sherry Lamonica, Qin Wang, Yu Xiang

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1 Overview

1.1	Machine Learning Overview	1
1.1.1	What Is Machine Learning?	1
1.1.2	Benefits of Machine Learning	2
1.1.3	Define Your Business Problem	3
1.1.4	What Do You Want to Do?	3
1.1.5	Discover More Through Interfaces	4
1.2	Machine Learning Process	5
1.2.1	Workflow	6
1.2.2	Define Business Goals	7
1.2.3	Understand Data	8
1.2.4	Prepare Data	8
1.2.5	Develop Models	9
1.2.6	Evaluate	10
1.2.7	Deploy	10
1.3	Machine Learning Techniques and Algorithms	11
1.3.1	What is a Machine Learning Algorithm	11
1.3.2	Supervised Learning	11
1.3.3	Unsupervised Learning	12

2 Get Started

2.1	Access OML Notebooks	1
2.1.1	Access Oracle Machine Learning User Interface	1
2.1.2	Create a Notebook from the Example Templates	2
2.1.3	Edit Your Notebook Classic	3
2.2	Access Autonomous Database	5
2.2.1	Provision an Autonomous Database	6
2.2.2	Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database	6
2.2.3	Create User	6
2.2.4	Add Existing Database User Account to Oracle Machine Learning Components	7

3 Use Cases

3.1	Classification Use Case	1
3.1.1	Load Data	3
3.1.2	Explore Data	4
3.1.3	Build Model	5
3.1.4	Evaluate	7
3.1.5	Deploy the Model	9
3.2	Clustering Use Case	11
3.2.1	Load Data	12
3.2.2	Explore Data	14
3.2.3	Build Model	18
3.2.4	Deploy the Model	21
3.3	Time Series Use Case	28
3.3.1	Access Data	29
3.3.2	Explore Data	29
3.3.3	Build Model	33
3.3.4	Evaluate	35

4 Reference

4.1	About Machine Learning Classes and Algorithms	1
4.2	About Model Settings	3
4.3	Shared Settings	3

Index

1

Overview

- [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.
- [Machine Learning Process](#)
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.
- [Machine Learning Techniques and Algorithms](#)
Machine learning problems are categorized into mining techniques. Each machine learning function specifies a class of problems that can be modeled and solved. An algorithm is a mathematical procedure for solving a specific kind of problem.

1.1 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.
- [Benefits of Machine Learning](#)
Machine learning is a powerful technology that can help you find patterns and relationships within your 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.
- [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.
- [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.

1.1.1 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.

1.1.2 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

1.1.3 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?).

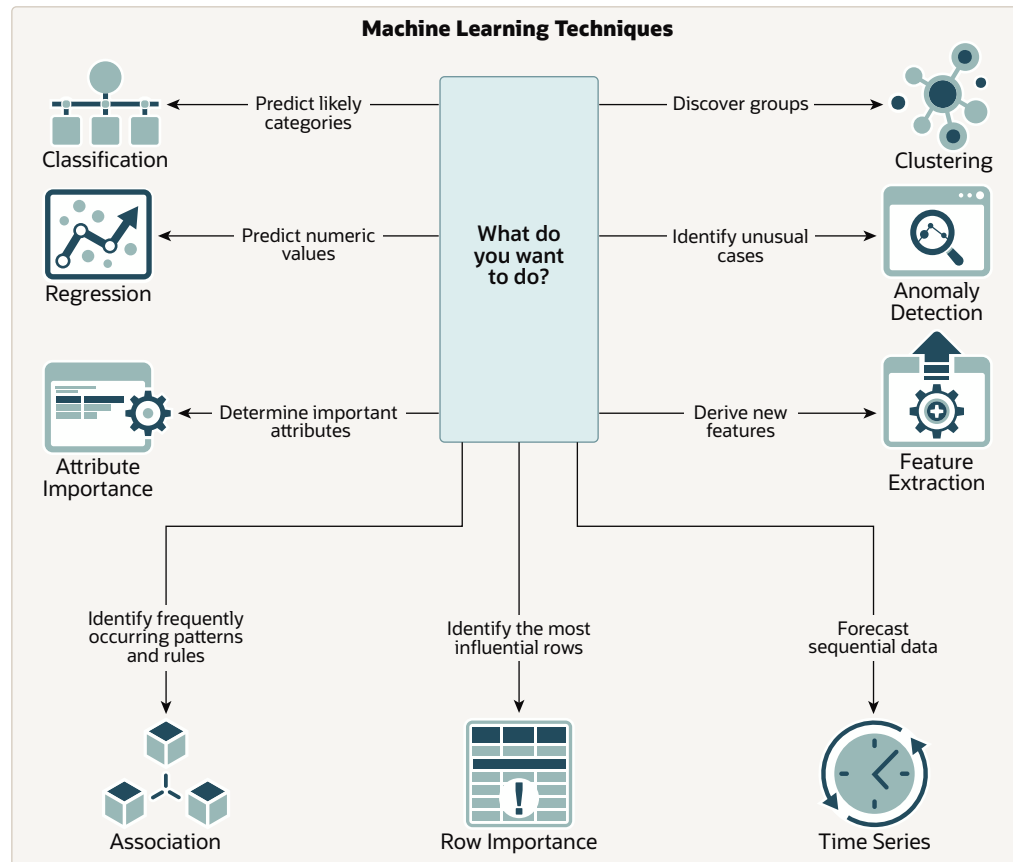
Other forms of machine learning identify groupings in the data. For example, a model might identify the segment of the population that has an income within a specified range, that has a good driving record, and that leases a new car on a yearly basis.

1.1.4 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.

1.1.5 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.

1.2 Machine Learning Process

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.
- [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.

- [Understand Data](#)
The data understanding phase involves data collection and exploration which includes loading the data and analyzing the data for your business problem.
- [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.
- [Develop Models](#)
In this phase, you select and apply various modeling techniques and tune the algorithm parameters, called *hyperparameters*, to desired values.
- [Evaluate](#)
At this stage of the project, it is time to evaluate how well the model satisfies the originally-stated business goal.
- [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.

1.2.1 Workflow

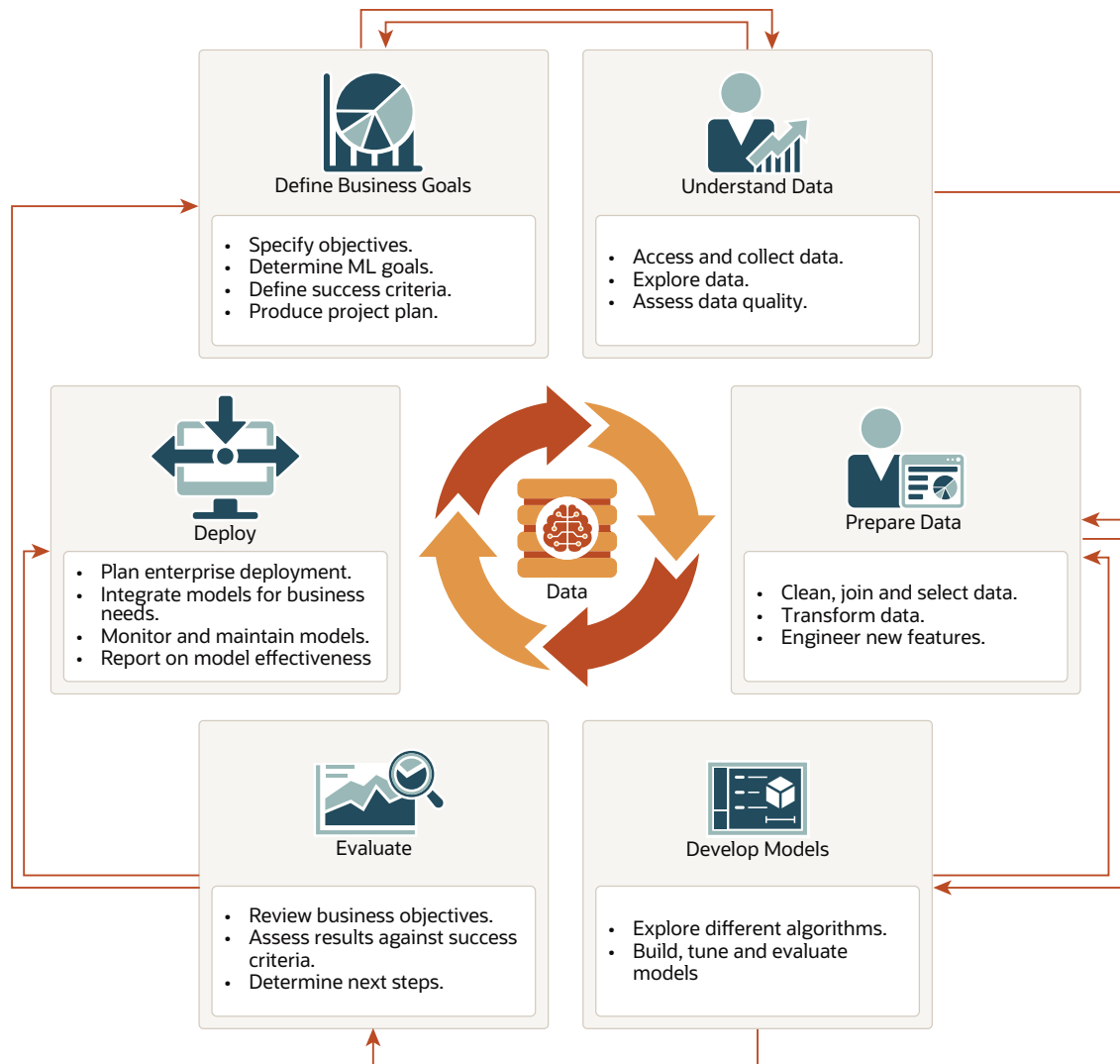
The machine learning process workflow illustration is based on the CRISP-DM methodology. Each stage in the workflow is illustrated with points that summarize the key tasks. The CRISP-DM methodology is the most commonly used methodology for machine learning.

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/>

1.2.2 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

1.2.3 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 address 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

1.2.4 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*

1.2.5 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*

1.2.6 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

1.2.7 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*

1.3 Machine Learning Techniques and Algorithms

Machine learning problems are categorized into mining techniques. Each machine learning function specifies a class of problems that can be modeled and solved. An algorithm is a mathematical procedure for solving a specific kind of problem.

- [What is a Machine Learning Algorithm](#)
An algorithm is a mathematical procedure for solving a specific kind of problem. For some machine learning techniques, you can choose among several algorithms.
- [Supervised Learning](#)
Supervised learning is also known as directed learning. The learning process is directed by a previously known dependent attribute or target.
- [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.

1.3.1 What is a Machine Learning Algorithm

An algorithm is a mathematical procedure for solving a specific kind of problem. For some machine learning techniques, 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.

1.3.2 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
Feature Selection or 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	<ul style="list-style-type: none"> • cur Matrix Decomposition • Expectation Maximization • Minimum Description Length
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	<ul style="list-style-type: none"> • Decision Tree • Explicit Semantic Analysis • XGBoost • 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	<ul style="list-style-type: none"> • XGBoost • 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	XGBoost
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

1.3.3 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.	<ul style="list-style-type: none"> • One-Class SVM • Multivariate State Estimation Technique - Sequential Probability Ratio Test
Association	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	<ul style="list-style-type: none"> • Expectation Maximization • k-Means • O-Cluster
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.	<ul style="list-style-type: none"> • Explicit Semantic Analysis • Non-Negative Matrix Factorization • PCA scoring • 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 Matrix Decomposition

2

Get Started

- [Access OML Notebooks](#)
To perform Oracle Machine Learning tasks, you can access Oracle Machine Learning Notebooks from Autonomous Database
- [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.

2.1 Access OML Notebooks

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

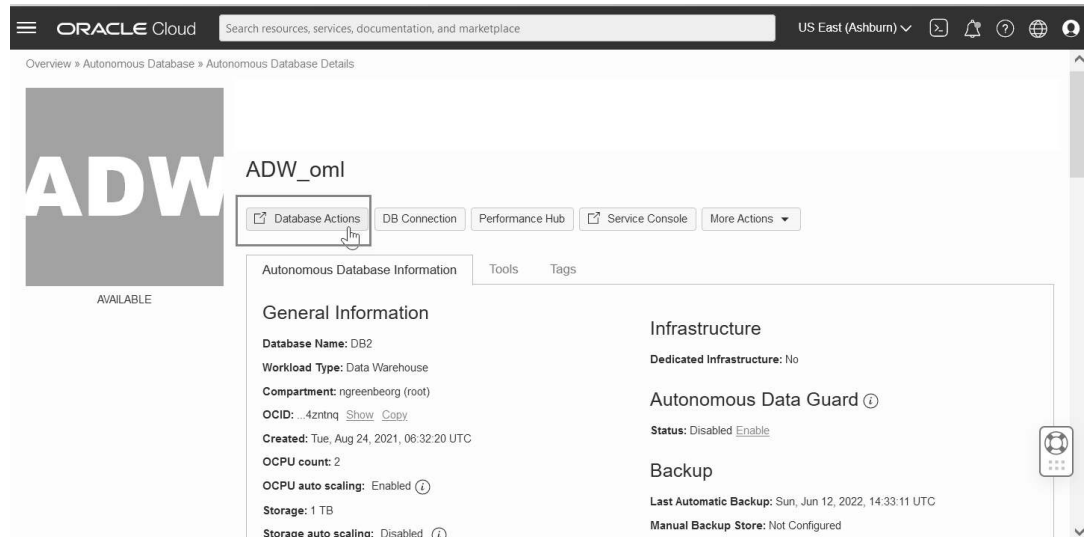
- [Access Oracle Machine Learning User Interface](#)
You can access Oracle Machine Learning **User Interface** from Autonomous Database.
- [Create a Notebook from the Example Templates](#)
Using the Oracle Machine Learning Example Templates, you can create a notebook from the available templates.
- [Edit Your Notebook Classic](#)
Upon creating an OML Notebook Classic, 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`, `%sql`, `%r`, `%md` or `%conda`.

2.1.1 Access Oracle Machine Learning User Interface

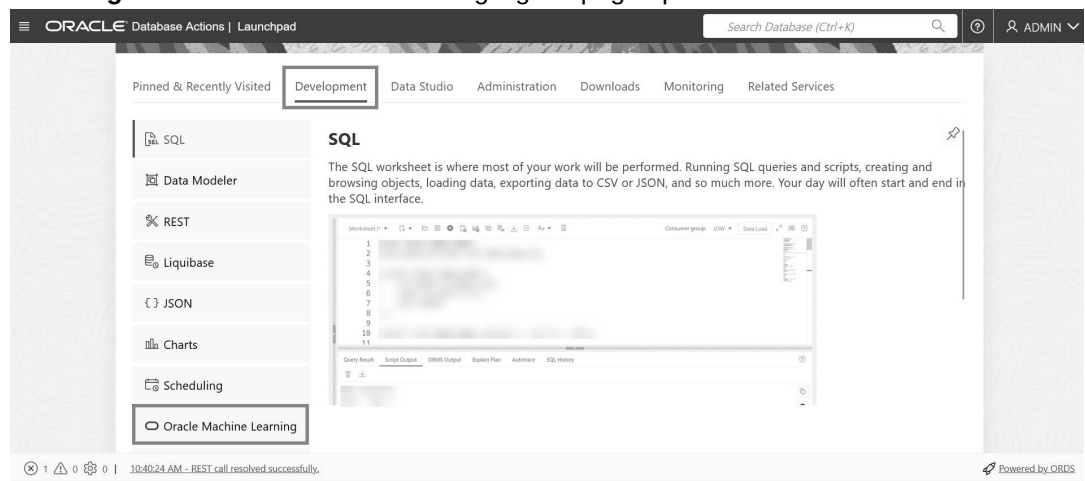
You can access Oracle Machine Learning **User Interface** from Autonomous Database.

To access Oracle Machine Learning User Interface (UI) from the Autonomous Database:

1. Select your Autonomous Database instance and on the Autonomous Database details page click **Database Actions**.



2. On the Database Actions page, go to the **Development** section and click **Oracle Machine Learning**. The Oracle Machine Learning sign in page opens.



3. On the Oracle Machine Learning sign in page, enter your username and password.
4. Click **Sign In**.


This opens the Oracle Machine Learning user application.

2.1.2 Create a Notebook from the Example Templates

Using the Oracle Machine Learning Example Templates, you can create a notebook from the available templates.

To create a notebook:

1. On the Example Templates page, select the template based on which you want to create a notebook.
2. Click **New Notebook**.
The Create Notebook dialog box opens.
3. In the **Create Notebook** dialog, the name of the selected template appears. In the **Name** field, you can change the notebook name.

4. In the **Comment** field, if any comment is available for the template, then it is displayed. You can edit the comment.
5. In the **Project** field, click the edit icon .
6. Select the project in which you want to save the notebook.
7. In the **Connection** field, the default connection is selected.
8. Click **OK**.

The notebook is created and is available on the Notebooks page.

2.1.3 Edit Your Notebook Classic

Upon creating an OML Notebook Classic, 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`, `%sql`, `%r`, `%md` or `%conda`.

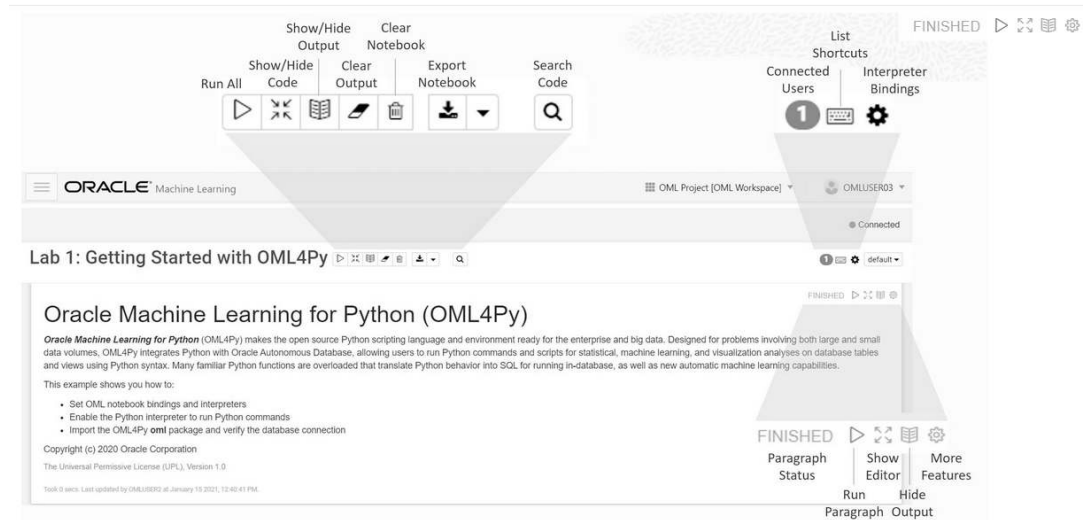
Set the context with a project with which your notebook is associated.

You can edit an existing Notebook Classic in your project. To edit an existing Notebook Classic:

1. On Oracle Machine Learning UI home page, select the project in which your notebook is available.
2. Go to the Oracle Machine Learning UI navigator, and select **Notebooks Classic**. 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 the edit mode, you can use the Oracle Machine Learning Notebooks Classic toolbar options to run code in paragraphs, for configuration settings, and display options.


Figure 2-1 Notebook toolbar


You can perform the following tasks:


- Write code to fetch data


- Click  to run one or all paragraphs in the notebook.


- Click  to hide all codes from all the paragraphs in the notebook. Click it again to display the codes.


- Click  to hide all outputs from all the paragraphs in the notebook. Click it again to view the outputs.


- Click  to remove all outputs from all the paragraphs in the notebook. To view the output, click the run icon again.

- Click  to delete all the paragraphs in the notebook.



- Click  to export the notebook.

- Click  to search any information in the codes present in the notebook.

- Click  to view the list of keyboard shortcuts.

- Click  to set the order for interpreter bindings for the notebook.



- Click  to select one of the three notebook display options.
 - Click **default** to view the codes, output, and metadata in all paragraphs in the notebook.
 - Click **Simple** to view only the code and output in all paragraphs in the notebook. In this view, the notebook toolbar and all edit options are hidden. You must hover your mouse to view the edit options.
 - Click **Report** to view only the output in all paragraphs in the notebook.
- Click  to access paragraph specific edit options such as clear output, remove paragraph, adjust width, font size, run all paragraphs above or below the selected paragraph and so on.
- 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 Notebooks Classic page.

2.2 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](#)
A LiveLabs workshop (a set of labs) that teaches you to manage and monitor Autonomous Database (ADB) is available. A part of the workshop aims to provision an Autonomous Database instance on Oracle Cloud.
- [Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database](#)
An administrator can add an existing database user account to use with Oracle Machine Learning components or create a new user account and user credentials with the Oracle Machine Learning User Management interface.
- [Create User](#)
An administrator creates new user accounts and user credentials for Oracle Machine Learning in Database Actions.

- [Add Existing Database User Account to Oracle Machine Learning Components](#)
As the ADMIN user you can add an existing database user account to provide access to Oracle Machine Learning components.

2.2.1 Provision an Autonomous Database

A LiveLabs workshop (a set of labs) that teaches you to manage and monitor Autonomous Database (ADB) is available. A part of the workshop aims to provision an Autonomous Database instance on Oracle Cloud.

[Manage and Monitor Autonomous Database](#)

2.2.2 Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database

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

2.2.3 Create User

An administrator creates new user accounts and user credentials for Oracle Machine Learning in Database Actions.

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, select **Database Actions** and click **Database Users**.
3. On the Database Users page, in the All Users area click **+ Create User**.
4. To create a new user enter a user name, a password, and enter the password again to confirm the password.
5. Select the options you want for the user and select **OML** to enable Oracle Machine Learning for the user.

Create User

User

5 Granted Roles

User Name

ADB_USER

Password

.....

Confirm Password

.....

Quota on tablespace DATA

100M

Password Expired (user must change)

☐

Account is Locked

☐

Graph

☐

Enable or disable Graph for user

Web Access

☒

Enable or disable a user to access from a web browser

OML

☒

Enable or disable Oracle Machine Learning for user

☐ ? Show code

Create User

Cancel

6. Click **Create User**.

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.

2.2.4 Add Existing Database User Account to Oracle Machine Learning Components

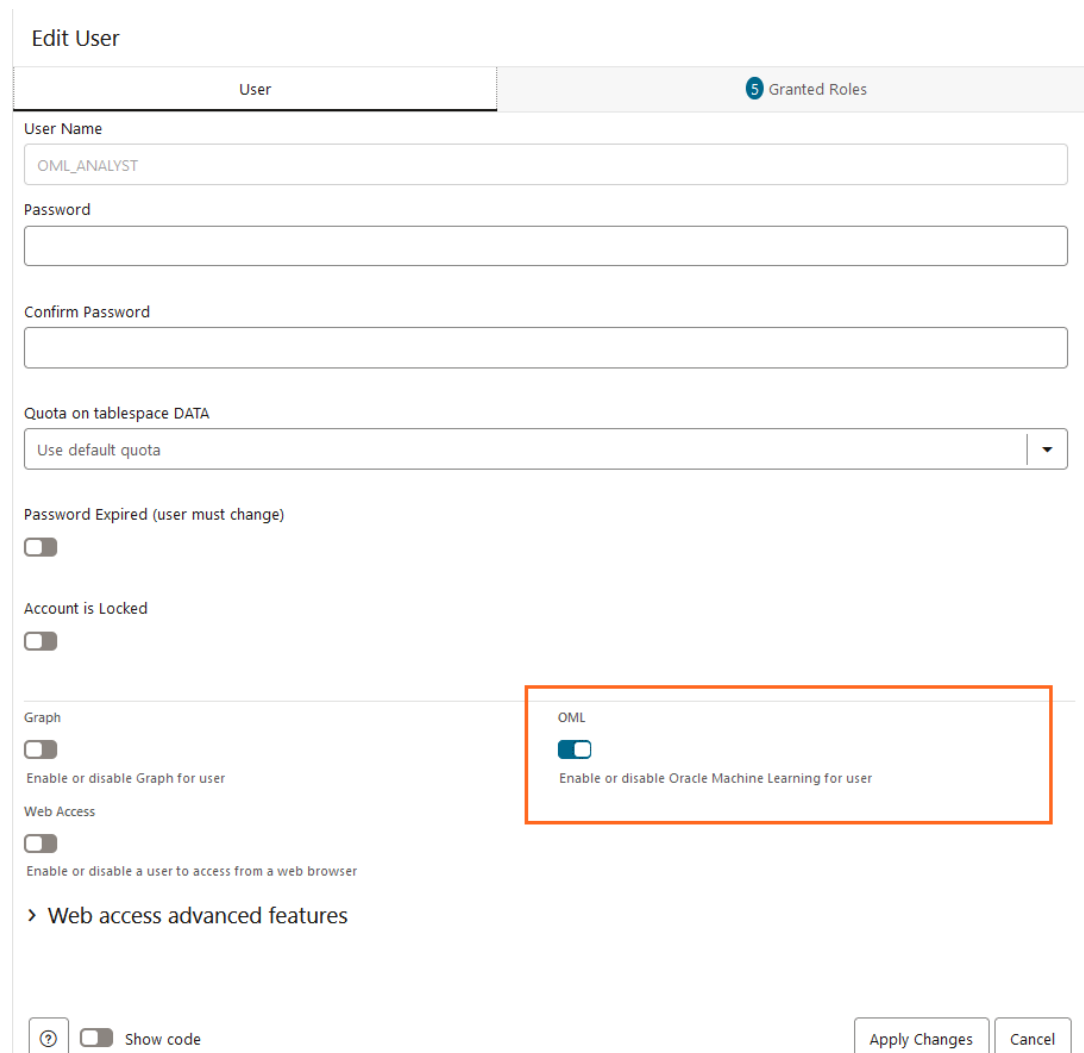
As the ADMIN user you can add an existing database user account to provide access to Oracle Machine Learning components.

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, select **Database Actions** and click **Database Users**.
3. In the All Users, search for the user of interest or select the user. For example, search the user **OML_ANALYST**.

4. In the user's card, click  and select **Edit**.
5. In the Edit User panel, select **OML**.

For example:



Edit User

User	5 Granted Roles
User Name OML_ANALYST	
Password 	
Confirm Password 	
Quota on tablespace DATA Use default quota	
Password Expired (user must change) <input type="checkbox"/>	
Account is Locked <input type="checkbox"/>	
Graph <input type="checkbox"/> Enable or disable Graph for user	OML <input checked="" type="checkbox"/> Enable or disable Oracle Machine Learning for user
Web Access <input type="checkbox"/> Enable or disable a user to access from a web browser	
Web access advanced features	
<input type="button" value="Show code"/> <input type="button" value="Apply Changes"/> <input type="button" value="Cancel"/>	

6. Click **Apply Changes**.

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.

3

Use Cases

- [Classification Use Case](#)
A retail store has information about its customers' behavior and the purchases they make. Now with the available data, they want you to analyze and identify the type of customers most likely to be positive responders to an Affinity Card loyalty program. High Affinity Card responders are defined as those customers who, when given a loyalty or affinity card, hyper-respond, that is, increase purchases more than the Affinity Card program's offered discount. In our data set, a responder is designated with value 1, and a non-responder with value 0. In this use case, you will demonstrate how to identify such customers using the Support Vector Machine model.
- [Clustering Use Case](#)
A retail store has information about its customers' behavior and the purchases they make. With that data, they would like you to analyze and identify if there are groups of customers with similar characteristics. Use Oracle Machine Learning to segment customers by finding clusters in the data set that can be then used to support targeted marketing campaigns to increase retail sales. In this use case, you will learn how to identify such segments using the k-Means algorithm.
- [Time Series Use Case](#)
You work in an electronics store, and sales of laptops and tablets have increased over the past two quarters. You want to forecast product sales for the next four quarters using historical timestamped data. To do this, you apply the Exponential Smoothing algorithm, which predicts future sales by analyzing patterns over evenly spaced time intervals in the historical data.

3.1 Classification Use Case

A retail store has information about its customers' behavior and the purchases they make. Now with the available data, they want you to analyze and identify the type of customers most likely to be positive responders to an Affinity Card loyalty program. High Affinity Card responders are defined as those customers who, when given a loyalty or affinity card, hyper-respond, that is, increase purchases more than the Affinity Card program's offered discount. In our data set, a responder is designated with value 1, and a non-responder with value 0. In this use case, you will demonstrate how to identify such customers using the Support Vector Machine model.

Related Contents

Topic	Link
OML4R GitHub Example	Classification Support Vector Machines (SVMs)
About Support Vector Machines (SVMs)	Classification Support Vector Machines (SVMs)
Shared Settings	Shared Settings

Before you start your OML4R 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.

- **Database**
Select or create 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](#)**
You will be using the SUPPLEMENTARY_DEMOGRAPHICS data set available in the SH schema. Use the `ore.sync` function to create an `ore.frame` proxy object in R that represents a database table, view, or query.
- **[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.
- **[Build Model](#)**
This model is designed to classify data into predefined categories by learning from training data.
- **[Evaluate](#)**
Before you make predictions using your model on new data, you should first evaluate model accuracy. You can evaluate the model using different methods.
- **[Deploy the Model](#)**
The machine learning model, `SVM_CLASSIFICATION_MODEL`, has been successfully trained and exists in your schema as a first-class database object. While you can use this model directly from R, for database applications, you can also run it directly from SQL queries.

Related Topics

- [Create a Notebook](#)
- [Edit your Notebook](#)

3.1.1 Load Data

You will be using the `SUPPLEMENTARY_DEMOGRAPHICS` data set available in the `SH` schema. Use the `ore.sync` function to create an `ore.frame` proxy object in R that represents a database table, view, or query.

Examine Data

Attribute Name	Information
CUST_ID	The ID of the customer
EDUCATION	Education level attained
OCCUPATION	Occupation of the customer
HOUSEHOLD_SIZE	Number of people living at residence
YRS_RESIDENCE	Number of years customer lived at current residence
AFFINITY_CARD	Indicates whether the customer holds an affinity card. 1 means Yes. 0 means No.
BULK_PACK_DISKETTES	Product. Indicates whether the customer purchased the bulk pack diskettes. 1 means Yes. 0 means No.
FLAT_PANEL_MONITOR	Product. Indicates whether the customer purchased flat panel monitor. 1 means Yes. 0 means No
HOME_THEATER_PACKAGE	Product. Indicates whether the customer purchased home theatre package. 1 means Yes. 0 means No
BOOKKEEPING_APPLICATION	Product. Indicates whether the customer purchased bookkeeping application. 1 means Yes. 0 means No
PRINTER_SUPPLIES	Product. Indicates whether the customer purchased printer supplies. 1 means Yes. 0 means No
Y_BOX_GAMES	Product. Indicates whether the customer purchased YBox Games. 1 means Yes. 0 means No
OS_DOC_SET_KANJI	Product. Indicates whether the customer purchased the Kanji character set for the operating system documentation. 1 means Yes. 0 means No
COMMENTS	Comments from customers

3.1.2 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.

Identify Target Variable

For this use case, the task is to train a Support Vector Machine model that predicts which customers most likely to be positive responders to an Affinity Card loyalty program. Therefore, the target variable is the attribute `AFFINITY_CARD`.

Data Understanding and Preparation

To access database data from R using OML4R, you must first create an `ore.frame` proxy object in R that represents a database table, view, or query. In this example, the proxy object is created using a query. Create proxy objects for `SUPPLEMENTARY_DEMOGRAPHICS` and then assess the data to identify data types and noise in the data. Look for missing values, outlier numeric values, or inconsistently labeled categorical values.

For data preparation and understanding run the following steps:

1. Run the following command in an R interpreter paragraph (using `%r`) to import the Oracle Machine Learning for R libraries and to suppress warning regarding row ordering:

```
library(ORE)
options(ore.warn.order=FALSE)
```

2. Use the `ore.sync` function to create the `ore.frame` object that is a proxy for the `SUPPLEMENTARY_DEMOGRAPHICS` table in the SH schema database table.

```
ore.sync(query = c("SUP_DEM" = "select * from
SH.SUPPLEMENTARY_DEMOGRAPHICS"))
ore.attach()
```

3. Run the following command to display few rows from `SUPPLEMENTARY_DEMOGRAPHICS` table

```
z.show(head(SUP_DEM))
```

CUST_ID	EDUCATION	OCCUPATION	HOUSEHOLD_SIZE	YRS_RESIDENCE	AFFINITY_CARD	BULK_PACK_DISKETTES	FLAT_PANEL_MONITOR	HOME_THEATER_PACKAGE	BOOKKEEPING_APPLICATION	PRINTER_SUPPLIES	Y_BORG_GAMES	OS_DOC_SET_XANX	COMMENTS
102547	10th	Other	1	0	0	1	1	0	0	1	1	0	NA
101050	10th	Other	1	0	0	1	1	0	0	1	1	0	NA
100040	11th	Sales	1	0	0	1	1	0	0	1	1	0	NA
102177	HS-grad	Farming	1	0	0	0	0	0	1	1	1	0	NA
101074	10th	Handler	1	1	0	1	1	0	0	1	1	0	NA

4. To display the number of rows and columns in the `ore.frame` object `SUPPLEMENTARY_DEMOGRAPHICS`, use `z.show(dim(SUP_DEM))`

```
z.show(dim(SUP_DEM))
```

```
(4500, 14)
```

5. View the data type of the columns in CUST_DF with the @desc operator.

```
SUP_DEM@desc
```

name <chr>	Sclass <chr>
CUST_ID	numeric
EDUCATION	character
OCCUPATION	character
HOUSEHOLD_SIZE	character
YRS_RESIDENCE	numeric
AFFINITY_CARD	numeric
BULK_PACK_DISKETTES	numeric
FLAT_PANEL_MONITOR	numeric
HOME_THEATER_PACKAGE	numeric
BOOKKEEPING_APPLICATION	numeric
PRINTER_SUPPLIES	numeric
Y_BOX_GAMES	numeric
OS_DOC_SET_KANJI	numeric
COMMENTS	character

6. Run the following command to check if there are any missing values in the data. The following code gives you the total number of missing values in the CUST_DF proxy object.

```
sum(is.na(SUP_DEM))
```

```
205
```

The value 205 indicates that there are missing values in the SUP_DEM proxy object.

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. You can enable ADP during the Build Model stage. The commonly used methods of data preparation are binning, normalization, and missing value treatment.

See [How ADP Transforms the Data](#) to understand how ADP prepares the data for some algorithms.

This completes the data understanding and data preparation stage.

3.1.3 Build Model

This model is designed to classify data into predefined categories by learning from training data.

Algorithm Selection

You can choose one of the following in-database algorithms to solve a classification problem:

- Decision Tree

- Generalized Linear Model
- Naive Bayes
- Neural Network
- Random Forest
- Support Vector Machine

Here you will be using the Support Vector Machine algorithms because the SVM classification is one of the algorithms that supports binary classification.

1. Split the data into train and test data sets. The train set is used to train the model so that it learns the hidden patterns and the test set is used to evaluate the trained model. Split the DEMO_DF data with 60 percent of the records for the train data set and 40 percent for the test data set.

```
sampleSize <- .4 * nrow(DEMO_DF)
index <- sample(1:nrow(DEMO_DF),sampleSize)
group <- as.integer(1:nrow(DEMO_DF) %in% index)

rownames(DEMO_DF) <- DEMO_DF$CUST_ID
DEMO_DF.train <- DEMO_DF[group==FALSE,]
class(DEMO_DF.train)

DEMO_DF.test <- DEMO_DF[group==TRUE,]
class(DEMO_DF.test)

'ore.frame'
'ore.frame'
```

2. After splitting the data, let's see the count of rows in train and test to see if any rows are left out in either of the datasets.

```
cat("\nTraining data: ")
dim(DEMO_DF.train)
cat("\nTest data: ")
dim(DEMO_DF.test)
```

```
Training data:  2700 13
Test data: 1800 13
```

3. Build your model using the ore.odmSVM function, which creates a Support Vector Machine model using the training data. The ore.odmSVM function is the R interface to the in-database SVM algorithm. Then we will make the prediction using this model for our test data.

```
ore.exec(
  "BEGIN DBMS_DATA_MINING.DROP_MODEL(model_name =>
'SVM_CLASSIFICATION_MODEL');
  EXCEPTION WHEN OTHERS THEN NULL; END;"
)

MOD <- ore.odmSVM(
  formula = AFFINITY_CARD ~ .,
  data = DEMO_DF.train,
```

```

type = "classification",
kernel.function = "system.determined",
odm.settings = list(model_name = "SVM_CLASSIFICATION_MODEL")
)

RES <- predict(
  object = MOD,
  data = DEMO_DF.test,
  type = c("raw", "class"),
  norm.votes = TRUE,
  cache.model = TRUE,
  supplemental.cols = c(
    "CUST_ID", "AFFINITY_CARD", "EDUCATION",
    "HOUSEHOLD_SIZE", "OCCUPATION", "YRS_RESIDENCE"
  )
)

```

3.1.4 Evaluate

Before you make predictions using your model on new data, you should first evaluate model accuracy. You can evaluate the model using different methods.

Show Model Accuracy

To check the accuracy of our model, we use a confusion matrix. The confusion matrix is a table that shows the correct model predictions and incorrect predictions for each class. After creating the confusion matrix, the code calculates the accuracy of the model by dividing the number of correct predictions by the total number of predictions.

```
CMATRIX <- with(RES, table(AFFINITY_CARD, PREDICTION))
```

```
CMATRIX
```

	PREDICTION	
AFFINITY_CARD	0	1
0	1206	145
1	180	269

To show the model accuracy, run the following statements:

```

ACCURACY <- CMATRIX / sum(CMATRIX)
round(sum(diag(ACCURACY)), 3) * 100

```

```
83.6
```

The result of the confusion matrix shows that the accuracy on the test set is 83.6%

Show Prediction Results

Here you will display the prediction results.

1. To display the prediction results, run the following code:

```
z.show(ore.sort(RES[(RES$"'1'" > 0.5)], by = c("'1'")))
```


X.0.	X.1.	CUST_ID	AFFINITY_CARD	EDUCATION	HOUSEHOLD_SIZE	OCCUPATION	YRS_RESIDENCE	PREDICTION
0.0293122324132142	0.970687767586786	104459	0	7th-8th	3	Sales	4	1
0.0293122321406716	0.970687767586328	103506	0	HS-grad	4.5	Cleric.	2	1
0.0293122320500647	0.970687767940035	101637	0	Assoc-V	3	TechSup	3	1
0.0293122320551593	0.970687767944841	103998	0	HS-grad	3	Cleric.	3	1
0.0293122320477496	0.97068776795225	100772	0	HS-grad	4.5	Cleric.	2	1
0.0293122320317564	0.970687767968244	102450	0	HS-grad	3	Sales	1	1
0.0293122320202633	0.970687767979737	100176	1	HS-grad	3	Protec.	4	1

2. To display the prediction result using ROC Curve, Lift Chart, and Distribution Chart, run the following code:

```
# BAR PLOT
res <- ore.pull(RES)
sensitivity <- res[order(res$'1',decreasing = TRUE), ]
sens <- sum(sensitivity$'0')/sum(sensitivity$'0') -
cumsum(sensitivity$'0')/sum(sensitivity$'0')
spec <- cumsum(sensitivity$'1')/sum(sensitivity$'1')

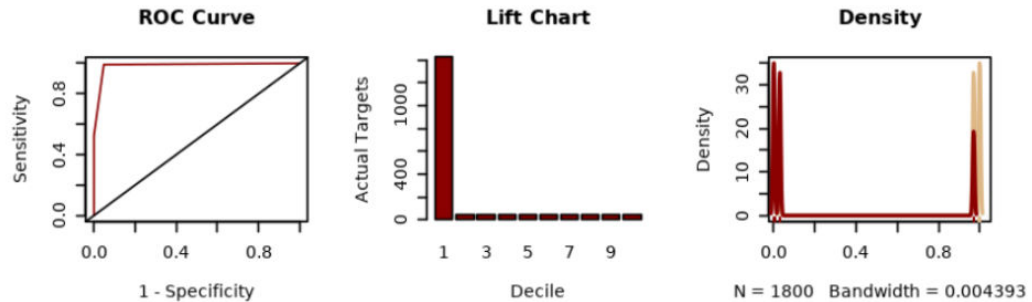
# LIFT CHART
decile2 <- quantile(sensitivity$'1', probs = seq(.1, .9, by = .1))
df_sens <- as.data.frame(sensitivity$'1', col.names = c("sens"))
df_sens$decile = as.numeric(cut(1-cumsum(df_sens$sens), breaks=10))

# DISTRIBUTION CHART
dx <- density(res$'0')
dx2 <- density(res$'1')

# PLOTS 3x1
par(mfrow=c(3,3))
plot(1 - spec, sens, type = "l", col = "darkred", ylab = "Sensitivity",
xlab = "1 - Specificity", main = 'ROC Curve')
abline(c(0,0),c(1,1))
paste("AUC: ", round(sum(spec*diff(c(0, 1 - sens))),3))

barplot(table(df_sens$decile), xlab = 'Decile', ylab = 'Actual Targets',
main = 'Lift Chart', col = "darkred")

plot(dx, lwd = 2, col = "burlywood",
      main = "Density")
lines(dx2, lwd = 2, col = "darkred")
# Add the data-points with noise in the X-axis
rug(jitter(res$'0'),col='burlywood')
rug(jitter(res$'1'),col='darkred')
```



3.1.5 Deploy the Model

The machine learning model, `SVM_CLASSIFICATION_MODEL`, has been successfully trained and exists in your schema as a first-class database object. While you can use this model directly from R, for database applications, you can also run it directly from SQL queries.

Using the SVM Model in SQL

To facilitate this, you will create a SQL table, `SVM_TEST_TABLE`, mirroring the structure of your R data frame, `DEMO_DF`. This will allow you to seamlessly integrate the model's predictions into your database work flows. To use the trained SVM model into your SQL environment, follow the steps below:

- Create a table that mirrors the structure of your R data frame, enabling seamless prediction workflows within the database.

Note

The data provided to the model through SQL queries must be prepared in the same manner as the data used to build the model in R.

```
ore.drop(table = "SVM_TEST_TABLE")
ore.create(DEMO_DF, table = "SVM_TEST_TABLE")
```

- Use the SQL Interface to score data and display the prediction results.

```
SELECT CUST_ID,
       round(PREDICTION_YRS_RES,3) PRED_YRS_RES,
       RTRIM(TRIM(SUBSTR(OUTPRED."Attribute1",17,100)), 'rank="1"/>')
FIRST_ATTRIBUTE,
       RTRIM(TRIM(SUBSTR(OUTPRED."Attribute2",17,100)), 'rank="2"/>')
SECOND_ATTRIBUTE,
       RTRIM(TRIM(SUBSTR(OUTPRED."Attribute3",17,100)), 'rank="3"/>')
THIRD_ATTRIBUTE
FROM (SELECT CUST_ID,
            PREDICTION(SVM_CLASSIFICATION_MODEL USING *)
PREDICTION_YRS_RES,
            PREDICTION_DETAILS(SVM_CLASSIFICATION_MODEL USING *) PD
FROM SVM_TEST_TABLE
WHERE CUST_ID < 100015
```

```

ORDER BY CUST_ID) OUT,
      XMLTABLE('/Details'
        PASSING OUT.PD
        COLUMNS
          "Attribute1" XMLType PATH 'Attribute[1]',
          "Attribute2" XMLType PATH 'Attribute[2]',
          "Attribute3" XMLType PATH 'Attribute[3]') OUTPRED

```

CUST_ID	PRED_YRS_RES	FIRST_ATTRIBUTE	SECOND_ATTRIBUTE	THIRD_ATTRIBUTE
100001	0	"HOUSEHOLD_SIZE" actualValue="2" weight=".941"	"OS_DOC_SET_KANJI" actualValue="0" weight="-.001"	"Y_BOX_GAMES" actualValue="0" weight="-.026"
100002	0	"HOUSEHOLD_SIZE" actualValue="2" weight=".941"	"OS_DOC_SET_KANJI" actualValue="0" weight="-.001"	"Y_BOX_GAMES" actualValue="0" weight="-.026"
100003	0	"HOUSEHOLD_SIZE" actualValue="2" weight=".941"	"OS_DOC_SET_KANJI" actualValue="0" weight="-.001"	"Y_BOX_GAMES" actualValue="0" weight="-.026"
100004	0	"HOUSEHOLD_SIZE" actualValue="2" weight=".941"	"OS_DOC_SET_KANJI" actualValue="0" weight="-.001"	"Y_BOX_GAMES" actualValue="0" weight="-.026"
100005	0	"OCCUPATION" actualValue="Crafts" weight=".941"	"OS_DOC_SET_KANJI" actualValue="0" weight="-.001"	"Y_BOX_GAMES" actualValue="0" weight="-.026"

The SQL code demonstrates how to deploy and use a trained SVM classification model (SVM_CLASSIFICATION_MODEL) within a database environment. It showcases the process of scoring new data, extracting predicted values (PREDICTION_YRS_RES), and retrieving relevant attributes (FIRST_ATTRIBUTE, SECOND_ATTRIBUTE, THIRD_ATTRIBUTE) using the PREDICTION_DETAILS function.

Using the SVM Model in R

You can also make predictions and obtain prediction details directly from R using the following code:

```

z.show(predict(
  object = MOD,
  newdata = DEMO_DF.test,
  supplemental.cols = c("CUST_ID"),
  topN.attrs = 3
))

```

The output appears as follows:

CUST_ID	PREDICTION	NAME_1	VALUE_1	WEIGHT_1	NAME_2	VALUE_2	WEIGHT_2	NAME_3	VALUE_3	WEIGHT_3
100002	0	HOUSEHOLD_SIZE	2	0.941	Y_BOX_GAMES	0	-0.026	OCCUPATION	Prof.	-0.029
100003	0	HOUSEHOLD_SIZE	2	0.941	Y_BOX_GAMES	0	-0.026	OCCUPATION	Sales	-0.029
100005	0	Y_BOX_GAMES	0	-0.026	NA	NA	NA	NA	NA	NA
100011	0	HOUSEHOLD_SIZE	2	0.029	NA	NA	NA	NA	NA	NA
100012	1	OCCUPATION	Prof.	0.941	Y_BOX_GAMES	0	0.183	NA	NA	NA

Deploying the Model to Other Databases or OML Services

To deploy the model to other databases or OML Services, follow these steps:

- **Export the Model:**

- Use the `DBMS_DATA_MINING.EXPORT_SERMODEL` procedure to export the model to a BLOB object.
- Save the BLOB object to a file or another storage location.
- **Import the Model into Another Database:**
 - In the target database, use `DBMS_DATA_MINING.IMPORT_SERMODEL` to import the model from the BLOB object.
- **Deploy the Model to OML Services:**
 - Use the OML REST API to upload the model and create a REST endpoint for scoring. Refer to the OML Services documentation for specific instructions.

For more information, see [DBMS_DATA_MINING Package](#) and [OML Services Documentation](#).

This use case identified customers most likely to be positive responders to an Affinity Card loyalty program using a Support Vector Machine (SVM) classification model. Thus, the model can be used to predict which customers are likely to become high-value customers with the Affinity Card program, allowing the store to focus their marketing resources more effectively.

3.2 Clustering Use Case

A retail store has information about its customers' behavior and the purchases they make. With that data, they would like you to analyze and identify if there are groups of customers with similar characteristics. Use Oracle Machine Learning to segment customers by finding clusters in the data set that can be then used to support targeted marketing campaigns to increase retail sales. In this use case, you will learn how to identify such segments using the k-Means algorithm.

Data Understanding

To understand the data, perform the following tasks:

1. Access data
 2. Explore data
- [Load Data](#)
Access the data set from the SH Schema and explore the data to understand the attributes.
 - [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.
 - [Build Model](#)
To evaluate a model's performance, it is common practice to split the data into training and test sets. This allows you to assess how well the model generalizes to unseen data. However, in unsupervised learning, such as clustering, there are no labels or predictors available to calculate accuracy or evaluate performance. As a result, you can use the entire dataset to build the model without the need to split it. Since there is no ground truth to compare the results against, the training-test split is neither applicable nor useful in unsupervised learning.
 - [Deploy the Model](#)
Here are several approaches to deploy your OML4R model and leverage its insights:

3.2.1 Load Data

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

Access Data

You will be using the CUSTOMERS and SUPPLEMENTARY_DEMOGRAPHICS tables available in the SH schema.

See [SH.CUSTOMERS](#) for information about the CUSTOMERS table in SH Schema.

The following table displays information about the attributes from SUPPLEMENTARY_DEMOGRAPHICS:

Attribute Name	Data Type	Information
CUST_ID	Numeric	The ID of the customer
EDUCATION	Character	Education level attained
OCCUPATION	Character	Occupation of the customer
HOUSEHOLD_SIZE	Character	Number of people living at residence
YRS_RESIDENCE	Numeric	Number of years customer lived at current residence
AFFINITY_CARD	Character	Indicates whether the customer holds an affinity card. 1 means Yes. 0 means No.
BULK_PACK_DISKETTES	Character	Product. Indicates whether the customer purchased the bulk pack diskettes. 1 means Yes. 0 means No.
FLAT_PANEL_MONITOR	Character	Product. Indicates whether the customer purchased flat panel monitor. 1 means Yes. 0 means No
HOME_THEATER_PACKAGE	Character	Product. Indicates whether the customer purchased home theatre package. 1 means Yes. 0 means No
BOOKKEEPING_APPLICATION	Character	Product. Indicates whether the customer purchased bookkeeping application. 1 means Yes. 0 means No
PRINTER_SUPPLIES	Character	Product. Indicates whether the customer purchased printer supplies. 1 means Yes. 0 means No
Y_BOX_GAMES	Character	Product. Indicates whether the customer purchased YBox Games. 1 means Yes. 0 means No

Attribute Name	Data Type	Information
OS_DOC_SET_KANJI	Character	Product. Indicates whether the customer purchased the Kanji character set for the operating system documentation. 1 means Yes. 0 means No
COMMENTS	Character	Comments from customers

To access database data from R using OML4R, you must first create a proxy object in R that represents a database table, view, or query. In this example, the proxy object is created using a query. Create proxy objects for SUPPLEMENTARY_DEMOGRAPHICS and CUSTOMERS and then merge them by inner join on a key column, in this case, CUST_ID. Assess the data to identify data types and data quality issues. Look for missing values, outlier numeric values, or inconsistently labeled categorical values.

1. Run the following command in an R interpreter paragraph (using %r) in an OML notebook (or similar notebook environment) to import the Oracle Machine Learning for R libraries and suppress warnings regarding row ordering. Alternatively, this code can be run from the R command line or tools like RStudio.

```
library(ORE)
options(ore.warn.order=FALSE)
```

2. Use the `ore.sync` function to create the `ore.frame` object that is a proxy for the CUSTOMERS table in the SH schema database table.

CUST_ID	CUST_GENDER	CUST_MARITAL_STATUS	CUST_YEAR_OF_BIRTH	CUST_INCOME_LEVEL	CUST_CREDIT_LIMIT
49671	M	married	1976	G: 130,000 - 149,999	1500
3228	M	NA	1964	G: 130,000 - 149,999	7000
6783	M	single	1942	G: 130,000 - 149,999	11000
10338	M	married	1977	G: 130,000 - 149,999	1500
13894	M	NA	1949	G: 130,000 - 149,999	9000
17449	M	single	1950	G: 130,000 - 149,999	9000

3. Use the `ore.sync` function to create the `ore.frame` object that is a proxy for the SUPPLEMENTARY_DEMOGRAPHICS table in the SH schema database table.

```
ore.sync(query = c("SUPPLEMENTARY_DEMOGRAPHICS" = "select CUST_ID,
HOUSEHOLD_SIZE, YRS_RESIDENCE, TO_CHAR(Y_BOX_GAMES) Y_BOX_GAMES from
SH.SUPPLEMENTARY_DEMOGRAPHICS"))
# The TO_CHAR function is used to have Y_BOX_GAMES treated as a
categorical variable, not a numeric variable.
z.show(head(SUPPLEMENTARY_DEMOGRAPHICS))
```

CUST_ID	HOUSEHOLD_SIZE	YRS_RESIDENCE	Y_BOX_GAMES
102547	1	0	1
101050	1	0	1
100040	1	0	1
102117	1	0	1
101074	1	1	1
104179	1	1	1

3.2.2 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.

To gain a broader understanding of the data and identify potential issues, we will now explore the dataset, focusing on data quality assessment and identifying missing or outlier values.

1. To determine the number of rows and columns in the `ore.frame` object `CUSTOMERS`, use `dim(CUSTOMERS)`.

```
dim(CUSTOMERS)
55500    6
```

2. To determine the number of rows and columns in the `ore.frame` object `SUPPLEMENTARY_DEMOGRAPHICS`, use `dim(SUPPLEMENTARY_DEMOGRAPHICS)`

```
dim(SUPPLEMENTARY_DEMOGRAPHICS)
4500    4
```

3. Create a new `ore.frame` object `CUST_DF` by merging the table `CUSTOMERS` and `SUPPLEMENTARY_DEMOGRAPHICS` with an inner join on the common column `CUST_ID`. The merge function joins one `ore.frame` to another `ore.frame`.

```
CUST_DF <- merge(SUPPLEMENTARY_DEMOGRAPHICS,CUSTOMERS, by="CUST_ID")
```

4. To display first 5 rows of `CUST_DF` data run the following code:

```
z.show(head(CUST_DF,5))
```

CUST_ID	CUST_GENDER	CUST_MARITAL_STATUS	CUST_YEAR_OF_BIRTH	CUST_INCOME_LEVEL	CUST_CREDIT_LIMIT	HOUSEHOLD_SIZE	YRS_RESIDENCE	Y_BOX_GAMES
100100	F	Married	1959	J. 190,000 - 249,999	10000	4.5	4	0
100200	M	NeverM	1983	L. 300.0 - 3 and above	9000	1	2	1
100300	M	Married	1981	G. 130,000 - 149,999	10000	3	4	0
100400	M	Divorc	1944	C. 50,000 - 89,999	9000	6.5	6	0
100500	M	NeverM	1983	H. 150,000 - 189,999	15000	1	2	1

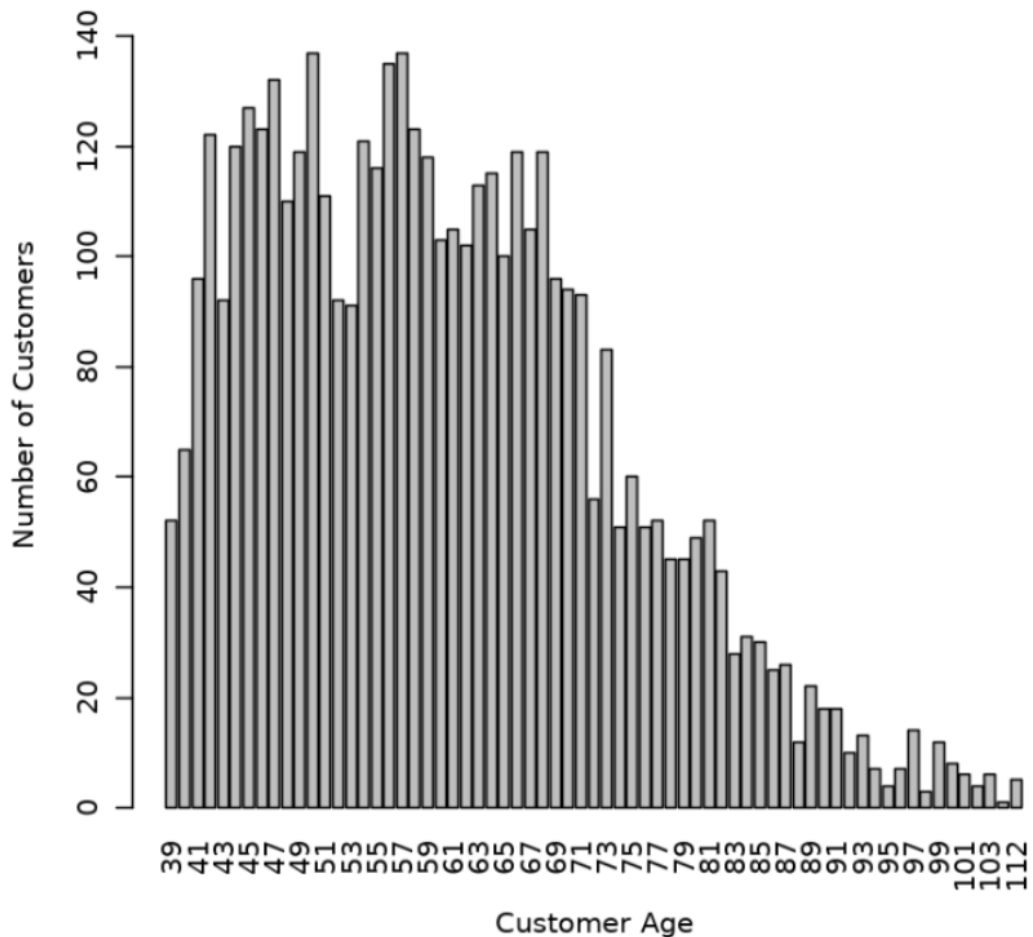
5. To get the dimensions using `CUST_DF` proxy object, use 'dim':

```
dim(CUST_DF)

4500    9
```

6. To transform the column `CUST_YEAR_OF_BIRTH` to `CUST_AGE` in the `CUST_DF` proxy object and produce a barplot of the distribution of customer ages, use the following code.

```
Date1 <- format(Sys.Date(), "%Y")
Date2 <- as.numeric(Date1)
CUST_DF$CUST_AGE <- Date2-CUST_DF$CUST_YEAR_OF_BIRTH
CUST_DF$CUST_YEAR_OF_BIRTH <- NULL
tbl <- with(CUST_DF, table(CUST_AGE))
barplot(tbl, ylim=c(0,150), ylab = "Number of Customers", xlab = "Customer
Age", las=3)
```



7. View the data type of the columns in `CUST_DF` with the `@desc` operator, which is crucial for understanding to understand your data and perform calculations accurately.

```
CUST_DF@desc
```


name	Sclass
<chr>	<chr>
CUST_ID	numeric
EDUCATION	character
OCCUPATION	character
HOUSEHOLD_SIZE	character
YRS_RESIDENCE	numeric
CUST_GENDER	character
CUST_MARITAL_STATUS	character
CUST_YEAR_OF_BIRTH	integer
CUST_INCOME_LEVEL	character
CUST_CREDIT_LIMIT	numeric

8. To check if there are any missing values in the data, run the following code. The following code gives you the total number of missing values in the `CUST_DF` proxy object.

```
sum(is.na(CUST_DF))
0
```

The value 0 indicates that there are no missing values in the `CUST_DF` proxy object.

9. Use the `crosstab` method to perform a cross-column analysis of the `ore.frame` object in the database. By default, it computes a frequency table for the columns unless a column and an aggregation function have been passed to it. In this example, the `crosstab` function displays the distribution of unique values of `CUST_CREDIT_LIMIT` along the x-axis and its occurrence frequency along the y-axis. In the output, click the Bar chart. In the Settings tab, choose “`CUST_CREDIT_LIMIT`” as the Group By column, and use “Last” as the Aggregate Duplicates function.

Settings

Setup

Customization

Height

auto

Aggregate Duplicates

Last

Series to Show

ORE.FREQ x

Group By

CUST_CREDIT_LIMIT x

```
ct <- ore.crosstab(~CUST_CREDIT_LIMIT, data=CUST_DF)
z.show(ct)
```



10. To compute the statistics of the CUST_DF table, use the summary function.

```
options(width = 80)
summary(subset(CUST_DF, select = -CUST_ID))
```

```
CUST_GENDER          CUST_MARITAL_STATUS    CUST_INCOME_LEVEL
Length:4500          Length:4500              Length:4500
Class :ore.character  Class :ore.character      Class :ore.character
Mode :character       Mode :character           Mode :character
```

```
CUST_CREDIT_LIMIT  HOUSEHOLD_SIZE          YRS_RESIDENCE      Y_BOX_GAMES
Min.   : 1500      Length:4500          Min.   : 0.000     Min.   :0.0000
1st Qu.: 5000      Class :ore.character  1st Qu.: 3.000     1st Qu.:0.0000
Median : 9000      Mode :character      Median : 4.000     Median :0.0000
Mean   : 7924                                Mean   : 4.022     Mean   :0.3124
3rd Qu.:11000                                3rd Qu.: 5.000     3rd Qu.:1.0000
Max.   :15000                                Max.   :14.000     Max.   :1.0000
```

```
CUST_AGE
Min.   : 39.00
1st Qu.: 49.00
Median : 59.00
Mean   : 60.38
3rd Qu.: 69.00
Max.   :112.00
```

This completes the data understanding stage.

Data Preparation

Before building the model you want to clean the data, if needed. Usually, data can contain outliers that may form a separate cluster, which can affect model quality. The command below defines the function `filter_outliers` to calculate the interquartile range for a dataframe object. The function `remove_outliers` uses a for loop to compute the interquartile range for the list of features. The user-defined function `remove_outliers` uses the interquartile range to find outliers in the data and remove them.

```
# create filter outliers function
filter_outliers <- function(x) {

  # calculate first quantile
```

```

Quantile1 <- quantile(x, probs=.25)

# calculate third quantile
Quantile3 <- quantile(x, probs=.75)

# calculate inter quartile range
IQR = Quantile3-Quantile1

# return true or false
x < Quantile3 + (IQR*1.5) & x > Quantile1 - (IQR*1.5)
}

# create remove outliers function
remove_outliers <- function(dataframe,
                             columns=names(dataframe)) {

  # for loop to traverse in columns vector
  for (col in columns) {

    # remove observation if it satisfies outlier function
    dataframe <- dataframe[filter_outliers(dataframe[[col]]),]
  }
  ore.pull(dataframe)
}

CUST_DF_CLEAN <- remove_outliers(CUST_DF, c('CUST_AGE', 'CUST_CREDIT_LIMIT',
'YRS_RESIDENCE', 'Y_BOX_GAMES'))
CUST_DF_CLEAN <- ore.push(CUST_DF_CLEAN)
dim(CUST_DF_CLEAN)

4233 9

```

This completes the data preparation stage.

3.2.3 Build Model

To evaluate a model's performance, it is common practice to split the data into training and test sets. This allows you to assess how well the model generalizes to unseen data. However, in unsupervised learning, such as clustering, there are no labels or predictors available to calculate accuracy or evaluate performance. As a result, you can use the entire dataset to build the model without the need to split it. Since there is no ground truth to compare the results against, the training-test split is neither applicable nor useful in unsupervised learning.

Algorithm Selection

Using OML4R, you can choose one of the following algorithms to solve a clustering problem:

1. K-Means (KM)
2. Expectation-Maximization (EM)
3. Orthogonal Partitioning Cluster (O-Cluster)

The *k*-Means (KM) algorithm is a distance-based clustering algorithm that partitions the data into a specified number of clusters. Distance-based algorithms are based on the concept that nearby data points are more related to each other than data points that are farther away. The algorithm iteratively tries to minimize the within-cluster variance with respect to its nearest

cluster centroid. The Expectation-Maximization(EM) algorithm uses a probabilistic clustering based on a density estimation algorithm. The Orthogonal Partitioning Cluster (O-Cluster) algorithm is a density-based clustering method designed for large, high-dimensional datasets.

A good starting point for clustering is the K-means algorithm. It works by assigning each data point to the closest cluster center (centroid). Unlike some methods, K-means doesn't make assumptions about the underlying shapes of the clusters. This simplicity makes it a user-friendly choice for many applications, and it will be the method we use for this use case.

We will use the elbow method to determine the number of clusters in the dataset. The elbow method uses 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. We will start with one cluster, and continue specifying one cluster through 8 clusters. We will look for the "elbow" in the resulting dispersion curve to assess which number of clusters seems best.

To specify model settings and build a k-Means model object that will segment the data, run the following command. The settings are given as key-value or dictionary pairs where it refers to parameters name and value setting respectively. Here are some of the settings specified:

KMNS_ITERATIONS, KMNS_RANDOM_SEED, KMNS_CONV_TOLERANCE, KMNS_NUM_BINS, KMNS_DETAILS, and PREP_AUTO. The k-Means algorithm uses the number of clusters (k) and other settings to configure the algorithm, as shown here:

```
settings = list(
  KMNS_ITERATIONS = 15,
  KMNS_RANDOM_SEED = 1,
  KMNS_CONV_TOLERANCE = 0.001,
  KMNS_NUM_BINS = 11,
  KMNS_DETAILS = "KMNS_DETAILS_HIERARCHY",
  CASE_ID_COLUMN_NAME = "CUST_ID"
)

KM.MOD <- ore.odmKMeans(
  formula = ~.-CUST_ID,
  data = CUST_DF_CLEAN,
  num.centers = 3,
  odm.settings = settings
)

KM.MOD
```

The following is the list of algorithm settings used in this example:

- **KMNS_ITERATIONS:** Specifies the maximum number of iterations for k-Means that are allowed. The default number of iterations is 20.
- **KMNS_RANDOM_SEED:** The random number generator uses a number called the random seed to initialize itself. The random number generator generates random numbers that are used by the k-Means algorithm to select the initial cluster centroid. This setting controls the seed of the random generator used during the k-Means initialization. It must be a non-negative integer value. The default is 0.
- **KMNS_CONV_TOLERANCE:** Convergence Tolerance is the threshold value for the change in the centroids between consecutive iterations of the algorithm. This setting is used to specify the minimum Convergence Tolerance for k-Means. The algorithm iterates until the minimum Convergence Tolerance is satisfied or until the maximum number of

iterations, specified in `KMNS_ITERATIONS`, is reached. Decreasing the Convergence Tolerance produces a more accurate solution but may result in longer run times. The default Convergence Tolerance is 0.001.

- `KMNS_NUM_BINS`: Number of bins in the attribute histogram produced by k-Means. The bin boundaries for each attribute are computed globally on the entire training data set. The binning method is equi-width. All attributes have the same number of bins with the exception of attributes with a single value that have only one bin.
- `KMNS_DETAILS`: This setting determines the level of cluster details that is computed during the build. The value `KMNS_DETAILS_ALL` means that the cluster hierarchy, record counts, and descriptive statistics (means, variances, modes, histograms, and rules) are computed and this is the default value. The value `KMNS_DETAILS_NONE` means no cluster details are computed and only the scoring information persisted. The value `KMNS_DETAILS_HIERARCHY` means cluster hierarchy and cluster record counts are computed.
- `PREP_AUTO`: Used to specify whether to use automatic data preparation or if the user is responsible for algorithm-specific data preparation. By default, it is enabled with a constant value as `'PREP_AUTO': PREP_AUTO_ON` and requires the `DBMS_DATA_MINING` package. Alternatively, it can also be specified as `'PREP_AUTO': 'ON'`.
- `~.-CUST_ID`: This argument is passed to the function to cluster the data in the `CUST_DF_CLEAN` data frame, excluding the `CUST_ID` column.
- `CUST_DF_CLEAN`: The data frame that needs to be clustered.
- `num.centers`: Defines the number of clusters for a clustering model. A value greater than or equal to 1. The default value is 10.
- `odm.settings`: A list to specify in-database algorithm parameter settings. This argument is applicable to building a model in Database 12.2 or later. Each list element's name and value refer to the parameter setting name and value, respectively. The setting value must be numeric or string.
The output appears as follows:

Call:

```
ore.odmKMeans(formula = ~. - CUST_ID, data = CUST_DF_CLEAN, num.centers =
3,
  odm.settings = settings)
```

Settings:

	value
<code>clus.num.clusters</code>	3
<code>block.growth</code>	2
<code>conv.tolerance</code>	0.001
<code>details</code>	<code>details.hierarchy</code>
<code>distance</code>	<code>euclidean</code>
<code>iterations</code>	15
<code>min.pct.attr.support</code>	0.1
<code>num.bins</code>	11
<code>random.seed</code>	1
<code>split.criterion</code>	<code>variance</code>
<code>odms.details</code>	<code>odms.enable</code>
<code>odms.missing.value.treatment</code>	<code>odms.missing.value.auto</code>
<code>odms.sampling</code>	<code>odms.sampling.disable</code>
<code>prep.auto</code>	<code>ON</code>

3.2.4 Deploy the Model

Here are several approaches to deploy your OML4R model and leverage its insights:

Prediction using R API:

This is the simplest method, ideal for quick analysis or prototyping. You can directly use the fitted model within your R environment to make predictions on new data.

```
pred <- predict(km.mod.ere, CUST_DF_CLEAN, supplemental.cols = "CUST_ID")

print(pred) # View predictions for new data
```

Deploy model in different database:

For production deployments within the different database, leverage the built-in functionalities:

- Export: Use DBMS_DATA_MINING.EXPORT_SERMODEL to export the trained model (CUST_CLUSTER_MODEL_ERE) to a BLOB object in Database 1.
- Transfer: Move the BLOB object (e.g., BFile) to Database 2.
- Import: Use DBMS_DATA_MINING.IMPORT_SERMODEL in Database 2 to import the model from the transferred BLOB object.

Running a user-defined R function from R and SQL, and on ADB REST:

For periodic model updates with new data, create a user-defined R function:

- Define model settings (number of clusters, distance metric, etc.).
- Drop any existing model named CUST_CLUSTER_MODEL_ERE (optional).
- Train the model using ore.odmKMeans.
- Optionally, generate predictions for new data and display them.

Schedule this script to run periodically using Oracle's scheduling features.

Example 3-1 Defining the R function in the script repository and running it from R

```
#suppress warnings#
options(warn=-1)

build.km.1 <- function(){
  settings = list('KMNS_ITERATIONS'='10',
    'KMNS_DISTANCE'='KMNS_EUCLIDEAN',
    'KMNS_NUM_BINS'='10',
    'KMNS_DETAILS'='KMNS_DETAILS_ALL',
    'PREP_AUTO'='ON',
    'MODEL_NAME'='CUST_CLUSTER_MODEL_ERE')

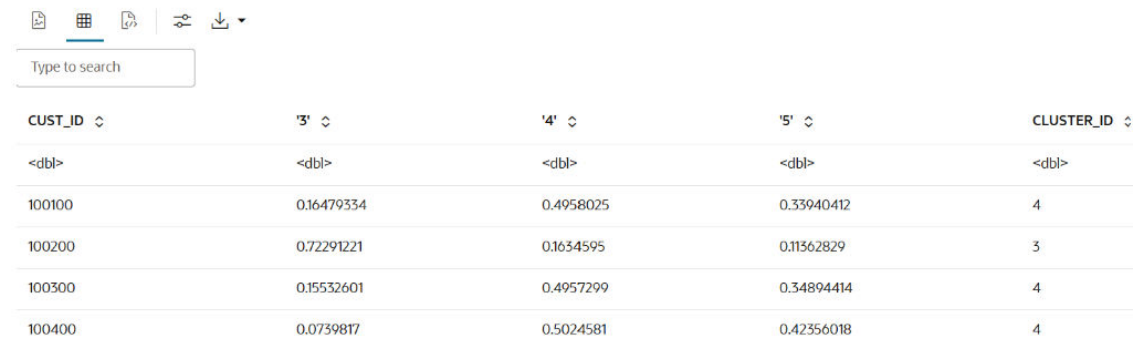
  ore.exec(paste("BEGIN
DBMS_DATA_MINING.DROP_MODEL('CUST_CLUSTER_MODEL_ERE'); EXCEPTION WHEN OTHERS
THEN NULL; END;", sep=""))

  km.mod.ere <- ore.odmKMeans(~ . -CUST_ID, CUST_DF_CLEAN, num.centers=3,
odm.settings=settings)
```

```
# Show predictions
pred <- predict(km.mod.ere, CUST_DF_CLEAN, supplemental.cols="CUST_ID")
pred
}

ore.doEval(FUN = build.km.1)
```

The output appears as follows:



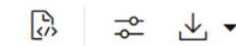
CUST_ID	'3'	'4'	'5'	CLUSTER_ID
100100	0.16479334	0.4958025	0.33940412	4
100200	0.72291221	0.1654595	0.11562829	3
100300	0.15532601	0.4957299	0.34894414	4
100400	0.0739817	0.5024581	0.42356018	4

Example 3-2 Running the user-defined R script from SQL

```
--set the access token
exec rqSetAuthToken('<access token>');

--run user-defined R script from SQL
SELECT *
FROM table(
  rqEval2(
    NULL,
    '{"CUST_ID": "NUMBER", "probability_of_cluster_3": "BINARY_DOUBLE",
"probability_of_cluster_4": "BINARY_DOUBLE",
"probability_of_cluster_5": "BINARY_DOUBLE", "CLUSTER_ID":
"NUMBER"}',
    'build.km.1'));
```

The SQL output appears as follows:



CUST_ID	probability_of_cluster_3	probability_of_cluster_4	probability_of_cluster_5	CLUSTER_ID
100100		0.4691	0.3084	4
100200		0.2713	0.1115	2
100300		0.4205	0.3704	4
100400		0.4054	0.476	5
100500		0.3005	0.0913	2
100600		0.3371	0.2168	2
100700		0.3992	0.4892	5
100800		0.3907	0.5131	5
100900		0.3333	0.4824	5
101000		0.3023	0.633	5
101100		0.4779	0.2069	4
101200		0.2571	0.0794	2
101300		0.48	0.3105	4
101400		0.3347	0.4575	5
CUST_ID	probability_of_cluster_3	probability_of_cluster_4	probability_of_cluster_5	CLUSTER_ID
101500		0.3101	0.0925	2
101600		0.3342	0.5093	5
101700		0.4421	0.1789	4
101800		0.3338	0.5092	5
101900		0.3446	0.5242	5
102000		0.4673	0.2904	4
102100		0.2961	0.104	2
102200		0.3715	0.3982	5
102300		0.3573	0.3542	4
102400		0.4282	0.3608	4
102500		0.4006	0.4896	5
102600		0.4298	0.1872	4
102700		0.4554	0.3814	4
102800		0.4425	0.224	4
CUST_ID	probability_of_cluster_3	probability_of_cluster_4	probability_of_cluster_5	CLUSTER_ID
102900		0.3129	0.1612	2
103000		0.4704	0.2455	4
103100		0.45	0.4085	4
103200		0.4257	0.2503	4
103300		0.3656	0.1272	2
103400		0.3212	0.5091	5
103500		0.533	0.2657	4
103600		0.2979	0.1003	2
103700		0.4694	0.2842	4
103800		0.2354	0.7056	5
103900		0.3667	0.4846	5
104000		0.4546	0.2471	4
104100		0.3163	0.1168	2
104200		0.276	0.207	2

Example 3-3 Running the R script from ADB REST using CURL command:

```
curl -i -X POST --header "Authorization: Bearer ${token}" \
--header 'Content-Type: application/json' --header 'Accept: application/json' \
-d '{}' \
"<oml-cloud-service-location-url>/oml/api/r-scripts/v1/do-eval/build.km.1"
```

The REST response appears as follows:

```
{
  "result": [
    {
```



```
"probability_of_cluster_5": 0.3084,  
"CUST_ID": 100100,  
"probability_of_cluster_4": 0.4691,  
"2": 0.2224,  
"CLUSTER_ID": 4  
},  
{  
  "probability_of_cluster_5": 0.1115,  
  "CUST_ID": 100200,  
  "probability_of_cluster_4": 0.2713,  
  "2": 0.6172,  
  "CLUSTER_ID": 2  
},  
.....  
{  
  "probability_of_cluster_5": 0.3974,  
  "CUST_ID": 104498,  
  "probability_of_cluster_4": 0.4256,  
  "2": 0.177,  
  "CLUSTER_ID": 4  
},  
{  
  "probability_of_cluster_5": 0.273,  
  "CUST_ID": 104499,  
  "probability_of_cluster_4": 0.4102,  
  "2": 0.3168,  
  "CLUSTER_ID": 4  
}
```

```
]
}
```

Persistent Table for SQL Integration:

There are two different approaches for creating persistent data structures: a dynamic view ('CUST_DF_VIEW') for accessing the latest data and a materialized table ('CUST_DF_CLEAN') for capturing a snapshot of the data.

Example 3-4 Creating a persistent data structure using a dynamic view

- Use the following code to create a view

```
ore.drop(view="CUST_DF_VIEW")
ore.create(CUST_DF,view="CUST_DF_VIEW")
```

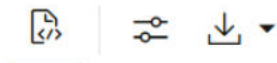
- Use the following SQL query to create a view named KM_PRED_VIEW. This view will dynamically score data based on the existing view CUST_DF_VIEW.

```
CREATE OR REPLACE VIEW KM_PRED_VIEW AS
  SELECT CUST_ID, CLUSTER_ID(CUST_CLUSTER_MODEL_ERE USING *) AS
  CLUSTER_ID, round (CLUSTER_PROBABILITY (CUST_CLUSTER_MODEL_ERE USING *),3)
  AS PROB
  FROM CUST_DF_VIEW;
```

- Use the following code to display first 20 rows of dynamic scoring view 'KM_PRED_VIEW'

```
select * from KM_PRED_VIEW
where rownum < 21;
```

The output appears as follows:



CUST_ID	CLUSTER_ID	PROB
100100	4	0.496
100200	3	0.723
100300	4	0.496
100400	4	0.502
100500	3	0.727
100600	3	0.542
100700	4	0.579
100800	4	0.569
100900	5	0.561
101000	5	0.491
101100	4	0.544
101200	3	0.778
101300	4	0.613
101400	5	0.549
CUST_ID	CLUSTER_ID	PROB
101500	3	0.716
101600	5	0.541
101700	4	0.415
101800	5	0.54
101900	5	0.506
102000	4	0.459

20 rows selected.

Example 3-5 Creating a persistent data structure using a materialized table

- Use the following code to create a table named `CUST_DF_CLEAN` to store the cleaned data in the database.

```
ore.drop(table="CUST_DF_CLEAN")
ore.create(CUST_DF_CLEAN,table="CUST_DF_CLEAN")
```

- Use the following code to create a table named `KM_SCORE_TABLE`, which will store a static snapshot of the scoring results based on the data in the `CUST_DF_CLEAN` table.

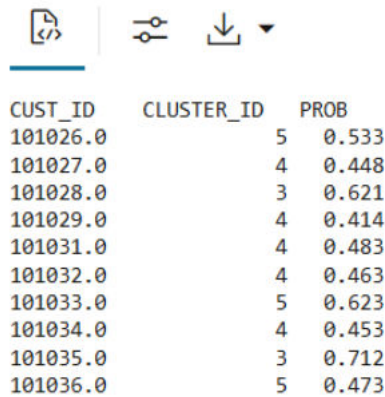
```
DROP TABLE KM_SCORE_TABLE;

CREATE TABLE KM_SCORE_TABLE AS
  SELECT CUST_ID,
         CLUSTER_ID(CUST_CLUSTER_MODEL_ERE USING *) AS CLUSTER_ID,
         round(CLUSTER_PROBABILITY (CUST_CLUSTER_MODEL_ERE USING *),3) AS
PROB
  FROM CUST_DF_CLEAN;
```

- Use the following code to display the first 10 rows of the scoring snapshot table.

```
select * from KM_SCORE_TABLE where rownum <= 10;
```

The output appears as follows:



CUST_ID	CLUSTER_ID	PROB
101026.0	5	0.533
101027.0	4	0.448
101028.0	3	0.621
101029.0	4	0.414
101031.0	4	0.483
101032.0	4	0.463
101033.0	5	0.623
101034.0	4	0.453
101035.0	3	0.712
101036.0	5	0.473

10 rows selected.

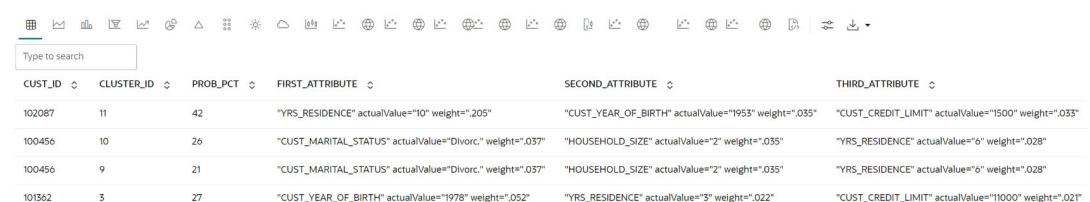
- Use the SQL Interface for scoring and then visualize the results using OML Notebooks. Use the following code to query the table, apply the CLUSTER_SET function for prediction, and extract details from the model output using XML parsing.

```

SELECT CUST_ID,
       CLUSTER_ID,
       ROUND(PROB*100,0) PROB_PCT,
       RTRIM(TRIM(SUBSTR(OUTPRED."Attribute1",17,100)),'rank="1"/>')
FIRST_ATTRIBUTE,
       RTRIM(TRIM(SUBSTR(OUTPRED."Attribute2",17,100)),'rank="2"/>')
SECOND_ATTRIBUTE,
       RTRIM(TRIM(SUBSTR(OUTPRED."Attribute3",17,100)),'rank="3"/>')
THIRD_ATTRIBUTE
FROM (SELECT CUST_ID, S.CLUSTER_ID, PROBABILITY PROB,
            CLUSTER_DETAILS(KM_CLUSTERING_MODEL USING T.*) DETAIL
      FROM (SELECT V.*, CLUSTER_SET(KM_CLUSTERING_MODEL, NULL, 0.2 USING
*) PSET
          FROM CUST_DF_KM V
          WHERE cust_id = ${CUST_ID = '101362','101362'|'102087'|
'100456'}) T,
          TABLE(T.PSET) S
      ORDER BY 2 DESC) OUT,
      XMLTABLE('/Details'
              PASSING OUT.DETAIL
              COLUMNS
                "Attribute1" XMLType PATH 'Attribute[1]',
                "Attribute2" XMLType PATH 'Attribute[2]',
                "Attribute3" XMLType PATH 'Attribute[3]') OUTPRED

```

The output appears as follows:



CUST_ID	CLUSTER_ID	PROB_PCT	FIRST_ATTRIBUTE	SECOND_ATTRIBUTE	THIRD_ATTRIBUTE
102087	11	42	"YRS_RESIDENCE" actualValue="10" weight=".205"	"CUST_YEAR_OF_BIRTH" actualValue="1953" weight=".035"	"CUST_CREDIT_LIMIT" actualValue="1500" weight=".035"
100456	10	26	"CUST_MARITAL_STATUS" actualValue="Divorc." weight=".037"	"HOUSEHOLD_SIZE" actualValue="2" weight=".035"	"YRS_RESIDENCE" actualValue="6" weight=".028"
100456	9	21	"CUST_MARITAL_STATUS" actualValue="Divorc." weight=".037"	"HOUSEHOLD_SIZE" actualValue="2" weight=".035"	"YRS_RESIDENCE" actualValue="6" weight=".028"
101362	3	27	"CUST_YEAR_OF_BIRTH" actualValue="1978" weight=".052"	"YRS_RESIDENCE" actualValue="3" weight=".022"	"CUST_CREDIT_LIMIT" actualValue="11000" weight=".021"

3.3 Time Series Use Case

You work in an electronics store, and sales of laptops and tablets have increased over the past two quarters. You want to forecast product sales for the next four quarters using historical timestamped data. To do this, you apply the Exponential Smoothing algorithm, which predicts future sales by analyzing patterns over evenly spaced time intervals in the historical data.

Table 3-1 Related Content

Topic	Link
About Time Series	About Time Series
About Model Setting	About Model Setting
Shared Settings	Shared Settings
Time Series Algorithm	Time Series Algorithm

Before you start your OML4R 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 or create a database using one 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**
Use OML Notebooks for Oracle Autonomous Database.

Topics:

- [Access Data](#)
Access the data set from the SH Schema and explore the data to understand the attributes.
- [Explore Data](#)
Explore the data to understand its structure and assess its quality. At this stage, identify the data types and detect any noise present. Check for missing values as well as numeric outliers.
- [Build Model](#)
To build a model using time series data, apply the Exponential Smoothing algorithm to the proxy object ESM_SH_DATA created during the exploratory stage.
- [Evaluate](#)
Evaluate your model by reviewing diagnostic metrics and performing quality checks.

Related Topics

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

3.3.1 Access Data

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

① 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 be using the `SALES` table data from the SH schema.

Examine Data

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

Attribute Name	Information
<code>PROD_ID</code>	The ID of the product
<code>CUST_ID</code>	The ID of the customer
<code>TIME_ID</code>	The timestamp of the purchase of the product in yyyy-mm-dd hh:mm:ss format
<code>CHANNEL_ID</code>	The channel ID of the channel sales data
<code>PROMO_ID</code>	The product promotion ID
<code>QUANTITY_SOLD</code>	The number of individual units or items sold.
<code>AMOUNT_SOLD</code>	The total monetary value of the sales (i.e., the revenue generated)

Identify Target Variable

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

3.3.2 Explore Data

Explore the data to understand its structure and assess its quality. At this stage, identify the data types and detect any noise present. Check for missing values as well as numeric outliers.

Note

Each record in the database is called a case, and each case is identified by a `case_id`. Here, the case id is `TIME_ID`, which serves as the independent variable. You are forecasting sales over evenly spaced time intervals.

The following steps will guide you through exploratory data analysis.

1. Import libraries

Run the following command in an R interpreter paragraph (using `%r`) in a notebook to import the Oracle Machine Learning for R libraries and suppress warnings regarding row ordering. This establishes a default connection to the database for use by OML4R.

Note

The OML4R functions are prefixed with "ore," which refers to the original product name "Oracle R Enterprise."

```
library(ORE)
options(ore.warn.order=FALSE)
```

2. Create a DataFrame proxy object on the SH.SALES table

Use the `ore.sync` function to create the a proxy object `ESM_SALES` for the `SH.SALES` database table. The `ore.sync` function returns an `ore.frame` object that represents the table in OML4R.

This script uses the `ore.sync` function to create a data.frame proxy object to the table `SH.SALES` in the object `ESM_SALES`. It then attaches the synchronized object to the R environment with `ore.attach()`, allowing direct access to `ESM_SALES` as if it were a local data frame. Finally, it displays the first few rows of this data using `head()` combined with ORE's formatted output function `z.show()`.

```
ore.sync(query = c(ESM_SALES = "select * from SH.SALES"))
ore.attach()
z.show(head(ESM_SALES))
```

The output appears as follows:

PROD_ID ◊	CUST_ID ◊	TIME_ID ◊	CHANNEL_ID ◊	PROMO_ID ◊	QUANTITY_SOLD ◊	AMOUNT_SOLD
13	524	1998-01-20 00:00:00	2	999	1	1205.99
13	2128	1998-04-05 00:00:00	2	999	1	1250.25
13	3212	1998-04-05 00:00:00	2	999	1	1250.25
13	3375	1998-04-05 00:00:00	2	999	1	1250.25
13	5204	1998-04-05 00:00:00	2	999	1	1250.25

This output displays the first 6 rows of the `SH.SALES` database table, including all available columns. It allows you to quickly inspect a sample of the sales data stored in the Oracle database directly from your OML4R session.

3. Sales dataset row and column count

To determine the number of rows and columns in the `ore.frame` object `SALES`, use the function `dim(SALES)`.

This script runs an R command inside a notebook to return the number of rows and columns of the `ESM_SALES` data frame or table.

```
%r
```

```
dim(ESM_SALES)
```



918843 · 7

The output shows the number of rows and columns in the `ESM_SALES` dataset to understand the shape and scale of the data you are working with in OML4R.

4. Sales Dataset Column Types

Use the following code to view the data type of each column.

This script retrieves and displays metadata for the `ESM_SALES` object in OML4R, providing details about the structure and columns of the linked Oracle database table.

```
%r
```

```
ESM_SALES@desc
```

The output appears as follows:



```
A data.frame: 7 x 2
  name      Sclass
  <chr>     <chr>
PROD_ID    numeric
CUST_ID    numeric
TIME_ID    POSIXct
CHANNEL_ID numeric
PROMO_ID    numeric
QUANTITY_SOLD numeric
AMOUNT_SOLD numeric
```


The output of `ESM_SALES@desc` is a structure of the `ESM_SALES` object. It includes Column names and Data types.

5. Count of missing values by column

To check if there are any missing values in the data, run the following code. It returns you the total number of missing values in the `ESM_SALES` proxy object to assess data completeness.

```
%r  
  
sum(is.na(ESM_SALES))
```

The output appears as 0, which indicates that there are no missing values in the `ESM_SALES` proxy object.

6. Prepare the data for sales forecasting, start by selecting the required columns from the `SH.SALES` table.

Use the `ore.sync()` function to create a proxy object named `SALES_TIME_AMOUNT` containing the `TIME_ID` and `AMOUNT_SOLD` columns..

This defines a new `ore.data.frame` object in R called `SALES_TIME_AMOUNT` by linking it to a query that selects `TIME_ID` and `AMOUNT_SOLD` columns from the `SH.SALES` table in an Oracle database. It helps in-database analytics without fully loading the data into R.

```
%r  
  
ore.sync(query = c(SALES_TIME_AMOUNT = "select TIME_ID, AMOUNT_SOLD from  
SH.SALES"))  
z.show(head(SALES_TIME_AMOUNT))
```

TIME_ID ↕	AMOUNT_SOLD ↕
1998-01-20	1205.99
1998-04-05	1250.25
1998-04-05	1250.25
1998-04-05	1250.25
1998-04-05	1250.25

The output shows the formatted table with the first 6 rows of the `SALES_TIME_AMOUNT` dataset. It includes two columns:

- `TIME_ID` – A time identifier (date).
- `AMOUNT_SOLD` – A numeric value showing the amount sold for each transaction.

7. Use the `dim()` function to view the dimensions (number of rows and columns) of the `SALES_TIME_AMOUNT` proxy object. This helps confirm the size of the dataset retrieved from the `SH.SALES` table.

The returns the number of rows and columns in the `SALES_TIME_AMOUNT` dataset, showing you the shape of the data queried from the database.

```
%r
```

```
dim(SALES_TIME_AMOUNT)
```



The output of `dim(SALES_TIME_AMOUNT)` shows that the dataset contains 918843 rows and 2 columns. This gives you a quick understanding of the dataset's size and structure.

This gives you a quick understanding of the dataset's size and structure.

3.3.3 Build Model

To build a model using time series data, apply the Exponential Smoothing algorithm to the proxy object `ESM_SH_DATA` created during the exploratory stage.

Oracle provides the Exponential Smoothing algorithm specifically for time series forecasting.

Exponential Smoothing is a forecasting technique that assigns exponentially decreasing weights to past observations. It is a type of moving average method. The Exponential Smoothing Model (ESM) includes components such as trend and seasonality, which can be modeled in either additive or multiplicative forms.

- **Trend** refers to the long-term increase or decrease in the data over time. It captures the general direction or movement of the series, whether upward, downward, or stable.
- **Seasonality** refers to regular, repeating patterns or cycles in the data that occur at fixed intervals, such as daily, monthly, or quarterly fluctuations caused by factors like holidays, weather, or business cycles.

In the additive form, the amplitude of variation (the size of the repeating seasonal fluctuations) is independent of the overall level of the time series, whereas in the multiplicative form, the seasonal variation changes proportionally with the level. Here, level refers to the baseline value or the underlying magnitude of the time series at a given point in time. This suggests a multiplicative model, where seasonal fluctuations are proportional to the level of the series—larger when the overall values are high and smaller when the values are low. Additive models assume that the error (or noise), trend, and seasonality are linear components. These components combine in a recursive way to form the final model.

To build a model using a supervised learning approach, it is common to split the dataset into training and test sets. However, time series modeling differs from classification and regression in that it predicts the next value in a series based on the preceding items in the series. In this case, splitting the dataset is unnecessary because the model always forecasts the current

value using only past data. Although it may appear that the model is trained and tested on the same data, each forecast is based solely on prior time points.

In this use case, you will use the proxy object `ESM_SH_DATA` to build the Exponential Smoothing model.

1. To get help on the Exponential Smoothing Model (ESM), run the following script:

```
%r

help(ore.odmESM)
```

The output of `help(ore.odmESM)` is the function reference for the `ore.odmESM`. It guides you on how to use it to build exponential smoothing models for time series forecasting using in-database analytics in Oracle. This is helpful for learning the syntax, understanding parameters, and seeing example usage.

2. Create a Holt-Winters model using the quarterly setting. The Holt-Winters model is a way to predict future values in a time series by looking at patterns like trends (overall direction) and seasonality (regular ups and downs). It updates its predictions step-by-step using only past data, never looking ahead.

To build the model using the `ESM_SH_DATA` proxy object, run the following statement:

This script deletes any existing model named `ESM_SALES_FORECAST_1`, then trains a Holt-Winters exponential smoothing model to forecast `AMOUNT_SOLD` on a quarterly basis for the next 4 quarters. It uses in-database modeling using `ore.odmESM`, and stores the trained model in the object `MOD` for further use.

```
%r

ore.drop(model = 'ESM_SALES_FORECAST_1')

settings = list(EXSM_INTERVAL = 'EXSM_INTERVAL_QTR',
               EXSM_PREDICTION_STEP = 4,
               EXSM_MODEL = 'EXSM_HW',
               EXSM_SEASONALITY = 4,
               EXSM_ACCUMULATE = 'EXSM_ACCU_TOTAL',
               MODEL_NAME='ESM_SALES_FORECAST_1',
               case_id_column_name = "TIME_ID")

MOD <- ore.odmESM(AMOUNT_SOLD~,
                 SALES_TIME_AMOUNT,
                 odm.settings= settings)
```

Examine the script:

- **EXSM_INTERVAL:** Specifies the interval of the dataset or the unit of interval size, such as day, week, month, etc. This setting applies only to the time column of datetime type. For example, if you want to predict quarterly sales, set this to `EXSM_INTERVAL_QTR`.
- **EXSM_PREDICTION_STEP:** Specifies how many future predictions to make. For example, if you want to predict one value per quarter, setting this to 4 will generate predictions for four quarters into the future.
- **EXSM_MODEL:** Specifies the type of exponential smoothing model to be used. As an example, `EXSM_WINTERS` represents the Holt-Winters triple exponential smoothing model with additive trend and multiplicative seasonality. This type of model considers

various combinations of additive and multiplicative trend, seasonality and error, with and without trend damping.

- **Additive trend:** A trend where the increase or decrease in values happens by adding a constant amount over time.
- **Multiplicative trend:** A trend where values grow or shrink by multiplying by a constant factor over time (e.g., percentage growth).
- **Seasonality:** Refers to regular, repeating patterns or fluctuations in a time series that happen at specific intervals—like daily, weekly, monthly, or yearly.
- **Error components:** The random noise or unexplained fluctuations in the data that aren't captured by trend or seasonality.
- **Trend damping:** A technique that slows down (reduces) the trend growth or decline over time, preventing it from continuing indefinitely at the same rate.
- **EXSM_SEASONALITY:** This parameter specifies the length of the seasonal cycle and must be a positive integer greater than 1. For example, if the value is 4, it means the seasonal pattern repeats every 4 time periods—such as the four quarters in a year. In this case, each group of four consecutive values represents one complete seasonal cycle.
- **EXSM_SETMISSING:** Specifies how to handle missing values. The special value **EXSM_MISS_AUTO** indicates that if the time series contains missing values, it should be treated as an irregular time series.

This completes the model building stage.

3.3.4 Evaluate

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

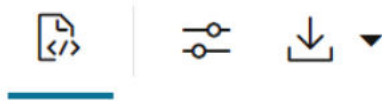
In some cases, querying dictionary views and model detail views may be sufficient to assess model performance. However, for a more thorough evaluation, you can compute test metrics such as Conditional Log-Likelihood, Average Mean Squared Error (AMSE), Akaike Information Criterion (AIC), and others.

Model Settings Information

Evaluate the model by examining the various statistics generated after model creation. These statistics provide insights into the model's quality.

```
%r
```

```
summary(MOD)
```



Call:

```
ore.odmESM(formula = AMOUNT_SOLD ~ ., data = SALES_TIME_AMOUNT,
            odm.settings = settings)
```

Settings:

	value
accumulate	accu.total
backcast.output	backcast.output.enable
confidence.level	.95
initvl.optimize	initvl.optimize.enable
interval	interval.qtr
model	winters
nmse	3
optimization.crit	opt.crit.lik
prediction.step	4
seasonality	4
setmissing	miss.auto

This script outputs a summary of a created exponential smoothing model, trained in the Oracle Database using the `SALES_TIME_AMOUNT` data. The summary confirms model type, input variables, target, forecast settings, and Oracle model metadata — all stored in the object `MOD` and available in the database as `ESM_SALES_FORECAST_1`.

Forecast

In this step, you will forecast sales for the next four quarters.

1. Forecast AMOUNT SOLD


The model `ESM_MOD` predicts four future values of `AMOUNT_SOLD`, along with lower and upper confidence bounds. The results are sorted in descending time order, displaying the most recent forecasted points first.

```
%r

modelDetails <- summary(MOD)

PRED <- modelDetails$predictions

z.show(ore.sort(PRED, by='CASE_ID', reverse=TRUE))
```



CASE_ID ↕	VALUE ↕	PREDICTION ↕	LOWER ↕	UPPER ↕
2002-10-01	NA	7821557.48371137	5462864.87683289	10180250.0905898
2002-07-01	NA	7907806.49947602	5843737.67151533	9971875.32743671
2002-04-01	NA	7151013.39786142	5591487.09771765	8710539.69800519
2002-01-01	NA	7819637.04127427	6484414.34513662	9154859.73741193
2001-10-01	7470897.51999978	7154087.56306759	NA	NA

The output is a table showing the forecasted values of `AMOUNT_SOLD` for the next 4 future `CASE_ID` periods (For example: quarters), sorted in descending order of time. This helps you view the most recent predictions at the top, providing a clear snapshot of expected future sales.

Explanation of each column in the output:

- **CASE_ID:**
A unique identifier for each input record (or row) used during prediction. Helps track which prediction corresponds to which original data row.
- **Value:**
The actual (observed) value from the dataset, typically the true target/output variable.
- **Prediction:**
The predicted value generated by the model for that record. This can be a class label (for classification) or a numeric value (for regression).
- **Lower:**
The lower bound of the confidence or prediction interval. Indicates the lowest expected value with a given level of confidence (e.g., 95%).
- **Upper:**
The upper bound of the confidence or prediction interval. Indicates the highest expected value with a given level of confidence.

2. Chart Forecasted `AMOUNT_SOLD` Values with Confidence Intervals

To visualize the forecasted values in OML Notebooks, run the same query as above with the following chart settings:

```
z.show(ESM_MOD.prediction.sort_values(by='TIME_SEQ'))
```

Settings

Setup

Customization

Height

auto

Aggregate Duplicates

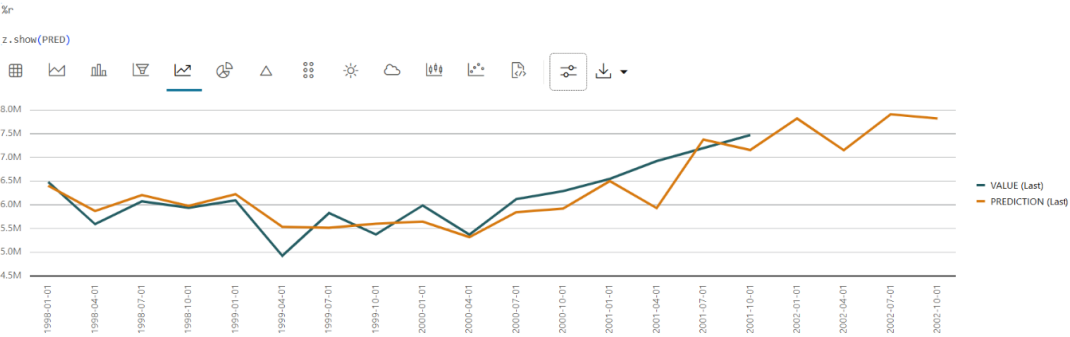
Last

Series to Show

VALUE × PREDICTION ×

Group By

CASE_ID ×



The line chart effectively illustrates how the model's predictions track the actual observed values (value) over time (time_id).

This completes the prediction step. The model has successfully forecasted sales for the next four quarters. These forecasts help track sales trends and provide valuable insights for inventory planning.

Reference

- [About Machine Learning Classes and Algorithms](#)
These classes provide access to in-database machine learning algorithms.
- [About Model Settings](#)
You can specify settings that affect the characteristics of a model.
- [Shared Settings](#)
These settings are common to all of the OML4R machine learning classes.

4.1 About Machine Learning Classes and Algorithms

These classes provide access to in-database machine learning algorithms.

Algorithm Classes

Class	Algorithm	Function of Algorithm	Description
<code>ore.odmAI</code>	Minimum Description Length	Attribute importance for classification or regression	Ranks attributes according to their importance in predicting a target.
<code>ore.odmAssocRules</code>	Apriori	Association rules	Performs market basket analysis by identifying co-occurring items (frequent itemsets) within a set.
<code>ore.odmDT</code>	Decision Tree	Classification	Extracts predictive information in the form of human-understandable rules. The rules are if-then-else expressions; they explain the decisions that lead to the prediction.
<code>ore.odmEM</code>	Expectation Maximization	Clustering	Performs probabilistic clustering based on a density estimation algorithm.
<code>ore.odmESA</code>	Explicit Semantic Analysis	Feature extraction	Extracts text-based features from a corpus of documents. Performs document similarity comparisons.
<code>ore.odmGLM</code>	Generalized Linear Model	Classification Regression	Implements logistic regression for classification of binary targets and linear regression for continuous targets.
<code>ore.odmKM</code>	<i>k</i> -Means	Clustering	Uses unsupervised learning to group data based on similarity into a predetermined number of clusters.
<code>ore.odmNB</code>	Naive Bayes	Classification	Makes predictions by deriving the probability of a prediction from the underlying evidence, as observed in the data.
<code>ore.odmNN</code>	Neural Network	Classification Regression	Learns from examples and tunes the weights of the connections among the neurons during the learning process.

Class	Algorithm	Function of Algorithm	Description
ore.odmRF	Random Forest	Classification	Provides an ensemble learning technique for classification of data.
ore.odmSVD	Singular Value Decomposition	Feature extraction	Performs orthogonal linear transformations that capture the underlying variance of the data by decomposing a rectangular matrix into three matrices.
ore.odmSVM	Support Vector Machine	Anomaly detection Classification Regression	Builds a model that is a profile of a class, which, when the model is applied, identifies cases that are somehow different from that profile.
ore.odmNMF	Non-Negative Matrix Factorization	Feature extraction	A state of the art feature extraction algorithm used when there are many attributes and the attributes are ambiguous or have weak predictability.
ore.odmXGB	XGBoost	Classification Regression	Can be used as a stand-alone predictor or incorporate it into real-world production pipelines for a wide range of problems such as ad click-through rate prediction, hazard risk prediction, web text classification, and so on.

Persisting Models

In-database models created through the OML4R API exist as temporary objects that are dropped when the database connection ends unless you take one of the following actions:

- Save a default-named model object in a datastore, as in the following example:

```
regr2 = ore.odmGLM("regression")
ore.save(regr2, name = 'regression2', overwrite=TRUE)
```

- Use the `model_name` parameter when building the model to explicitly name in-database model proxy object, as in the following example:

```
ore.drop(model='RF_CLASSIFICATION_MODEL')
settings = list(RFOR_MTRY = 3, model_name="RF_CLASSIFICATION_MODEL")
MOD2 <- ore.odmRF(AFFINITY_CARD~., DEMO_DF.train, odm.settings= settings)
MOD2$name
```

- Change the name of an existing model using the `model_name` function of the model, as in the following example:

```
regr2(model_name = 'myRegression2')
```

To drop a persistent named model, use the `oml.drop` function.

Scoring New Data with a Model

For most of the OML4R machine learning classes, you can use the `predict` method of the model object to score new data.

For in-database models, you can use the SQL `PREDICTION` function on model proxy objects, which scores directly in the database. You can use in-database models directly from SQL if you

prepare the data properly. For open source models, you can use Embedded R Execution and enable data-parallel execution for performance and scalability.

Deploying Models Through a REST API

The [REST API for Oracle Machine Learning Services](#) provides REST endpoints hosted on an Oracle Autonomous Database instance. These endpoints allow you to store OML models along with their metadata, and to create scoring endpoints for the models.

4.2 About Model Settings

You can specify settings that affect the characteristics of a model.

Some settings are general, some are specific to an Oracle Machine Learning function, and some are specific to an algorithm.

All settings have default values. If you want to override one or more of the settings for a model, then you must specify the settings with the `**params` parameter when instantiating the model or later by using the `set_params` method of the model.

If a parameter is specified by both OML4R algorithm parameters and `odm.settings`, the value in `odm.settings` is used.

Example 4-1 Specifying Model Settings

This example shows the creation of an Expectation Maximization (EM) model and the changing of a setting.

```
settings = list(  
  EMCS_NUM_ITERATIONS= 20,  
  EMCS_RANDOM_SEED= 7)  
EM.MOD <- ore.odmEM(~.-CUST_ID, CUST_DF, num.centers = 3, odm.settings =  
settings)
```

4.3 Shared Settings

These settings are common to all of the OML4R machine learning classes.

The following table lists the settings that are shared by all Oracle Machine Learning for R models.

Table 4-1 Shared Model Settings

Setting Name	Setting Value	Description
ODMS_DETAILS	ODMS_ENABLE	Helps to control model size in the database. Model details can consume significant disk space, especially for partitioned models. The default value is <code>ODMS_ENABLE</code> . If the setting value is <code>ODMS_ENABLE</code> , then model detail tables and views are created along with the model. You can query the model details using SQL. If the value is <code>ODMS_DISABLE</code> , then model detail tables are not created and tables relevant to model details are also not created. The reduction in the space depends on the algorithm. Model size reduction can be on the order of 10x .
	ODMS_DISABLE	

Table 4-1 (Cont.) Shared Model Settings

Setting Name	Setting Value	Description
ODMS_MAX_PARTITIONS	1 < value <= 1000000	Controls the maximum number of partitions allowed for a partitioned model. The default is 1000.
ODMS_MISSING_VALUE_TREATMENT	ODMS_MISSING_VALUE_AUTO ODMS_MISSING_VALUE_MEAN_MODE ODMS_MISSING_VALUE_DELETE_ROW	<p>Indicates how to treat missing values in the training data. This setting does not affect the scoring data. The default value is ODMS_MISSING_VALUE_AUTO.</p> <p>ODMS_MISSING_VALUE_MEAN_MODE replaces missing values with the mean (numeric attributes) or the mode (categorical attributes) both at build time and apply time where appropriate. ODMS_MISSING_VALUE_AUTO performs different strategies for different algorithms.</p> <p>When ODMS_MISSING_VALUE_TREATMENT is set to ODMS_MISSING_VALUE_DELETE_ROW, the rows in the training data that contain missing values are deleted. However, if you want to replicate this missing value treatment in the scoring data, then you must perform the transformation explicitly.</p> <p>The value ODMS_MISSING_VALUE_DELETE_ROW is applicable to all algorithms.</p>
ODMS_PARTITION_BUILD_TYPE	ODMS_PARTITION_BUILD_INTRA ODMS_PARTITION_BUILD_INTER ODMS_PARTITION_BUILD_HYBRID	<p>Controls the parallel building of partitioned models.</p> <p>ODMS_PARTITION_BUILD_INTRA builds each partition in parallel using all slaves.</p> <p>ODMS_PARTITION_BUILD_INTER builds each partition entirely in a single slave, but multiple partitions may be built at the same time because multiple slaves are active.</p> <p>ODMS_PARTITION_BUILD_HYBRID combines the other two types and is recommended for most situations to adapt to dynamic environments. This is the default value.</p>
ODMS_PARTITION_COLUMNS	Comma separated list of machine learning attributes	Requests the building of a partitioned model. The setting value is a comma-separated list of the machine learning attributes to be used to determine the in-list partition key values. These attributes are taken from the input columns, unless an XFORM_LIST parameter is passed to the model. If XFORM_LIST parameter is passed to the model, then the attributes are taken from the attributes produced by these transformations.
ODMS_TABLESPACE_NAME	<i>tablespace_name</i>	<p>Specifies the tablespace in which to store the model.</p> <p>If you explicitly set this to the name of a tablespace (for which you have sufficient quota), then the specified tablespace storage creates the resulting model content. If you do not provide this setting, then the your default tablespace creates the resulting model content.</p>
ODMS_SAMPLE_SIZE	0 < value	Determines how many rows to sample (approximately). You can use this setting only if ODMS_SAMPLING is enabled. The default value is system determined.
ODMS_SAMPLING	ODMS_SAMPLING_ENABLE ODMS_SAMPLING_DISABLE	Allows the user to request sampling of the build data. The default is ODMS_SAMPLING_DISABLE.
ODMS_TEXT_MAX_FEATURES	1 <= value	The maximum number of distinct features, across all text attributes, to use from a document set passed to the model. The default is 3000. An oml.esa model has the default value of 300000.

Table 4-1 (Cont.) Shared Model Settings

Setting Name	Setting Value	Description
ODMS_TEXT_MIN_DOCUMENTS	Non-negative value	This text processing setting controls how many documents a token needs to appear in to be used as a feature. The default is 1. An <code>oml.esa</code> model has the default value of 3.
ODMS_TEXT_POLICY_NAME	The name of an Oracle Text POLICY created using <code>CTX_DDL.CREATE_POLICY</code> .	Affects how individual tokens are extracted from unstructured text. For details about <code>CTX_DDL.CREATE_POLICY</code> , see <i>Oracle Text Reference</i> .
PREP_AUTO	PREP_AUTO_ON PREP_AUTO_OFF	This data preparation setting enables fully automated data preparation. The default is <code>PREP_AUTO_ON</code> .
PREP_SCALE_2DNUM	pPREP_SCALE_STDDEV PREP_SCALE_RANGE	This data preparation setting enables scaling data preparation for two-dimensional numeric columns. <code>PREP_AUTO</code> must be <code>OFF</code> for this setting to take effect. The following are the possible values: PREP_SCALE_STDDEV: A request to divide the column values by the standard deviation of the column and is often provided together with <code>PREP_SHIFT_MEAN</code> to yield z-score normalization. PREP_SCALE_RANGE: A request to divide the column values by the range of values and is often provided together with <code>PREP_SHIFT_MIN</code> to yield a range of [0,1].
PREP_SCALE_NNUM	PREP_SCALE_MAXABS	This data preparation setting enables scaling data preparation for nested numeric columns. <code>PREP_AUTO</code> must be <code>OFF</code> for this setting to take effect. If specified, then the valid value for this setting is <code>PREP_SCALE_MAXABS</code> , which yields data in the range of [-1,1].
PREP_SHIFT_2DNUM	PREP_SHIFT_MEAN PREP_SHIFT_MIN	This data preparation setting enables centering data preparation for two-dimensional numeric columns. <code>PREP_AUTO</code> must be <code>OFF</code> for this setting to take effect. The following are the possible values: PREP_SHIFT_MEAN: Results in subtracting the average of the column from each value. PREP_SHIFT_MIN: Results in subtracting the minimum of the column from each value.
ODMS_BOXCOX	ODMS_BOXCOX_ENABLE ODMS_BOXCOX_DISABLE	This setting enables the Box-Cox variance-stabilization transformation. It is useful when the variance increases as the target value increases. It reduces variance and transforms a multiplicative relationship with the target, with a simpler additive relationship. This setting is applicable only to the Exponential Smoothing algorithm. When a value for <code>EXSM_MODEL</code> setting is not specified, the default value is <code>ODMS_BOXCOX_ENABLE</code> and when a value for the <code>EXSM_MODEL</code> setting is provided, the default value is <code>ODMS_BOXCOX_DISABLE</code> .

Table 4-1 (Cont.) Shared Model Settings

Setting Name	Setting Value	Description
ODMS_EXPLOSION_MIN_SUPP	X >= 0	It is the minimum required support for categorical values that must be included in the explosion mapping. It removes categorical values with insufficient row instances to have a statistically significant effect on the model, because, they could potentially degrade performance or exhaust memory. The default is system determined depending on the number of rows in the dataset. A value of 1 results into mapping all categorical values.

Glossary

Index

A

algorithms

- machine learning, [1](#)
- settings common to all, [3](#)

C

classes

- machine learning, [1](#)

clustering

- use case, [11](#), [12](#), [14](#), [18](#)

K

- k-means algorithm, [11](#), [12](#), [14](#), [18](#)

M

machine learning

- classes, [1](#)

models

- persisting, [1](#)

S

- scoring new data, [1](#)

settings

- about model, [3](#)
- shared algorithm, [3](#)