Oracle® Machine Learning for SQL

Concepts

21c
F31930-05
June 2021

Oracle Machine Learning for SQL Concepts, 21c

F31930-05

Primary Author: Sarika Surampudi

Contributors: David McDermid, Boriana Milanova

Contents

4 Regression

5 Classification

6 Anomaly Detection

18 Explicit Semantic Analysis

19 Exponential Smoothing

20 Generalized Linear Model

27 O-Cluster

28 R Extensibility

29 Random Forest

30 Singular Value Decomposition

31 Support Vector Machine

Preface

This manual describes Oracle Machine Learning for SQL (OML4SQL), a comprehensive, state-of-the-art machine learning capability within Oracle Database, previously known as Oracle Data Mining. This manual presents the concepts that underlie the procedural information that is presented in *Oracle Machine Learning for SQL User's Guide*.

The preface contains these topics:

Technology Rebrand

Oracle is rebranding the suite of products and components that support machine learning with Oracle Database and Big Data. This technology is now known as Oracle Machine Learning (OML).

The OML application programming interfaces (APIs) for SQL include PL/SQL packages, SQL functions, and data dictionary views. Using these APIs is described in publications, previously under the name Oracle Data Mining, that are now named Oracle Machine Learning for SQL (OML4SQL).

Audience

Oracle Machine Learning for SQL Concepts is intended for anyone who wants to learn about Oracle Machine Learning for SQL.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/

lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

Related documentation for Oracle Machine Learning for SQL (OML4SQL) includes publications and web pages.

The following publications document OML4SQL:

- *Oracle Machine Learning for SQL Concepts* (this publication)
- *Oracle Machine Learning for SQL User's Guide*
- *Oracle Machine Learning for SQL API Guide*

> **Note:**
>
> *Oracle Machine Learning for SQL API Guide* combines key passages from *Oracle Machine Learning for SQL Concepts* and *Oracle Machine Learning for SQL User's Guide* with related reference documentation from *Oracle Database PL/SQL Packages and Types Reference*, *Oracle Database Reference*, and *Oracle Database SQL Language Reference*.

- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*

For other information and resources about OML4SQL, see the Oracle Machine Learning for SQL web page.

Oracle Machine Learning for SQL Resources on the Oracle Technology Network

The Oracle Machine Learning for SQL page on the Oracle Technology Network (OTN) provides a wealth of information, including white papers, demonstrations, blogs, discussion forums, and Oracle By Example tutorials.

You can download Oracle Data Miner, the graphical user interface to Oracle Machine Learning for SQL, from this site:

`Oracle Data Miner`

Application Development and Database Administration Documentation

Refer to the documentation to assist you in developing database applications and in administering Oracle Database.

- *Oracle Database Concepts*
- *Oracle Database Administrator's Guide*
- *Oracle Database Development Guide*
- *Oracle Database Performance Tuning Guide*
- *Oracle Database VLDB and Partitioning Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Machine Learning for SQL Concepts

Changes in this release for *Oracle Machine Learning for SQL Concepts*.

Changes in Oracle Machine Learning for SQL 21c

Changes in *Oracle Machine Learning for SQL Concepts* for Oracle Database 21c.

New Features in 21c

Oracle Machine Learning for SQL features new in Oracle Database 21c.

New Algorithms

- MSET-SPRT

  The Multivariate State Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT) algorithm is a nonlinear, nonparametric anomaly detection machine learning technique designed for monitoring critical processes. It detects subtle anomalies while also producing minimal false alarms.

  The algorithm calibrates an expected behavior from available, historical data from the normal operational sequence of monitored signals. The learned behavior of the system is then incorporated into a persistent Oracle Machine Learning for SQL MSET-SPRT model that captures expected normal behavior and can be applied to new records to detect anomalous behaviors.

- XGBoost

  XGBoost is machine learning algorithm for regression and classification that makes available the XGBoost open source package. Oracle Machine Learning for SQL XGBoost prepares training data, invokes XGBoost, builds and persists a model, and applies the model for prediction.

New Algorithm Setting

Adam Optimization Solver

Adam is an extension to stochastic gradient descent that uses mini-batch optimization. The Adam solver can make progress faster by seeing less data than the L-BFGS solver. Adam is computationally efficient, with little memory requirements, and is well-suited for problems that are large in terms of data or parameters or both.

Enhancements

Neural Network Algorithm Settings

The Neural Network algorithm setting `NNET_ACTIVATIONS` now accepts the value `NNET_ACTIVATIONS_RELU`. Rectified Linear Units is a commonly used activation function for deep learning models that addresses the vanishing gradient problem in large neural networks.

The algorithm has a new setting, `NNET_SOLVER`, that specifies the method of optimization, either L-BFGS or Adam.

For the `NNET_NODES_PER_LAYER` and `NNET_ACTIVATIONS` settings, you can now specify a single value that is then applied to each hidden layer.

The `NNET_ITERATIONS` setting has a default value for the LBFGS solver and now has a default value for the Adam solver. The default values are different for each solver.

Other Changes

The following are additional changes in *Oracle Machine Learning for SQL Concepts* for 21*c*:

- Added "Anomaly Detection for Time Series Data" topic in the Anomaly Detection chapter.
- Updated machine learning process illustration and added process descriptions. See The Oracle Machine Learning Process.
- Added machine learning functions overview. See Machine Learning Functions.

Part I
Introductions

Introduces Oracle Machine Learning for SQL.

Provides a high-level overview for those who are new to OML4SQL technology.

- What Is Machine Learning?
- Introduction to Oracle Machine Learning for SQL
- Oracle Machine Learning Basics

1

What Is Machine Learning?

Orientation to machine learning technology.

- What Is Machine Learning?
- What Can Machine Learning Do and Not Do?
- The Oracle Machine Learning Process

> **Note:**
>
> Information about machine learning is widely available. No matter what your level of expertise, you can find helpful books and articles on machine learning.

Related Topics

- https://en.wikipedia.org/wiki/Machine_learning

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.

1.1.1 Automatic Discovery

Machine learning is performed by a model that uses an algorithm to act on a set of data.

Machine learning models can be used to mine the data on which they are built, but most types of models are generalizable to new data. The process of applying a model to new data is known as **scoring**.

1.1.2 Prediction

Many forms of machine learning are predictive. For example, a model can predict income 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?).

1.1.3 Grouping

Other forms of machine learning identify natural 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 Actionable Information

Machine learning can derive actionable information from large volumes of data. For example, a town planner might use a model that predicts income based on demographics to develop a plan for low-income housing. A car leasing agency might use a model that identifies customer segments to design a promotion targeting high-value customers.

Machine Learning and Statistics

There is a great deal of overlap between machine learning and statistics. In fact most of the techniques used in machine learning can be placed in a statistical framework. However, machine learning techniques are not the same as traditional statistical techniques.

Statistical models usually make strong assumptions about the data and, based on those assumptions, they make strong statements about the results. However, if the assumptions are flawed, the validity of the model becomes questionable. By contrast, the machine learning methods typically make weak assumptions about the data. As a result, machine learning cannot generally make such strong statements about the results. Yet machine learning can produce very good results regardless of the data.

Traditional statistical methods, in general, require a great deal of user interaction in order to validate the correctness of a model. As a result, statistical methods can be difficult to automate. Statistical methods rely on testing hypotheses or finding correlations based on smaller, representative samples of a larger population.

Less user interaction and less knowledge of the data is required for machine learning. The user does not need to massage the data to guarantee that a method is valid for a given data set. Oracle Machine Learning techniques are easier to automate than traditional statistical techniques.

Oracle Machine Learning and OLAP

On-Line Analytical Processing (OLAP) can be defined as fast analysis of multidimensional data. OLAP and Oracle Machine Learning are different but complementary activities.

OLAP supports activities such as data summarization, cost allocation, time series analysis, and what-if analysis. However, most OLAP systems do not have inductive inference capabilities beyond the support for time-series forecast. Inductive inference, the process of reaching a general conclusion from specific examples, is a characteristic of machine learning. Inductive inference is also known as computational learning.

OLAP systems provide a multidimensional view of the data, including full support for hierarchies. This view of the data is a natural way to analyze businesses and organizations.

Oracle Machine Learning and OLAP can be integrated in a number of ways. OLAP can be used to analyze machine learning results at different levels of granularity. Machine learning can help you construct more interesting and useful cubes. For example, the results of predictive machine learning can be added as custom measures to a cube. Such measures can provide information such as "likely to default" or "likely to buy" for each customer. OLAP processing can then aggregate and summarize the probabilities.

Oracle Machine Learning and Data Warehousing

Data can be mined whether it is stored in flat files, spreadsheets, database tables, or some other storage format. The important criteria for the data is not the storage format, but its applicability to the problem to be solved.

Proper data cleansing and preparation are very important for machine learning, and a data warehouse can facilitate these activities. However, a data warehouse is of no use if it does not contain the data you need to solve your problem.

1.2 What Can Machine Learning Do and Not Do?

Machine learning is a powerful tool that can help you find patterns and relationships within your data. But machine learning does not work by itself. It does not eliminate the need to know your business, to understand your data, or to understand analytical methods. Machine learning discovers hidden information in your data, but it cannot tell you the value of the information to your organization.

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 discernible through simple observation.

It is important to remember that the predictive relationships discovered through machine learning 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. However, you must not assume that the population identified through machine learning buys the product *because* they belong to this population.

Machine learning yields probabilities, not exact answers. It is important to keep in mind that rare events *can* happen; they do not happen very often.

1.2.1 Asking the Right Questions

Machine learning does not automatically discover information without guidance. The patterns you find through machine learning are very different depending on how you formulate the problem.

To obtain meaningful results, you must learn how to ask the right questions. For example, rather than trying to learn how to "improve the response to a direct mail solicitation," you might try to find the characteristics of people who have responded to your solicitations in the past.

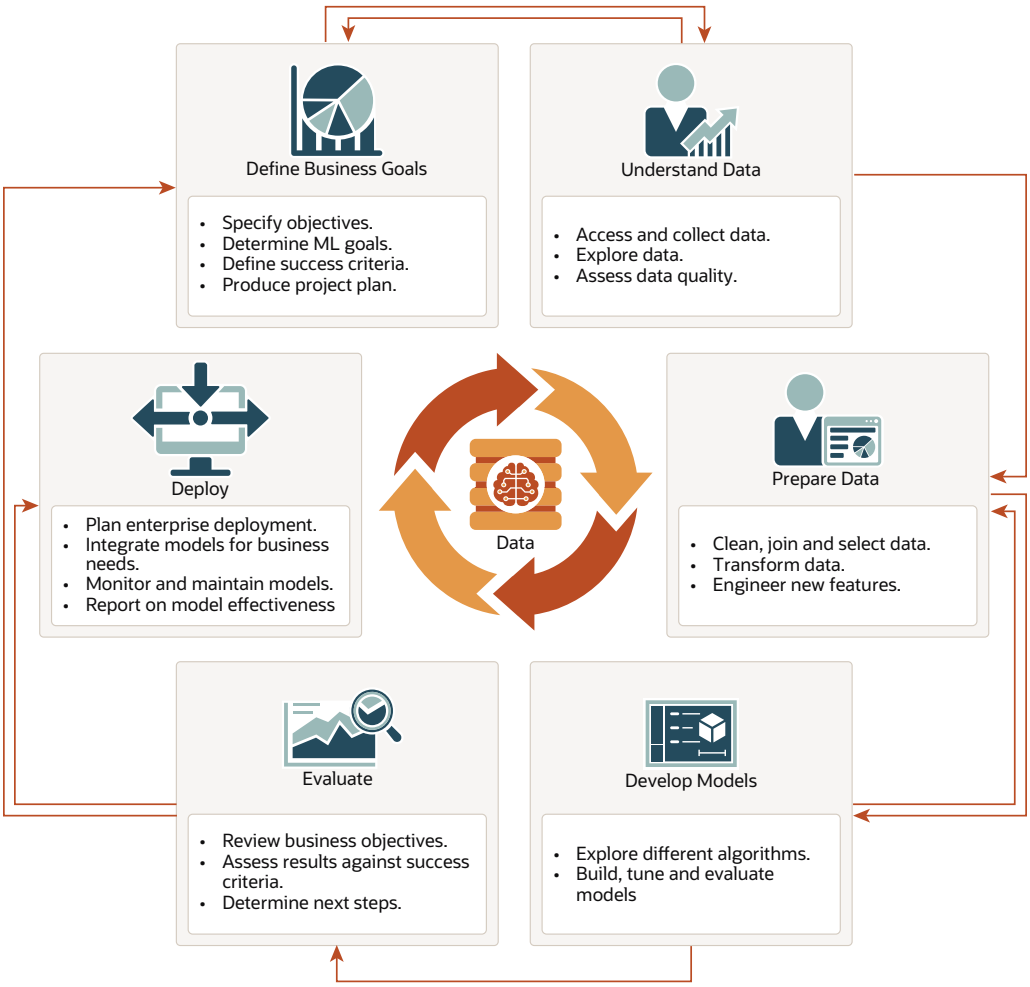
1.2.2 Understanding Your Data

To ensure meaningful machine learning results, you must understand your data. Machine learning algorithms are often sensitive to specific characteristics of the data: outliers (data values that are very different from the typical values in your database), irrelevant columns, columns that vary together (such as age and date of birth), data coding, and data that you choose to include or exclude. Oracle Machine Learning can automatically perform much of the data preparation required by the algorithm. But some of the data preparation is typically specific to the domain or the machine learning problem. At any rate, you need to understand the data that was used to build the model to properly interpret the results when the model is applied.

1.3 The Oracle Machine Learning Process

The following figure illustrates the phases, and the iterative nature, of a machine learning project. The process flow shows that a machine learning project does not stop when a particular solution is deployed. The results trigger new business questions, which in turn can be used to develop more focused models.

Figure 1-1 The Oracle Machine Learning Process



1.3.1 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

1.3.2 Understand Data

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

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

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

To summarize, in this phase, you will:

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

1.3.3 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. column to ; you can insert the median income in cases where the column is null.

Additionally you can add new computed attributes in an effort to tease information closer to the surface of the data referred to 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*

1.3.4 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.

Automatic 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*

1.3.5 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

1.3.6 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*

2

Introduction to Oracle Machine Learning for SQL

Introduces Oracle Machine Learning for SQL to perform a variety of machine learning tasks.

2.1 About Oracle Machine Learning for SQL

Understand the uses of Oracle Machine Learning for SQL and learn about different machine learning techniques.

OML4SQL provides a powerful, state-of-the-art machine learning capability within Oracle Database. You can use OML4SQL to build and deploy predictive and descriptive machine learning applications, to add intelligent capabilities to existing applications, and to generate predictive queries for data exploration.

OML4SQL offers a comprehensive set of in-database algorithms for performing a variety of machine learning tasks, such as classification, regression, anomaly detection, feature extraction, clustering, and market basket analysis. The algorithms can work on standard case data, transactional data, star schemas, and text and other forms of unstructured data. OML4SQL is uniquely suited to the analysis of very large data sets.

Oracle Machine Learning for SQL is a component of the Oracle Database Enterprise Edition. Another component is Oracle Machine Learning for R, which integrates R, the open-source statistical environment, with Oracle Database. Together, OML4SQL and Oracle Machine Learning for R provide a comprehensive advanced analytics platform for big data analytics.

2.2 Oracle Machine Learning for SQL in the Database Kernel

Learn about the implementation of Oracle Machine Learning for SQL (OML4SQL) in Oracle Database kernel and its advantages.

OML4SQL is implemented in the Oracle Database kernel. OML4SQL models are first class database objects. Oracle Machine Learning for SQL processes use built-in features of Oracle Database to maximize scalability and make efficient use of system resources.

OML4SQL within Oracle Database offers many advantages:

- No Data Movement: Some machine learning products require that the data be exported from a corporate database and converted to a specialized format. With OML4SQL, no data movement or conversion is needed. This makes the entire process less complex, time-consuming, and error-prone, and it allows for the analysis of very large data sets.
- Security: Your data is protected by the extensive security mechanisms of Oracle Database. Moreover, specific database privileges are needed for different machine learning activities. Only users with the appropriate privileges can define, manipulate, or apply machine learning model objects.
- Data Preparation and Administration: Most data must be cleansed, filtered, normalized, sampled, and transformed in various ways before it can be mined. Up to 80% of the effort in a machine learning project is often devoted to data preparation. OML4SQL can automatically manage key steps in the data preparation process. Additionally, Oracle Database provides extensive administrative tools for preparing and managing data.
- Ease of Data Refresh: Machine learning processes within Oracle Database have ready access to refreshed data. OML4SQL can easily deliver machine learning results based on current data, thereby maximizing its timeliness and relevance.
- Oracle Database Analytics: Oracle Database offers many features for advanced analytics and business intelligence. You can easily integrate machine learning with other analytical features of the database, such as statistical analysis and OLAP.
- Oracle Technology Stack: You can take advantage of all aspects of Oracle's technology stack to integrate machine learning within a larger framework for business intelligence or scientific inquiry.
- Domain Environment: Machine learning models have to be built, tested, validated, managed, and deployed in their appropriate application domain environments. Machine learning results may need to be post-processed as part of domain specific computations (for example, calculating estimated risks and response probabilities) and then stored into permanent repositories or data warehouses. With OML4SQL, the pre- and post-machine learning activities can all be accomplished within the same environment.
- Application Programming Interfaces: The PL/SQL API and SQL language operators provide direct access to OML4SQL functionality in Oracle Database.

Related Topics

- Overview of Database Analytics
  An overview of native analytics supported by Oracle Database.

2.3 Oracle Machine Learning for SQL in Oracle Exadata

Understand how complex scoring and algorithmic processing is done using Oracle Exadata.

Scoring refers to the process of applying a OML4SQL model to data to generate predictions. The scoring process may require significant system resources. Vast amounts of data may be involved, and algorithmic processing may be very complex.

With OML4SQL, scoring can be off-loaded to intelligent Oracle Exadata Storage Servers where processing is extremely performant.

Oracle Exadata Storage Servers combine Oracle's smart storage software and Oracle's industry-standard Sun hardware to deliver the industry's highest database storage performance. For more information about Oracle Exadata, visit the Oracle Technology Network.

Related Topics

- http://www.oracle.com/us/products/database/exadata/index.htm

2.4 About Partitioned Models

Introduces partitioned models to organize and represent multiple models.

When you build a model on your data set and apply it to new data, sometimes the prediction may be generic that performs badly when run on new and evolving data. To overcome this, the data set can be divided into different parts based on some characteristics. Oracle Machine Learning for SQL supports partitioned model. Partitioned models allow users to build a type of ensemble model for each data partition. The top-level model has sub models that are automatically produced. The sub models are based on the attribute options. For example, if your data set has an attribute called MARITAL with four values and you have defined it as the partitioned attribute. Then, four sub models are created for this attribute. The sub models are automatically managed and used as a single model. The partitioned model automates a typical machine learning task and can potentially achieve better accuracy through multiple targeted models.

The partitioned model and its sub models reside as first class, persistent database objects. Persistent means that the partitioned model has an on-disk representation.

To create a partitioned model, include the `ODMS_PARTITION_COLUMNS` setting. To define the number of partitions, include the `ODMS_MAX_PARTITIONS` setting. When you are making predictions, you must use the top-level model. The correct sub model is selected automatically based on the attribute, the attribute options, and the partition setting. You must include the partition columns as part of the `USING` clause when scoring. The `GROUPING` hint is an optional hint that applies to machine learning scoring functions when scoring partitioned models.

The partition names, key values, and the structure of the partitioned model are available in the `ALL_MINING_MODEL_PARTITIONS` view.

Related Topics

- *Oracle Database Reference*

> **See Also:**
>
> *Oracle Database SQL Language Reference* on how to use `GROUPING` hint.
> *Oracle Machine Learning for SQL User's Guide* to understand more about partitioned models.

2.5 Interfaces to Oracle Machine Learning for SQL

Introduces supported interfaces for Oracle Machine Learning for SQL.

The programmatic interfaces to Oracle Machine Learning for SQL are PL/SQL for building and maintaining models and a family of SQL functions for scoring. OML4SQL also supports a graphical user interface, which is implemented as an extension to Oracle SQL Developer.

Oracle Predictive Analytics, a set of simplified OML4SQL routines, is built on top of OML4SQL and is implemented as a PL/SQL package.

2.5.1 PL/SQL API

Includes PL/SQL package for Oracle Machine Learning for SQL.

The OML4SQL PL/SQL API is implemented in the `DBMS_DATA_MINING` PL/SQL package, which contains routines for building, testing, and maintaining machine learning models. A batch apply operation is also included in this package.

The following example shows part of a simple PL/SQL script for creating an SVM classification model called SVMC_SH_Clas_sample. The model build uses weights, specified in a weights table, and settings, specified in a settings table. The weights influence the weighting of target classes. The settings override default behavior. The model uses Automatic Data Preparation (`prep_auto_on` setting). The model is trained on the data in mining_data_build_v.

Example 2-1 Creating a Classification Model

```
----------------------  CREATE AND POPULATE A CLASS WEIGHTS TABLE  ------------
CREATE TABLE svmc_sh_sample_class_wt (
  target_value NUMBER,
  class_weight NUMBER);
INSERT INTO svmc_sh_sample_class_wt VALUES (0,0.35);
INSERT INTO svmc_sh_sample_class_wt VALUES (1,0.65);
COMMIT;
----------------------  CREATE AND POPULATE A SETTINGS TABLE ------------------
CREATE TABLE svmc_sh_sample_settings (
  setting_name  VARCHAR2(30),
  setting_value VARCHAR2(4000));
BEGIN
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.svms_kernel_function, dbms_data_mining.svms_linear);
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.clas_weights_table_name, 'svmc_sh_sample_class_wt');
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.prep_auto, dbms_data_mining.prep_auto_on);
```

```
END;
/
-----------------------  CREATE THE MODEL ------------------------------------
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'SVMC_SH_Clas_sample',
    mining_function     => dbms_data_mining.classification,
    data_table_name     => 'mining_data_build_v',
    case_id_column_name => 'cust_id',
    target_column_name  => 'affinity_card',
    settings_table_name => 'svmc_sh_sample_settings');
END;
/
```

2.5.2 SQL Functions

Oracle Machine Learning for SQL supports SQL functions for performing prediction, clustering, and feature extraction.

The functions score data by applying an OML4SQL model object or by running an analytic clause that performs dynamic scoring.

The following example shows a query that applies the classification model `svmc_sh_clas_sample` to the data in the view `mining_data_apply_v`. The query returns the average age of customers who are likely to use an affinity card. The results are broken out by gender.

Example 2-2 The PREDICTION Function

```
SELECT cust_gender,
       COUNT(*) AS cnt,
       ROUND(AVG(age)) AS avg_age
  FROM mining_data_apply_v
 WHERE PREDICTION(svmc_sh_clas_sample USING *) = 1
GROUP BY cust_gender
ORDER BY cust_gender;

C        CNT    AVG_AGE
- ---------- ----------
F         59         41
M        409         45
```

Related Topics

- In-Database Scoring
  Scoring is the application of a machine learning algorithm to new data. In Oracle Machine Learning for SQL scoring engine and the data both reside within the database.

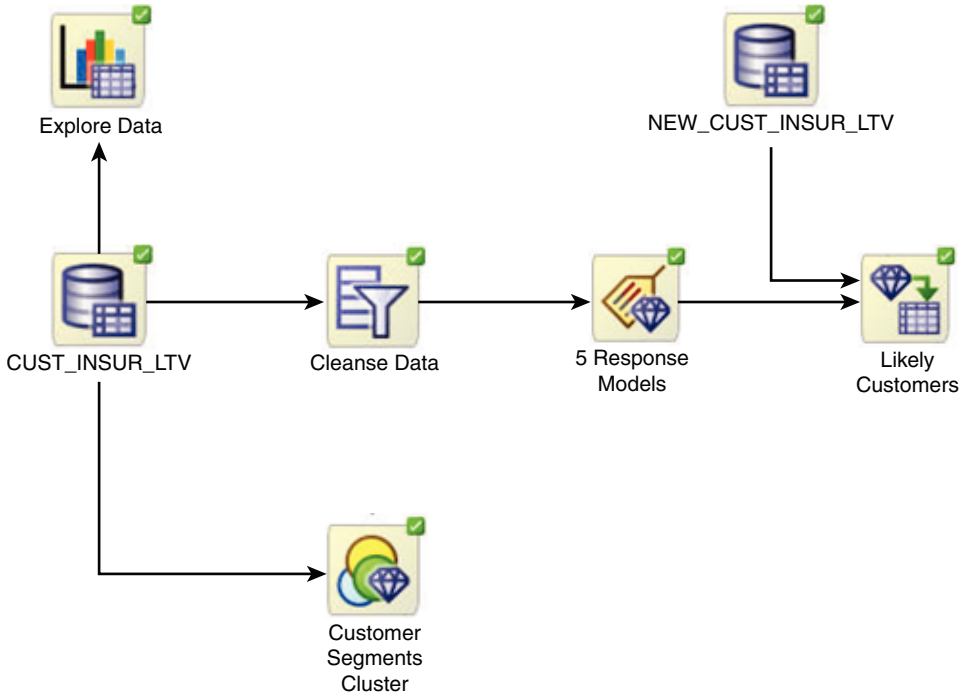
2.5.3 Oracle Data Miner

Oracle Machine Learning for SQL supports a graphical interface called Oracle Data Miner.

Oracle Data Miner is a graphical interface to OML4SQL. Oracle Data Miner is an extension to Oracle SQL Developer, which is available for download free of charge on the Oracle Technology Network.

Oracle Data Miner uses a work flow paradigm to capture, document, and automate the process of building, evaluating, and applying OML4SQL models. Within a work flow, you can specify data transformations, build and evaluate multiple models, and score multiple data sets. You can then save work flows and share them with other users.

Figure 2-1 An Oracle Data Miner Workflow



For information about Oracle Data Miner, including installation instructions, visit Oracle Technology Network.

Related Topics

- `Oracle Data Miner`

2.5.4 Predictive Analytics

Predictive analytics is a technology that captures Oracle Machine Learning for SQL processes in simple routines.

Sometimes called "one-click machine learning," predictive analytics simplifies and automates the machine learning process.

Predictive analytics uses OML4SQL technology, but knowledge of OML4SQL is not needed to use predictive analytics. You can use predictive analytics by specifying an operation to perform on your data. You do not need to create or use OML4SQL models or understand the OML4SQL functions and algorithms summarized in "Oracle Machine Learning for SQL Basics ".

Oracle Machine Learning for SQL predictive analytics operations are described in the following table:

Table 2-1 Oracle Predictive Analytics Operations

Operation	Description
`EXPLAIN`	Explains how individual predictors (columns) affect the variation of values in a target column
`PREDICT`	For each case (row), predicts the values in a target column
`PROFILE`	Creates a set of rules for cases (rows) that imply the same target value

The Oracle predictive analytics operations are implemented in the `DBMS_PREDICTIVE_ANALYTICS` PL/SQL package. They are also available in Oracle Data Miner.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

2.6 Overview of Database Analytics

An overview of native analytics supported by Oracle Database.

Oracle Database supports an array of native analytical features. Since all these features are part of a common server it is possible to combine them efficiently. The results of analytical processing can be integrated with Oracle Business Intelligence Suite Enterprise Edition and other BI tools and applications.

The possibilities for combining different analytics are virtually limitless. Example 2-3 shows Oracle Machine Learning for SQL and text processing within a single SQL query. The query selects all customers who have a high propensity to attrite (> 80% chance), are valuable customers (customer value rating > 90), and have had a recent conversation with customer services regarding a Checking Plus account. The propensity to attrite information is computed using a OML4SQL model called `tree_model`. The query uses the Oracle Text `CONTAINS` operator to search call center notes for references to Checking Plus accounts.

Some of the native analytics supported by Oracle Database are described in the following table:

Table 2-2 Oracle Database Native Analytics

Analytical Feature	Description	Documented In...
Complex data transformations	Data transformation is a key aspect of analytical applications and ETL (extract, transform, and load). You can use SQL expressions to implement data transformations, or you can use the `DBMS_DATA_MINING_TRANSFORM` package. `DBMS_DATA_MINING_TRANSFORM` is a flexible data transformation package that includes a variety of missing value and outlier treatments, as well as binning and normalization capabilities.	*Oracle Database PL/SQL Packages and Types Reference*

Table 2-2 (Cont.) Oracle Database Native Analytics

Analytical Feature	Description	Documented In...
Statistical functions	Oracle Database provides a long list of SQL statistical functions with support for: hypothesis testing (such as t-test, F-test), correlation computation (such as pearson correlation), cross-tab statistics, and descriptive statistics (such as median and mode). The `DBMS_STAT_FUNCS` package adds distribution fitting procedures and a summary procedure that returns descriptive statistics for a column.	*Oracle Database SQL Language Reference* and *Oracle Database PL/SQL Packages and Types Reference*
Window and analytic SQL functions	Oracle Database supports analytic and windowing functions for computing cumulative, moving, and centered aggregates. With windowing aggregate functions, you can calculate moving and cumulative versions of `SUM`, `AVERAGE`, `COUNT`, `MAX`, `MIN`, and many more functions.	*Oracle Database Data Warehousing Guide*
Linear algebra	The `UTL_NLA` package exposes a subset of the popular `BLAS` and `LAPACK` (Version 3.0) libraries for operations on vectors and matrices represented as VARRAYs. This package includes procedures to solve systems of linear equations, invert matrices, and compute eigenvalues and eigenvectors.	*Oracle Database PL/SQL Packages and Types Reference*
OLAP	Oracle OLAP supports multidimensional analysis and can be used to improve performance of multidimensional queries. Oracle OLAP provides functionality previously found only in specialized OLAP databases. Moving beyond drill-downs and roll-ups, Oracle OLAP also supports time-series analysis, modeling, and forecasting.	*Oracle OLAP User's Guide*
Spatial analytics	Oracle Spatial provides advanced spatial features to support high-end GIS and LBS solutions. Oracle Spatial's analysis and machine learning capabilities include functions for binning, detection of regional patterns, spatial correlation, colocation machine learning, and spatial clustering. Oracle Spatial also includes support for topology and network data models and analytics. The topology data model of Oracle Spatial allows one to work with data about nodes, edges, and faces in a topology. It includes network analysis functions for computing shortest path, minimum cost spanning tree, nearest-neighbors analysis, traveling salesman problem, among others.	*Oracle Spatial Developer's Guide*
Graph	The Property Graph delivers advanced graph query and analytics capabilities in Oracle Database. The in-memory graph server (PGX) provides a machine learning library, which supports graph-empowered machine learning algorithms. The machine learning library supports DeepWalk, supervised GraphWise, and Pg2vec algorithms.	*Oracle Database Graph Developer's Guide for Property Graph*

Table 2-2 (Cont.) Oracle Database Native Analytics

Analytical Feature	Description	Documented In...
Text Analysis	Oracle Text uses standard SQL to index, search, and analyze text and documents stored in the Oracle database, in files, and on the web. Oracle Text also supports automatic classification and clustering of document collections. Many of the analytical features of Oracle Text are layered on top of Oracle Machine Learning functionality.	*Oracle Text Application Developer's Guide*

Example 2-3 SQL Query Combining Oracle Machine Learning for SQL and Oracle Text

```
SELECT A.cust_name, A.contact_info
  FROM customers A
 WHERE PREDICTION_PROBABILITY(tree_model,
           'attrite' USING A.*) > 0.8
   AND A.cust_value > 90
   AND A.cust_id IN
       (SELECT B.cust_id
          FROM call_center B
         WHERE B.call_date BETWEEN '01-Jan-2005'
                               AND '30-Jun-2005'
         AND CONTAINS(B.notes, 'Checking Plus', 1) > 0);
```

3

Oracle Machine Learning Basics

Understand the basic concepts of Oracle Machine Learning.

3.1 Machine Learning Functions

Each machine learning **function** specifies a class of problems that can be modeled and solved.

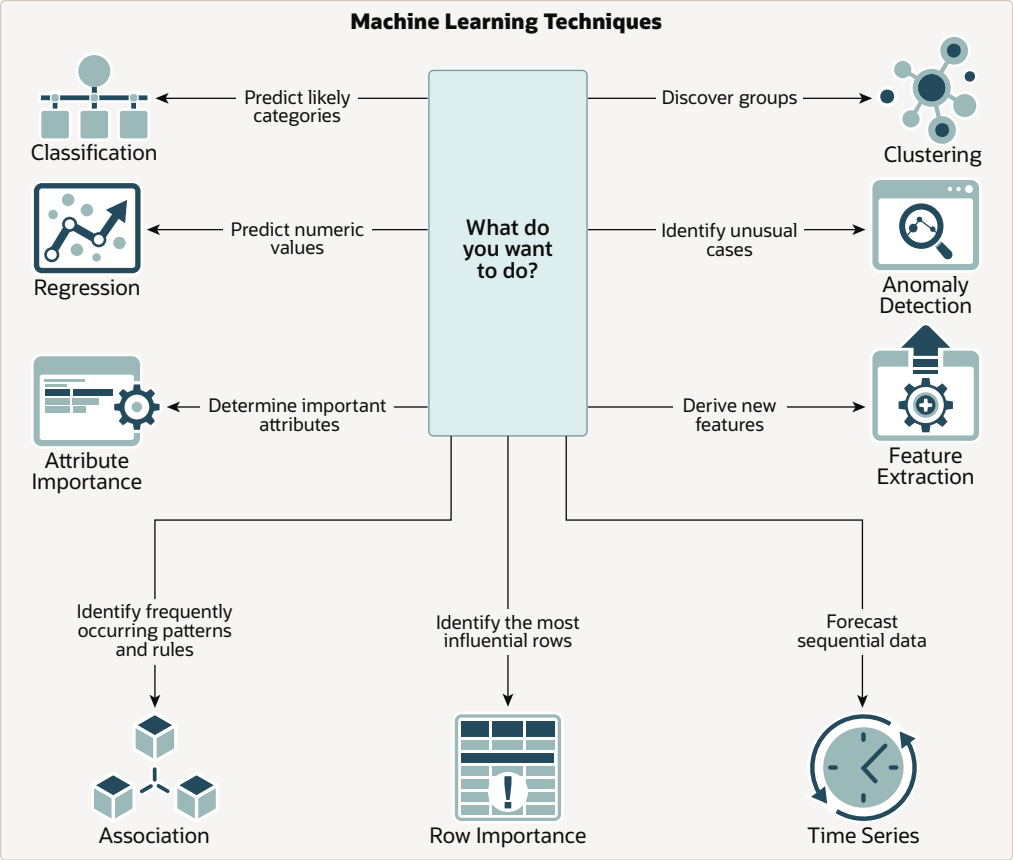
A basic understanding of machine learning functions and algorithms is required for using Oracle Machine Learning.

Machine learning functions fall generally into two categories: **supervised** and **unsupervised**. Notions of supervised and unsupervised learning are derived from the science of machine learning, which has been called a sub-area of artificial intelligence.

Artificial intelligence refers to the implementation and study of systems that exhibit autonomous intelligence or behavior of their own. Machine learning deals with techniques that enable devices to learn from their own performance and modify their own functioning.

The following illustration provides an idea of how to use Oracle machine learning functions.

Figure 3-1 How to Use Machine Learning Functions



Related Topics

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

3.1.1 Supervised Machine Learning

Overview of supervised machine learning.

Supervised learning is also known as directed learning. The learning process is directed by a previously known dependent attribute or target. Directed Oracle 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. This is in contrast to unsupervised learning where the goal is pattern detection.

3.1.1.1 Supervised Learning: Training

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 logic for making the prediction. 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. Separate data sets are required for building (training) and testing some predictive models. The build data (training data) and test data must have the same column structure. Typically, one large table or view is split into two data sets: one for building the model, and the other for testing the model.

3.1.1.2 Supervised Learning: Testing

The process of applying the model to test data helps to determine whether the model, built on one chosen sample, is generalizable to other data. In other words, test data is used for scoring.

In particular, it helps to avoid the phenomenon of overfitting, which can occur when the logic of the model fits the build data too well and therefore has little predictive power.

3.1.1.3 Supervised Learning: Scoring

Learn about scoring in supervised learning.

Apply data, also called scoring data, is the actual population to which a model is applied. For example, you might build a model that identifies the characteristics of customers who frequently buy a certain product. To obtain a list of customers who shop at a certain store and are likely to buy a related product, you might apply the model to the customer data for that store. In this case, the store customer data is the scoring data.

Most supervised learning can be applied to a population of interest. The principal supervised machine learning techniques, **classification** and **regression**, can both be used for scoring.

Oracle Machine Learning does not support the scoring operation for **attribute importance**, another supervised function. Models of this type are built on a population of interest to obtain information about that population; they cannot be applied to separate data. An attribute importance model returns and ranks the attributes that are most important in predicting a target value.

Oracle Machine Learning supports the supervised machine learning functions described in the following table:

Table 3-1 Oracle Machine Learning Supervised Functions

Function	Description	Sample Problem
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
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
Regression	Approximates and forecasts continuous values	Given demographic and purchasing data about a set of customers, predict customers' age

3.1.2 Unsupervised Machine Learning

Overview of unsupervised machine 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. It can also be used to make predictions.

3.1.2.1 Unsupervised Learning: Scoring

Introduces unsupervised learning, supported scoring operations, and unsupervised machine learning functions.

Although unsupervised machine learning does not specify a target, most unsupervised learning can be applied to a population of interest. For example, clustering models use descriptive machine learning techniques, but they can be applied to classify cases according to their cluster assignments. **Anomaly Detection**, although unsupervised, is typically used to predict whether a data point is typical among a set of cases.

Oracle Machine Learning supports the scoring operation for **Clustering** and **Feature Extraction**, both unsupervised machine learning functions. Oracle Machine Learning does not support the scoring operation for **Association Rules**, another unsupervised function. Association models are built on a population of interest to obtain information about that population; they cannot be applied to separate data. An association model returns rules that explain how items or events are associated with each other. The association rules are returned with statistics that can be used to rank them according to their probability.

OML supports the unsupervised functions described in the following table:

Table 3-2 Oracle Machine Learning Unsupervised Functions

Function	Description	Sample Problem
Anomaly Detection	Identifies items (outliers) that do not satisfy the characteristics of "normal" data	Given demographic data about a set of customers, identify customer purchasing behavior that is significantly different from the norm
Association Rules	Finds items that tend to co-occur in the data and specifies the rules that govern their co-occurrence	Find the items that tend to be purchased together and specify their relationship
Clustering	Finds natural groupings in the data	Segment demographic data into clusters and rank the probability that an individual belongs to a given cluster
Feature Extraction	Creates new attributes (features) using linear combinations of the original attributes	Given demographic data about a set of customers, group the attributes into general characteristics of the customers

Related Topics

- Machine Learning Functions
  Part II provides basic conceptual information about machine learning functions that the Oracle Machine Learning for SQL supports.
- In-Database Scoring
  Scoring is the application of a machine learning algorithm to new data. In Oracle Machine Learning for SQL scoring engine and the data both reside within the database.

3.2 Algorithms

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

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

3.2.1 Oracle Machine Learning Supervised Algorithms

Oracle Machine Learning for SQL (OML4SQL) supports the supervised machine learning algorithms described in the following table.

Table 3-3 Oracle Machine Learning Algorithms for Supervised Functions

Algorithm	Function	Description
Decision Tree	Classification	Decision trees extract 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.
Explicit Semantic Analysis	Classification	Explicit Semantic Analysis (ESA) is designed to make predictions for text data. This algorithm can address use cases with hundreds of thousands of classes. In Oracle Database 12*c* Release 2, ESA was introduced as Feature Extraction algorithm.
Exponential Smoothing	Time Series	Exponential Smoothing (ESM) provides forecasts for time series data. Forecasts are made for each time period within a user-specified forecast window. ESM provides a total of 14 different time series models, including all the most popular estimates of trend and seasonal effects. Choice of model is controlled by user settings. ESM provides confidence bounds on its forecasts.
Generalized Linear Model	Classification and Regression	Generalized Linear Model (GLM) implements logistic regression for classification of binary targets and linear regression for continuous targets. GLM classification supports confidence bounds for prediction probabilities. GLM regression supports confidence bounds for predictions.
Minimum Description Length	Attribute Importance	Minimum Description Length (MDL) is an information theoretic model selection principle. MDL assumes that the simplest, most compact representation of data is the best and most probable explanation of the data.

Table 3-3 (Cont.) Oracle Machine Learning Algorithms for Supervised Functions

Algorithm	Function	Description
Naive Bayes	Classification	Naive Bayes makes predictions using Bayes' Theorem, which derives the probability of a prediction from the underlying evidence, as observed in the data.
Neural Network	Classification and Regression	Neural Network in machine learning is an artificial algorithm inspired from biological neural network and is used to estimate or approximate functions that depend on a large number of generally unknown inputs. Neural Network is designed for classification and regression.
Random Forest	Classification	Random Forest is a powerful machine learning algorithm. The Random Forest algorithm builds a number of Decision Tree models and predicts using the ensemble of trees.
Support Vector Machine	Classification and Regression	Distinct versions of the Support Vector Machine (SVM) algorithm use different kernel functions to handle different types of data sets. Linear and Gaussian (nonlinear) kernels are supported. SVM classification attempts to separate the target classes with the widest possible margin. SVM regression tries to find a continuous function such that the maximum number of data points lie within an epsilon-wide tube around it.
XGBoost	Classification and Regression	XGBoost is machine learning algorithm for regression and classification that makes available the XGBoost open source package. Oracle Machine Learning for SQL XGBoost prepares training data, invokes XGBoost, builds and persists a model, and applies the model for prediction.

3.2.2 Oracle Machine Learning Unsupervised Algorithms

Oracle Machine Learning for SQL (OML4SQL) supports the unsupervised machine learning algorithms described in the following table.

Table 3-4 Oracle Machine Learning Algorithms for Unsupervised Functions

Algorithm	Function	Description
Apriori	Association	Apriori performs market basket analysis by identifying co-occurring items (frequent itemsets) within a set. Apriori finds rules with support greater than a specified minimum support and confidence greater than a specified minimum confidence.
CUR Matrix Decomposition	Attribute Importance	CUR Matrix Decomposition is an alternative to Support Vector Machine (SVM) and Principal Component Analysis (PCA) and an important tool for exploratory data analysis. This algorithm performs analytical processing and singles out important columns and rows.

Table 3-4 (Cont.) Oracle Machine Learning Algorithms for Unsupervised Functions

Algorithm	Function	Description
Expectation Maximization	Clustering	Expectation Maximization (EM) is a density estimation algorithm that performs probabilistic clustering. In density estimation, the goal is to construct a density function that captures how a given population is distributed. The density estimate is based on observed data that represents a sample of the population. Oracle Machine Learning supports probabilistic clustering and data frequency estimates and other applications of Expectation Maximization.
Explicit Semantic Analysis	Feature Extraction	Explicit Semantic Analysis (ESA) uses existing knowledge base as features. An attribute vector represents each feature or a concept. ESA creates a reverse index that maps every attribute to the knowledge base concepts or the concept-attribute association vector value.
k-Means	Clustering	*k*-Means is a distance-based clustering algorithm that partitions the data into a predetermined number of clusters. Each cluster has a centroid (center of gravity). Cases (individuals within the population) that are in a cluster are close to the centroid. OML4SQL supports an enhanced version of *k*-Means. It goes beyond the classical implementation by defining a hierarchical parent-child relationship of clusters.
Multivariate State Estimation Technique - Sequential Probability Ratio Test	Anomaly Detection	The Multivariate State Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT) algorithm is a nonlinear, nonparametric anomaly detection machine learning technique designed for monitoring critical processes. It detects subtle anomalies while also producing minimal false alarms.
Non-Negative Matrix Factorization	Feature Extraction	Non-Negative Matrix Factorization (NMF) generates new attributes using linear combinations of the original attributes. The coefficients of the linear combinations are non-negative. During model apply, an NMF model maps the original data into the new set of attributes (features) discovered by the model.
One Class Support Vector Machine	Anomaly Detection	One-class SVM builds a profile of one class. When the model is applied, it identifies cases that are somehow different from that profile. This allows for the detection of rare cases that are not necessarily related to each other.
Orthogonal Partitioning Clustering	Clustering	Orthogonal Partitioning Clustering (O-Cluster) creates a hierarchical, grid-based clustering model. The algorithm creates clusters that define dense areas in the attribute space. A sensitivity parameter defines the baseline density level.
Singular Value Decomposition and Principal Component Analysis	Feature Extraction	Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) are orthogonal linear transformations that are optimal at capturing the underlying variance of the data. This property is extremely useful for reducing the dimensionality of high-dimensional data and for supporting meaningful data visualization. In addition to dimensionality reduction, SVD and PCA have a number of other important applications, such as data de-noising (smoothing), data compression, matrix inversion, and solving a system of linear equations.

Related Topics

- Algorithms
  Oracle Machine Learning for SQL supports the algorithms listed in Part III. Part III provides basic conceptual information about the algorithms. There is at least one algorithm for each of the machine learning functions.

3.3 Data Preparation

Preparing the data is a valuable step in solving machine learning problems.

The quality of a model depends to a large extent on the quality of the data used to build (train) it. Much of the time spent in any given machine learning project is devoted to data preparation. The data must be carefully inspected, cleansed, and transformed, and algorithm-appropriate data preparation methods must be applied.

The process of data preparation is further complicated by the fact that any data to which a model is applied, whether for testing or for scoring, must undergo the same transformations as the data used to train the model.

3.3.1 Oracle Machine Learning for SQL Simplifies Data Preparation

Learn about various features of Oracle Machine Learning for SQL for data preparation.

OML4SQL offers several features that significantly simplify the process of data preparation:

- Embedded data preparation: The transformations used in training the model are embedded in the model and automatically run whenever the model is applied to new data. If you specify transformations for the model, you only have to specify them once.
- Automatic Data Preparation (ADP): Oracle Machine Learning for SQL supports an automated data preparation mode. When ADP is active, Oracle Machine Learning for SQL automatically performs the data transformations required by the algorithm. The transformation instructions are embedded in the model along with any user-specified transformation instructions.
- Automatic management of missing values and sparse data: Oracle Machine Learning for SQL uses consistent methodology across machine learning algorithms to handle sparsity and missing values.
- Transparency: Oracle Machine Learning for SQL provides model details, which are a view of the attributes that are internal to the model. This insight into the inner details of the model is possible because of reverse transformations, which map the transformed attribute values to a form that can be interpreted by a user. Where possible, attribute values are reversed to the original column values. Reverse transformations are also applied to the target of a supervised model, thus the results of scoring are in the same units as the units of the original target.
- Tools for custom data preparation: Oracle Machine Learning for SQL provides many common transformation routines in the `DBMS_DATA_MINING_TRANSFORM` PL/SQL package. You can use these routines, or develop your own routines in SQL, or both. The SQL language is well suited for implementing transformations in the database. You can use custom transformation instructions along with ADP or instead of ADP.

3.3.2 Case Data

Learn the importance of case data in machine learning.

Most machine learning algorithms act on single-record case data, where the information for each case is stored in a separate row. The data attributes for the cases are stored in the columns.

When the data is organized in transactions, the data for one case (one transaction) is stored in many rows. An example of transactional data is market basket data. With the single exception of Association Rules, which can operate on native transactional data, Oracle Machine Learning for SQL algorithms require single-record case organization.

3.3.2.1 Nested Data

Learn how nested columns are treated in Oracle Machine Learning for SQL.

OML4SQL supports attributes in nested columns. A transactional table can be cast as a nested column and included in a table of single-record case data. Similarly, star schemas can be cast as nested columns. With nested data transformations, Oracle Machine Learning for SQL can effectively mine data originating from multiple sources and configurations.

3.3.3 Text Data

Prepare and transform unstructured text data for machine learning.

Oracle Machine Learning for SQL interprets `CLOB` columns and long `VARCHAR2` columns automatically as unstructured text. Additionally, you can specify columns of short `VARCHAR2`, `CHAR`, `BLOB`, and `BFILE` as unstructured text. Unstructured text includes data items such as web pages, document libraries, Power Point presentations, product specifications, emails, comment fields in reports, and call center notes.

OML4SQL uses Oracle Text utilities and term weighting strategies to transform unstructured text for analysis. In text transformation, text terms are extracted and given numeric values in a text index. The text transformation process is configurable for the model and for individual attributes. Once transformed, the text can by mined with a OML4SQL algorithm.

Related Topics

- Prepare the Data
- Machine Learning Operations on Unstructured Text

3.4 In-Database Scoring

Scoring is the application of a machine learning algorithm to new data. In Oracle Machine Learning for SQL scoring engine and the data both reside within the database.

In traditional machine learning, models are built using specialized software on a remote system and deployed to another system for scoring. This is a cumbersome, error-prone process open to security violations and difficulties in data synchronization.

With OML4SQL, scoring is simple and secure. The scoring engine and the data both reside within the database. Scoring is an extension to the SQL language, so the results of machine learning can easily be incorporated into applications and reporting systems.

3.4.1 Parallel Execution and Ease of Administration

All Oracle Machine Learning for SQL scoring routines support parallel execution for scoring large data sets.

In-database scoring provides performance advantages. All Oracle Machine Learning for SQL scoring routines support parallel execution, which significantly reduces the time required for executing complex queries and scoring large data sets.

In-database machine learning minimizes the IT effort needed to support OML4SQL initiatives. Using standard database techniques, models can easily be refreshed (re-created) on more recent data and redeployed. The deployment is immediate since the scoring query remains the same; only the underlying model is replaced in the database.

Related Topics

- *Oracle Database VLDB and Partitioning Guide*

3.4.2 SQL Functions for Model Apply and Dynamic Scoring

In Oracle Machine Learning for SQL, scoring is performed by SQL language functions. Understand the different ways of scoring using SQL functions.

The functions perform prediction, clustering, and feature extraction. The functions can be invoked in two different ways: By applying a machine learning model object (Example 3-1), or by executing an analytic clause that computes the machine learning analysis dynamically and applies it to the data (Example 3-2). Dynamic scoring, which eliminates the need for a model, can supplement, or even replace, the more traditional methodology described in "The Machine Learning Process".

In Example 3-1, the `PREDICTION_PROBABILITY` function applies the model svmc_sh_clas_sample, created in Example 2-1, to score the data in `mining_data_apply_v`. The function returns the ten customers in Italy who are most likely to use an affinity card.

In Example 3-2, the functions `PREDICTION` and `PREDICTION_PROBABILITY` use the analytic syntax (the `OVER` () clause) to dynamically score the data in `mining_data_apply_v`. The query returns the customers who currently do not have an affinity card with the probability that they are likely to use.

Example 3-1 Applying a Oracle Machine Learning for SQL Model to Score Data

```
SELECT cust_id FROM
  (SELECT cust_id,
        rank() over (order by PREDICTION_PROBABILITY(svmc_sh_clas_sample, 1
                     USING *) DESC, cust_id) rnk
   FROM mining_data_apply_v
   WHERE country_name = 'Italy')
WHERE rnk <= 10
ORDER BY rnk;

   CUST_ID
```

```
----------
    101445
    100179
    100662
    100733
    100554
    100081
    100344
    100324
    100185
    101345
```

Example 3-2 Executing an Analytic Function to Score Data

```
SELECT cust_id, pred_prob FROM
  (SELECT cust_id, affinity_card,
    PREDICTION(FOR TO_CHAR(affinity_card) USING *) OVER () pred_card,
    PREDICTION_PROBABILITY(FOR TO_CHAR(affinity_card),1 USING *) OVER ()
pred_prob
   FROM mining_data_build_v)
WHERE affinity_card = 0
AND pred_card = 1
ORDER BY pred_prob DESC;

   CUST_ID PRED_PROB
---------- ---------
    102434       .96
    102365       .96
    102330       .96
    101733       .95
    102615       .94
    102686       .94
    102749       .93
    .
    .
    .
    101656       .51
```

Related Topics

- *Oracle Database SQL Language Reference*
- *Oracle Machine Learning for SQL User's Guide*

Part II
Machine Learning Functions

Part II provides basic conceptual information about machine learning functions that the Oracle Machine Learning for SQL supports.

Machine learning functions represent a class of problems that can be solved using OML4SQL algorithms.

Part II contains these chapters:

- Regression
- Classification
- Anomaly Detection
- Ranking
- Clustering
- Association
- Feature Selection
- Feature Extraction
- Row Importance
- Time Series

> **Note:**
>
> The term machine learning function has no relationship to a SQL language function.

Related Topics

- Algorithms
  Oracle Machine Learning for SQL supports the algorithms listed in Part III. Part III provides basic conceptual information about the algorithms. There is at least one algorithm for each of the machine learning functions.
- *Oracle Database SQL Language Reference*

4
Regression

Learn how to predict a continuous numerical target through regression - the supervised machine learning function.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

4.1 About Regression

Regression is an Oracle Machine Learning for SQL function that predicts numeric values along a continuum.

Profit, sales, mortgage rates, house values, square footage, temperature, or distance can be predicted using Regression techniques. For example, a regression model can be used to predict the value of a house based on location, number of rooms, lot size, and other factors.

A regression task begins with a data set in which the target values are known. For example, a regression model that predicts house values can be developed based on observed data for many houses over a period of time. In addition to the value, the data can track the age of the house, square footage, number of rooms, taxes, school district, proximity to shopping centers, and so on. House value can be the target, the other attributes are the predictors, and the data for each house constitutes a case.

In the model build (training) process, a regression algorithm estimates the value of the target as a function of the predictors for each case in the build data. These relationships between predictors and target are summarized in a model, which can then be applied to a different data set in which the target values are unknown.

Regression models are tested by computing various statistics that measure the difference between the predicted values and the expected values. The historical data for a regression project is typically divided into two data sets: one for building the model, the other for testing the model.

Regression modeling has many applications in trend analysis, business planning, marketing, financial forecasting, time series prediction, biomedical and drug response modeling, and environmental modeling.

4.1.1 How Does Regression Work?

Understand regression as a mathematical expression.

You do not need to understand the mathematics used in regression analysis to develop and use quality regression models for Oracle Machine Learning for SQL. However, it is helpful to understand a few basic concepts.

Regression analysis seeks to determine the values of parameters for a function that cause the function to best fit a set of data observations that you provide. The following equation expresses these relationships in symbols. It shows that regression is the process of estimating the value of a continuous target (y) as a function (F) of one or more predictors ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$), a set of parameters (θ_1 , θ_2 , ..., θ_n), and a measure of error (e).

$$y = F(\mathbf{x}, \theta) + e$$

The predictors can be understood as independent variables and the target as a dependent variable. The error, also called the **residual**, is the difference between the expected and predicted value of the dependent variable. The regression parameters are also known as **regression coefficients**.

The process of training a regression model involves finding the parameter values that minimize a measure of the error, for example, the sum of squared errors.

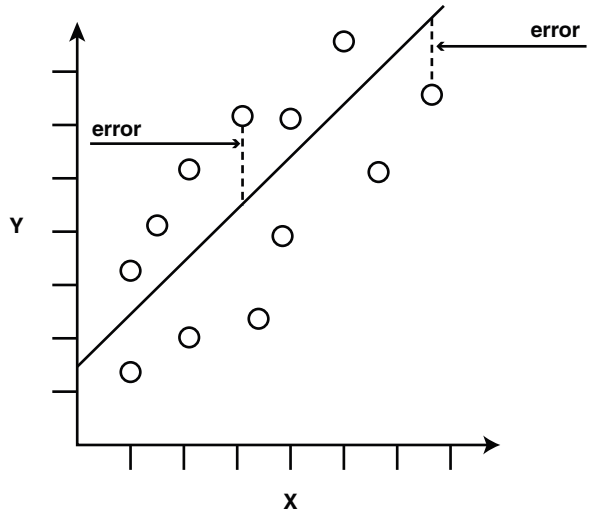
There are different families of regression functions and different ways of measuring the error.

4.1.1.1 Linear Regression

A linear regression technique can be used if the relationship between the predictors and the target can be approximated with a straight line.

Regression with a single predictor is the easiest to visualize. Simple linear regression with a single predictor is shown in the following figure:

Figure 4-1 Linear Regression With a Single Predictor



Linear regression with a single predictor can be expressed with the following equation.

$$y = \theta_2 \mathbf{x} + \theta_1 + e$$

The regression parameters in simple linear regression are:

- The **slope** of the line ($_2$) — the angle between a data point and the regression line
- The ***y* intercept** ($_1$) — the point where **x** crosses the *y* axis (**x** = 0)

4.1.1.2 Multivariate Linear Regression

The term **multivariate linear regression** refers to linear regression with two or more predictors ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$). When multiple predictors are used, the regression line cannot be visualized in two-dimensional space. However, the line can be computed by expanding the equation for single-predictor linear regression to include the parameters for each of the predictors.

$$y = \theta_1 + \theta_2 \mathbf{x}_1 + \theta_3 \mathbf{x}_2 + \ldots\ldots \theta_n \mathbf{x}_{n-1} + e$$

4.1.1.3 Regression Coefficients

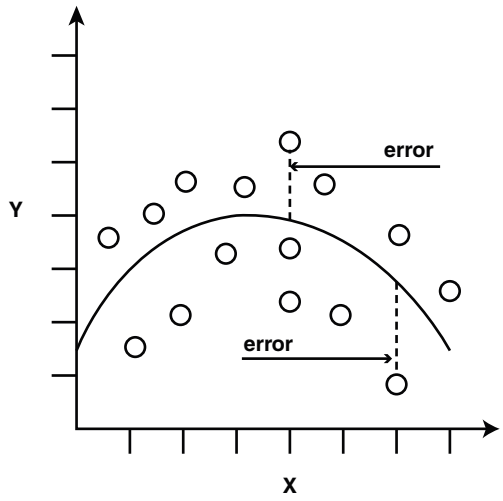
In multivariate linear regression, the regression parameters are often referred to as coefficients. When you build a multivariate linear regression model, the algorithm computes a coefficient for each of the predictors used by the model. The coefficient is a measure of the impact of the predictor **x** on the target *y*. Numerous statistics are available for analyzing the regression coefficients to evaluate how well the regression line fits the data.

4.1.1.4 Nonlinear Regression

Often the relationship between **x** and ***y*** cannot be approximated with a straight line. In this case, a nonlinear regression technique can be used. Alternatively, the data can be preprocessed to make the relationship linear.

Nonlinear regression models define ***y*** as a function of **x** using an equation that is more complicated than the linear regression equation. In the following figure, **x** and ***y*** have a nonlinear relationship.

Figure 4-2 Nonlinear Regression With a Single Predictor



4.1.1.5 Multivariate Nonlinear Regression

The term **multivariate nonlinear regression** refers to nonlinear regression with two or more predictors ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$). When multiple predictors are used, the nonlinear relationship cannot be visualized in two-dimensional space.

4.1.1.6 Confidence Bounds

A regression model predicts a numeric target value for each case in the scoring data. In addition to the predictions, some regression algorithms can identify confidence bounds, which are the upper and lower boundaries of an interval in which the predicted value is likely to lie.

When a model is built to make predictions with a given confidence, the confidence interval is produced along with the predictions. For example, a model predicts the value of a house to be $500,000 with a 95% confidence that the value is between $475,000 and $525,000.

4.2 Testing a Regression Model

A regression model is tested by applying it to test data with known target values and comparing the predicted values with the known values.

The test data must be compatible with the data used to build the model and must be prepared in the same way that the build data was prepared. Typically the build data and test data come from the same historical data set. A percentage of the records is used to build the model; the remaining records are used to test the model.

Test metrics are used to assess how accurately the model predicts these known values. If the model performs well and meets the business requirements, it can then be applied to new data to predict the future.

4.2.1 Regression Statistics

The Root Mean Squared Error and the Mean Absolute Error are commonly used statistics for evaluating the overall quality of a regression model. Different statistics may also be available depending on the regression methods used by the algorithm.

4.2.1.1 Root Mean Squared Error

The Root Mean Squared Error (RMSE) is the square root of the average squared distance of a data point from the fitted line.

This SQL expression calculates the RMSE.

```
SQRT(AVG((predicted_value - actual_value) * (predicted_value -
actual_value)))
```

This formula shows the RMSE in mathematical symbols. The large sigma character represents summation; *j* represents the current predictor, and *n* represents the number of predictors.

Figure 4-3 Room Mean Squared Error

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

4.2.1.2 Mean Absolute Error

The Mean Absolute Error (MAE) is the average of the absolute value of the residuals (error). The MAE is very similar to the RMSE but is less sensitive to large errors.

This SQL expression calculates the MAE.

```
AVG(ABS(predicted_value - actual_value))
```

This formula shows the MAE in mathematical symbols. The large sigma character represents summation; *j* represents the current predictor, and *n* represents the number of predictors.

Figure 4-4 Mean Absolute Error

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

4.3 Regression Algorithms

Oracle Machine Learning for SQL supports these algorithms for regression: Generalized Linear Model (GLM), Neural Network (NN), Support Vector Machine (SVM), and XGBoost.

GLM and SVM algorithms are particularly suited for analysing data sets that have very high dimensionality (many attributes), including transactional and unstructured data.

- **Generalized Linear Model**

  GLM is a popular statistical technique for linear modeling. Oracle Machine Learning for SQL implements GLM for regression and for binary classification. GLM provides extensive coefficient statistics and model statistics, as well as row diagnostics. GLM also supports confidence bounds.

- **Neural Network**

  Neural Network is a powerful algorithm that can learn arbitrary nonlinear regression functions.

- **Support Vector Machine**

  SVM is a powerful, state-of-the-art algorithm for linear and nonlinear regression. OML4SQL implements SVM for regression, classification, and anomaly detection. SVM regression supports two kernels: the Gaussian kernel for nonlinear regression and the linear kernel for linear regression.

  > **Note:**
  >
  > OML4SQL uses the linear kernel SVM as the default regression algorithm.

- **XGBoost**

  XGBoost is machine learning algorithm for regression and classification that makes available the XGBoost open source package. Oracle Machine Learning for SQL XGBoost prepares training data, invokes XGBoost, builds and persists a model, and applies the model for prediction.

Related Topics

- Generalized Linear Model
  Learn how to use Generalized Linear Model (GLM) statistical technique for linear modeling.
- Neural Network
  Learn about the Neural Network algorithms for regression and classification machine learning functions.
- Support Vector Machine
  Learn how to use Support Vector Machine (SVM), a powerful algorithm based on statistical learning theory.

- XGBoost
  XGBoost is highly-efficient, scalable machine learning algorithm for regression and classification that makes available the XGBoost Gradient Boosting open source package.

5
Classification

Learn how to predict a categorical target through classification - the supervised machine learning function.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

5.1 About Classification

Classification is a machine learning function that assigns items in a collection to target categories or classes.

The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model can be used to identify loan applicants as low, medium, or high credit risks.

A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk can be developed based on observed data for many loan applicants over a period of time. In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating is the target, the other attributes are the predictors, and the data for each customer constitutes a case.

Classification are discrete and do not imply order. Continuous, floating-point values indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm.

The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating.

In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

Classification models are tested by comparing the predicted values to known target values in a set of test data. The historical data for a classification project is typically divided into two data sets: one for building the model; the other for testing the model.

Applying a classification model results in class assignments and probabilities for each case. For example, a model that classifies customers as low, medium, or high value also predicts the probability of each classification for each customer.

Classification has many applications in customer segmentation, business modeling, marketing, credit analysis, and biomedical and drug response modeling.

5.2 Testing a Classification Model

A classification model is tested by applying it to test data with known target values and comparing the predicted values with the known values.

The test data must be compatible with the data used to build the model and must be prepared in the same way that the build data was prepared. Typically the build data and test data come from the same historical data set. A percentage of the records is used to build the model; the remaining records are used to test the model.

Test metrics are used to assess how accurately the model predicts the known values. If the model performs well and meets the business requirements, it can then be applied to new data to predict the future.

5.2.1 Confusion Matrix

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

The following figure shows a confusion matrix for a binary classification model. The rows present the number of actual classifications in the test data. The columns present the number of predicted classifications made by the model.

Figure 5-1 Confusion Matrix for a Binary Classification Model

		PREDICTED CLASS	
		affinity_card = 1	**affinity_card = 0**
ACTUAL CLASS	**affinity_card = 1**	**516**	**25**
	affinity_card = 0	**10**	**725**

In this example, the model correctly predicted the positive class (also called true positive (TP)) for `affinity_card` 516 times and incorrectly predicted (also called false negative (FN)) it 25 times. The model correctly predicted the negative class (also called true negative (TN)) for `affinity_card` 725 times and incorrectly predicted (also called false positive (FP)) it 10 times. The following can be computed from this confusion matrix:

- The model made 1241 correct predictions, that is, TP + TN, (516 + 725).
- The model made 35 incorrect predictions, that is, FN + FP, (25 + 10).
- There are 1276 total scored cases, (516 + 25 + 10 + 725).
- The error rate is 35/1276 = 0.0274. (FN+FP/Total)
- The overall accuracy rate is 1241/1276 = 0.9725 (TP+TN)/Total).

Precision and Recall

Consider the same example, the accuracy rate shows 0.97. However, there are cases where the model has incorrectly predicted. **Precision** (positive predicted value) is the ability of a classification model to return only relevant cases. Precision can be calculated as TP/TP+FP. **Recall** (sensitivity or true positive rate) is the ability of a classification model to return relevant cases. Recall can be calculated as TP/TP+FN. The precision in this example is 516/526 = 0.98. The recall in this example is 516/541 = 0.95. Ideally, the model is good when both precision and recall are 1. This can happen when the numerator and the denominator are equal. That means, for precision, FP is zero and for recall, FN is zero.

5.2.2 Lift

Lift measures the degree to which the predictions of a classification model are better than randomly-generated predictions.

Lift applies to binary classification only, and it requires the designation of a positive class. If the model itself does not have a binary target, you can compute lift by designating one class as positive and combining all the other classes together as one negative class.

Numerous statistics can be calculated to support the notion of lift. Basically, lift can be understood as a ratio of two percentages: the percentage of correct positive classifications made by the model to the percentage of actual positive classifications in the test data. For example, if 40% of the customers in a marketing survey have responded favorably (the positive classification) to a promotional campaign in the past and the model accurately predicts 75% of them, the lift is obtained by dividing .75 by .40. The resulting lift is 1.875.

Lift is computed against quantiles that each contain the same number of cases. The data is divided into quantiles after it is scored. It is ranked by probability of the positive class from highest to lowest, so that the highest concentration of positive predictions is in the top quantiles. A typical number of quantiles is 10.

Lift is commonly used to measure the performance of response models in marketing applications. The purpose of a response model is to identify segments of the population with potentially high concentrations of positive responders to a marketing campaign. Lift reveals how much of the population must be solicited to obtain the highest percentage of potential responders.

Related Topics

- Positive and Negative Classes
  Discusses the importance of positive and negative classes in a confusion matrix.

5.2.2.1 Lift Statistics

Learn the different Lift statistics that Oracle Machine Learning for SQL can compute.

Oracle Machine Learning for SQL computes the following lift statistics:

- **Probability threshold** for a quantile *n* is the minimum probability for the positive target to be included in this quantile or any preceding quantiles (quantiles *n*-1, *n*-2,..., 1). If a cost matrix is used, a cost threshold is reported instead. The cost threshold is the maximum cost for the positive target to be included in this quantile or any of the preceding quantiles.
- **Cumulative gain** is the ratio of the cumulative number of positive targets to the total number of positive targets.
- **Target density** of a quantile is the number of true positive instances in that quantile divided by the total number of instances in the quantile.
- **Cumulative target density** for quantile *n* is the target density computed over the first *n* quantiles.
- **Quantile lift** is the ratio of the target density for the quantile to the target density over all the test data.
- **Cumulative percentage of records** for a quantile is the percentage of all cases represented by the first *n* quantiles, starting at the end that is most confidently positive, up to and including the given quantile.
- **Cumulative number of targets** for quantile *n* is the number of true positive instances in the first *n* quantiles.
- **Cumulative number of nontargets** is the number of actually negative instances in the first *n* quantiles.
- **Cumulative lift** for a quantile is the ratio of the cumulative target density to the target density over all the test data.

Related Topics

- Costs

5.2.3 Receiver Operating Characteristic (ROC)

ROC is a metric for comparing predicted and actual target values in a classification model.

ROC, like Lift, applies to binary classification and requires the designation of a positive class.

You can use ROC to gain insight into the decision-making ability of the model. How likely is the model to accurately predict the negative or the positive class?

ROC measures the impact of changes in the **probability threshold**. The probability threshold is the decision point used by the model for classification. The default probability threshold for binary classification is 0.5. When the probability of a prediction is 50% or more, the model predicts that class. When the probability is less than 50%, the other class is predicted. (In multiclass classification, the predicted class is the one predicted with the highest probability.)

Related Topics

- Positive and Negative Classes
  Discusses the importance of positive and negative classes in a confusion matrix.

5.2.3.1 The ROC Curve

ROC can be plotted as a curve on an X-Y axis. The **false positive rate** is placed on the X axis. The **true positive rate** is placed on the Y axis.

The top left corner is the optimal location on an ROC graph, indicating a high true positive rate and a low false positive rate.

5.2.3.2 Area Under the Curve

The area under the ROC curve (AUC) measures the discriminating ability of a binary classification model. The larger the AUC, the higher the likelihood that an actual positive case is assigned, and a higher probability of being positive than an actual negative case. The AUC measure is especially useful for data sets with unbalanced target distribution (one target class dominates the other).

5.2.3.3 ROC and Model Bias

The ROC curve for a model represents all the possible combinations of values in its confusion matrix.

Changes in the probability threshold affect the predictions made by the model. For instance, if the threshold for predicting the positive class is changed from 0.5 to 0.6, then fewer positive predictions are made. This affects the distribution of values in the confusion matrix: the number of true and false positives and true and false negatives differ.

You can use ROC to find the probability thresholds that yield the highest overall accuracy or the highest per-class accuracy. For example, if it is important to you to accurately predict the positive class, but you don't care about prediction errors for the

negative class, then you can lower the threshold for the positive class. This can bias the model in favor of the positive class.

A cost matrix is a convenient mechanism for changing the probability thresholds for model scoring.

Related Topics

- Costs

5.2.3.4 ROC Statistics

Oracle Machine Learning for SQL computes the following ROC statistics:

- **Probability threshold:** The minimum predicted positive class probability resulting in a positive class prediction. Different threshold values result in different hit rates and different false alarm rates.
- **True negatives:** Negative cases in the test data with predicted probabilities strictly less than the probability threshold (correctly predicted).
- **True positives:** Positive cases in the test data with predicted probabilities greater than or equal to the probability threshold (correctly predicted).
- **False negatives:** Positive cases in the test data with predicted probabilities strictly less than the probability threshold (incorrectly predicted).
- **False positives:** Negative cases in the test data with predicted probabilities greater than or equal to the probability threshold (incorrectly predicted).
- **True positive fraction**: Hit rate. (true positives/(true positives + false negatives))
- **False positive fraction:** False alarm rate. (false positives**/**(false positives + true negatives))

5.3 Biasing a Classification Model

Costs, prior probabilities, and class weights are methods for biasing classification models.

5.3.1 Costs

A cost matrix is a mechanism for influencing the decision making of a model. A cost matrix can cause the model to minimize costly misclassifications. It can also cause the model to maximize beneficial accurate classifications.

For example, if a model classifies a customer with poor credit as low risk, this error is costly. A cost matrix can bias the model to avoid this type of error. The cost matrix can also be used to bias the model in favor of the correct classification of customers who have the worst credit history.

ROC is a useful metric for evaluating how a model behaves with different probability thresholds. You can use ROC to help you find optimal costs for a given classifier given different usage scenarios. You can use this information to create cost matrices to influence the deployment of the model.

5.3.1.1 Costs Versus Accuracy

Compares Cost matrix and Confusion matrix for costs and accuracy to evaluate model quality.

Like a confusion matrix, a cost matrix is an *n*-by-*n* matrix, where *n* is the number of classes. Both confusion matrices and cost matrices include each possible combination of actual and predicted results based on a given set of test data.

A confusion matrix is used to measure accuracy, the ratio of correct predictions to the total number of predictions. A cost matrix is used to specify the relative importance of accuracy for different predictions. In most business applications, it is important to consider costs in addition to accuracy when evaluating model quality.

Related Topics

- Confusion Matrix

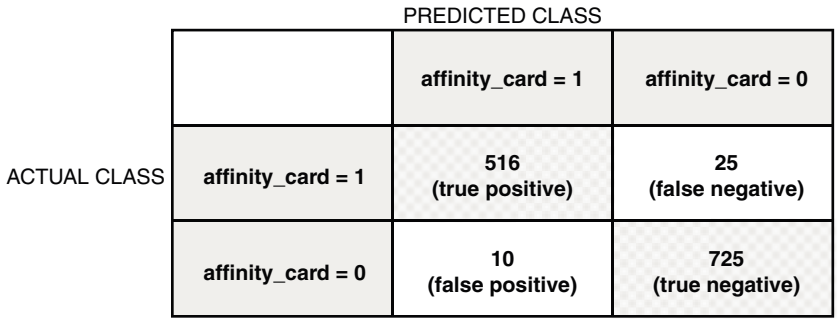
5.3.1.2 Positive and Negative Classes

Discusses the importance of positive and negative classes in a confusion matrix.

The positive class is the class that you care the most about. Designation of a positive class is required for computing Lift and ROC.

In the confusion matrix, in the following figure, the value 1 is designated as the positive class. This means that the creator of the model has determined that it is more important to accurately predict customers who increase spending with an affinity card (`affinity_card`=1) than to accurately predict non-responders (`affinity_card`=0). If you give affinity cards to some customers who are not likely to use them, there is little loss to the company since the cost of the cards is low. However, if you overlook the customers who are likely to respond, you miss the opportunity to increase your revenue.

Figure 5-2 Positive and Negative Predictions



The true and false positive rates in this confusion matrix are:

- False positive rate — 10/(10 + 725) =.01

- True positive rate — 516/(516 + 25) =.95

Related Topics

- Lift
  Lift measures the degree to which the predictions of a classification model are better than randomly-generated predictions.
- Receiver Operating Characteristic (ROC)
  ROC is a metric for comparing predicted and actual target values in a classification model.

5.3.1.3 Assigning Costs and Benefits

In a cost matrix, positive numbers (costs) can be used to influence negative outcomes. Since negative costs are interpreted as benefits, negative numbers (benefits) can be used to influence positive outcomes.

Suppose you have calculated that it costs your business $1500 when you do not give an affinity card to a customer who can increase spending. Using the model with the confusion matrix shown in Figure 5-2, each false negative (misclassification of a responder) costs $1500. Misclassifying a non-responder is less expensive to your business. You estimate that each false positive (misclassification of a non-responder) only costs $300.

You want to keep these costs in mind when you design a promotion campaign. You estimate that it costs $10 to include a customer in the promotion. For this reason, you associate a benefit of $10 with each true negative prediction, because you can eliminate those customers from your promotion. Each customer that you eliminate represents a savings of $10. In your cost matrix, you specify this benefit as -10, a negative cost.

The following figure shows how you would represent these costs and benefits in a cost matrix:

Figure 5-3 Cost Matrix Representing Costs and Benefits

		PREDICTED	
		affinity_card = 1	**affinity_card = 0**
ACTUAL	**affinity_card = 1**	**0**	**1500**
	affinity_card = 0	**300**	**-10**

With Oracle Machine Learning for SQL you can specify costs to influence the scoring of any classification model. Decision Tree models can also use a cost matrix to influence the model build.

5.3.2 Priors and Class Weights

Learn about Priors and Class Weights in a classification model to produce a useful result.

With Bayesian models, you can specify **Prior** probabilities to offset differences in distribution between the build data and the real population (scoring data). With other forms of classification, you are able to specify **Class Weights**, which have the same biasing effect as priors.

In many problems, one target value dominates in frequency. For example, the positive responses for a telephone marketing campaign is 2% or less, and the occurrence of fraud in credit card transactions is less than 1%. A classification model built on historic data of this type cannot observe enough of the rare class to be able to distinguish the characteristics of the two classes; the result can be a model that when applied to new data predicts the frequent class for every case. While such a model can be highly accurate, it is not be very useful. This illustrates that it is not a good idea to rely solely on accuracy when judging the quality of a classification model.

To correct for unrealistic distributions in the training data, you can specify priors for the model build process. Other approaches to compensating for data distribution issues include stratified sampling and anomaly detection.

Related Topics

- Anomaly Detection
  Learn how to detect rare cases in the data through anomaly detection - an unsupervised function.

5.4 Classification Algorithms

Learn the different classification algorithms used in Oracle Machine Learning for SQL.

Oracle Machine Learning for SQL provides the following algorithms for classification:

- **Decision Tree**

  Decision trees automatically generate rules, which are conditional statements that reveal the logic used to build the tree.

- **Explicit Semantic Analysis**

  Explicit Semantic Analysis (ESA) is designed to make predictions for text data. This algorithm can address use cases with hundreds of thousands of classes.

- **Generalized Linear Model**

  Generalized Linear Model (GLM) is a popular statistical technique for linear modeling. OML4SQL implements GLM for binary classification and for regression. GLM provides extensive coefficient statistics and model statistics, as well as row diagnostics. GLM also supports confidence bounds.

- **Naive Bayes**

  Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

- **Random Forest**

Random Forest is a powerful and popular machine learning algorithm that brings significant performance and scalability benefits.

- **Support Vector Machine**

  Support Vector Machine (SVM) is a powerful, state-of-the-art algorithm based on linear and nonlinear regression. OML4SQL implements SVM for binary and multiclass classification.

- **XGBoost**
  XGBoost is machine learning algorithm for regression and classification that makes available the XGBoost open source package. Oracle Machine Learning for SQL XGBoost prepares training data, invokes XGBoost, builds and persists a model, and applies the model for prediction.

> **Note:**
>
> OML4SQL uses Naive Bayes as the default classification algorithm.

Related Topics

- Decision Tree
  Oracle Machine Learning for SQL supports Decision Tree as one of the classification algorithms. This chapter provides an overview of the Decision Tree algorithm.
- Explicit Semantic Analysis
  Learn how to use Explicit Semantic Analysis (ESA) as an unsupervised algorithm for feature extraction function and as a supervised algorithm for classification.
- Generalized Linear Model
  Learn how to use Generalized Linear Model (GLM) statistical technique for linear modeling.
- Multivariate State Estimation Technique - Sequential Probability Ratio Test
  The Multivariate State Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT) algorithm monitors critical processes and detects subtle anomalies.
- Naive Bayes
  Learn how to use the Naive Bayes classification algorithm.
- Random Forest
  Learn how to use Random Forest as a classification algorithm.
- Support Vector Machine
  Learn how to use Support Vector Machine (SVM), a powerful algorithm based on statistical learning theory.
- XGBoost
  XGBoost is highly-efficient, scalable machine learning algorithm for regression and classification that makes available the XGBoost Gradient Boosting open source package.

6
Anomaly Detection

Learn how to detect rare cases in the data through anomaly detection - an unsupervised function.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

> **See Also:**
>
> - Campos, M.M., Milenova, B.L., Yarmus, J.S., "Creation and Deployment of Data Mining-Based Intrusion Detection Systems in Oracle Database 10*g*"
> - K. C. Gross, V. Bhardwaj and R. Bickford, "Proactive detection of software aging mechanisms in performance critical computers," 27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings., Greenbelt, MD, USA, 2002, pp. 17-23, doi: 10.1109/SEW.2002.1199445.

6.1 About Anomaly Detection

The goal of anomaly detection is to identify items, events, or observations that are unusual within data that is seemingly 'normal'. This data may consist of traditional enterprise data or Internet of Things (IoT) sensor data.

Anomaly detection is an important tool for detecting, for example, fraud, network intrusions, enterprise computing service interruptions, sensor time series prognostics, and other rare events that can have great significance but are hard to find. Anomaly detection can be used to solve problems like the following:

- A law enforcement agency compiles data about illegal activities, but nothing about legitimate activities. How can a suspicious activity be flagged?

  The law enforcement data is all of one class. There are no counter-examples.
- An insurance agency processes millions of insurance claims, knowing that a very small number are fraudulent. How can the fraudulent claims be identified?

The claims data contains very few counter-examples. They are outliers.

- An IT department encounters compute resource performance anomalies. How can such anomalies be detected along with their source causes, such as resource-contention issues and complex memory leaks?

  The data contains sensor output from thousands of sensors.

- An oil and gas enterprise or utility company requires proactive maintenance of business-critical assets, such as oil rigs or smart meters, to reduce operations and maintenance costs, improve up-time of revenue-generating assets, and improve safety margins for life-critical systems.

6.1.1 Anomaly Detection as a form of One-Class Classification

Learn about anomaly detection as one-class classification in training data.

When applied to traditional data, anomaly detection can be viewed as a form of one-class classification, because ideally only one class is represented in the training data. An anomaly detection model predicts whether a data point is typical for a given distribution or not. An atypical data point can be either an outlier or an example of a previously unseen class.

Normally, a classification model must be trained on data that includes both examples and counterexamples for each class so that the model can learn to distinguish between them. For example, a model that predicts the side effects of a medication must be trained on data that includes a wide range of responses to the medication.

A one-class classifier develops a profile that generally describes a typical case in the training data. Deviation from the profile is identified as an anomaly. One-class classifiers are sometimes referred to as positive security models, because they seek to identify "good" behaviors and assume that all other behaviors are bad.

In single-class data, all the cases have the same classification. Counterexamples, instances of another class, are hard to specify or expensive to collect. For instance, in text document classification, it is easy to classify a document under a given topic. However, the universe of documents outside of this topic can be very large and diverse. Thus, it is not feasible to specify other types of documents as counterexamples. Anomaly detection can be used to find unusual instances of a particular type of document.

> **Note:**
>
> Solving a one-class classification problem can be difficult. The accuracy of one-class classifiers cannot usually match the accuracy of standard classifiers built with meaningful counter examples.
>
> The goal of this type of anomaly detection is to provide some useful information where no information was previously attainable. However, if there are enough of the "rare" cases so that stratified sampling produces a training set with enough counterexamples for a standard classification model, then the classification may be a better solution.

Related Topics

- About Classification
  Classification is a machine learning function that assigns items in a collection to target categories or classes.

6.1.2 Anomaly Detection for Time Series Data

Learn about anomaly detection in IoT sensor data.

With the growing number of sensors in the internet of things, the ability to identify anomalous events among potentially thousands of sensors is essential. For example, in the early detection of anomalies in business-critical enterprise computing servers and software systems, storage systems, and networks.

Enterprises require high anomaly detection accuracy, which implies lower false-alarm probabilities, lower missed-alarm probabilities, and lower overhead compute cost. The ability to distinguish between a real problem and sensor malfunction can significantly reduce costs in problem solution.

Building a model involves supplying historical, error-free operating data from, for example, monitored equipment. The resulting model is used to score new sensor data, also referred to as the *monitoring phase*, to estimate the expected sensor values.

6.2 Anomaly Detection Algorithms

For anomaly detection, Oracle Machine Learning for SQL has the following algorithms.

- Multivariate state Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT)
- One-Class Support Vector Machine (SVM)

Anomaly detection is a form of classification. When you create a model using the MSET-SPRT and One-Class SVM algorithms, specify the classification machine learning function. These algorithms do not use a target.

Related Topics

- Multivariate State Estimation Technique - Sequential Probability Ratio Test
  The Multivariate State Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT) algorithm monitors critical processes and detects subtle anomalies.
- One-Class SVM
  Support Vector Machine (SVM) as a one-class classifier is used for detecting anomalies.

7
Ranking

Ranking is a regression machine learning technique.

7.1 About Ranking

Ranking is a machine learning technique to rank items.

Ranking is useful for many applications in information retrieval such as e-commerce, social networks, recommendation systems, and so on. For example, a user searches for an article or an item to buy online. To build a recommendation system, it becomes important that similar articles or items of relevance appear to the user such that the user clicks or purchases the item. A simple regression model can predict the probability of a user to click an article or buy an item. However, it is more practical to use ranking technique and be able to order or rank the articles or items to maximize the chances of getting a click or purchase. The prioritization of the articles or the items influence the decision of the users.

The ranking technique directly ranks items by training a model to predict the ranking of one item over another item. In the training model, it is possible to have items, ranking one over the other by having a "score" for each item. Higher ranked items have higher scores and lower ranked items have lower scores. Using these scores, a model is built to predict which item ranks higher than the other.

7.2 Ranking Methods

Oracle Machine Learning supports pairwise and listwise ranking methods through XGBoost.

For a training data set, in a number of sets, each set consists of objects and labels representing their ranking. A ranking function is constructed by minimizing a certain loss function on the training data. Using test data, the ranking function is applied to get a ranked list of objects. Ranking is enabled for XGBoost using the regression function. OML4SQL supports pairwise and listwise ranking methods through XGBoost.

Pairwise ranking: This approach regards a pair of objects as the learning instance. The pairs and lists are defined by supplying the same `case_id` value. Given a pair of objects, this approach gives an optimal ordering for that pair. Pairwise losses are defined by the order of the two objects. In OML4SQL, the algorithm uses LambdaMART to perform pairwise ranking with the goal of minimizing the average number of inversions in ranking.

Listwise ranking: This approach takes multiple lists of ranked objects as learning instance. The items in a list must have the same `case_id`. The algorithm uses LambdaMART to perform list-wise ranking.

See Also:

- "Ranking Measures and Loss Functions in Learning to Rank" a research paper presentation at https://www.researchgate.net/
- *Oracle Database PL/SQL Packages and Types Reference* for a listing and explanation of the available model settings for XGBoost.

Note:

The term hyperparameter is also interchangeably used for model setting.

Related Topics

- XGBoost
  XGBoost is highly-efficient, scalable machine learning algorithm for regression and classification that makes available the XGBoost Gradient Boosting open source package.
- DBMS_DATA_MINING — Algorithm Settings: XGBoost

7.3 Ranking Algorithms

Ranking falls under the Regression function.

OML4SQL supports XGBoost algorithm for ranking.

Related Topics

- XGBoost
  XGBoost is highly-efficient, scalable machine learning algorithm for regression and classification that makes available the XGBoost Gradient Boosting open source package.

8
Clustering

Learn how to discover natural groupings in the data through clustering - the unsupervised machine learning function.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

8.1 About Clustering

Clustering analysis finds clusters of data objects that are similar to one another.

The members of a cluster are more like each other than they are like members of other clusters. Different clusters can have members in common. The goal of clustering analysis is to find high-quality clusters such that the inter-cluster similarity is low and the intra-cluster similarity is high.

Clustering, like classification, is used to segment the data. Unlike classification, clustering models segment data into groups that were not previously defined. Classification models segment data by assigning it to previously-defined classes, which are specified in a target. Clustering models do not use a target.

Clustering is useful for exploring data. You can use clustering algorithms to find natural groupings when there are many cases and no obvious groupings.

Clustering can serve as a useful data-preprocessing step to identify homogeneous groups on which you can build supervised models.

You can also use clustering for anomaly detection. Once you segment the data into clusters, you find that some cases do not fit well into any clusters. These cases are anomalies or outliers.

8.1.1 How are Clusters Computed?

There are several different approaches to the computation of clusters. Oracle Machine Learning for SQL supports the methods listed here.

- **Density-based**: This type of clustering finds the underlying distribution of the data and estimates how areas of high density in the data correspond to peaks in the distribution. High-density areas are interpreted as clusters. Density-based cluster estimation is probabilistic.
- **Distance-based**: This type of clustering uses a distance metric to determine similarity between data objects. The distance metric measures the distance between actual cases in the cluster and the prototypical case for the cluster. The prototypical case is known as the **centroid**.
- **Grid-based**: This type of clustering divides the input space into hyper-rectangular cells and identifies adjacent high-density cells to form clusters.

8.1.2 Scoring New Data

Although clustering is an unsupervised machine learning function, Oracle Machine Learning for SQL supports the scoring operation for clustering.

New data is scored probabilistically.

8.1.3 Hierarchical Clustering

Oracle Machine Learning for SQL supports clustering algorithms that perform hierarchical clustering.

The leaf clusters are the final clusters generated by the algorithm. Clusters higher up in the hierarchy are intermediate clusters.

8.1.3.1 Rules

Rules describe the data in each cluster.

A rule is a conditional statement that captures the logic used to split a parent cluster into child clusters. A rule describes the conditions for a case to be assigned with some probability to a cluster.

8.1.3.2 Support and Confidence

Support and **confidence** are metrics that describe the relationships between clustering rules and cases.

Support is the percentage of cases for which the rule holds. Confidence is the probability that a case described by this rule is actually assigned to the cluster.

8.2 Evaluating a Clustering Model

Since known classes are not used in clustering, the interpretation of clusters can present difficulties. How do you know if the clusters can reliably be used for business decision making?

Oracle Machine Learning for SQL clustering models support a high degree of model transparency. You can evaluate the model by examining information generated by the clustering algorithm: for example, the centroid of a distance-based cluster. Moreover, because the clustering process is hierarchical, you can evaluate the rules and other information related to each cluster's position in the hierarchy.

8.3 Clustering Algorithms

Learn different clustering algorithms used in Oracle Machine Learning for SQL.

Oracle Machine Learning for SQL supports these clustering algorithms:

- **Expectation Maximization**

  Expectation Maximization is a probabilistic, density-estimation clustering algorithm.

- **k-Means**

  k-Means is a distance-based clustering algorithm. OML4SQL supports an enhanced version of *k*-Means.

- **Orthogonal Partitioning Clustering (O-Cluster)**

  O-Cluster is a proprietary, grid-based clustering algorithm.

> **See Also:**
>
> Campos, M.M., Milenova, B.L., "O-Cluster: Scalable Clustering of Large High Dimensional Data Sets", Oracle Data Mining Technologies, 10 Van De Graaff Drive, Burlington, MA 01803.

The main characteristics of the two algorithms are compared in the following table.

Table 8-1 Clustering Algorithms Compared

Feature	k-Means	O-Cluster	Expectation Maximization
Clustering methodolgy	Distance-based	Grid-based	Distribution-based
Number of cases	Handles data sets of any size	More appropriate for data sets that have more than 500 cases. Handles large tables through active sampling	Handles data sets of any size
Number of attributes	More appropriate for data sets with a low number of attributes	More appropriate for data sets with a high number of attributes	Appropriate for data sets with many or few attributes
Number of clusters	User-specified	Automatically determined	Automatically determined
Hierarchical clustering	Yes	Yes	Yes
Probabilistic cluster assignment	Yes	Yes	Yes

> **Note:**
>
> OML4SQL uses *k*-Means as the default clustering algorithm.

Related Topics

- Oracle Machine Learning for SQL
- Expectation Maximization
  Learn how to use expectation maximization clustering algorithm.
- *k*-Means
  Oracle Machine Learning for SQL supports enhanced *k*-Means clustering algorithm. Learn how to use the algorithm.
- O-Cluster
  Learn how to use orthogonal partitioning clustering (O-Cluster), an Oracle-proprietary clustering algorithm.

9
Association

Learn how to discover association rules through association - an unsupervised machine learning function.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

9.1 About Association

Association is a Oracle Machine Learning for SQL function that discovers the probability of the co-occurrence of items in a collection.

The relationships between co-occurring items are expressed as **Association Rules**.

9.1.1 Association Rules

Identifies the pattern of association within the data.

The results of an association model are the rules that identify patterns of association within the data. Oracle Machine Learning for SQL does not support the scoring operation for association modeling.

Association rules can be applied as follows:

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

9.1.2 Market-Basket Analysis

Association rules are often used to analyze sales transactions. For example, it is noted that customers who buy cereal at the grocery store often buy milk at the same time. In fact, association analysis find that 85% of the checkout sessions that include cereal also include milk. This relationship can be formulated as the following rule:

```
Cereal implies milk with 85% confidence
```

This application of association modeling is called **market-basket analysis**. It is valuable for direct marketing, sales promotions, and for discovering business trends. Market-basket analysis can also be used effectively for store layout, catalog design, and cross-sell.

9.1.3 Association Rules and eCommerce

Learn about application of association rules in other domains.

Association modeling has important applications in other domains as well. For example, in e-commerce applications, association rules may be used for Web page personalization. An association model might find that a user who visits pages A and B is 70% likely to also visit page C in the same session. Based on this rule, a dynamic link can be created for users who are likely to be interested in page C. The association rule is expressed as follows:

```
A and B imply C with 70% confidence
```

Related Topics

- Confidence
  The confidence of a rule indicates the probability of both the antecedent and the consequent appearing in the same transaction.

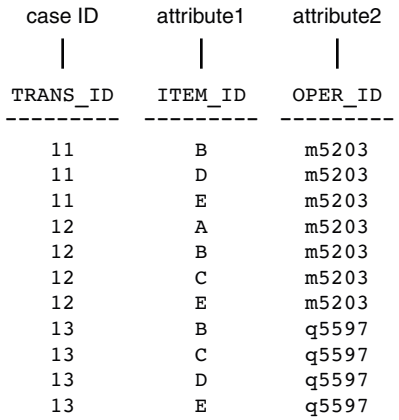
9.2 Transactional Data

Learn about transactional data, also known as market-basket data.

Unlike other machine learning functions, association is transaction-based. In transaction processing, a case includes a collection of items such as the contents of a market basket at the checkout counter. The collection of items in the transaction is an attribute of the transaction. Other attributes might be a timestamp or user ID associated with the transaction.

Transactional data, also known as **market-basket data**, is said to be in **multi-record case** format because a set of records (rows) constitute a case. For example, in the following figure, case 11 is made up of three rows while cases 12 and 13 are each made up of four rows.

Figure 9-1 Transactional Data



Non transactional data is said to be in a **single-record case** format because a single record (row) constitutes a case. In Oracle Machine Learning for SQL, association models can be built using either transactional or non transactional or two-dimensional data formats. If the data is non transactional, it is possible to transform to a nested column to make it transactional before association machine learning activities can be performed. Transactional format is the usual format but, the association rules model does accept two-dimensional input format. For non transactional input format, each distinct combination of the content in all columns other than the case ID column is treated as a unique item.

Related Topics

- *Oracle Machine Learning for SQL User's Guide*
- Data Preparation for Apriori

9.3 Association Algorithm

Oracle Machine Learning for SQL uses the Apriori algorithm to calculate association rules for items in frequent itemsets.

Related Topics

- Apriori
  Learn how to calculate association rules using the Apriori algorithm.

10
Feature Selection

Learn how to perform feature selection and attribute importance.

Oracle Machine Learning for SQL supports attribute importance as a supervised and unsurpervised machine learning function .

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

10.1 Finding the Attributes

Find the attributes by using preprocessing steps to reduce the effect of noise, correlation, and high-dimensionality.

Sometimes too much information can reduce the effectiveness of OML4SQL. Some of the columns of data attributes assembled for building and testing a model in a supervised learning do not contribute meaningful information to the model. Some actually detract from the quality and accuracy of the model.

For example, you want to collect a great deal of data about a given population because you want to predict the likelihood of a certain illness within this group. Some of this information, perhaps much of it, has little or no effect on susceptibility to the illness. It is possible that attributes such as the number of cars per household do not have effect whatsoever.

Irrelevant attributes add noise to the data and can affect model accuracy. Noise increases the size of the model and the time and system resources needed for model building and scoring.

Data sets with many attributes can contain groups of attributes that are correlated. These attributes actually measure the same underlying feature. Their presence together in the build data can skew the patterns found by algorithm and affect the accuracy of the model.

Wide data (many attributes) typically results in more processing by machine learning algorithms. Model attributes are the dimensions of the processing space used by the algorithm. The higher the dimensionality of the processing space, the higher the computation cost involved in algorithmic processing.

To minimize the effects of noise, correlation, and high dimensionality, some form of dimension reduction is often a desirable preprocessing step. Feature selection

involves identifying those attributes that are most predictive and selecting among those to provide the algorithm for model building. By removing attributes that add little or no value to a model has these benefits - potentially increasing model accuracy while reducing compute time since fewer attributes need to be processed. Informative and representative samples are best suited in feature selection. Sometimes you can represent the variables that are important than to represent the linear combination of variables. You can single-out and measure the "importance" of a column or a row in a data matrix in an unsupervised manner (a low-rank matrix decomposition).

Feature selection optimization is performed in the Decision Tree algorithm and within Naive Bayes as an algorithm behavior. The Generalized Linear Model (GLM) algorithm can be configured to perform feature selection through model setting.

10.2 About Feature Selection and Attribute Importance

Finding the most significant predictors is the goal of some machine learning projects. For example, a model might seek to find the principal characteristics of clients who pose a high credit risk.

Oracle Machine Learning for SQL supports the **attribute importance** machine learning function, which ranks attributes according to their importance. Attribute importance does not actually select the features, but ranks them as to their relevance to predicting the result. It is up to the user to review the ranked features and create a data set to include the desired features.

Feature selection is useful as a preprocessing step to improve computational efficiency in predictive modeling.

10.2.1 Attribute Importance and Scoring

The results of attribute importance are the attributes of the build data ranked according to their influence.

The ranking and the measure of importance can be used in selecting training data for classification and regression models. Also, used for selecting data for unsupervised algorithm like CUR matrix decomposition. Oracle Machine Learning for SQL does not support the scoring operation for attribute importance.

10.3 Algorithms for Attribute Importance

Understand the algorithms used for attribute importance.

Oracle Machine Learning for SQL supports the following algorithms for attribute importance:

- Minimum Description Length
- CUR Matrix Decomposition

Related Topics

- CUR Matrix Decomposition
  Learn how to use CUR decomposition based algorithm for attribute importance.
- Minimum Description Length
  Learn how to use Minimum Description Length, the supervised technique for calculating attribute importance.

11
Feature Extraction

Learn how to perform attribute reduction using feature extraction as an unsupervised function.

Oracle Machine Learning for SQL supports feature extraction as an unsupervised machine learning function.

Related Topics

- Oracle Machine Learning Basics
  Understand the basic concepts of Oracle Machine Learning.

11.1 About Feature Extraction

Feature extraction is a dimensionality reduction technique. Unlike feature selection, which selects and retains the most significant attributes, feature extraction actually transforms the attributes. The transformed attributes, or **features**, are linear combinations of the original attributes.

The feature extraction technique results in a much smaller and richer set of attributes. The maximum number of features can be user-specified or determined by the algorithm. By default, the algorithm determines it.

Models built on extracted features can be of higher quality, because fewer and more meaningful attributes describe the data.

Feature extraction projects a data set with higher dimensionality onto a smaller number of dimensions. As such it is useful for data visualization, since a complex data set can be effectively visualized when it is reduced to two or three dimensions.

Some applications of feature extraction are latent semantic analysis, data compression, data decomposition and projection, and pattern recognition. Feature extraction can also be used to enhance the speed and effectiveness of machine learning algorithms.

Feature extraction can be used to extract the themes of a document collection, where documents are represented by a set of key words and their frequencies. Each theme (feature) is represented by a combination of keywords. The documents in the collection can then be expressed in terms of the discovered themes.

11.1.1 Feature Extraction and Scoring

Oracle Machine Learning for SQL supports the scoring operation for feature extraction. As an unsupervised machine learning function, feature extraction does not involve a target. When applied, a feature extraction model transforms the input into a set of features.

11.2 Algorithms for Feature Extraction

Understand the algorithms used for feature extraction.

OML4SQL supports these feature extraction algorithms:

- **Explicit Semantic Analysis** (ESA).
- **Non-Negative Matrix Factorization** (NMF).
- **Singular Value Decomposition** (SVD) and **Prediction Component Analysis** (PCA).

> **Note:**
>
> OML4SQL uses NMF as the default feature extraction algorithm.

Related Topics

- Explicit Semantic Analysis
  Learn how to use Explicit Semantic Analysis (ESA) as an unsupervised algorithm for feature extraction function and as a supervised algorithm for classification.
- Non-Negative Matrix Factorization
  Learn how to use Non-Negative Matrix Factorization (NMF), an unsupervised algorithm, that Oracle Machine Learning for SQL uses for feature extraction.
- Singular Value Decomposition
  Learn how to use Singular Value Decomposition, an unsupervised algorithm for feature extraction.
- PCA scoring
  Learn about configuring Singular Value Decomposition (SVD) to perform Principal Component Analysis (PCA) projections.

12
Row Importance

Row importance is an unsupervised machine learning technique that can be applied to data as a preprocessing step prior to model building using other mining functions and algorithms.

12.1 About Row Importance

Row importance captures the influence of the rows or cases in a data set.

Row importance technique is used in dimensionality reduction of large data sets. Row importance identifies the most influential rows of the data matrix. The rows with high importance are ranked by their importance scores. The "importance" of a row is determined by high statistical leverage scores. In CUR matrix decomposition, row importance is often combined with column (attribute) importance. Row importance can serve as a data preprocessing step prior to model building using regression, classification, and clustering.

Related Topics

- CUR Matrix Decomposition
  Learn how to use CUR decomposition based algorithm for attribute importance.
- Statistical Leverage Score
  Leverage scores are statistics that determine which column (or rows) are most representative with respect to a rank subspace of a matrix. The statistical leverage scores represent the column (or attribute) and row importance.
- CUR Matrix Decomposition Algorithm Configuration
  Configure the CUR Matrix Decomposition algorithm setting to build your model.

12.2 Row Importance Algorithms

Oracle Machine Learning for SQL supports CUR matrix decomposition algorithm for row and column (attribute) importance.

Popular algorithms for dimensionality reduction are Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and CUR Matrix Decomposition. All these algorithms apply low-rank matrix decomposition.

In CUR matrix decomposition, the attributes include 2-Dimensional numerical columns, levels of exploded 2D categorical columns, and attribute name or subname or value pairs for nested columns. To arrive at row importance or selection, the algorithm computes singular vectors, calculates leverage scores, and then selects rows. Row importance is performed when users specify `CURS_ROW_IMP_ENABLE` for

the `CURS_ROW_IMPORTANCE` parameter in the settings table and the `case_id` column is present. Unless users explicitly specify, row importance is not performed.

Related Topics

- Singular Value Decomposition
  Learn how to use Singular Value Decomposition, an unsupervised algorithm for feature extraction.
- CUR Matrix Decomposition
  Learn how to use CUR decomposition based algorithm for attribute importance.

13
Time Series

Learn about time series as an Oracle Machine Learning for SQL regression function.

13.1 About Time Series

Time series is a machine learning function that forecasts target value based solely on a known history of target values. It is a specialized form of regression, known in the literature as auto-regressive modeling.

The input to time series analysis is a sequence of target values. A case id column specifies the order of the sequence. The case id can be of type `NUMBER` or a date type (date, datetime, timestamp with timezone, or timestamp with local timezone). Regardless of case id type, the user can request that the model include trend, seasonal effects or both in its forecast computation. When the case id is a date type, the user must specify a time interval (for example, month) over which the target values are to be aggregated, along with an aggregation procedure (for example, sum). Aggregation is performed by the algorithm prior to constructing the model.

The time series model provide estimates of the target value for each step of a time window that can include up to 30 steps beyond the historical data. Like other regression models, time series models compute various statistics that measure the goodness of fit to historical data.

Forecasting is a critical component of business and governmental decision making. It has applications at the strategic, tactical and operation level. The following are the applications of forecasting:

- Projecting return on investment, including growth and the strategic effect of innovations
- Addressing tactical issues such as projecting costs, inventory requirements and customer satisfaction
- Setting operational targets and predicting quality and conformance with standards

Related Topics

- Regression
  Learn how to predict a continuous numerical target through regression - the supervised machine learning function.

13.2 Choosing a Time Series Model

Learn how to select a time series model.

Time series data may contain patterns that can affect predictive accuracy. For example, during a period of economic growth, there may be an upward trend in sales. Sales may increase in specific seasons (bathing suits in summer). To accommodate such series, it can be useful to choose a model that incorporates trend, seasonal effects, or both.

Trend can be difficult to estimate, when you must represent trend by a single constant. For example, if there is a grow rate of 10%, then after 7 steps, the value doubles. Local growth rates, appropriate to a few time steps can easily approach such levels, but thereafter drop. **Damped trend** models can more accurately represent such data, by reducing cumulative trend effects. Damped trend models can better represent variability in trend effects over the historical data. Damped trend models are a good choice when the data have significant, but variable trend.

Since modeling attempts to reduce error, how error is measured can affect model predictions. For example, data that exhibit a wide range of values may be better represented by error as fraction of level. An error of a few hundred feet in the measurement of the height of a mountain may be equivalent to an error of an inch or two in the measurement of the height of a child. Errors that are measured relative to value are called **multiplicative errors**. Errors that are the same across values are called **additive errors**. If there are multiplicative effects in the model, then the error type is multiplicative. If there are no explicit multiplicative effects, error type is left to user specification. The type need not be the same across individual effects. For example, trend can be additive while seasonality is multiplicative. This particular mixed type effect combination defines the popular Holt-Winters model.

> **Note:**
>
> Multiplicative error is not an appropriate choice for data that contain zeros or negative values. Thus, when the data contains such values, it is best not to choose a model with multiplicative effects or to set error type to be multiplicative.

13.3 Time Series Statistics

Learn to evaluate model quality by applying commonly used statistics.

As with other regression functions, there are commonly used statistics for evaluating the overall model quality. An expert user can also specify one of these figures of merit as criterion to optimize by the model build process. Choosing an optimization criterion is not required because model-specific defaults are available.

13.3.1 Conditional Log-Likelihood

Log-likelihood is a figure of merit often used as an optimization criterion for models that provide probability estimates for predictions which depend on the values of the model's parameters.

The model probability estimates for the actual values in the training data then yields an estimate of the likelihood of the parameter values. Parameter values that yield high probabilities for the observed target values have high likelihood, and therefore indicate a good model. The calculation of log-likelihood depends on the form of the model.

Conditional log-likelihood breaks the parameters into two groups. One group is assumed to be correct and the other is assumed the source of any errors. Conditional log-likelihood is the log-likelihood of the latter group conditioned on the former group. For example, Exponential Smoothing (ESM) models make an estimate of the initial model state. The conditional log-likelihood of an ESM model is conditional on that initial model state (assumed to be correct). The ESM conditional log-likelihood is as follows:

$$L^*(\theta, X_0) = n \ln\left(\sum_{t=1}^{n} e_t^2 / k^2(x_{t-1})\right) + 2\sum_{t=1}^{n} \ln|k(x_{t-1})|$$

where e_t is the error at time `t` and `k(x(t-1) )` is `1` for ESM models with additive errors and is the estimated level at the previous time step in models with multiplicative error.

13.3.2 Mean Square Error (MSE) and Other Error Measures

Another time series figure of merit, that can also be used as an optimization criterion, is Mean Square Error (MSE).

The mean square error is computed as:

$$MSE = \sum_{t=1}^{n} {e_t}^2 / n$$

where the error at time *t* is the difference between the actual and model one step ahead forecast value at time *t* for models with additive error and that difference divided by the one-step ahead forecast for models with multiplicative error.

> **Note:**
>
> These "forecasts" are for over periods already observed and part of the input time series.

Since time series models can forecast for each of multiple steps ahead, time series can measure the error associated with such forecasts. Average Mean Square Error (AMSE), another figure of merit, does exactly that. For each period in the input time series, it computes a multi-step forecast, computes the error of those forecasts and averages the errors. AMSE computes the individual errors exactly as MSE does taking cognizance of error type (additive or multiplicative). The number of steps, *k*, is determined by the user (default 3). The formula is as follows:

$$AMSE = \sum_{t=1}^{n}\left(\sum_{i=0}^{k-1} e_{t+i}^{2}/k\right)/n$$

Other figure of merit relatives of MSE include the Residual Standard Error (RMSE), which is the square root of MSE, and the Mean Absolute Error (MAE) which is the average of the absolute value of the errors.

13.3.3 Irregular Time Series

Irregular time series are time series data where the time intervals between observed values are not equally spaced.

One common practice is for the time intervals between adjacent steps to be equally spaced. However, it is not always convenient or realistic to force such spacing on time series. Irregular time series do not make the assumption that time series are equally spaced, but instead use the case id's date and time values to compute the intervals between observed values. Models are constructed directly on the observed values with their observed spacing. Oracle time series analysis handles irregular time series.

13.3.4 Build and Apply

A new time series model is built when new data arrives.

Many of the Oracle Machine Learning for SQL functions have separate build and apply operations, because you can construct and potentially apply a model to many different sets of input data. However, time series input consists of the target value history only. Thus, there is only one set of appropriate input data. When new data arrive, good practice dictates that a new model be built. Since the model is only intended to be used once, the model statistics and forecasts are produced during model build and are available through the model views.

13.4 Time Series Algorithm

Oracle Machine Learning for SQL uses the Exponential Smoothing algorithm to forecast from time series data.

Related Topics

- Exponential Smoothing
  Learn about the Exponential Smoothing algorithm.

Part III
Algorithms

Oracle Machine Learning for SQL supports the algorithms listed in Part III. Part III provides basic conceptual information about the algorithms. There is at least one algorithm for each of the machine learning functions.

Part III contains these chapters:

- Apriori
- CUR Matrix Decomposition
- Decision Tree
- Expectation Maximization
- Explicit Semantic Analysis
- Exponential Smoothing
- Generalized Linear Model
- k-Means
- Minimum Description Length
- Multivariate State Estimation Technique - Sequential Probability Ratio Test
- Naive Bayes
- Neural Network
- Non-Negative Matrix Factorization
- O-Cluster
- R Extensibility
- Random Forest
- Singular Value Decomposition
- Support Vector Machine
- XGBoost

Related Topics

- Machine Learning Functions
  Part II provides basic conceptual information about machine learning functions that the Oracle Machine Learning for SQL supports.

14
Apriori

Learn how to calculate association rules using the Apriori algorithm.

Related Topics

- Association
  Learn how to discover association rules through association - an unsupervised machine learning function.
- DBMS_DATA_MINING - Model Settings
- Machine Learning Function Settings
- Automatic Data Preparation
- Model Detail Views for Association Rules
- OML4SQL Examples
- OML4R Association Rules Example
- OML4R Code Examples

14.1 About Apriori

Learn how to find associations involving rare events in a large number of items using Apriori.

An association machine learning problem can be decomposed into the following subproblems:

- Find all combinations of items in a set of transactions that occur with a specified minimum frequency. These combinations are called **frequent itemsets**.
- Calculate rules that express the probable co-occurrence of items within frequent itemsets.

Apriori calculates the probability of an item being present in a frequent itemset, given that another item or items is present.

Association rule machine learning is not recommended for finding associations involving rare events in problem domains with a large number of items. Apriori discovers patterns with frequencies above the minimum support threshold. Therefore, to find associations involving rare events, the algorithm must run with very low minimum support values. However, doing so potentially explodes the number of

enumerated itemsets, especially in cases with a large number of items. This increases the execution time significantly. Classification or anomaly detection is more suitable for discovering rare events when the data has a high number of attributes.

The build process for Apriori supports parallel execution.

Related Topics

- Example: Calculating Rules from Frequent Itemsets
  Example to calculating rules from frequent itemsets.
- *Oracle Database VLDB and Partitioning Guide*

14.2 Association Rules and Frequent Itemsets

The Apriori algorithm calculates rules that express probabilistic relationships between items in frequent itemsets. For example, a rule derived from frequent itemsets containing A, B, and C might state that if A and B are included in a transaction, then C is likely to also be included.

An association rule states that an item or group of items implies the presence of another item with some probability. Unlike decision tree rules, which predict a target, association rules express correlation.

14.2.1 Antecedent and Consequent

Defines antecedent and consequent in an Apriori algorithm.

The IF component of an association rule is known as the **antecedent**. The THEN component is known as the **consequent**. The antecedent and the consequent are disjoint; they have no items in common.

Oracle Machine Learning for SQL supports association rules that have one or more items in the antecedent and a single item in the consequent.

14.2.2 Confidence

Rules have an associated confidence, which is the conditional probability that the consequent occurs given the occurrence of the antecedent. You can specify the minimum confidence for rules.

14.3 Data Preparation for Apriori

Association models are designed to use transactional data. In transactional data, there is a one-to-many relationship between the case identifier and the values for each case. Each case ID/value pair is specified in a separate record (row).

14.3.1 Native Transactional Data and Star Schemas

Learn about storage format of transactional data.

Transactional data may be stored in native transactional format, with a non-unique case ID column and a values column, or it may be stored in some other configuration, such as a star schema. If the data is not stored in native transactional format, it must be transformed to a nested column for processing by the Apriori algorithm.

Related Topics

- Transactional Data
  Learn about transactional data, also known as market-basket data.
- *Oracle Machine Learning for SQL User's Guide*

14.3.2 Items and Collections

In transactional data, a collection of items is associated with each case. The collection theoretically includes all possible members of the collection. For example, all products can theoretically be purchased in a single market-basket transaction. However, in actuality, only a tiny subset of all possible items are present in a given transaction; the items in the market-basket represent only a small fraction of the items available for sale in the store.

14.3.3 Sparse Data

Understand how sparse data is used in the Apriori algorithm.

Missing items in a collection indicate **sparsity**. Missing items may be present with a null value, or they may be missing.

Nulls in transactional data are assumed to represent values that are known but not present in the transaction. For example, three items out of hundreds of possible items might be purchased in a single transaction. The items that were not purchased are known but not present in the transaction.

Oracle Machine Learning for SQL assumes sparsity in transactional data. The Apriori algorithm is optimized for processing sparse data.

> **Note:**
>
> Apriori is not affected by Automatic Data Preparation.

Related Topics

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

14.3.4 Improved Sampling

Association rules (AR) can use a good sample size with performance guarantee, based on the work of Riondato and Upfal.

The AR algorithm computes the sample size by the following inputs:

- *d*-index of the dataset
- Absolute error ε
- Confidence level γ

d-index is defined as the maximum integer *d* such that the dataset contains at least *d* transactions of length *d* at the minimum. It is the upper bound of Vapnik-Chervonenkis (VC) dimension. The AR algorithm computes *d*-index of the dataset by scanning the length of all transactions in the dataset.

Users specify absolute error ε and confidence level γ parameters. A large d-index, small AR support, small ε or large γ can cause a large sample size. The sample size theoretically guarantees that the absolute error of both the support and confidence of the approximated AR (from sampling) is less than ε compared to the exact AR with probability (or confidence level) at least γ. In this document this sample size is called AR-specific sample size.

14.3.4.1 Sampling Implementation

The sample size is only computed when users turn on the sampling (`ODMS_SAMPLING` is set as `ODMS_SAMPLING_ENABLE`) and do not specify the sample size (`ODMS_SAMPLE_SIZE` is unspecified).

Usage Notes

1. If `ODMS_SAMPLING` is unspecified or set as `ODMS_SAMPLING_DISABLE`, the sampling is not performed for AR and the exact AR is obtained.
2. If `ODMS_SAMPLING` is set as `ODMS_SAMPLING_ENABLE` and if `ODMS_SAMPLE_SIZE` is specified as positive integer number then the user-specified sample size (`ODMS_SAMPLE_SIZE`) is utilized. The sampling is performed in the general data preparation stage before the AR algorithm. The AR-specific sample size is not computed. The approximated AR is obtained.
3. If `ODMS_SAMPLING` is set as `ODMS_SAMPLING_ENABLE` and `ODMS_SAMPLE_SIZE` is not specified, the AR-specified sample size is computed and then sampling is performed in the AR algorithm. The approximated AR is obtained.

> **Note:**
>
> If the computed AR-specific sample size is larger than or equal to the total transaction size in the dataset, the sampling is not performed and the exact AR is obtained.

If users do not have a good idea on the choice of sample size for AR, it is suggested to leave `ODMS_SAMPLE_SIZE` unspecified, only specify proper values for sampling parameters and let AR algorithm compute the suitable AR-specific sample size.

> **See Also:**
>
> DBMS_DATA_MINING — Machine Learning Function Settings for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

14.4 Calculating Association Rules

The first step in association analysis is the enumeration of **itemsets**. An itemset is any combination of two or more items in a transaction.

14.4.1 Itemsets

Learn about itemsets.

The maximum number of items in an itemset is user-specified. If the maximum is two, then all the item pairs are counted. If the maximum is greater than two, then all the item pairs, all the item triples, and all the item combinations up to the specified maximum are counted.

The following table shows the itemsets derived from the transactions shown in the following example, assuming that maximum number of items in an itemset is set to 3.

Table 14-1 Itemsets

Transaction	Itemsets
11	(B,D) (B,E) (D,E) (B,D,E)
12	(A,B) (A,C) (A,E) (B,C) (B,E) (C,E) (A,B,C) (A,B,E) (A,C,E) (B,C,E)
13	(B,C) (B,D) (B,E) (C,D) (C,E) (D,E) (B,C,D) (B,C,E) (B,D,E) (C,D,E)

Example 14-1 Sample Transactional Data

```
TRANS_ID   ITEM_ID
---------  -------------------
11         B
11         D
11         E
12         A
12         B
12         C
12         E
13         B
13         C
13         D
13         E
```

14.4.2 Frequent Itemsets

Learn about frequent itemsets and support.

Association rules are calculated from itemsets. If rules are generated from all possible itemsets, there can be a very high number of rules and the rules may not be very meaningful. Also, the model can take a long time to build. Typically it is desirable to only generate rules from itemsets that are well-represented in the data. **Frequent itemsets** are those that occur with a minimum frequency specified by the user.

The minimum frequent itemset **support** is a user-specified percentage that limits the number of itemsets used for association rules. An itemset must appear in at least this percentage of all the transactions if it is to be used as a basis for rules.

The following table shows the itemsets from Table 14-1 that are frequent itemsets with support > 66%.

Table 14-2 Frequent Itemsets

Frequent Itemset	Transactions	Support
(B,C)	2 of 3	67%
(B,D)	2 of 3	67%
(B,E)	3 of 3	100%
(C,E)	2 of 3	67%
(D,E)	2 of 3	67%
(B,C,E)	2 of 3	67%
(B,D,E)	2 of 3	67%

Related Topics

- Apriori
  Learn how to calculate association rules using the Apriori algorithm.

14.4.3 Example: Calculating Rules from Frequent Itemsets

Example to calculating rules from frequent itemsets.

The following tables show the itemsets and frequent itemsets that were calculated in "Association". The frequent itemsets are the itemsets that occur with a minimum support of 67%; at least 2 of the 3 transactions must include the itemset.

Table 14-3 Itemsets

Transaction	Itemsets
11	(B,D) (B,E) (D,E) (B,D,E)
12	(A,B) (A,C) (A,E) (B,C) (B,E) (C,E) (A,B,C) (A,B,E) (A,C,E) (B,C,E)
13	(B,C) (B,D) (B,E) (C,D) (C,E) (D,E) (B,C,D) (B,C,E) (B,D,E) (C,D,E)

Table 14-4 Frequent Itemsets with Minimum Support 67%

Itemset	Transactions	Support
(B,C)	12 and 13	67%
(B,D)	11 and 13	67%
(B,E)	11, 12, and 13	100%
(C,E)	12 and 13	67%
(D,E)	11 and 13	67%
(B,C,E)	12 and 13	67%
(B,D,E)	11 and 13	67%

A rule expresses a conditional probability. Confidence in a rule is calculated by dividing the probability of the items occurring together by the probability of the occurrence of the antecedent.

For example, if B (antecedent) is present, what is the chance that C (consequent) is also present? What is the confidence for the rule "IF B, THEN C"?

As shown in Table 14-3:

- All 3 transactions include B (3/3 or 100%)
- Only 2 transactions include both B and C (2/3 or 67%)
- Therefore, the confidence of the rule "IF B, THEN C" is 67/100 or 67%.

The following table the rules that can be derived from the frequent itemsets in Table 14-4.

Table 14-5 Frequent Itemsets and Rules

Frequent Itemset	Rules	prob(antecedent and consequent) / prob(antecedent)	Confidence
(B,C)	(If B then C) (If C then B)	67/100 67/67	67% 100%
(B,D)	(If B then D) (If D then B)	67/100 67/67	67% 100%
(B,E)	(If B then E) (If E then B)	100/100 100/100	100% 100%
(C,E)	(If C then E) (If E then C)	67/67 67/100	100% 67%
(D,E)	(If D then E) I(f E then D)	67/67 67/100	100% 67%
(B,C,E)	(If B and C then E) (If B and E then C) (If C and E then B)	67/67 67/100 67/67	100% 67% 100%
(B,D,E)	(If B and D then E) (If B and E then D) (If D and E then B)	67/67 67/100 67/67	100% 67% 100%

If the minimum confidence is 70%, ten rules are generated for these frequent itemsets. If the minimum confidence is 60%, sixteen rules are generated.

> **Tip:**
>
> Increase the minimum confidence if you want to decrease the build time for the model and generate fewer rules.

Related Topics

- Association
  Learn how to discover association rules through association - an unsupervised machine learning function.

14.4.4 Aggregates

Aggregates refer to the quantities associated with each item that the user opts for association rules model to aggregate.

There can be more than one aggregate. For example, the user can specify the model to aggregate both profit and quantity.

14.4.5 Example: Calculating Aggregates

This example shows how to calculate aggregates using the customer grocery purchase and profit data.

Calculating Aggregates for Grocery Store Data

Assume a grocery store has the following data:

Table 14-6 Grocery Store Data

Customer	Item A	Item B	Item C	Item D
Customer 1	Buys (Profit $5.00)	Buys (Profit $3.20)	Buys (Profit $12.00)	NA
Customer 2	Buys (Profit $4.00)	NA	Buys (Profit $4.20)	NA
Customer 3	Buys (Profit $3.00)	Buys (Profit $10.00)	Buys (Profit $14.00)	Buys (Profit $8.00)
Customer 4	Buys (Profit $2.00)	NA	NA	Buys (Profit $1.00)

The basket of each customer can be viewed as a transaction. The manager of the store is interested in not only the existence of certain association rules, but also in the aggregated profit if such rules exist.

In this example, one of the association rules can be (A, B)=>C for customer 1 and customer 3. Together with this rule, the store manager may want to know the following:

- The total profit of item A appearing in this rule
- The total profit of item B appearing in this rule
- The total profit for consequent C appearing in this rule
- The total profit of all items appearing in the rule

For this rule, the profit for item A is $5.00 + $3.00 = $8.00, for item B the profit is $3.20 + $10.00 = $13.20, for consequent C, the profit is $12.00 + $14.00 = $26.00, for the antecedent itemset (A, B) is $8.00 + $13.20 = $21.20. For the whole rule, the profit is $21.20 + $26.00 = $47.40.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

14.4.6 Including and Excluding Rules

Explains including rules and excluding rules used in association.

Including rules enables a user to provide a list of items such that at least one item from the list must appear in the rules that are returned. Excluding rules enables a user to provide a list of items such that no item from the list can appear in the rules that are returned.

> **Note:**
>
> Since each association rule includes both antecedent and consequent, a set of including or excluding rules can be specified for antecedent while another set of including or excluding rules can be specified for consequent. Including or excluding rules can also be defined for the association rule.

Related Topics

- *Oracle Machine Learning for SQL User's Guide*
- *Oracle Database PL/SQL Packages and Types Reference*

14.4.7 Performance Impact for Aggregates

Aggregate function requires more memory usage and longer execution time.

For each item, the user may supply several columns to aggregate. It requires more memory to buffer the extra data and more time to compute the aggregate values.

14.5 Evaluating Association Rules

Evaluate association rules by using support and confidence.

Minimum support and confidence are used to influence the build of an association model. Support and confidence are also the primary metrics for evaluating the quality of the rules generated by the model. Additionally, Oracle Machine Learning for SQL supports lift for association rules. These statistical measures can be used to rank the rules and hence the usefulness of the predictions.

14.5.1 Support

The support of a rule indicates how frequently the items in the rule occur together. For example, cereal and milk might appear together in 40% of the transactions. If so, the following rules each have a support of 40%:

```
cereal implies milk
milk implies cereal
```

Support is the ratio of transactions that include all the items in the antecedent and consequent to the number of total transactions.

Support can be expressed in probability notation as follows:

```
support(A implies B) = P(A, B)
```

14.5.2 Minimum Support Count

Minimum support count defines minimum threshold in transactions that each rule must satisfy.

When the number of transactions is unknown, the support percentage threshold parameter can be tricky to set appropriately. For this reason, support can also be expressed as a count of transactions, with the greater of the two thresholds being used to filter out infrequent itemsets. The default is `1` indicating that this criterion is not applied.

Related Topics

- Association Rules
  Identifies the pattern of association within the data.
- *Oracle Machine Learning for SQL User's Guide*
- Frequent Itemsets
  Learn about frequent itemsets and support.

14.5.3 Confidence

The confidence of a rule indicates the probability of both the antecedent and the consequent appearing in the same transaction.

Confidence is the conditional probability of the consequent given the antecedent. For example, cereal appears in 50 transactions; 40 of the 50 might also include milk. The rule confidence is:

```
cereal implies milk with 80% confidence
```

Confidence is the ratio of the rule support to the number of transactions that include the antecedent.

Confidence can be expressed in probability notation as follows.

```
confidence (A implies B) = P (B/A), which is equal to P(A, B) / P(A)
```

Related Topics

- Confidence
- Frequent Itemsets
  Learn about frequent itemsets and support.

14.5.4 Reverse Confidence

The reverse confidence of a rule is defined as the number of transactions in which the rule occurs divided by the number of transactions in which the consequent occurs.

Reverse confidence eliminates rules that occur because the consequent is frequent. The default is `0`.

Related Topics

- Confidence
- Example: Calculating Rules from Frequent Itemsets
  Example to calculating rules from frequent itemsets.
- *Oracle Machine Learning for SQL User's Guide*
- *Oracle Database PL/SQL Packages and Types Reference*

14.5.5 Lift

Both support and confidence must be used to determine if a rule is valid. However, there are times when both of these measures may be high, and yet still produce a rule that is not useful. For example:

```
Convenience store customers who buy orange juice also buy milk with
a 75% confidence.
The combination of milk and orange juice has a support of 30%.
```

This at first sounds like an excellent rule, and in most cases, it would be. It has high confidence and high support. However, what if convenience store customers in general buy milk 90% of the time? In that case, orange juice customers are actually *less* likely to buy milk than customers in general.

A third measure is needed to evaluate the quality of the rule. Lift indicates the strength of a rule over the random co-occurrence of the antecedent and the consequent, given their individual support. It provides information about the improvement, the increase in probability of the consequent given the antecedent. Lift is defined as follows.

```
(Rule Support) /(Support(Antecedent) * Support(Consequent))
```

This can also be defined as the confidence of the combination of items divided by the support of the consequent. So in our milk example, assuming that 40% of the customers buy orange juice, the improvement would be:

```
30% / (40% * 90%)
```

which is 0.83 – an improvement of less than 1.

Any rule with an improvement of less than 1 does not indicate a real cross-selling opportunity, no matter how high its support and confidence, because it actually offers less ability to predict a purchase than does random chance.

> **Tip:**
>
> Decrease the maximum rule length if you want to decrease the build time for the model and generate simpler rules.

Tip:

Increase the minimum support if you want to decrease the build time for the model and generate fewer rules.

15
CUR Matrix Decomposition

Learn how to use CUR decomposition based algorithm for attribute importance.

- About CUR Matrix Decomposition
- Singular Vectors
- Statistical Leverage Score
- Column (Attribute) Selection and Row Selection
- CUR Matrix Decomposition Algorithm Configuration

Related Topics

- Feature Selection
  Learn how to perform feature selection and attribute importance.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: CUR Matrix Decomposition
- Automatic Data Preparation
- Model Detail Views for CUR Matrix Decomposition
- OML4SQL Examples

15.1 About CUR Matrix Decomposition

CUR Matrix Decomposition is a low-rank matrix decomposition algorithm that is explicitly expressed in a small number of actual columns and/or actual rows of data matrix.

CUR Matrix Decomposition was developed as an alternative to Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). CUR Matrix Decomposition selects columns and rows that exhibit high **statistical leverage** or large **influence** from the data matrix. By implementing the CUR Matrix Decomposition algorithm, a small number of most important attributes and/or rows can be identified from the original data matrix. Therefore, CUR Matrix Decomposition is an important tool for exploratory data analysis. CUR Matrix Decomposition can be applied to a variety of areas and facilitates regression, classification, and clustering.

Related Topics

- Data Preparation for SVD
  Oracle Machine Learning for SQL implements Singular Value Decomposition (SVD) for numerical data and categorical data.

15.2 Singular Vectors

Singular Value Decomposition (SVD) is the first step in CUR Matrix Decomposition.

SVD returns left and right singular vectors for calculating column and row leverage scores. Perform SVD on the following matrix:

$A \varepsilon \mathbf{R}^{mxn}$

The matrix is factorized as follows:

$A=U\Sigma V^{T}$

where $U = [u^1\ u^2 \ldots u^m]$ and $V = [v^1\ v^2 \ldots v^n]$ are orthogonal matrices.

Σ is a diagonal m × n matrix with non-negative real numbers $\sigma 1, \ldots, \sigma_\rho$ on the diagonal, where $\rho = \min\{m,n\}$ and σ_ξ is the ξ^{th} singular value of A.

Let u^ξ and v^ξ be the ξ^{th} left and right singular vector of A, the j^{th} column of A can thus be approximated by the top k singular vectors and corresponding singular values as:

$$A^j \approx \sum_{\xi=1}^{k} \left(\sigma_\xi u^\xi\right) v_j^\xi$$

where v^ξ_j is the j^{th} coordinate of the ξ^{th} right singular vector.

15.3 Statistical Leverage Score

Leverage scores are statistics that determine which column (or rows) are most representative with respect to a rank subspace of a matrix. The statistical leverage scores represent the column (or attribute) and row importance.

The normalized statistical leverage scores for all columns are computed from the top k right singular vectors as follows:

$$\pi_j = \frac{1}{k}\sum_{\xi=1}^{k}(v_j^\xi)^2$$

where k is called rank parameter and $j = 1, \ldots, n$. Given that π_j>=0 and

$$\sum_{j=1}^{n}\pi_j = 1$$

, these scores form a probability distribution over the n columns.

Similarly, the normalized statistical leverage scores for all rows are computed from the top k left singular vectors as:

$$\pi_i' = \frac{1}{k}\sum_{\xi=1}^{k}(u_i^\xi)^2$$

where $i = 1,...,m$.

15.4 Column (Attribute) Selection and Row Selection

The CUR matrix decomposition in OML4SQL is designed for attribute and/or row importance. It returns attributes and rows with high importance that are ranked by their leverage (importance) scores. Column (Attribute) selection and row selection is the final stage in CUR Matrix Decomposition.

Attribute selection: Selects attributes with high leverage scores and reports their names, scores (as importance) and ranks (by importance).

Row selection: Selects rows with high leverage scores and reports their names, scores (as importance) and ranks (by importance).

1. CUR Matrix Decomposition first selects the j^{th} column (or attribute) of A with probability $p_j = \min \{1, c\pi_j\}$ for all $j \in \{1,...,n\}$
2. If users enable row selection, select i^{th} row of A with probability $p_i = \min \{1, r\pi_i\}$ for all $i \in \{1,...,m\}$
3. Report the name (or ID) and leverage score (as importance) for all selected attributes (if row importance is disabled) or for all selected attributes and rows (if row importance is enabled).

c is the approximated (or expected) number of columns that users want to select, and r is the approximated (or expected) number of rows that users want to select.

To realize column and row selections, you need to calculate the probability to select each column and row.

Calculate the probability for each column as follows:

$p_j = \min \{1, c\pi_j\}$

Calculate the probability for each row as follows:

$p_i = \min\{1, c\pi_i\}$.

A column or row is selected if the probability is greater than some threshold.

15.5 CUR Matrix Decomposition Algorithm Configuration

Configure the CUR Matrix Decomposition algorithm setting to build your model.

Create a model with the algorithm specific settings. Define the algorithm name as `ALGO_CUR_DECOMPOSITION` and mining function as `ATTRIBUTE_IMPORTANCE`.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: CUR Matrix Decomposition for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Row Selection

To use this feature, specify the row importance setting `CURS_ROW_IMPORTANCE` to `CURS_ROW_IMP_ENABLE`.

> **Note:**
>
> The row selection is performed only when users specify that row importance is enabled and the `CASE_ID` column is present.

16
Decision Tree

Oracle Machine Learning for SQL supports Decision Tree as one of the classification algorithms. This chapter provides an overview of the Decision Tree algorithm.

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Decision Tree
- Automatic Data Preparation
- Model Detail Views for Decision Tree
- OML4SQL Examples
- OML4R Decision Tree Example
- OML4R Code Examples

16.1 About Decision Tree

Decision tree is a supervised machine learning algorithm used for classifying data. Decision tree has a tree structure built top-down that has a root node, branches, and leaf nodes.

In some applications of Oracle Machine Learning for SQL, the reason for predicting one outcome or another may not be important in evaluating the overall quality of a model. In others, the ability to explain the reason for a decision can be crucial. You can use decision tree rules to validate models in such problems. The Decision Tree algorithm, like Naive Bayes, is based on conditional probabilities. Unlike Naive Bayes, decision trees generate **rules**. A rule is a conditional statement that can be understood by humans and used within a database to identify a set of records.

For example, a Marketing professional requires complete descriptions of customer segments to launch a successful marketing campaign. The Decision Tree algorithm is ideal for this type of application.

Use decision tree rules to validate models. If the rules make sense to a subject matter expert, then this validates the model.

16.1.1 Decision Tree Rules

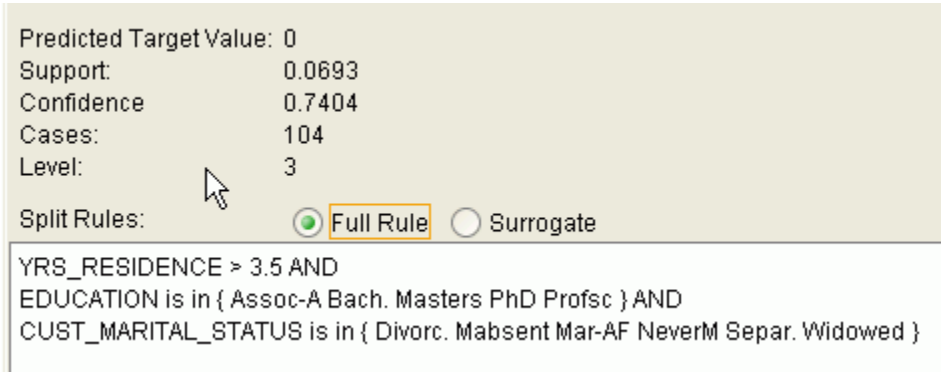
Introduces decision tree rules.

Oracle Machine Learning for SQL supports several algorithms that provide rules. In addition to decision trees, clustering algorithms provide rules that describe the conditions shared by the members of a cluster, and association rules provide rules that describe associations between attributes.

Rules provide **model transparency**, a window on the inner workings of the model. Rules show the basis for the model's predictions. Oracle Machine Learning for SQL supports a high level of model transparency. While some algorithms provide rules, *all* algorithms provide **model details**. You can examine model details to determine how the algorithm handles the attributes internally, including transformations and reverse transformations. Transparency is discussed in the context of data preparation and in the context of model building in *Oracle Machine Learning for SQL User's Guide*.

The following figure shows a rule generated by a Decision Tree model. This rule comes from a decision tree that predicts the probability that customers increase spending if given a loyalty card. A target value of 0 means not likely to increase spending; 1 means likely to increase spending.

Figure 16-1 Sample Decision Tree Rule



The rule shown in the figure represents the conditional statement:

```
IF
          (current residence > 3.5 and has college degree and is single)
THEN
          predicted target value = 0
```

This rule is a full rule. A surrogate rule is a related attribute that can be used at apply time if the attribute needed for the split is missing.

Related Topics

- Understanding Reverse Transformations
- Model Detail Views for Decision Tree
- Clustering
  Learn how to discover natural groupings in the data through clustering - the unsupervised machine learning function.

- Association
  Learn how to discover association rules through association - an unsupervised machine learning function.

16.1.1.1 Confidence and Support

Confidence and support are properties of rules. These statistical measures can be used to rank the rules and hence the predictions.

Support: The number of records in the training data set that satisfy the rule.

Confidence: The likelihood of the predicted outcome, given that the rule has been satisfied.

For example, consider a list of 1000 customers (1000 cases). Out of all the customers, 100 satisfy a given rule. Of these 100, 75 are likely to increase spending, and 25 are not likely to increase spending. The **support of the rule** is 100/1000 (10%). The **confidence of the prediction** (likely to increase spending) for the cases that satisfy the rule is 75/100 (75%).

16.1.2 Advantages of Decision Trees

Learn about the advantages of the Decision Tree algorithm.

The Decision Tree algorithm produces accurate and interpretable models with relatively little user intervention. The algorithm can be used for both binary and multiclass classification problems.

The algorithm is fast, both at build time and apply time. The build process for Decision Tree supports parallel execution. (Scoring supports parallel execution irrespective of the algorithm.)

Decision Tree scoring is especially fast. The tree structure, created in the model build, is used for a series of simple tests, (typically 2-7). Each test is based on a single predictor. It is a membership test: either IN or NOT IN a list of values (categorical predictor); or LESS THAN or EQUAL TO some value (numeric predictor).

Related Topics

- *Oracle Database VLDB and Partitioning Guide*

16.1.3 XML for Decision Tree Models

Learn about generating XML representation of Decision Tree models.

You can generate XML representing a Decision Tree model; the generated XML satisfies the definition specified in the Predictive Model Markup Language (PMML) version 2.1 specification.

Related Topics

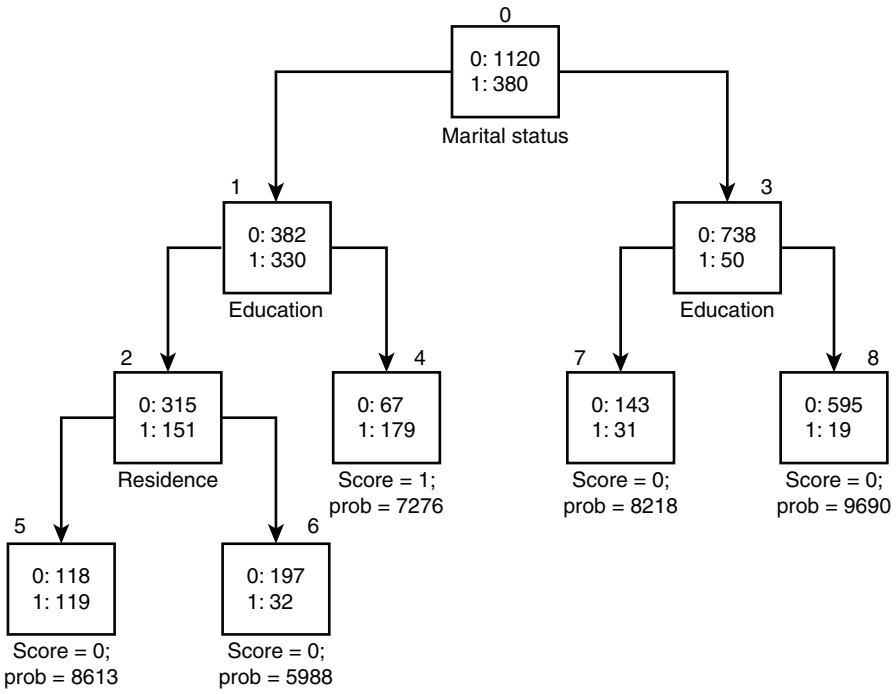
- http://www.dmg.org

16.2 Growing a Decision Tree

Predict a target value by a sequence of questions to form or grow a decision tree. A sample here shows how to grow a decision tree.

A decision tree predicts a target value by asking a sequence of questions. At a given stage in the sequence, the question that is asked depends upon the answers to the previous questions. The goal is to ask questions that, taken together, uniquely identify specific target values. Graphically, this process forms a tree structure.

Figure 16-2 Sample Decision Tree



The figure is a decision tree with nine nodes (and nine corresponding rules). The target attribute is binary: 1 if the customer increases spending, 0 if the customer does not increase spending. The first split in the tree is based on the `CUST_MARITAL_STATUS` attribute. The root of the tree (node 0) is split into nodes 1 and 3. Married customers are in node 1; single customers are in node 3.

The rule associated with node 1 is:

```
Node 1 recordCount=712,0 Count=382, 1 Count=330
CUST_MARITAL_STATUS isIN  "Married",surrogate:HOUSEHOLD_SIZE isIn "3""4-5"
```

Node 1 has 712 records (cases). In all 712 cases, the `CUST_MARITAL_STATUS` attribute indicates that the customer is married. Of these, 382 have a target of 0 (not likely to increase spending), and 330 have a target of 1 (likely to increase spending).

16.2.1 Splitting

The Decision Tree algorithm offers metrics for splitting the cases (records).

During the training process, the Decision Tree algorithm must repeatedly find the most efficient way to split a set of cases (records) into two child nodes. Oracle Machine Learning for SQL offers two homogeneity metrics, **gini** and **entropy**, for calculating the splits. The default metric is gini.

Homogeneity metrics asses the quality of alternative split conditions and select the one that results in the most homogeneous child nodes. Homogeneity is also called **purity**; it refers to the degree to which the resulting child nodes are made up of cases with the same target value. The objective is to maximize the purity in the child nodes. For example, if the target can be either yes or no (does or does not increase spending), the objective is to produce nodes where most of the cases either increase spending or most of the cases do not increase spending.

16.2.2 Cost Matrix

Learn about a cost matrix for the Decision Tree algorithm.

All classification algorithms, including Decision Tree, support a cost-benefit matrix at apply time. You can use the same cost matrix for building and scoring a decision tree model, or you can specify a different cost/benefit matrix for scoring.

Related Topics

- Costs
- Priors and Class Weights
  Learn about Priors and Class Weights in a classification model to produce a useful result.

16.2.3 Preventing Over-Fitting

Understand over-fitting in trees and what can you do to resolve over-fitting.

In principle, the Decision Tree algorithm can grow each branch of the tree deeply enough to perfectly classify the training examples. While this is sometimes a reasonable strategy, in fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that over-fit the training examples. Over-fit is a condition where a model is able to accurately predict the data used to create the model, but does poorly on new data presented to it.

To prevent over-fitting, Oracle Machine Learning for SQL supports automatic **pruning** and configurable **limit conditions** that control tree growth. Limit conditions prevent further splits once the conditions have been satisfied. Pruning removes branches that have insignificant predictive power.

16.3 Tuning the Decision Tree Algorithm

Fine tune the Decision Tree algorithm with various parameters.

The Decision Tree algorithm is implemented with reasonable defaults for splitting and termination criteria. However several build settings are available for fine tuning.

You can specify a homogeneity metric for finding the optimal split condition for a tree. The default metric is gini. The entropy metric is also available.

Settings for controlling the growth of the tree are also available. You can specify the maximum depth of the tree, the minimum number of cases required in a child node, the minimum number of cases required in a node in order for a further split to be possible, the minimum number of cases in a child node, and the minimum number of cases required in a node in order for a further split to be possible.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

The training data attributes are binned as part of the algorithm's data preparation. You can alter the number of bins used by the binning step. There is a trade-off between the number of bins used and the time required for the build.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Decision Tree for a listing and description of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

16.4 Data Preparation for Decision Tree

The Decision Tree algorithm manages its own data preparation internally. It does not require pretreatment of the data.

Decision Tree is not affected by Automatic Data Preparation (ADP).

Related Topics

- Prepare the Data

17

Expectation Maximization

Learn how to use expectation maximization clustering algorithm.

Related Topics

- Clustering Algorithms
  Learn different clustering algorithms used in Oracle Machine Learning for SQL.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Expectation Maximization
- Automatic Data Preparation
- Model Detail Views for Expectation Maximization
- OML4SQL Examples
- OML4R Expectation Maximization Example
- OML4R Code Examples

17.1 About Expectation Maximization

Expectation maximization (EM) estimation of mixture models is a popular probability density estimation technique that is used in a variety of applications.

Oracle Machine Learning for SQL uses EM to implement a distribution-based clustering algorithm (EM-clustering).

17.1.1 Expectation Step and Maximization Step

The two steps to compute the likelihood of the current model and to maximize the likelihood defines the algorithm.

Expectation maximization is an iterative method. It starts with an initial parameter guess. The parameter values are used to compute the likelihood of the current model. This is the Expectation step. The parameter values are then recomputed to maximize the likelihood. This is the Maximization step. The new parameter estimates are used to compute a new expectation and then they are optimized again to maximize the likelihood. This iterative process continues until model convergence.

17.1.2 Probability Density Estimation

Compute reliable cluster assignment using probability density.

In density estimation, the goal is to construct a density function that captures how a given population is distributed. In probability density estimation, the density estimate is based on observed data that represents a sample of the population. Areas of high data density in the model correspond to the peaks of the underlying distribution.

Density-based clustering is conceptually different from distance-based clustering (for example *k*-Means) where emphasis is placed on minimizing inter-cluster and maximizing the intra-cluster distances. Due to its probabilistic nature, density-based clustering can compute reliable probabilities in cluster assignment. It can also handle missing values automatically.

17.2 Algorithm Enhancements

Expectation Maximization (EM) is enhanced to resolve some challenges in it's standard form.

Although EM is well established as a distribution-based clustering algorithm, it presents some challenges in its standard form. The Oracle Machine Learning for SQL implementation includes significant enhancements, such as scalable processing of large volumes of data and automatic parameter initialization. The strategies that OML4SQL uses to address the inherent limitations of EM clustering are described further.

> **Note:**
>
> The EM abbreviation is used here to refer to EM-clustering.

Limitations of Standard Expectation Maximization:

- Scalability: EM has linear scalability with the number of records and attributes. The number of iterations to convergence tends to increase with growing data size (both rows and columns). EM convergence can be slow for complex problems and can place a significant load on computational resources.
- High dimensionality: EM has limited capacity for modeling high dimensional (wide) data. The presence of many attributes slows down model convergence, and the algorithm becomes less able to distinguish between meaningful attributes and noise. The algorithm is thus compromised in its ability to find correlations.
- Number of components: EM typically requires the user to specify the number of components. In most cases, this is not information that the user can know in advance.
- Parameter initialization: The choice of appropriate initial parameter values can have a significant effect on the quality of the model. Initialization strategies that have been used for EM have generally been computationally expensive.
- From components to clusters: EM model components are often treated as clusters. This approach can be misleading since cohesive clusters are often

modeled by multiple components. Clusters that have a complex shape need to be modeled by multiple components.

17.2.1 Scalability

Expectation Maximization (EM) in Oracle Machine Learning for SQL, uses database parallel processing to achieve excellent scalability.

The OML4SQL implementation of Expectation Maximization uses database parallel processing to achieve excellent scalability. EM computations naturally lend themselves to row parallel processing, and the partial results are easily aggregated. The parallel implementation efficiently distributes the computationally intensive work across slave processes and then combines the partial results to produce the final solution.

Related Topics

- *Oracle Database VLDB and Partitioning Guide*

17.2.2 High Dimensionality

Process high dimensional data through Expectation Maximization.

The Oracle Machine Learning for SQL implementation of Expectation Maximization (EM) can efficiently process high-dimensional data with thousands of attributes. This is achieved through a two-fold process:

- The data space of single-column (not nested) attributes is analyzed for pair-wise correlations. Only attributes that are significantly correlated with other attributes are included in the EM mixture model. The algorithm can also be configured to restrict the dimensionality to the *M* most correlated attributes.
- High-dimensional (nested) numerical data that measures events of similar type is projected into a set of low-dimensional features that are modeled by EM. Some examples of high-dimensional, numerical data are: text, recommendations, gene expressions, and market basket data.

17.2.3 Number of Components

The number of EM components are automatically determined.

Typical implementations of Expectation Maximization (EM) require the user to specify the number of model components. This is problematic because users do not generally know the correct number of components. Choosing too many or too few components can lead to over-fitting or under-fitting, respectively.

When model search is enabled, the number of EM components is automatically determined. The algorithm uses a held-aside sample to determine the correct number of components, except in the cases of very small data sets when Bayesian Information Criterion (BIC) regularization is used.

17.2.4 Parameter Initialization

Choosing appropriate initial parameter values can have a significant effect on the quality of the solution.

Expectation maximization (EM) is not guaranteed to converge to the global maximum of the likelihood function but may instead converge to a local maximum. Therefore

different initial parameter values can lead to different model parameters and different model quality.

In the process of model search, the EM model is grown independently. As new components are added, their parameters are initialized to areas with poor distribution fit.

17.2.5 From Components to Clusters

Expectation Maximization produces assignment of model components to high-level clusters.

Expectation Maximization (EM) model components are often treated as clusters. However, this approach can be misleading. Cohesive clusters are often modeled by multiple components. The shape of the probability density function used in EM effectively predetermines the shape of the identified clusters. For example, Gaussian density functions can identify single peak symmetric clusters. Clusters of more complex shape need to be modeled by multiple components.

Ideally, high density areas of arbitrary shape must be interpreted as single clusters. To accomplish this, the Oracle Machine Learning for SQL implementation of EM builds a component hierarchy that is based on the overlap of the individual components' distributions. OML4SQL EM uses agglomerative hierarchical clustering. Component distribution overlap is measured using the Bhattacharyya distance function. Choosing an appropriate cutoff level in the hierarchy automatically determines the number of high-level clusters.

The OML4SQL implementation of EM produces an assignment of the model components to high-level clusters. Statistics like means, variances, modes, histograms, and rules additionally describe the high-level clusters. The algorithm can be configured to either produce clustering assignments at the component level or at the cluster level.

17.3 Configuring the Algorithm

Configure Expectation Maximization (EM).

In Oracle Machine Learning for SQL, EM can effectively model very large data sets (both rows and columns) without requiring the user to supply initialization parameters or specify the number of model components. While the algorithm offers reasonable defaults, it also offers flexibility.

The following list describes some of the configurable aspects of EM:

- Whether or not independent non-nested column attributes are included in the model. The choice is system-determined by default.
- Whether to use Bernoulli or Gaussian distribution for numerical attributes. By default, the algorithm chooses the most appropriate distribution, and individual attributes may use different distributions. When the distribution is user-specified, it is used for all numerical attributes.
- Whether the convergence criterion is based on a held-aside data set or on Bayesian Information Criterion (BIC). The convergence criterion is system-determined by default.
- The percentage improvement in the value of the log likelihood function that is required to add a new component to the model. The default percentage is 0.001.

- Whether to define clusters as individual components or groups of components. Clusters are associated to groups of components by default.
- The maximum number of components in the model. If model search is enabled, the algorithm determines the number of components based on improvements in the likelihood function or based on regularization (BIC), up to the specified maximum.
- Whether the linkage function for the agglomerative clustering step uses the nearest distance within the branch (single linkage), the average distance within the branch (average linkage), or the maximum distance within the branch (complete linkage). By default the algorithm uses single linkage.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Expectation Maximization for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- DBMS_DATA_MINING - Global Settings

17.4 Data Preparation for Expectation Maximization

Learn how to prepare data for Expectation Maximization (EM).

If you use Automatic Data Preparation (ADP), you do not need to specify additional data preparation for Expectation Maximization. ADP normalizes numerical attributes (in non-nested columns) when they are modeled with Gaussian distributions. ADP applies a topN binning transformation to categorical attributes.

Missing value treatment is not needed since Oracle Machine Learning for SQL algorithms handle missing values automatically. The EM algorithm replaces missing values with the mean in single-column numerical attributes that are modeled with Gaussian distributions. In other single-column attributes (categoricals and numericals modeled with Bernoulli distributions), NULLs are not replaced; they are treated as a distinct value with its own frequency count. In nested columns, missing values are treated as zeros.

Related Topics

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

18
Explicit Semantic Analysis

Learn how to use Explicit Semantic Analysis (ESA) as an unsupervised algorithm for feature extraction function and as a supervised algorithm for classification.

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- Feature Extraction
  Learn how to perform attribute reduction using feature extraction as an unsupervised function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: Explicit Semantic Analysis
- Automatic Data Preparation
- Model Detail Views for Explicit Semantic Analysis
- OML4SQL Examples
- OML4R Explicit Semantic Analysis Example
- OML4R Code Examples

18.1 About Explicit Semantic Analysis

In Oracle Database 12*c* Release 2, Explicit Semantic Analysis (ESA) was introduced as an unsupervised algorithm for feature extraction. Starting from Oracle Database 18c, ESA is enhanced as a supervised algorithm for classification.

As a feature extraction algorithm, ESA does not discover latent features but instead uses explicit features represented in an existing knowledge base. As a feature extraction algorithm, ESA is mainly used for calculating semantic similarity of text documents and for explicit topic modeling. As a classification algorithm, ESA is primarily used for categorizing text documents. Both the feature extraction and classification versions of ESA can be applied to numeric and categorical input data as well.

The input to ESA is a set of attributes vectors. Every attribute vector is associated with a concept. The concept is a feature in the case of feature extraction or a target class in the case of classification. For feature extraction, only one attribute vector may be associated with any feature. For classification, the training set may contain multiple

attribute vectors associated with any given target class. These rows related to one target class are aggregated into one by the ESA algorithm.

The output of ESA is a sparse attribute-concept matrix that contains the most important attribute-concept associations. The strength of the association is captured by the weight value of each attribute-concept pair. The attribute-concept matrix is stored as a reverse index that lists the most important concepts for each attribute.

> **Note:**
>
> For feature extraction the ESA algorithm does not project the original feature space and does not reduce its dimensionality. ESA algorithm filters out features with limited or uninformative set of attributes.

The scope of classification tasks that ESA handles is different than the classification algorithms such as Naive Bayes and Support Vector Machine. ESA can perform large scale classification with the number of distinct classes up to hundreds of thousands. The large scale classification requires gigantic training data sets with some classes having significant number of training samples whereas others are sparsely represented in the training data set.

18.1.1 ESA for Text Analysis

Learn how Explicit Semantic Analysis (ESA) can be used for machine learning operations on text.

Explicit knowledge often exists in text form. Multiple knowledge bases are available as collections of text documents. These knowledge bases can be generic, for example, Wikipedia, or domain-specific. Data preparation transforms the text into vectors that capture attribute-concept associations. ESA is able to quantify semantic relatedness of documents even if they do not have any words in common. The function `FEATURE_COMPARE` can be used to compute semantic relatedness.

Related Topics

- *Oracle Database SQL Language Reference*

18.2 Data Preparation for ESA

Automatic Data Preparation normalizes input vectors to a unit length for Explicit Semantic Analysis (ESA).

When there are missing values in columns with simple data types (not nested), ESA replaces missing categorical values with the mode and missing numerical values with the mean. When there are missing values in nested columns, ESA interprets them as sparse. The algorithm replaces sparse numeric data with zeros and sparse categorical data with zero vectors. The Oracle Machine Learning for SQL data preparation transforms the input text into a vector of real numbers. These numbers represent the importance of the respective words in the text.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Explicit Semantic Analysis for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

18.3 Scoring with ESA

A typical feature extraction application of Explicit Semantic Analysis (ESA) is to identify the most relevant features of a given input and score their relevance. Scoring an ESA model produces data projections in the concept feature space.

If an ESA model is built from an arbitrary collection of documents, then each one is treated as a feature. You can then identify the most relevant documents in the collection. The feature extraction functions are: `FEATURE_DETAILS`, `FEATURE_ID`, `FEATURE_SET`, `FEATURE_VALUE`, and `FEATURE_COMPARE`.

A typical classification application of ESA is to predict classes of a given document and estimate the probabilities of the predictions. As a classification algorithm, ESA implements the following scoring functions: `PREDICTION`, `PREDICTION_PROBABILITY`, `PREDICTION_SET`, `PREDICTION_DETAILS`, `PREDICTION_COST`.

Related Topics

- *Oracle Machine Learning for SQL User's Guide*
- *Oracle Database SQL Language Reference*

18.3.1 Scoring Large ESA Models

Building an Explicit Semantic Analysis (ESA) model on a large collection of text documents can result in a model with many features or titles.

The model information for scoring is loaded into System Global Area (SGA) as a shared (shared pool size) library cache object. Different SQL predictive queries can reference this object. When the model size is large, it is necessary to set the SGA parameter in the database to a sufficient size that accommodates large objects. If the SGA is too small, the model may need to be re-loaded every time it is referenced which is likely to lead to performance degradation.

18.4 Terminologies in Explicit Semantic Analysis

Discusses the terms associated with Explicit Semantic Analysis (ESA).

Multi-target Classification

The training items in these large scale classifications belong to several classes. The goal of classification in such case is to detect possible multiple target classes for one item. This kind of classification is called multi-target classification. The target

column for ESA-based classification is extended. Collections are allowed as target column values. The collection type for the target in ESA-based classification is `ORA_MINING_VARCHAR2_NT`.

Large-scale classification

Large-scale classification applies to ontologies that contain gigantic numbers of categories, usually ranging in tens or hundreds of thousands. This large-scale classification also requires gigantic training datasets which are usually unbalanced, that is, some classes may have significant number of training samples whereas others may be sparsely represented in the training dataset. Large-scale classification normally results in multiple target class assignments for a given test case.

Topic modeling

Topic modelling refers to derivation of the most important topics of a document. Topic modeling can be explicit or latent. Explicit topic modeling results in the selection of the most relevant topics from a pre-defined set, for a given document. Explicit topics have names and can be verbalized. Latent topic modeling identifies a set of latent topics characteristic for a collection of documents. A subset of these latent topics is associated with every document under examination. Latent topics do not have verbal descriptions or meaningful interpretation.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

19
Exponential Smoothing

Learn about the Exponential Smoothing algorithm.

- About Exponential Smoothing
- Data Preparation for Exponential Smoothing Models

Related Topics

- Time Series
  Learn about time series as an Oracle Machine Learning for SQL regression function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: Exponential Smoothing
- Automatic Data Preparation
- Model Detail Views for Exponential Smoothing
- OML4SQL Examples
- OML4R Code Examples

19.1 About Exponential Smoothing

Exponential smoothing is a forecasting method for time-series data. It is a moving average method where exponentially decreasing weights are assigned to past observations.

Exponential smoothing methods have been widely used in forecasting for over half a century. It has applications at the strategic, tactical, and operation level. For example, at a strategic level, forecasting is used for projecting return on investment, growth and the effect of innovations. At a tactical level, forecasting is used for projecting costs, inventory requirements, and customer satisfaction. At an operational level, forecasting is used for setting targets and predicting quality and conformance with standards.

In its simplest form, exponential smoothing is a moving average method with a single parameter which models an exponentially decreasing effect of past levels on future values. With a variety of extensions, exponential smoothing covers a broader class of models than competitors, such as the Box-Jenkins auto-regressive integrated moving average (ARIMA) approach. Oracle Machine Learning for SQL implements exponential smoothing using a state of the art state space method that incorporates a single source of error (SSOE) assumption which provides theoretical and performance advantages.

Exponential smoothing is extended to the following:

- A matrix of models that mix and match error type (additive or multiplicative), trend (additive, multiplicative, or none), and seasonality (additive, multiplicative, or none)
- Models with damped trends.

- Models that directly handle irregular time series and time series with missing values.

> **Note:**
>
> For more information, see Ord, J.K., et al, *Time Series Forecasting: The Case for the Single Source of Error State Space Approach, Working Paper*, Department of Econometrics and Business Statistics, Monash University, VIC 3800, Australia, April 2, 2005.

19.1.1 Exponential Smoothing Models

Exponential Smoothing models are a broad class of forecasting models that are intuitive, flexible, and extensible.

Members of this class include simple, single parameter models that predict the future as a linear combination of a previous level and a current shock. Extensions can include parameters for linear or non-linear trend, trend damping, simple or complex seasonality, related series, various forms of non-linearity in the forecasting equations, and handling of irregular time series.

Exponential smoothing assumes that a series extends infinitely into the past, but that influence of past on future, decays smoothly and exponentially fast. The smooth rate of decay is expressed by one or more smoothing constants. The **smoothing constants** are parameters that the model estimates. The assumption is made practical for modeling real world data by using an equivalent recursive formulation that is only expressed in terms of an estimate of the current level based on prior history and a shock to that estimate dependent on current conditions only.The procedure requires an estimate for the time period just prior to the first observation, that encapsulates all prior history. This initial observation is an additional model parameter whose value is estimated by the modeling procedure.

Components of ESM such as trend and seasonality extensions, can have an additive or multiplicative form. The simpler additive models assume that shock, trend, and seasonality are linear effects within the recursive formulation.

19.1.2 Simple Exponential Smoothing

Simple exponential smoothing assumes the data fluctuates around a stationary mean, with no trend or seasonal pattern.

In a simple Exponential Smoothing model, each forecast (smoothed value) is computed as the weighted average of the previous observations, where the weights decrease exponentially depending on the value of smoothing constant α. Values of the smoothing constant, α, near one, put almost all weight on the most recent observations. Values of α near zero allows the distant past observations to have a large influence.

19.1.3 Models with Trend but No Seasonality

The preferred form of additive (linear) trend is sometimes called Holt's method or double exponential smoothing.

Models with trend add a smoothing parameter γ and optionally a damping parameter φ. The damping parameter smoothly dampens the influence of past linear trend on future estimates of level, often improving accuracy.

19.1.4 Models with Seasonality but No Trend

When the time series average does not change over time (stationary), but is subject to seasonal fluctuations, the appropriate model has seasonal parameters but no trend.

Seasonal fluctuations are assumed to balance out over periods of length *m*, where *m* is the number of seasons, For example, *m*=4 might be used when the input data are aggregated quarterly. For models with additive errors, the seasonal parameters must sum to zero. For models with multiplicative errors, the product of seasonal parameters must be one.

19.1.5 Models with Trend and Seasonality

Holt and Winters introduced both trend and seasonality in an Exponential Smoothing model.

The original model, also known as Holt-Winters or triple exponential smoothing, considered an additive trend and multiplicative seasonality. Extensions include models with various combinations of additive and multiplicative trend, seasonality and error, with and without trend damping.

19.1.6 Prediction Intervals

To compute prediction intervals, an Exponential Smoothing (ESM) model is divided into three classes.

The simplest class is the class of linear models, which include, among others, simple ESM, Holt's method, and additive Holt-Winters. Class 2 models (multiplicative error, additive components) make an approximate correction for violations of the Normality assumption. Class 3 modes use a simple simulation approach to calculate prediction intervals.

19.2 Data Preparation for Exponential Smoothing Models

Learn about preparing the data for an Exponential Smoothing (ESM) model.

To build an ESM model, you must supply the following :

- Input data
- An aggregation level and method, if the case id is a date type
- Partitioning column, if the data are partitioned

In addition, for a greater control over the build process, the user may optionally specify model build parameters, all of which have defaults:

- Model
- Error type
- Optimization criterion
- Forecast Window

- Confidence level for forecast bounds
- Missing value handling
- Whether the input series is evenly spaced

Related Topics

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

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Exponential Smoothing Models for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

19.2.1 Input Data

Time series analysis, requires ordered input data. Hence, each data row must consist of an [index, value] pair, where the index specifies the ordering.

When the `CREATE_MODEL` procedure is used to initiate an Exponential Smoothing (ESM) model build, the `CASE_ID_COLUMN_NAME` specifies the column used to compute the indices of the input and the `TARGET_COLUMN_NAME` specifies the column used to compute the observed time series values. The time column bears Oracle number, or Oracle date, timestamp, timestamp with time zone, or timestamp with local time zone. The input time series are sorted according to the values of `CASE_ID` (time label). The case id column cannot contain missing values. The value column can contain missing values indicated as `NULL`. ESM also supports partitioned models and in such cases, the input table contains an extra column specifying the partition. All [index, value] pairs with the same partition ID form one complete time series. The Exponential Smoothing algorithm constructs models for each partition independently, although all models use the same model settings.

Properties of the data can result in a warning message or settings are ignored. Settings are ignored when If the user specifies a model with either multiplicative trend, multiplicative seasonality or both and the data contains values $Y_t <= 0$, then the model type is set to the default. If the series contain fewer values than the number of user-specified seasons, then the seasonality specifications are ignored with a warning.

19.2.2 Accumulation

For the Exponential Smoothing algorithm, the accumulation procedure is applied when the column is a date type (`date`, `datetime`, `timestamp`, `timestamp with timezone`, or `timestamp with local timezone`).

The case id can be a `NUMBER` column whose sort index represents the position of the value in the time series sequence of values. The case id column can also be a date

type. A date type is accumulated in accordance with a user specified accumulation window. Regardless of type, the case id is used to transform the column into an equally spaced time series. No accumulation is applied for a case id of type `NUMBER`. As an example, consider a time series about promotion events. The time column contains the date of each event, and the dates can be unequally spaced. The user must specify the spacing interval, which is the spacing of the accumulated or transformed equally spaced time series. In the example, if the user specifies the interval to be month, then an equally spaced time series with profit for each calendar month is generated from the original time series. Setting `EXSM_INTERVAL` is used to specify the spacing interval. The user must also specify a value for `EXSM_ACCUMULATE`, for example, `EXSM_ACCU_MAX`, in which case the equally spaced monthly series would contain the maximum profit over all events that month as the observed time series value.

19.2.3 Missing Value

Input time series can contain missing values. A `NULL` entry in the target column indicates a missing value. When the time column is of the type datetime, the accumulation procedure can also introduce missing values. The setting `EXSM_SETMISSING` can be used to specify how to handle missing values. The special value `EXSM_MISS_AUTO` indicates that, if the series contains missing values it is to be treated as an irregular time series.

> **Note:**
>
> Missing value handling setting must be compatible with model setting, otherwise an error is thrown.

19.2.4 Prediction

An Exponential Smoothing (ESM) model can be applied to make predictions by specifying the prediction window.

Setting `EXSM_PREDICTION_STEP` can be used to specify the prediction window. The prediction window is expressed in terms of number of intervals (setting `EXSM_INTERVAL`), when the time column is of the type datetime. If the time column is a number then the prediction window is the number of steps to forecast. Regardless of whether the time series is regular or irregular, `EXSM_PREDICTION_STEP` specifies the prediction window.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

19.2.5 Parallellism by Partition

Oracle Machine Learning for SQL supports parallellism by partition.

For example, a user can choose `PRODUCT_ID` as one partition column and can generate forecasts for different products in a model build. Although a distinct smoothing model is built for each partition, all partitions share the same model settings. For example, if setting `EXSM_MODEL` is set to `EXSM_SIMPLE`, all partition models will be simple Exponential Smoothing models. Time series from different partitions can be distributed to different processes and processed in parallel. The model for each time series is built serially.

20
Generalized Linear Model

Learn how to use Generalized Linear Model (GLM) statistical technique for linear modeling.

Oracle Machine Learning for SQL supports GLM for regression and binary classification.

Related Topics

- Regression
  Learn how to predict a continuous numerical target through regression - the supervised machine learning function.
- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- Feature Selection
  Learn how to perform feature selection and attribute importance.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Generalized Linear Models
- Automatic Data Preparation
- Model Detail Views for Generalized Linear Model
- OML4SQL Examples
- OML4R Generalized Linear Model Example
- OML4R Code Examples

20.1 About Generalized Linear Model

Learn about Generalized Linear Model (GLM) models include and extend the class of linear models which address and accommodate some restrictive assumptions of the linear models.

Linear models make a set of restrictive assumptions, most importantly, that the target (dependent variable *y*) is normally distributed conditioned on the value of predictors

with a constant variance regardless of the predicted response value. The advantage of linear models and their restrictions include computational simplicity, an interpretable model form, and the ability to compute certain diagnostic information about the quality of the fit.

GLM relaxes these restrictions, which are often violated in practice. For example, binary (yes/no or 0/1) responses do not have same variance across classes. Furthermore, the sum of terms in a linear model typically can have very large ranges encompassing very negative and very positive values. For the binary response example, we would like the response to be a probability in the range [0,1].

GLM accommodates responses that violate the linear model assumptions through two mechanisms: a link function and a variance function. The link function transforms the target range to potentially -infinity to +infinity so that the simple form of linear models can be maintained. The variance function expresses the variance as a function of the predicted response, thereby accommodating responses with non-constant variances (such as the binary responses).

Oracle Machine Learning for SQL includes two of the most popular members of the GLM family of models with their most popular link and variance functions:

- **Linear regression** with the identity link and variance function equal to the constant 1 (constant variance over the range of response values).
- **Logistic regression** with the logit link and binomial variance functions.

Related Topics

- Linear Regression
- Linear Regression
- Logistic Regression

20.2 GLM in Oracle Machine Learning for SQL

Learn how Oracle Machine Learning for SQL implements the Generalized Linear Model (GLM) algorithm.

GLM is a parametric modeling technique. Parametric models make assumptions about the distribution of the data. When the assumptions are met, parametric models can be more efficient than non-parametric models.

The challenge in developing models of this type involves assessing the extent to which the assumptions are met. For this reason, quality diagnostics are key to developing quality parametric models.

20.2.1 Interpretability and Transparency

You can interpret and understand key characteristics of Generalized Linear Model (GLM) model through model details and global details.

You can interpret Oracle Machine Learnings' GLM with ease. Each model build generates many statistics and diagnostics. Transparency is also a key feature: model details describe key characteristics of the coefficients, and global details provide high-level statistics.

Related Topics

- Tuning and Diagnostics for GLM

20.2.2 Wide Data

Generalized Linear Model(GLM) in Oracle Machine Learning for SQL is uniquely suited for handling wide data. The algorithm can build and score quality models that use a virtually limitless number of predictors (attributes). The only constraints are those imposed by system resources.

20.2.3 Confidence Bounds

Predict confidence bounds through the Generalized Linear Model (GLM) algorithm.

GLM have the ability to predict confidence bounds. In addition to predicting a best estimate and a probability (classification only) for each row, GLM identifies an interval wherein the prediction (regression) or probability (classification) lies. The width of the interval depends upon the precision of the model and a user-specified confidence level.

The confidence level is a measure of how sure the model is that the true value lies within a confidence interval computed by the model. A popular choice for confidence level is 95%. For example, a model might predict that an employee's income is $125K, and that you can be 95% sure that it lies between $90K and $160K. Oracle Machine Learning for SQL supports 95% confidence by default, but that value can be configured.

> **Note:**
>
> Confidence bounds are returned with the coefficient statistics. You can also use the `PREDICTION_BOUNDS` SQL function to obtain the confidence bounds of a model prediction.

Related Topics

- *Oracle Database SQL Language Reference*

20.2.4 Ridge Regression

Understand the use of ridge regression for singularity (exact multicollinearity) in data.

The best regression models are those in which the predictors correlate highly with the target, but there is very little correlation between the predictors themselves. **Multicollinearity** is the term used to describe multivariate regression with correlated predictors.

Ridge regression is a technique that compensates for multicollinearity. Oracle Machine Learning for SQL supports ridge regression for both regression and classification machine learning functions. The algorithm automatically uses ridge if it detects singularity (exact multicollinearity) in the data.

Information about singularity is returned in the global model details.

Related Topics

- Global Model Statistics for Linear Regression
- Global Model Statistics for Logistic Regression

20.2.4.1 Configuring Ridge Regression

Configure ridge regression through build settings.

You can choose to explicitly enable ridge regression by specifying a build setting for the model. If you explicitly enable ridge, you can use the system-generated ridge parameter or you can supply your own. If ridge is used automatically, the ridge parameter is also calculated automatically.

The configuration choices are summarized as follows:

- Whether or not to override the automatic choice made by the algorithm regarding ridge regression
- The value of the ridge parameter, used only if you specifically enable ridge regression.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

20.2.4.2 Ridge and Confidence Bounds

Models built with ridge regression do not support confidence bounds.

Related Topics

- Confidence Bounds
  Predict confidence bounds through the Generalized Linear Model (GLM) algorithm.

20.2.4.3 Ridge and Data Preparation

Learn about preparing data for ridge regression.

When ridge regression is enabled, different data preparation is likely to produce different results in terms of model coefficients and diagnostics. Oracle recommends that you enable Automatic Data Preparation for Generalized Linear Model models, especially when ridge regression is used.

Related Topics

- Data Preparation for GLM
  Learn about preparing data for the Generalized Linear Model (GLM) algorithm.

20.3 Scalable Feature Selection

Oracle Machine Learning for SQL supports a highly scalable and automated version of feature selection and generation for the Generalized Linear Model algorithm.

This scalable and automated capability can enhance the performance of the algorithm and improve accuracy and interpretability. Feature selection and generation are available for both linear regression and binary logistic regression.

20.3.1 Feature Selection

Feature selection is the process of choosing the terms to be included in the model. The fewer terms in the model, the easier it is for human beings to interpret its meaning. In addition, some columns may not be relevant to the value that the model is trying to predict. Removing such columns can enhance model accuracy.

20.3.1.1 Configuring Feature Selection

Feature selection is a build setting for Generalized Linear Model models. It is not enabled by default. When configured for feature selection, the algorithm automatically determines appropriate default behavior, but the following configuration options are available:

- The feature selection criteria can be AIC, SBIC, RIC, or α-investing. When the feature selection criteria is α-investing, feature acceptance can be either strict or relaxed.
- The maximum number of features can be specified.
- Features can be pruned in the final model. Pruning is based on t-statistics for linear regression or wald statistics for logistic regression.

20.3.1.2 Feature Selection and Ridge Regression

Feature selection and ridge regression are mutually exclusive. When feature selection is enabled, the algorithm can not use ridge.

> **Note:**
>
> If you configure the model to use both feature selection and ridge regression, then you get an error.

20.3.2 Feature Generation

Feature generation is the process of adding transformations of terms into the model. Feature generation enhances the power of models to fit more complex relationships between target and predictors.

20.3.2.1 Configuring Feature Generation

Learn about configuring feature generation.

Feature generation is only possible when feature selection is enabled. Feature generation is a build setting. By default, feature generation is not enabled.

The feature generation method can be either quadratic or cubic. By default, the algorithm chooses the appropriate method. You can also explicitly specify the feature generation method.

The following options for feature selection also affect feature generation:

- Maximum number of features
- Model pruning

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

20.4 Tuning and Diagnostics for GLM

The process of developing a Generalized Linear Model model typically involves a number of model builds. Each build generates many statistics that you can evaluate to determine the quality of your model. Depending on these diagnostics, you may want to try changing the model settings or making other modifications.

20.4.1 Build Settings

Specify the build settings for Generalized Linear Model (GLM).

You can use specify build settings.

Additional build settings are available to:

- Control the use of ridge regression.
- Specify the handling of missing values in the training data.
- Specify the target value to be used as a reference in a logistic regression model.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Generalized Linear Models for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- Ridge Regression
  Understand the use of ridge regression for singularity (exact multicollinearity) in data.
- Data Preparation for GLM
  Learn about preparing data for the Generalized Linear Model (GLM) algorithm.

- Logistic Regression

20.4.2 Diagnostics

A Generalized Linear Model model generates many metrics to help you evaluate the quality of the model.

20.4.2.1 Coefficient Statistics

Learn about coeffficient statistics for linear and logistic regression.

The same set of statistics is returned for both linear and logistic regression, but statistics that do not apply to the machine learning function are returned as NULL.

Coefficient statistics are returned by the model detail views for a Generalized Linear Model (GLM) model.

Related Topics

- Coefficient Statistics for Linear Regression
- Coefficient Statistics for Logistic Regression
- *Oracle Machine Learning for SQL User's Guide*

20.4.2.2 Global Model Statistics

Learn about high-level statistics describing the model.

Separate high-level statistics describing the model as a whole, are returned for linear and logistic regression. When ridge regression is enabled, fewer global details are returned.

Global statistics are returned by the model detail views for a Generalized Linear Model model.

Related Topics

- Global Model Statistics for Linear Regression
- Global Model Statistics for Logistic Regression
- Ridge Regression
  Understand the use of ridge regression for singularity (exact multicollinearity) in data.
- *Oracle Machine Learning for SQL User's Guide*

20.4.2.3 Row Diagnostics

Generate row-statistics by configuring the Generalized Linear Model (GLM) algorithm.

GLM generates per-row statistics if you specify the name of a diagnostics table in the build setting `GLMS_DIAGNOSTICS_TABLE_NAME`.

GLM requires a case ID to generate row diagnostics. If you provide the name of a diagnostic table but the data does not include a case ID column, an exception is raised.

Related Topics

- Row Diagnostics for Linear Regression
- Row Diagnostics for Logistic Regression

20.5 GLM Solvers

Generalized Linear Model (GLM) algorithm applies different solvers. These solvers employ different approaches for optimization.

The GLM algorithm supports four different solvers: Cholesky, QR, Stochastic Gradient Descent (SGD),and Alternating Direction Method of Multipliers (ADMM) (on top of L-BFGS). The Cholesky and QR solvers employ classical decomposition approaches. The Cholesky solver is faster compared to the QR solver but less stable numerically. The QR solver handles better rank deficient problems without the help of regularization.

The SGD and ADMM (on top of L-BFGS) solvers are best suited for large scale data. The SGD solver employs the stochastic gradient descent optimization algorithm while ADMM (on top of L-BFGS) uses the Broyden-Fletcher-Goldfarb-Shanno optimization algorithm within an Alternating Direction Method of Multipliers framework. The SGD solver is fast but is sensitive to parameters and requires suitable scaled data to achieve good convergence. The L-BFGS algorithm solves unconstrained optimization problems and is more stable and robust than SGD. Also, L-BFGS uses ADMM in conjunction, which, results in an efficient distributed optimization approach with low communication cost.

Related Topics

- DBMS_DATA_MINING - Algorithm Settings: Neural Network
- DBMS_DATA_MINING — Algorithm Settings: Generalized Linear Models
- DBMS_DATA_MINING — Algorithm Settings: ADMM
- DBMS_DATA_MINING — Algorithm Settings: LBFGS

20.6 Data Preparation for GLM

Learn about preparing data for the Generalized Linear Model (GLM) algorithm.

Automatic Data Preparation (ADP) implements suitable data transformations for both linear and logistic regression.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Generalized Linear Models for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting. Oracle recommends that you use ADP with GLM.

Related Topics

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

20.6.1 Data Preparation for Linear Regression

Learn about Automatic Data Preparation (ADP) for the Generalized Linear Model (GLM) algorithm.

When ADP is enabled, the algorithm chooses a transformation based on input data properties and other settings. The transformation can include one or more of the following for numerical data: subtracting the mean, scaling by the standard deviation, or performing a correlation transformation (Neter, et. al, 1990). If the correlation transformation is applied to numeric data, it is also applied to categorical attributes.

Prior to standardization, categorical attributes are exploded into N-1 columns where N is the attribute cardinality. The most frequent value (mode) is omitted during the explosion transformation. In the case of highest frequency ties, the attribute values are sorted alpha-numerically in ascending order, and the first value on the list is omitted during the explosion. This explosion transformation occurs whether or not ADP is enabled.

In the case of high cardinality categorical attributes, the described transformations (explosion followed by standardization) can increase the build data size because the resulting data representation is dense. To reduce memory, disk space, and processing requirements, use an alternative approach. Under these circumstances, the VIF statistic must be used with caution.

Related Topics

- Ridge and Data Preparation
  Learn about preparing data for ridge regression.

See Also:

- Neter, J., Wasserman, W., and Kutner, M.H., "Applied Statistical Models", Richard D. Irwin, Inc., Burr Ridge, IL, 1990.

20.6.2 Data Preparation for Logistic Regression

Categorical attributes are exploded into *N*-1 columns where *N* is the attribute cardinality. The most frequent value (mode) is omitted during the explosion transformation. In the case of highest frequency ties, the attribute values are sorted alpha-numerically in ascending order and the first value on the list is omitted during the explosion. This explosion transformation occurs whether or not Automatic Data Preparation (ADP) is enabled.

When ADP is enabled, numerical attributes are scaled by the standard deviation. This measure of variability is computed as the standard deviation per attribute with respect to the origin (not the mean) (Marquardt, 1980).

> **See Also:**
>
> Marquardt, D.W., "A Critique of Some Ridge Regression Methods: Comment", Journal of the American Statistical Association, Vol. 75, No. 369 , 1980, pp. 87-91.

20.6.3 Missing Values

When building or applying a model, Oracle Machine Learning for SQL automatically replaces missing values of numerical attributes with the mean and missing values of categorical attributes with the mode.

You can configure the Generalized Linear Model algorithm to override the default treatment of missing values. With the `ODMS_MISSING_VALUE_TREATMENT` setting, you can cause the algorithm to delete rows in the training data that have missing values instead of replacing them with the mean or the mode. However, when the model is applied, OML4SQL performs the usual mean/mode missing value replacement. As a result, it is possible that the statistics generated from scoring does not match the statistics generated from building the model.

If you want to delete rows with missing values in the scoring the model, you must perform the transformation explicitly. To make build and apply statistics match, you must remove the rows with NULLs from the scoring data before performing the apply operation. You can do this by creating a view.

```
CREATE VIEW viewname AS SELECT * from tablename
     WHERE column_name1 is NOT NULL
     AND   column_name2 is NOT NULL
     AND   column_name3 is NOT NULL .....
```

> **Note:**
>
> In OML4SQL, missing values in nested data indicate sparsity, not values missing at random.
>
> The value `ODMS_MISSING_VALUE_DELETE_ROW` is only valid for tables without nested columns. If this value is used with nested data, an exception is raised.

20.7 Linear Regression

Oracle Machine Learning for SQL supports linear regression as the Generalized Linear Model regression algorithm. The algorithm assumes no target transformation and constant variance over the range of target values.

20.7.1 Coefficient Statistics for Linear Regression

Generalized Linear Model regression models generate the following coefficient statistics:

- Linear coefficient estimate

- Standard error of the coefficient estimate
- t-value of the coefficient estimate
- Probability of the t-value
- Variance Inflation Factor (VIF)
- Standardized estimate of the coefficient
- Lower and upper confidence bounds of the coefficient

20.7.2 Global Model Statistics for Linear Regression

Generalized Linear Model regression models generate the following statistics that describe the model as a whole:

- Model degrees of freedom
- Model sum of squares
- Model mean square
- Model *F* statistic
- Model *F* value probability
- Error degrees of freedom
- Error sum of squares
- Error mean square
- Corrected total degrees of freedom
- Corrected total sum of squares
- Root mean square error
- Dependent mean
- Coefficient of variation
- R-Square
- Adjusted R-Square
- Akaike's information criterion
- Schwarz's Baysian information criterion
- Estimated mean square error of the prediction
- Hocking Sp statistic
- JP statistic (the final prediction error)
- Number of parameters (the number of coefficients, including the intercept)
- Number of rows
- Whether or not the model converged
- Whether or not a covariance matrix was computed

20.7.3 Row Diagnostics for Linear Regression

For linear regression, the diagnostics table has the columns described in the following table. All the columns are NUMBER, except the CASE_ID column, which preserves the type from the training data.

Table 20-1 Diagnostics Table for GLM Regression Models

Column	Description
CASE_ID	Value of the case ID column
TARGET_VALUE	Value of the target column
PREDICTED_VALUE	Value predicted by the model for the target
HAT	Value of the diagonal element of the hat matrix
RESIDUAL	Measure of error
STD_ERR_RESIDUAL	Standard error of the residual
STUDENTIZED_RESIDUAL	Studentized residual
PRED_RES	Predicted residual
COOKS_D	Cook's D influence statistic

20.8 Logistic Regression

Oracle Machine Learning for SQL supports binary logistic regression as a Generalized Linear Model classification algorithm. The algorithm uses the logit link function and the binomial variance function.

20.8.1 Reference Class

You can use the build setting GLMS_REFERENCE_CLASS_NAME to specify the target value to be used as a reference in a binary logistic regression model. Probabilities are produced for the other (non-reference) class. By default, the algorithm chooses the value with the highest prevalence. If there are ties, the attributes are sorted alpha-numerically in an ascending order.

20.8.2 Class Weights

You can use the build setting CLAS_WEIGHTS_TABLE_NAME to specify the name of a class weights table. Class weights influence the weighting of target classes during the model build.

20.8.3 Coefficient Statistics for Logistic Regression

Generalized Linear Model classification models generate the following coefficient statistics:

- Name of the predictor
- Coefficient estimate

- Standard error of the coefficient estimate
- Wald chi-square value of the coefficient estimate
- Probability of the Wald chi-square value
- Standardized estimate of the coefficient
- Lower and upper confidence bounds of the coefficient
- Exponentiated coefficient
- Exponentiated coefficient for the upper and lower confidence bounds of the coefficient

20.8.4 Global Model Statistics for Logistic Regression

Generalized Linear Model classification models generate the following statistics that describe the model as a whole:

- Akaike's criterion for the fit of the intercept only model
- Akaike's criterion for the fit of the intercept and the covariates (predictors) model
- Schwarz's criterion for the fit of the intercept only model
- Schwarz's criterion for the fit of the intercept and the covariates (predictors) model
- -2 log likelihood of the intercept only model
- -2 log likelihood of the model
- Likelihood ratio degrees of freedom
- Likelihood ratio chi-square probability value
- Pseudo R-square Cox an Snell
- Pseudo R-square Nagelkerke
- Dependent mean
- Percent of correct predictions
- Percent of incorrect predictions
- Percent of ties (probability for two cases is the same)
- Number of parameters (the number of coefficients, including the intercept)
- Number of rows
- Whether or not the model converged
- Whether or not a covariance matrix was computed.

20.8.5 Row Diagnostics for Logistic Regression

For logistic regression, the diagnostics table has the columns described in the following table. All the columns are `NUMBER`, except the `CASE_ID` and `TARGET_VALUE` columns, which preserve the type from the training data.

Table 20-2 Row Diagnostics Table for Logistic Regression

Column	Description
CASE_ID	Value of the case ID column
TARGET_VALUE	Value of the target value
TARGET_VALUE_PROB	Probability associated with the target value
HAT	Value of the diagonal element of the hat matrix
WORKING_RESIDUAL	Residual with respect to the adjusted dependent variable
PEARSON_RESIDUAL	The raw residual scaled by the estimated standard deviation of the target
DEVIANCE_RESIDUAL	Contribution to the overall goodness of fit of the model
C	Confidence interval displacement diagnostic
CBAR	Confidence interval displacement diagnostic
DIFDEV	Change in the deviance due to deleting an individual observation
DIFCHISQ	Change in the Pearson chi-square

21
k-Means

Oracle Machine Learning for SQL supports enhanced *k*-Means clustering algorithm. Learn how to use the algorithm.

Related Topics

- Clustering Algorithms
  Learn different clustering algorithms used in Oracle Machine Learning for SQL.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: k-Means
- Automatic Data Preparation
- Model Detail Views for k-Means
- OML4SQL Examples
- OML4R k-Means Example
- OML4R Code Examples

21.1 About *k*-Means

The *k*-Means algorithm is a distance-based clustering algorithm that partitions the data into a specified number of clusters.

Distance-based algorithms rely on a distance function to measure the similarity between cases. Cases are assigned to the nearest cluster according to the distance function used.

21.1.1 Oracle Machine Learning for SQL Enhanced *k*-Means

Implementation of *k*-Means in Oracle Machine Learning for SQL.

OML4SQL implements an enhanced version of the *k*-Means algorithm with the following features:

- **Distance function**: The algorithm supports Euclidean and Cosine distance functions. The default is Euclidean.
- **Scalable Parallel Model build**: The algorithm uses a very efficient method of initialization based on *Bahmani, Bahman, et al. "Scalable k-means++." Proceedings of the VLDB Endowment 5.7 (2012): 622-633*.

- **Cluster properties**: For each cluster, the algorithm returns the centroid, a histogram for each attribute, and a rule describing the hyperbox that encloses the majority of the data assigned to the cluster. The centroid reports the mode for categorical attributes and the mean and variance for numerical attributes.

This approach to *k*-Means avoids the need for building multiple *k*-Means models and provides clustering results that are consistently superior to the traditional *k*-Means.

21.1.2 Centroid

Defines a centroid in a cluster.

The **centroid** represents the most typical case in a cluster. For example, in a data set of customer ages and incomes, the centroid of each cluster would be a customer of average age and average income in that cluster. The centroid is a prototype. It does not necessarily describe any given case assigned to the cluster.

The attribute values for the centroid are the mean of the numerical attributes and the mode of the categorical attributes.

21.2 *k*-Means Algorithm Configuration

Learn about configuring the *k*-Means algorithm.

The Oracle Machine Learning for SQL enhanced *k*-Means algorithm supports several build-time settings. All the settings have default values. There is no reason to override the defaults unless you want to influence the behavior of the algorithm in some specific way.

You can configure *k*-Means by specifying the following considerations:

- Number of clusters
- Distance Function. The default distance function is Euclidean.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: k-Means for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

21.3 Data Preparation for *k*-Means

Learn about preparing data for *k*-Means algorithm.

Normalization is typically required by the *k*-Means algorithm. Automatic Data Preparation performs normalization for *k*-Means. If you do not use ADP, you must normalize numeric attributes before creating or applying the model.

When there are missing values in columns with simple data types (not nested), *k*-Means interprets them as missing at random. The algorithm replaces missing categorical values with the mode and missing numerical values with the mean.

When there are missing values in nested columns, *k*-Means interprets them as sparse. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*
- Prepare the Data

22
Minimum Description Length

Learn how to use Minimum Description Length, the supervised technique for calculating attribute importance.

Related Topics

- Feature Selection
  Learn how to perform feature selection and attribute importance.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Automatic Data Preparation
- Model Detail Views for Minimum Description Length
- OML4SQL Examples
- OML4R Code Examples

22.1 About MDL

Minimum Description Length (MDL) is an information theoretic model selection principle.

Information theoretic model selection principle is an important concept in information theory (the study of the quantification of information) and in learning theory (the study of the capacity for generalization based on empirical data).

MDL assumes that the simplest, most compact representation of the data is the best and most probable explanation of the data. The MDL principle is used to build Oracle Machine Learning for SQL attribute importance models.

The build process for attribute importance supports parallel execution.

Related Topics

- *Oracle Database VLDB and Partitioning Guide*

22.1.1 Compression and Entropy

Data compression is the process of encoding information using fewer **bits** than what the original representation uses. The MDL Principle is based on the notion that the shortest description of the data is the most probable. In typical instantiations of this principle, a model is used to compress the data by reducing the uncertainty (entropy) as discussed below. The description of the data includes a description of the model and the data as described by the model.

Entropy is a measure of uncertainty. It quantifies the uncertainty in a random variable as the information required to specify its value. **Information** in this sense is defined as the number of yes/no questions known as **bits** (encoded as 0 or 1) that must be answered for a complete specification. Thus, the information depends upon the number of values that variable can assume.

For example, if the variable represents the sex of an individual, then the number of possible values is two: female and male. If the variable represents the salary of individuals expressed in whole dollar amounts, then the values can be in the range $0-$10B, or billions of unique values. Clearly it takes more information to specify an exact salary than to specify an individual's sex.

22.1.1.1 Values of a Random Variable: Statistical Distribution

Information (the number of bits) depends on the statistical distribution of the values of the variable as well as the number of values of the variable. If we are judicious in the choice of Yes/No questions, then the amount of information for salary specification cannot be as much as it first appears. Most people do not have billion dollar salaries. If most people have salaries in the range $32000-$64000, then most of the time, it requires only 15 questions to discover their salary, rather than the 30 required, if every salary from $0-$1000000000 were equally likely. In the former example, if the persons were known to be pregnant, then their sex is known to be female. There is no uncertainty, no Yes/No questions need be asked. The entropy is 0.

22.1.1.2 Values of a Random Variable: Significant Predictors

Suppose that for some random variable there is a predictor that when its values are known reduces the uncertainty of the random variable. For example, knowing whether a person is pregnant or not, reduces the uncertainty of the random variable sex-of-individual. This predictor seems like a valuable feature to include in a model. How about name? Imagine that if you knew the name of the person, you would also know the person's sex. If so, the name predictor would seemingly reduce the uncertainty to zero. However, if names are unique, then what was gained? Is the person named Sally? Is the person named George?... We would have as many Yes/No predictors in the name model as there are people. Therefore, specifying the name model would require as many bits as specifying the sex of each person.

22.1.1.3 Total Entropy

For a random variable, X, the **total entropy** is defined as minus the Probability(X) multiplied by the log to the base 2 of the Probability(X). This can be shown to be the variable's most efficient encoding.

22.1.2 Model Size

A Minimum Description Length (MDL) model takes into consideration the size of the model as well as the reduction in uncertainty due to using the model. Both model size and entropy are measured in bits. For our purposes, both numeric and categorical predictors are binned. Thus the size of each single predictor model is the number of predictor bins. The uncertainty is reduced to the within-bin target distribution.

22.1.3 Model Selection

Minimum Description Length (MDL) considers each attribute as a simple predictive model of the target class. **Model selection** refers to the process of comparing and ranking the single-predictor models.

MDL uses a communication model for solving the model selection problem. In the communication model there is a sender, a receiver, and data to be transmitted.

These single predictor models are compared and ranked with respect to the MDL metric, which is the relative compression in bits. MDL penalizes model complexity to avoid over-fit. It is a principled approach that takes into account the complexity of the predictors (as models) to make the comparisons fair.

22.1.4 The MDL Metric

Attribute importance uses a two-part code as the metric for transmitting each unit of data. The first part (preamble) transmits the model. The parameters of the model are the target probabilities associated with each value of the prediction.

For a target with j values and a predictor with k values, n_i (i= 1,..., k) rows per value, there are C_i, the combination of j-1 things taken n_i-1 at a time possible conditional probabilities. The size of the preamble in bits can be shown to be $\text{Sum}(\log_2(C_i))$, where the sum is taken over k. Computations like this represent the penalties associated with each single prediction model. The second part of the code transmits the target values using the model.

It is well known that the most compact encoding of a sequence is the encoding that best matches the probability of the symbols (target class values). Thus, the model that assigns the highest probability to the sequence has the smallest target class value transmission cost. In bits, this is the $\text{Sum}(\log_2(p_i))$, where the p_i are the predicted probabilities for row $_i$ associated with the model.

The predictor rank is the position in the list of associated description lengths, smallest first.

22.2 Data Preparation for MDL

Learn about preparing data for Minimum Description Length (MDL).

Automatic Data Preparation performs supervised binning for MDL. Supervised binning uses decision trees to create the optimal bin boundaries. Both categorical and numerical attributes are binned.

MDL handles missing values naturally as missing at random. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors. Missing values in nested columns are interpreted as sparse. Missing values in columns with simple data types are interpreted as missing at random.

If you choose to manage your own data preparation, keep in mind that MDL usually benefits from binning. However, the discriminating power of an attribute importance model can be significantly reduced when there are outliers in the data and external equal-width binning is used. This technique can cause most of the data to concentrate in a few bins (a single bin in extreme cases). In this case, quantile binning is a better solution.

> **See Also:**
>
> DBMS_DATA_MINING — Automatic Data Preparation for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- Prepare the Data

23

Multivariate State Estimation Technique - Sequential Probability Ratio Test

The Multivariate State Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT) algorithm monitors critical processes and detects subtle anomalies.

- About Multivariate State Estimation Technique - Sequential Probability Ratio Test
- Score an MSET-SPRT Model

Related Topics

- Anomaly Detection
  Learn how to detect rare cases in the data through anomaly detection - an unsupervised function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Multivariate State Estimation Technique - Sequential Probability Ratio Test
- Automatic Data Preparation
- Model Detail View for Multivariate State Estimation Technique - Sequential Probability Ratio Test
- OML4SQL Examples
- OML4R Code Examples

23.1 About Multivariate State Estimation Technique - Sequential Probability Ratio Test

Multivariate state Estimation Technique - Sequential Probability Ratio Test (MSET-SPRT) is an algorithm for anomaly detection and statistical testing.

MSET is a nonlinear, nonparametric anomaly detection machine learning technique that calibrates the expected behavior of a system based on historical data from the normal operational sequence of monitored signals. It incorporates the learned behavior of a system into a persistent model that represents the normal estimated behavior. You can deploy the model to evaluate a subsequent stream of live signal vectors using OML4SQL scoring functions. To form a hypothesis as to the overall health of the system, these functions calculate the difference between the estimated and the actual signal values (residuals) and use SPRT calculations to determine whether any of the signals have become degraded.

To build a good model, MSET requires sufficient historical data that adequately captures all normal modes of behavior of the system. Incomplete data results in false alerts when the system enters a mode of operation that was poorly represented in the historical data. MSET assumes that the characteristics of the data being monitored do not change over time. Once deployed, MSET is a stationary model and does not evolve as it monitors a data stream.

Both MSET and SPRT operate on continuous time-ordered sensor data. If the raw data stream needs to be pre-processed or sampled, you must do that before you pass the data to the MSET-SPRT model.

The `ALGO_MSET_SPRT` algorithm is designated as a classification machine learning function. It generates a model in which each data row is labeled as either normal or anomalous. For anomalous predictions, the prediction details provide a list of the sensors that show the anomaly and a weight.

When creating an MSET-SPRT model with the `DBMS_DATA_MINING.CREATE_MODEL` function, use the `case_id` argument to provide a unique row identifier for the time-ordered data that the algorithm requires. The build is then able to sort the training data and create windows for sampling and variance estimation. If you do not provide a `case_id`, then an exception occurs.

MSET-SPRT supports only numeric data. An exception occurs if other column types are in the build data.

When the number of sensors is very high, MSET-SPRT leverages random projections to improve the scalability and robustness of the algorithm. Random projections is a technique that reduces dimensionality while preserving pairwise distances. By randomly projecting the sensor data, the problem is solved in a distance-preserving, lower-dimension space. The MSET hypothesis testing approach is applied on the projected data where each random projection can be viewed as a Monte Carlo simulation of system health. The overall probability of an anomaly follows a binomial distribution with the number of projections as the number of trials and the number of alerting projections as the number of successes.

> **Note:**
>
> An MSET-SPRT model with random projections does not produce prediction details. When random projections are employed, the nature of the prediction output changes. The prediction captures the global health of the system and it is not possible to attribute the cause to individual attributes. Therefore, `PREDICTION_DETAILS` returns an empty list.

> **See Also:**
>
> DBMS_DATA_MINING - Algorithm Settings: Multivariate State Estimation Technique - Sequential Probability Ratio Test for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- DBMS_DATA_MINING - Algorithm Settings: Multivariate State Estimation Technique - Sequential Probability Ratio Test

23.2 Score an MSET-SPRT Model

Scoring data with MSET-SPRT models is similar to scoring with classification algorithms, except that the SPRT methodology relies on ordered data because it tracks gradual shifts over multiple MSET predictions.

This is different than the typical usage of Oracle Database SQL prediction functions, which do not keep state information between rows.

The following functions are supported: `PREDICTION`, `PREDICTION_COST`, `PREDICTION_DETAILS`, `PREDICTION_PROBABILITY`, and `PREDICTION_SET`. These functions have syntax new in Oracle Database 21c for scoring MSET-SPRT models. That syntax has an `ORDER BY` clause to order and window the historical data.

The prediction functions return the following information:

- `PREDICTION` indicates whether the record is flagged as anomalous. It uses the same automatically generated labels as one-class SVM models: 1 for normal and 0 for anomalous.
- `PREDICTION_COST` performs an auto-cost analysis or a user-specified cost. A user-specified cost typically assigns a higher cost to false positives than to false negatives.
- `PREDICTION_DETAILS` specify the signals that support the prediction along with a weight.
- `PREDICTION_PROBABILITY` conveys a measure of certainty based on the consolidation logic.
- `PREDICTION_SET` returns the set of predictions (0, 1) and the corresponding prediction probabilities for each observation.

> **Note:**
>
> If the values in one or more of the columns specified in the `ORDER BY` clause are not unique, or do not represent a true chronology of data sample values, the SPRT predictions are not guaranteed to be meaningful or consistent between query executions.

Unlike other classification models, an MSET-SPRT model has no obvious probability measure associated with the anomalous label for the record as a whole. However, the consolidation logic can produce a measure of uncertainty in place of probability. For example, if an alert is raised for 2 anomalies over a window of 5 observations, a certainty of 0.5 is reported when 2 anomalies are seen within the 5 observation window. The certainty increases if more than 3 anomalies are seen and decreases if no anomalies are seen.

The `PREDICTION_DETAILS` function accommodates output of varying forms and can convey the required information regarding the individual signals that triggered an alarm. When random projections are engaged, only the overall `PREDICTION` and `PREDICTION_PROBABILITY` are computed and `PREDICTION_DETAILS` are not reported.

You must score the historical data in order to tune the SPRT parameters, such as false alerts and miss rates or consolidation logic, before you deploy the MSET model.

The SPRT parameters are embedded in the model object to facilitate deployment. While scoring in the database is needed for parameter tuning and forensic analysis on historical data, monitoring a stream of sensor data is more easily done outside of the database in an IoT service or on the edge device itself.

You can build and score an MSET-SPRT model as a partitioned model if the same columns that you use to build the model are present in the input scoring data set. If those columns are not present, the query results in an error.

Related Topics

- SQL Scoring Functions
- SQL Scoring Functions
- MSET_SPRT example on GitHub

24
Naive Bayes

Learn how to use the Naive Bayes classification algorithm.

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Naive Bayes
- Automatic Data Preparation
- Model Detail Views for Naive Bayes
- OML4SQL Examples
- OML4R Naive Bayes Example
- OML4R Code Examples

24.1 About Naive Bayes

Naive Bayes algorithm is based on conditional probabilities. It uses Bayes' theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

Bayes' theorem finds the probability of an event occurring given the probability of another event that has already occurred. If `B` represents the dependent event and `A` represents the prior event, Bayes' theorem can be stated as follows.

> **Note:**
>
> Prob(B given A) = Prob(A and B)/Prob(A)

To calculate the probability of `B` given `A`, the algorithm counts the number of cases where `A` and `B` occur together and divides it by the number of cases where `A` occurs alone.

Example 24-1 Use Bayes' Theorem to Predict an Increase in Spending

Suppose you want to determine the likelihood that a customer under 21 increases spending. In this case, the prior condition (A) is "under 21," and the dependent condition (B) is "increase spending."

If there are 100 customers in the training data and 25 of them are customers under 21 who have increased spending, then:

Prob(A and B) = 25%

If 75 of the 100 customers are under 21, then:

Prob(A) = 75%

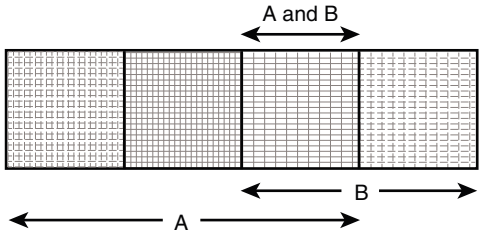
Bayes' theorem predicts that 33% of customers under 21 are likely to increase spending (25/75).

The cases where both conditions occur together are referred to as **pairwise**. In Example 24-1, 25% of all cases are pairwise.

The cases where only the prior event occurs are referred to as **singleton**. In Example 24-1, 75% of all cases are singleton.

A visual representation of the conditional relationships used in Bayes' theorem is shown in the following figure.

Figure 24-1 Conditional Probabilities in Bayes' Theorem



P(A) = 3/4
P(B) = 2/4
P(A and B) = P(AB) = 1/4
P(A|B) = P(AB) / P(B) = (1/4) / (2/4) = 1/2
P(B|A) = P(AB) / P(A) = (1/4) / (3/4) = 1/3

For purposes of illustration, Example 24-1 and Figure 24-1 show a dependent event based on a single independent event. In reality, the Naive Bayes algorithm must usually take many independent events into account. In Example 24-1, factors such as income, education, gender, and store location might be considered in addition to age.

Naive Bayes makes the assumption that each predictor is conditionally independent of the others. For a given target value, the distribution of each predictor is independent of the other predictors. In practice, this assumption of independence, even when violated, does not degrade the model's predictive accuracy significantly, and makes the difference between a fast, computationally feasible algorithm and an intractable one.

Sometimes the distribution of a given predictor is clearly not representative of the larger population. For example, there might be only a few customers under 21 in the

training data, but in fact there are many customers in this age group in the wider customer base. To compensate for this, you can specify **prior probabilities** when training the model.

Related Topics

- Priors and Class Weights
  Learn about Priors and Class Weights in a classification model to produce a useful result.

24.1.1 Advantages of Naive Bayes

Learn about the advantages of Naive Bayes.

The Naive Bayes algorithm affords fast, highly scalable model building and scoring. It scales linearly with the number of predictors and rows.

The build process for Naive Bayes supports parallel execution. (Scoring supports parallel execution irrespective of the algorithm.)

Naive Bayes can be used for both binary and multiclass classification problems.

Related Topics

- *Oracle Database VLDB and Partitioning Guide*

24.2 Tuning a Naive Bayes Model

Introduces about probability calculation of pairwise occurrences and percentage of singleton occurrences.

Naive Bayes calculates a probability by dividing the percentage of pairwise occurrences by the percentage of singleton occurrences. If these percentages are very small for a given predictor, they probably do not contribute to the effectiveness of the model. Occurrences below a certain threshold can usually be ignored.

The following build settings are available for adjusting the probability thresholds. You can specify:

- The minimum percentage of pairwise occurrences required for including a predictor in the model.
- The minimum percentage of singleton occurrences required for including a predictor in the model .

The default thresholds work well for most models, so you need not adjust these settings.

> **See Also:**
>
> DBMS_DATA_MINING — Algorithm Settings: Naive Bayes for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

24.3 Data Preparation for Naive Bayes

Learn about preparing the data for Naive Bayes.

Automatic Data Preparation (ADP) performs supervised binning for Naive Bayes. Supervised binning uses decision trees to create the optimal bin boundaries. Both categorical and numeric attributes are binned.

Naive Bayes handles missing values naturally as missing at random. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors. Missing values in nested columns are interpreted as sparse. Missing values in columns with simple data types are interpreted as missing at random.

If you choose to manage your own data preparation, keep in mind that Naive Bayes usually requires binning. Naive Bayes relies on counting techniques to calculate probabilities. Columns must be binned to reduce the cardinality as appropriate. Numerical data can be binned into ranges of values (for example, low, medium, and high), and categorical data can be binned into meta-classes (for example, regions instead of cities). Equi-width binning is not recommended, since outliers cause most of the data to concentrate in a few bins, sometimes a single bin. As a result, the discriminating power of the algorithms is significantly reduced

Related Topics

- Prepare the Data

25
Neural Network

Learn about the Neural Network algorithms for regression and classification machine learning functions.

- About Neural Network
- Data Preparation for Neural Network
- Neural Network Algorithm Configuration
- Scoring with Neural Network

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- Regression
  Learn how to predict a continuous numerical target through regression - the supervised machine learning function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: Neural Network
- Automatic Data Preparation
- Model Detail Views for Neural Network
- OML4SQL Examples
- OML4R Neural Network Example
- OML4R Code Examples

25.1 About Neural Network

The Neural Network algorithm in Oracle Machine Learning for SQL is designed for machine learning functions like classification and regression.

In machine learning, an artificial neural network is an algorithm inspired from biological neural network and is used to estimate or approximate functions that depend on a large number of generally unknown inputs. An artificial neural network is composed of a large number of interconnected neurons which exchange messages between each other to solve specific problems. They learn by examples and tune the weights of the connections among the neurons during the learning process. The Neural Network algorithm is capable of solving a wide variety of tasks such as computer vision, speech recognition, and various complex business problems.

Related Topics

- Regression
  Learn how to predict a continuous numerical target through regression - the supervised machine learning function.

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.

25.1.1 Neurons and Activation Functions

Neurons are the building blocks of a neural network.

A neuron takes one or more inputs having different weights and has an output which depends on the inputs. The output is achieved by adding up inputs of each neuron with weights and feeding the sum into the activation function.

A Sigmoid function is usually the most common choice for activation function but other non-linear functions, piecewise linear functions or step functions are also used. The Rectified Linear Units function `NNET_ACTIVATIONS_RELU` is a commonly used activation function that addresses the vanishing gradient problem for larger neural networks.

The following are some examples of activation functions:

- Logistic Sigmoid function
- Linear function
- Tanh function
- Arctan function
- Bipolar sigmoid function
- Rectified Linear Units

25.1.2 Loss or Cost function

A loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event.

An optimization problem seeks to minimize a loss function. The form of loss function is chosen based on the nature of the problem and mathematical needs.

The following are the different loss functions for different scenarios:

- Binary classification: cross entropy function.
- Multi-class classification: softmax function.
- Regression: squared error function.

25.1.3 Forward-Backward Propagation

Understand forward-backward propagation.

Forward propagation computes the loss function value by weighted summing the previous layer neuron values and applying activation functions. Backward propagation calculates the gradient of a loss function with respect to all the weights in the network. The weights are initialized with a set of random numbers uniformly distributed within a region specified by user (by setting weights boundaries), or region defined by the number of nodes in the adjacent layers (data driven). The gradients are fed to an optimization method which in turn uses them to update the weights, in an attempt to minimize the loss function.

25.1.4 Optimization Solvers

An optimization solver is a function that searches for the optimal solution of the loss function to find the extreme value (maximum or minimum) of the loss (cost) function.

Oracle Machine Learning implements Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) together with line search and the Adam solver.

Limited-memory Broyden–Fletcher–Goldfarb–Shanno Solver

L-BFGS is a Quasi-Newton method. This method uses rank-one updates specified by gradient evaluations to approximate a Hessian matrix. This method only needs a limited amount of memory. L-BFGS is used to find the descent direction and line search is used to find the appropriate step size. The number of historical copies kept in the L-BFGS solver is defined by the `LBFGS_HISTORY_DEPTH` solver setting. When the number of iterations is smaller than the history depth, the Hessian computed by L-BFGS is accurate. When the number of iterations is larger than the history depth, the Hessian computed by L-BFGS is an approximation. Therefore, the history depth should not be too small or too large to avoid making the computation too slow. Typically, the value is between `3` and `10`.

Adam Solver

Adam is an extension to stochastic gradient descent that uses mini-batch optimization. The L-BFGS solver may be a more stable solver whereas the Adam solver can make progress faster by seeing less data. Adam is computationally efficient, with little memory requirements, and is well-suited for problems that are large in terms of data or parameters or both.

25.1.5 Regularization

Understand regularization.

Regularization refers to a process of introducing additional information to solve an ill-posed problem or to prevent over-fitting. Ill-posed or over-fitting can occur when a statistical model describes random errors or noise instead of the underlying relationship. Typical regularization techniques include L1-norm regularization, L2-norm regularization, and held-aside.

Held-aside is usually used for large training date sets whereas L1-norm regularization and L2-norm regularization are mostly used for small training date sets.

25.1.6 Convergence Check

This checks if the optimal solution has been reached and if the iterations of the optimization has come to an end.

In L-BFGS solver, the convergence criteria includes maximum number of iterations, infinity norm of gradient, and relative error tolerance. For held-aside regularization, the convergence criteria checks the loss function value of the test data set, as well as the best model learned so far. The training is terminated when the model becomes worse for a specific number of iterations (specified by `NNET_HELDASIDE_MAX_FAIL`), or the loss function is close to zero, or the relative error on test data is less than the tolerance.

25.1.7 LBFGS_SCALE_HESSIAN

Defines `LBFGS_SCALE_HESSIAN`.

It specifies how to set the initial approximation of the inverse Hessian at the beginning of each iteration. If the value is set to be `LBFGS_SCALE_HESSIAN_ENABLE`, then we approximate the initial inverse Hessian with Oren-Luenberger scaling. If it is set to be `LBFGS_SCALE_HESSIAN_DISABLE`, then we use identity as the approximation of the inverse Hessian at the beginning of each iteration.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

25.1.8 NNET_HELDASIDE_MAX_FAIL

Defines `NNET_HELDASIDE_MAX_FAIL`.

Validation data (held-aside) is used to stop training early if the network performance on the validation data fails to improve or remains the same for `NNET_HELDASIDE_MAX_FAIL` epochs in a row.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

25.2 Data Preparation for Neural Network

Learn about preparing data for the Neural Network algorithm.

The algorithm automatically "explodes" categorical data into a set of binary attributes, one per category value. Oracle Machine Learning for SQL algorithms automatically handle missing values and therefore, missing value treatment is not necessary.

The algorithm automatically replaces missing categorical values with the mode and missing numerical values with the mean. The algorithm requires the normalization of numeric input and it uses z-score normalization. The normalization occurs only for two-dimensional numeric columns (not nested). Normalization places the values of numeric attributes on the same scale and prevents attributes with a large original scale from biasing the solution. Neural Network scales the numeric values in nested columns by the maximum absolute value seen in the corresponding columns.

Related Topics

- Prepare the Data

25.3 Neural Network Algorithm Configuration

Configure the Neural Network algorithm.

Specify Nodes Per Layer

```
INSERT INTO SETTINGS_TABLE (setting_name, setting_value) VALUES
                   ('NNET_NODES_PER_LAYER', '2,3');
```

Specify Activation Functions Per Layer

`NNET_ACTIVATIONS` setting specifies the activation functions or hidden layers.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Neural Network for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

25.4 Scoring with Neural Network

Learn to score with a Neural Network algorithm.

Scoring with Neural Network is the same as any other classification or regression algorithm. The following functions are supported: `PREDICTION`, `PREDICTION_PROBABILITY`, `PREDICTION_COST`, `PREDICTION_SET`, and `PREDICTION_DETAILS`.

Related Topics

- *Oracle Database SQL Language Reference*

26
Non-Negative Matrix Factorization

Learn how to use Non-Negative Matrix Factorization (NMF), an unsupervised algorithm, that Oracle Machine Learning for SQL uses for feature extraction.

Related Topics

- Feature Extraction
  Learn how to perform attribute reduction using feature extraction as an unsupervised function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: Non-Negative Matrix Factorization
- Automatic Data Preparation
- Model Detail Views for Non-Negative Matrix Factorization
- OML4SQL Examples
- OML4R Non-Negative Matrix Factorization Example
- OML4R Code Examples

> **See Also:**
>
> Paper "Learning the Parts of Objects by Non-Negative Matrix Factorization" by D. D. Lee and H. S. Seung in *Nature* (401, pages 788-791, 1999)

26.1 About NMF

Non-Negative Matrix Factorization is useful when there are many attributes and the attributes are ambiguous or have weak predictability. By combining attributes, NMF can produce meaningful patterns, topics, or themes. NMF is a feature extraction algorithm.

Each feature created by NMF is a linear combination of the original attribute set. Each feature has a set of coefficients, which are a measure of the weight of each attribute on the feature. There is a separate coefficient for each numerical attribute and for each distinct value of each categorical attribute. The coefficients are all non-negative.

26.1.1 Matrix Factorization

Non-Negative Matrix Factorization uses techniques from multivariate analysis and linear algebra. It decomposes the data as a matrix *M* into the product of two lower ranking matrices *W* and *H*. The sub-matrix *W* contains the NMF basis; the sub-matrix *H* contains the associated coefficients (weights).

The algorithm iteratively modifies of the values of *W* and *H* so that their product approaches *M*. The technique preserves much of the structure of the original data and guarantees that both basis and weights are non-negative. The algorithm terminates when the approximation error converges or a specified number of iterations is reached.

The NMF algorithm must be initialized with a seed to indicate the starting point for the iterations. Because of the high dimensionality of the processing space and the fact that there is no global minimization algorithm, the appropriate initialization can be critical in obtaining meaningful results. Oracle Machine Learning for SQL uses a random seed that initializes the values of W and H based on a uniform distribution. This approach works well in most cases.

26.1.2 Scoring with NMF

Non-Negative Matrix Factorization (NMF) can be used as a pre-processing step for dimensionality reduction in classification, regression, clustering, and other machine learning tasks. Scoring an NMF model produces data projections in the new feature space. The magnitude of a projection indicates how strongly a record maps to a feature.

The SQL scoring functions for feature extraction support NMF models. When the functions are invoked with the analytical syntax, the functions build and apply a transient NMF model. The feature extraction functions are: `FEATURE_DETAILS`, `FEATURE_ID`, `FEATURE_SET`, and `FEATURE_VALUE`.

Related Topics

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

26.1.3 Text Analysis with NMF

Learn about text analysis with Non-Negative Matrix Factorization (NMF).

NMF is especially well-suited for analyzing text. In a text document, the same word can occur in different places with different meanings. For example, "hike" can be applied to the outdoors or to interest rates. By combining attributes, NMF introduces context, which is essential for explanatory power:

- "hike" + "mountain" -> "outdoor sports"
- "hike" + "interest" -> "interest rates"

Related Topics

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

26.2 Tuning the NMF Algorithm

Learn about configuring parameters for Non-Negative Matrix Factorization (NMF).

Oracle Machine Learning for SQL supports five configurable parameters for NMF. All of them have default values which are appropriate for most applications of the algorithm. The NMF settings are:

- Number of features. By default, the number of features is determined by the algorithm.
- Convergence tolerance. The default is .05.
- Number of iterations. The default is 50.
- Random seed. The default is -1.
- Non-negative scoring. You can specify whether negative numbers must be allowed in scoring results. By default they are allowed.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Non-Negative Matrix Factorization for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

26.3 Data Preparation for NMF

You can use Automatic Data Preparation (ADP) or supply your transformation like binning or normalization to prepare the data for Non-Negative Matrix Factorization (NMF).

ADP normalizes numerical attributes for NMF.

When there are missing values in columns with simple data types (not nested), NMF interprets them as missing at random. The algorithm replaces missing categorical values with the mode and missing numerical values with the mean.

When there are missing values in nested columns, NMF interprets them as sparse. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors.

If you choose to manage your own data preparation, keep in mind that outliers can significantly impact NMF. Use a clipping transformation before binning or normalizing. NMF typically benefits from normalization. However, outliers with min-max normalization cause poor matrix factorization. To improve the matrix factorization, you need to decrease the error tolerance. This in turn leads to longer build times.

Related Topics

- Prepare the Data

27
O-Cluster

Learn how to use orthogonal partitioning clustering (O-Cluster), an Oracle-proprietary clustering algorithm.

Related Topics

- Clustering Algorithms
  Learn different clustering algorithms used in Oracle Machine Learning for SQL.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: O-Cluster
- Automatic Data Preparation
- Model Detail Views for O-Cluster
- OML4SQL Examples
- OML4R O-Cluster Example
- OML4R Code Examples

> **See Also:**
>
> Campos, M.M., Milenova, B.L., "Clustering Large Databases with Numeric and Nominal Values Using Orthogonal Projections", Oracle Data Mining Technologies, Oracle Corporation.

27.1 About O-Cluster

O-Cluster is a fast, scalable grid-based clustering algorithm well-suited for analysing large, high-dimensional data sets. The algorithm can produce high quality clusters without relying on user-defined parameters.

The objective of O-Cluster is to identify areas of high density in the data and separate the dense areas into clusters. It uses axis-parallel uni-dimensional (orthogonal) data projections to identify the areas of density. The algorithm looks for splitting points that result in distinct clusters that do not overlap and are balanced in size.

O-Cluster operates recursively by creating a binary tree hierarchy. The number of leaf clusters is determined automatically. The algorithm can be configured to limit the maximum number of clusters.

27.1.1 Partitioning Strategy

Partitioning strategy refers to the process of discovering areas of density in the attribute histograms. The process differs for numerical and categorical data. When both are present in the data, the algorithm performs the searches separately and then compares the results.

In choosing a partition, the algorithm balances two objectives: finding well separated clusters, and creating clusters that are balanced in size. The following paragraphs detail how partitions for numerical and categorical attributes are identified.

27.1.1.1 Partitioning Numerical Attributes

To find the best valid cutting plane, O-Cluster searches the attribute histograms for bins of low density (valleys) between bins of high density (peaks).

O-Cluster attempts to find a pair of peaks with a valley between them where the difference between the peak and valley histogram counts is statistically significant.

A **sensitivity** level parameter specifies the lowest density that may be considered a peak. Sensitivity is an optional parameter for numeric data. It may be used to filter the splitting point candidates.

27.1.1.2 Partitioning Categorical Attributes

Categorical values do not have an intrinsic order associated with them. Therefore it is impossible to apply the notion of histogram peaks and valleys that is used to partition numerical values. Instead the counts of individual values form a histogram.

Bins with large counts are interpreted as regions with high density. The clustering objective is to separate these high-density areas and effectively decrease the entropy (randomness) of the data.

O-Cluster identifies the histogram with highest entropy along the individual projections. Entropy is measured as the number of bins above **sensitivity** level. O-Cluster places the two largest bins into separate partitions, thereby creating a splitting predicate. The remainder of the bins are assigned randomly to the two resulting partitions.

27.1.2 Active Sampling

The O-Cluster algorithm operates on a data buffer of a limited size. It uses an active sampling mechanism to handle data sets that do not fit into memory.

After processing an initial random sample, O-Cluster identifies cases that are of no further interest. Such cases belong to *frozen* partitions where further splitting is highly unlikely. These cases are replaced with examples from *ambiguous* regions where further information (additional cases) is needed to find good splitting planes and continue partitioning. A partition is considered ambiguous if a valid split can only be found at a lower confidence level.

Cases associated with frozen partitions are marked for deletion from the buffer. They are replaced with cases belonging to ambiguous partitions. The histograms of the ambiguous partitions are updated and splitting points are reevaluated.

27.1.3 Process Flow

At a high level, O-Cluster algorithm evaluates, splits the data into new partition, and searches for cutting planes inside the new partitions.

The O-Cluster algorithm evaluates possible splitting points for all projections in a partition, selects the best one, and splits the data into two new partitions. The algorithm proceeds by searching for good cutting planes inside the newly created partitions. Thus, O-Cluster creates a binary tree structure that divides the input space into rectangular regions with no overlaps or gaps.

The main processing stages are:

1. Load the buffer. Assign all cases from the initial buffer to a single active root partition.
2. Compute histograms along the orthogonal uni-dimensional projections for each active partition.
3. Find the best splitting points for active partitions.
4. Flag ambiguous and frozen partitions.
5. When a valid separator exists, split the active partition into two new active partitions and start over at step 2.
6. Reload the buffer after all recursive partitioning on the current buffer is completed. Continue loading the buffer until either the buffer is filled again, or the end of the data set is reached, or until the number of cases is equal to the data buffer size.

> **Note:**
>
> O-Cluster requires at most one pass through the data

27.1.4 Scoring

The clusters discovered by O-Cluster are used to generate a Bayesian probability model that can be used to score new data.

The generated probability model is a mixture model where the mixture components are represented by a product of independent normal distributions for numerical attributes and multinomial distributions for categorical attributes.

27.2 Tuning the O-Cluster Algorithm

You can configure build-time settings for O-Cluster.

The O-Cluster algorithm supports two build-time settings. Both settings have default values. There is no reason to override the defaults unless you want to influence the behavior of the algorithm in some specific way.

You can configure O-Cluster by specifying any of the following:

- **Buffer size** — Size of the processing buffer.

- **Sensitivity factor** — A fraction that specifies the peak density required for separating a new cluster.

> **See Also:**
>
> DBMS_DATA_MINING — Algorithm Settings: O-Cluster for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- Active Sampling
  The O-Cluster algorithm operates on a data buffer of a limited size. It uses an active sampling mechanism to handle data sets that do not fit into memory.
- Partitioning Strategy

27.3 Data Preparation for O-Cluster

Use Automatic Data Preparation (ADP) to prepare the data for O-Cluster.

ADP bins numerical attributes for O-Cluster. It uses a specialized form of equi-width binning that computes the number of bins per attribute automatically. Numerical columns with all nulls or a single value are removed. O-Cluster handles missing values naturally as missing at random.

> **Note:**
>
> O-Cluster does not support nested columns, sparse data, or unstructured text.

Related Topics

- Prepare the Data

27.3.1 User-Specified Data Preparation for O-Cluster

You can prepare the data for O-Cluster by considering the points listed here.

Keep the following in mind if you choose to prepare the data for O-Cluster:

- O-Cluster does not necessarily use all the input data when it builds a model. It reads the data in batches (the default batch size is 50000). It only reads another batch if it believes, based on statistical tests, that uncovered clusters can still exist.
- Binary attributes must be declared as categorical.

- Automatic equi-width binning is highly recommended. The bin identifiers are expected to be positive consecutive integers starting at 1.
- The presence of outliers can significantly impact clustering algorithms. Use a clipping transformation before binning or normalizing. Outliers with equi-width binning can prevent O-Cluster from detecting clusters. As a result, the whole population appears to fall within a single cluster.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

28
R Extensibility

 This topic applies only to Oracle on-premises.

Learn how to build an analytics model and score in R. The R extensible algorithms are enhanced to support and register additional algorithms for users who use SQL and graphical user interface.

- Oracle Machine Learning for SQL with R Extensibility
- Scoring with R
- About Algorithm Metadata Registration

Related Topics

- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: ALGO_EXTENSIBLE_LANG
- OML4SQL Examples
- OML4R Extensible R Example
- OML4R Code Examples

28.1 Oracle Machine Learning for SQL with R Extensibility

Learn how you can use Oracle Machine Learning for SQL to build, score, and view machine learning models as well as R models.

The OML4SQL framework is enhanced extending the OML4SQL algorithm set with algorithms from the open source R ecosystem. Oracle Machine Learning for SQL is implemented in the Oracle Database kernel. The OML4SQL models are Database schema objects. With the extensibility enhancement, the OML4SQL framework can build, score, and view both OML4SQL models and R models.

Registration of R scripts

The R engine on the database server runs the R scripts to build, score, and view R models. These R scripts must be registered with the database beforehand by a privileged user with `rqAdmin` role. You must first install Oracle Machine Learning for R to register the R scripts.

Functions of Oracle Machine Learning for SQL with R Model

The following functions are supported for an R model:

- OML4SQL `DBMS_DATA_MINING` package is enhanced to support R model. For example, `CREATE_MODEL` and `DROP_MODEL`.
- `MODEL VIEW` to get the R model details about a single model and a partitioned model.

- OML4SQL SQL functions are enhanced to operate with the R model functions. For example, `PREDICTION` and `CLUSTER_ID`.

R model extensibility supports the following OML4SQL functions:

- Association
- Attribute Importance
- Regression
- Classification
- Clustering
- Feature Extraction

28.2 About Algorithm Metadata Registration

Algorithm metadata registration allows for a uniform and consistent approach of registering new algorithm functions and their settings.

Users have the ability to add new R-based algorithms through the registration process. The new algorithms appear as available within Oracle Machine Learning for R and within the appropriate machine learning functions. Based on the registration metadata, the settings page is dynamically rendered. The advantages are as follows:

- Manage R-based algorithms more easily
- Specify R-based algorithm for model build
- Clean individual properties in JSON structure
- Share R-based algorithm across user

Algorithm metadata registration extends the machine learning model capability of Oracle Machine Learning for SQL.

> **See Also:**
>
> DBMS_DATA_MINING — Algorithm Settings: ALGO_EXTENSIBLE_LANG for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- Create Model Using Registration Information
- FETCH_JSON_SCHEMA Procedure
- REGISTER_ALGORITHM Procedure
- JSON Schema for R Extensible Algorithm

28.3 Scoring with R

Learn how to build and score with an Oracle Machine Learning for R model.

For more information, see *Oracle Machine Learning for SQL User's Guide*

29
Random Forest

Learn how to use Random Forest as a classification algorithm.

- About Random Forest
- Building a Random Forest
- Scoring with Random Forest

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Random Forest
- Automatic Data Preparation
- Model Detail Views for Random Forest
- OML4SQL Examples
- OML4R Random Forest Example
- OML4R Code Examples

29.1 About Random Forest

Random Forest is a classification algorithm that builds an **ensemble** (also called **forest**) of trees.

The algorithm builds a number of Decision Tree models and predicts using the ensemble. An individual decision tree is built by choosing a random sample from the training data set as the input. At each node of the tree, only a random sample of predictors is chosen for computing the split point. This introduces variation in the data used by the different trees in the forest. The parameters `RFOR_SAMPLING_RATIO` and `RFOR_MTRY` are used to specify the sample size and number of predictors chosen at each node. Users can use `ODMS_RANDOM_SEED` to set the random seed value before running the algorithm.

Related Topics

- Decision Tree
  Oracle Machine Learning for SQL supports Decision Tree as one of the classification algorithms. This chapter provides an overview of the Decision Tree algorithm.
- Splitting
  The Decision Tree algorithm offers metrics for splitting the cases (records).

- Data Preparation for Decision Tree
  The Decision Tree algorithm manages its own data preparation internally. It does not require pretreatment of the data.

29.2 Building a Random Forest

The Random Forest is built upon existing infrastructure and Application Programming Interfaces (APIs) of Oracle Machine Learning for SQL.

Random forest models provide attribute importance ranking of predictors. The model is built by specifying parameters in the existing APIs. The scoring is performed using the same SQL queries and APIs as the existing classification algorithms. OML4SQL implements a variant of classical Random Forest algorithm. This implementation supports big data sets. The implementation of the algorithm differs in the following ways:

- OML4SQL does not support bagging and instead provides sampling without replacement
- Users have the ability to specify the depth of the tree. Trees are not built to maximum depth.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

Related Topics

- DBMS_DATA_MINING — Algorithm Settings: Random Forest

29.3 Scoring with Random Forest

Learn to score with the Random Forest algorithm.

Scoring with Random Forest is the same as any other classification algorithm. The following functions are supported: `PREDICTION`, `PREDICTION_PROBABILITY`, `PREDICTION_COST`, `PREDICTION_SET`, and `PREDICTION_DETAILS`.

Related Topics

- *Oracle Database SQL Language Reference*

30
Singular Value Decomposition

Learn how to use Singular Value Decomposition, an unsupervised algorithm for feature extraction.

Related Topics

- Feature Extraction
  Learn how to perform attribute reduction using feature extraction as an unsupervised function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Constants and Settings: Singular Value Decomposition
- Automatic Data Preparation
- Model Detail Views for Singular Value Decompisition
- OML4SQL Examples
- OML4R Singular Value Decomposition Example
- OML4R Code Examples

30.1 About Singular Value Decomposition

SVD and the closely-related PCA are well established feature extraction methods that have a wide range of applications. Oracle Machine Learning for SQL implements Singular Value Decomposition (SVD) as a feature extraction algorithm and Principal Component Analysis (PCA) as a special scoring method for SVD models.

SVD and PCA are orthogonal linear transformations that are optimal at capturing the underlying variance of the data. This property is very useful for reducing the dimensionality of high-dimensional data and for supporting meaningful data visualization.

SVD and PCA have a number of important applications in addition to dimensionality reduction. These include matrix inversion, data compression, and the imputation of unknown data values.

30.1.1 Matrix Manipulation

Singular Value Decomposition (SVD) is a factorization method that decomposes a rectangular matrix **X** into the product of three matrices: **U**, **S**, and **V**.

Figure 30-1 Matrix Manipulation

$$\mathbf{X} = \mathbf{USV'}$$

The **U** matrix consists of a set of 'left' orthonormal bases
The **S** matrix is a diagonal matrix
The **V** matrix consists of set of 'right' orthonormal bases

The values in **S** are called singular values. They are non-negative, and their magnitudes indicate the importance of the corresponding bases (components). The singular values reflect the amount of data variance captured by the bases. The first basis (the one with largest singular value) lies in the direction of the greatest data variance. The second basis captures the orthogonal direction with the second greatest variance, and so on.

SVD essentially performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance in the data. This is a useful procedure under the assumption that the observed data has a high signal-to-noise ratio and that a large variance corresponds to interesting data content while a lower variance corresponds to noise.

SVD makes the assumption that the underlying data is Gaussian distributed and can be well described in terms of means and covariances.

30.1.2 Low Rank Decomposition

Singular Value Decomposition (SVD) keeps lower-order bases (the ones with the largest singular values) and ignores higher-order bases (the ones with the smallest singular values) to capture the most important aspects of the data.

To reduce dimensionality, SVD keeps lower-order bases and ignores higher-order bases. The rationale behind this strategy is that the low-order bases retain the characteristics of the data that contribute most to its variance and are likely to capture the most important aspects of the data.

Given a data set **X** (*n*x*m*), where *n* is the number of rows and *m* is the number of attributes, a low-rank SVD uses only *k* components (*k* <= **min**(*m*, *n*)). In typical implementations of SVD, the value of *k* requires a visual inspection of the ranked singular values associated with the individual components. In OML4SQL, SVD automatically estimates the cutoff point, which corresponds to a significant drop in the explained variance.

SVD produces two sets of orthonormal bases (**U** and **V**). Either of these bases can be used as a new coordinate system. In OML4SQL, SVD, **V** is the new coordinate system, and **U** represents the projection of **X** in this coordinate system. The algorithm computes the projection of new data as follows:

Figure 30-2 Computing Projection of New Data

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{V}_k\mathbf{S}_k^{-1}$$

where **X** (*nxk*) is the projected data in the reduced data space, defined by the first *k* components, and $\mathbf{V}_k$ and $\mathbf{S}_k$ define the reduced component set.

30.1.3 Scalability

In Oracle Machine Learning for SQL, Singular Value Decomposition (SVD) can process data sets with millions of rows and thousands of attributes. Oracle Machine Learning for SQL automatically recommends an appropriate number of features, based on the data, for dimensionality reduction.

SVD has linear scalability with the number of rows and cubic scalability with the number of attributes when a full decomposition is computed. A low-rank decomposition is typically linear with the number of rows and linear with the number of columns. The scalability with the reduced rank depends on how the rank compares to the number of rows and columns. It can be linear when the rank is significantly smaller or cubic when it is on the same scale.

30.2 Configuring the Algorithm

Several options are available for configuring the Singular Value Decomposition (SVD) algorithm.

Among several options are: settings to control model size and performance, and whether to score with SVD projections or Principal Component Analysis (PCA) projections.

> **See Also:**
>
> DBMS_DATA_MINING — Algorithm Constants and Settings: Singular Value Decomposition for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

30.2.1 Model Size

Learn how a model size is decided based on the rows in the build data and algorithm-specific setting.

The **U** matrix in Singular Value Decomposition has as many rows as the number of rows in the build data. To avoid creating a large model, the **U** matrix persists only when an algorithm-specific setting is enabled. By default the **U** matrix does not persist.

30.2.2 Performance

Singular Value Decomposition can use approximate computations to improve performance.

Approximation may be appropriate for data sets with many columns. An approximate low-rank decomposition provides good solutions at a reasonable computational cost. The quality of the approximation is dependent on the characteristics of the data.

30.2.3 PCA scoring

Learn about configuring Singular Value Decomposition (SVD) to perform Principal Component Analysis (PCA) projections.

SVD models can be configured to perform PCA projections. PCA is closely related to SVD. PCA computes a set of orthonormal bases (principal components) that are ranked by their corresponding explained variance. The main difference between SVD and PCA is that the PCA projection is not scaled by the singular values. The PCA projection to the new coordinate system is given by:

Figure 30-3 PCA Projection Calculation

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{V}_k$$

where

$$\overline{X}$$

(*nxk*) is the projected data in the reduced data space, defined by the first *k* components, and $\mathbf{V}_k$ defines the reduced component set.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

30.3 Data Preparation for SVD

Oracle Machine Learning for SQL implements Singular Value Decomposition (SVD) for numerical data and categorical data.

When the build data is scored with SVD, Automatic Data Preparation does nothing. When the build data is scored with Principal Component Analysis (PCA), Automatic Data Preparation shifts the numerical data by mean.

Missing value treatment is not needed, because OML4SQL algorithms handle missing values automatically. SVD replaces numerical missing values with the mean and categorical missing values with the mode. For sparse data (missing values in nested columns), SVD replaces missing values with zeros.

Related Topics

- Prepare the Data

31
Support Vector Machine

Learn how to use Support Vector Machine (SVM), a powerful algorithm based on statistical learning theory.

Oracle Machine Learning for SQL implements SVM for classification, regression, and anomaly detection.

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- Regression
  Learn how to predict a continuous numerical target through regression - the supervised machine learning function.
- Anomaly Detection
  Learn how to detect rare cases in the data through anomaly detection - an unsupervised function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: Support Vector Machine
- Automatic Data Preparation
- Model Detail View for Support Vector Machine
- OML4SQL Examples
- OML4R Support Vector Machine Example
- OML4R Code Examples
- `Oracle Machine Learning for SQL`

> **See Also:**
>
> Milenova, B.L., Yarmus, J.S., Campos, M.M., "Support Vector Machines in Oracle Database 10*g*: Removing the Barriers to Widespread Adoption of Support Vector Machines", Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

31.1 About Support Vector Machine

Support Vector Machine (SVM) is a powerful, state-of-the-art algorithm with strong theoretical foundations based on the Vapnik-Chervonenkis theory.

SVM has strong **regularization** properties. Regularization refers to the generalization of the model to new data.

31.1.1 Advantages of SVM

Support Vector Machine (SVM) implements solvers for scalability and handling large volumes of data.

Oracle Machine Learning for SQL SVM implementation includes two types of solvers, an Interior Point Method (IPM) solver and a Sub-Gradient Descent (SGD) solver. The IPM solver provides stable and accurate solutions, however, it may not be able to handle data of high dimensionality. For high-dimensional and/or large data, for example, text, ratings, and so on, the SGD solver is a better choice. Both solvers have highly scalable parallel implementations and can handle large volumes of data.

31.1.2 Advantages of SVM in Oracle Machine Learning for SQL

Describes advantages of using the Support Vector Machine (SVM) algorithm.

Oracle Machine Learning for SQL has its own proprietary implementation of SVM, which exploits the many benefits of the algorithm while compensating for some of the limitations inherent in the SVM framework. OML4SQL SVM provides the scalability and usability that are needed in a production quality OML4SQL system.

31.1.2.1 Usability

Explains usability for Support Vector Machine (SVM) in Oracle Machine Learning for SQL.

Usability is a major enhancement, because SVM has often been viewed as a tool for experts. The algorithm typically requires data preparation, tuning, and optimization. Oracle Machine Learning minimizes these requirements. You do not need to be an expert to build a quality SVM model in OML4SQL. For example:

- Data preparation is not required in most cases.
- Default tuning parameters are generally adequate.

Related Topics

- Data Preparation for SVM
  Support Vector Machine (SVM) uses normalization and missing value treatment for data preparation.
- Tuning an SVM Model
  The Support Vector Machine (SVM) algorithm has built-in mechanisms that automatically choose appropriate settings based on the data.

31.1.2.2 Scalability

Learn how to scale the data for Support Vector Machine (SVM).

When dealing with very large data sets, sampling is often required. However, sampling is not required with Oracle Machine Learning for SQL SVM, because the algorithm itself uses stratified sampling to reduce the size of the training data as needed.

OML4SQL SVM is highly optimized. It builds a model incrementally by optimizing small working sets toward a global solution. The model is trained until convergence on the current working set, then the model adapts to the new data. The process continues iteratively until the convergence conditions are met. The Gaussian kernel uses caching techniques to manage the working sets.

Related Topics

- Kernel-Based Learning
  Learn about kernal-based functions to transform the input data for Support Vector Machine (SVM).

31.1.3 Kernel-Based Learning

Learn about kernal-based functions to transform the input data for Support Vector Machine (SVM).

SVM is a kernel-based algorithm. A **kernel** is a function that transforms the input data to a high-dimensional space where the problem is solved. Kernel functions can be linear or nonlinear.

Oracle Machine Learning for SQL supports linear and Gaussian (nonlinear) kernels.

In OML4SQL, the **linear kernel** function reduces to a linear equation on the original attributes in the training data. A linear kernel works well when there are many attributes in the training data.

The **Gaussian kernel** transforms each case in the training data to a point in an *n*-dimensional space, where *n* is the number of cases. The algorithm attempts to separate the points into subsets with homogeneous target values. The Gaussian kernel uses nonlinear separators, but within the kernel space it constructs a linear equation.

> **Note:**
>
> Active Learning is not relevant in Oracle Database 12*c* Release 2 and later. A setting similar to Active Learning is `ODMS_SAMPLING`.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

31.2 Tuning an SVM Model

The Support Vector Machine (SVM) algorithm has built-in mechanisms that automatically choose appropriate settings based on the data.

You may need to override the system-determined settings for some domains.

Settings pertain to regression, classification, and anomaly detection unless otherwise specified.

> **See Also:**
>
> DBMS_DATA_MINING —Algorithm Settings: Support Vector Machine for a listing and explanation of the available model settings.

> **Note:**
>
> The term hyperparameter is also interchangeably used for model setting.

31.3 Data Preparation for SVM

Support Vector Machine (SVM) uses normalization and missing value treatment for data preparation.

The SVM algorithm operates natively on numeric attributes. SVM uses z-score normalization on numeric attributes. The normalization occurs only for two-dimensional numeric columns (not nested). The algorithm automatically "explodes" categorical data into a set of binary attributes, typically one per category value. For example, a character column for marital status with values `married` or `single` is transformed to two numeric attributes: `married` and `single`. The new attributes can have the value `1` (true) or `0` (false).

When there are missing values in columns with simple data types (not nested), SVM interprets them as missing at random. The algorithm automatically replaces missing categorical values with the mode and missing numerical values with the mean.

When there are missing values in the nested columns, SVM interprets them as sparse. The algorithm automatically replaces sparse numerical data with zeros and sparse categorical data with zero vectors.

31.3.1 Normalization

Transform data through normalization for Support Vector Machine (SVM).

SVM require the normalization of numeric input. Normalization places the values of numeric attributes on the same scale and prevents attributes with a large original scale from biasing the solution. Normalization also minimizes the likelihood of overflows and underflows.

31.3.2 SVM and Automatic Data Preparation

You can prepare data by treating and transforming data manually or through Automatic Data Preparation (ADP) for Support Vector Machine (SVM).

The SVM algorithm automatically handles missing value treatment and the transformation of categorical data, but normalization and outlier detection must be handled by ADP or prepared manually. ADP performs min-max normalization for SVM.

> **Note:**
>
> Oracle recommends that you use ADP with SVM. The transformations performed by ADP are appropriate for most models.

Related Topics

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

31.4 SVM Classification

Support Vector Machine (SVM) classification is based on the concept of decision planes that define decision boundaries.

A decision plane is one that separates between a set of objects having different class memberships. SVM finds the vectors ("support vectors") that define the separators giving the widest separation of classes.

SVM classification supports both binary, multiclass, and multitarget classification. Multitarget alllows multiple class labels to be associated with a single row. The target type is a collection of type `ORA_MINING_VARCHAR2_NT`.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

31.4.1 Class Weights

Learn when to implement class weights to a data in Support Vector Machine (SVM).

In SVM classification, weights are a biasing mechanism for specifying the relative importance of target values (classes).

SVM models are automatically initialized to achieve the best average prediction across all classes. However, if the training data does not represent a realistic distribution, you can bias the model to compensate for class values that are under-represented. If you increase the weight for a class, then the percent of correct predictions for that class must increase.

Related Topics

- Priors and Class Weights
  Learn about Priors and Class Weights in a classification model to produce a useful result.

31.5 One-Class SVM

Support Vector Machine (SVM) as a one-class classifier is used for detecting anomalies.

Oracle Machine Learning for SQL uses SVM as the one-class classifier for anomaly detection. When SVM is used for anomaly detection, it has the classification machine learning function but no target.

One-class SVM models, when applied, produce a prediction and a probability for each case in the scoring data. If the prediction is 1, the case is considered typical. If the prediction is 0, the case is considered anomalous. This behavior reflects the fact that the model is trained with normal data.

You can specify the percentage of the data that you expect to be anomalous with the `SVMS_OUTLIER_RATE` build setting. If you have some knowledge that the number of "suspicious" cases is a certain percentage of your population, then you can set the outlier rate to that percentage. The model approximately identifies that many "rare" cases when applied to the general population.

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING - Algorithm Settings: Support Vector Machine
- Automatic Data Preparation
- Model Detail View for Support Vector Machine
- OML4SQL Examples
- OML4R SVM Example
- OML4R Code Examples

31.6 SVM Regression

Learn how to use epsilon-insensitivity loss function to solve regression problems in Support Vector Machine (SVM).

SVM uses an epsilon-insensitive loss function to solve regression problems.

SVM regression tries to find a continuous function such that the maximum number of data points lie within the epsilon-wide insensitivity tube. Predictions falling within epsilon distance of the true target value are not interpreted as errors.

The epsilon factor is a regularization setting for SVM regression. It balances the margin of error with model robustness to achieve the best generalization to new data.

Related Topics

- SVM Model Settings

32
XGBoost

XGBoost is highly-efficient, scalable machine learning algorithm for regression and classification that makes available the XGBoost Gradient Boosting open source package.

- About XGBoost
- Scoring with XGBoost

Related Topics

- Classification
  Learn how to predict a categorical target through classification - the supervised machine learning function.
- Regression
  Learn how to predict a continuous numerical target through regression - the supervised machine learning function.
- Ranking
  Ranking is a regression machine learning technique.
- DBMS_DATA_MINING - Model Settings
- DBMS_DATA_MINING — Algorithm Settings: XGBoost
- Automatic Data Preparation
- Model Detail Views for XGBoost
- OML4SQL Examples

32.1 About XGBoost

Oracle Machine Learning for SQL XGBoost prepares training data, invokes XGBoost, builds and persists a model, and applies the model for prediction.

OML4SQL XGBoost is a scalable gradient tree boosting system that supports both classification and regression. It makes available the open source gradient boosting framework.

You can use XGBoost 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.

The OML4SQL XGBoost algorithm takes three types of parameters: general parameters, booster parameters, and task parameters. You set the parameters through the model settings table. The algorithm supports most of the settings of the open source project.

Through XGBoost, OML4SQL supports a number of different classification and regression specifications, ranking models, and survival models. Binary and multiclass models are supported under the classification machine learning function while

regression, ranking, count, and survival are supported under the regression machine learning function.

XGBoost also supports partitioned models and internalizes the data preparation.

Related Topics

- DBMS_DATA_MINING — Algorithm Settings: XGBoost
- XGBoost: A Scalable Tree Boosting System, by Tianqi Chen and Carlos Guestrin
- XGBoost on GitHub

32.2 Scoring with XGBoost

Learn how to score with XGBoost.

The SQL scoring functions supported for a classification XGBoost model are `PREDICTION`, `PREDICTION_COST`, `PREDICTION_DETAILS`, `PREDICTION_PROBABILITY`, and `PREDICTION_SET`.

The scoring functions supported for a regression XGBoost model are `PREDICTION` and `PREDICTION_DETAILS`.

The prediction functions return the following information:

- `PREDICTION` returns the predicted value.
- `PREDICTION_COST` returns a measure of cost for a given prediction as an Oracle NUMBER. (classification only)
- `PREDICTION_DETAILS` returns the SHAP (SHapley Additive exPlanation) contributions.
- `PREDICTION_PROBABILITY` returns the probability for a given prediction. (classification only)
- `PREDICTION_SET` returns the prediction and the corresponding prediction probability for each observation. (classification only)

Related Topics

- SQL Scoring Functions

Glossary

ADP

See Automatic Data Preparation.

aggregation

The process of consolidating data values into a smaller number of values. For example, sales data collected on a daily basis can be totaled to the week level.

algorithm

A sequence of steps for solving a problem. See Oracle Machine Learning for SQL algorithm. The Oracle Machine Learning for SQL API supports the following algorithms: Apriori, Decision Tree, k-Means, MDL, Naive Bayes, GLM, O-Cluster, Support Vector Machines, Expectation Maximization, and Singular Value Decomposition.

algorithm settings

The settings that specify algorithm-specific behavior for model building.

anomaly detection

The detection of outliers or atypical cases. Oracle Machine Learning for SQL implements anomaly detection as one-class SVM.

apply

The machine learning operation that scores data. Scoring is the process of applying a model to new data to predict results.

Apriori

The algorithm that uses frequent itemsets to calculate associations.

association

A machine learning technique that identifies relationships among items.

association rules

A machine learning function that captures co-occurrence of items among transactions. A typical rule is an implication of the form A -> B, which means that the presence of itemset A implies the presence of itemset B with certain support and confidence. The support of the rule is the ratio of the number of transactions where the itemsets A and B are present to the total number of transactions. The confidence of the rule is the ratio of the number of transactions where the itemsets A and B are present to the number of transactions where itemset A is present. Oracle Machine Learning for SQL uses the Apriori algorithm for association models.

attribute

An attribute is a predictor in a predictive model or an item of descriptive information in a descriptive model. **Data attributes** are the columns of data that are used to build a model. Data attributes undergo transformations so that they can be used as categoricals or numericals by the model. Categoricals and numericals are **model attributes**. See also target.

attribute importance

A machine learning function that provides a measure of the importance of an attribute and predicts a specified target. The measure of different attributes of a training data table enables users to select the attributes that are found to be most relevant to a machine learning model. A smaller set of attributes results in a faster model build; the resulting model could be more accurate. Oracle Machine Learning for SQL uses the Minimum Description Length to discover important attributes. Sometimes referred to as *feature selection* or *key fields*.

Automatic Data Preparation

machine learning models can be created with Automatic Data Preparation (ADP), which transforms the build data according to the requirements of the algorithm and embeds the transformation instructions in the model. The embedded transformations are executed whenever the model is applied to new data.

bagging

Combine independently trained models on bootstrap samples (bagging is bootstrap aggregating).

binning

See discretization.

build data

Data used to build (train) a model. Also called *training data*.

case

All the data collected about a specific transaction or related set of values. A data set is a collection of cases. Cases are also called *records* or *examples*. In the simplest situation, a case corresponds to a row in a table.

case table

A table or view in single-record case format. All the data for each case is contained in a single row. The case table may include a case ID column that holds a unique identifier for each row. Machine learning data must be presented as a case table.

categorical attribute

An attribute whose values correspond to discrete categories. For example, *state* is a categorical attribute with discrete values (CA, NY, MA). Categorical attributes are either non-ordered (nominal) like state or gender, or ordered (ordinal) such as high, medium, or low temperatures.

centroid

See cluster centroid.

classification

A machine learning function for predicting categorical target values for new records using a model built from records with known target values. Oracle Machine Learning for SQL supports the following algorithms for classification: Naive Bayes, Decision Tree, Generalized Linear Model, Explicit Semantic Analysis, Random Forest, Support Vector Machine, and XGBoost.

clipping

See trimming.

cluster centroid

The vector that encodes, for each attribute, either the mean (if the attribute is numerical) or the mode (if the attribute is categorical) of the cases in the training data assigned to a cluster. A cluster centroid is often referred to as "the centroid."

clustering

A machine learning function for finding naturally occurring groupings in data. More precisely, given a set of data points, each having a set of attributes, and a similarity measure among them, clustering is the process of grouping the data points into different clusters such that data points in the same cluster are more similar to one another and data points in different clusters are less similar to one another. Oracle Machine Learning for SQL supports three algorithms for clustering, k-Means, Orthogonal Partitioning Clustering, and Expectation Maximization.

confusion matrix

Measures the correctness of predictions made by a model from a test task. The row indexes of a confusion matrix correspond to *actual values* observed and provided in the test data. The column indexes correspond to *predicted values* produced by applying the model to the test data. For any pair of actual/predicted indexes, the value indicates the number of records classified in that pairing.

When predicted value equals actual value, the model produces correct predictions. All other entries indicate errors.

cost matrix

An *n* by *n* table that defines the cost associated with a prediction versus the actual value. A cost matrix is typically used in classification models, where *n* is the number of distinct values in the target, and the columns and rows are labeled with target values. The rows are the actual values; the columns are the predicted values.

counterexample

Negative instance of a target. Counterexamples are required for classification models, except for one-class Support Vector Machines.

machine learning

Machine learning is the practice of automatically searching large stores of data to discover patterns and trends from experience that go beyond simple analysis. Machine learning uses sophisticated mathematical algorithms to segment the data and evaluate the probability of future events. Machine learning is also known as *Knowledge Discovery in Data* (KDD).

A machine learning model implements a machine learning algorithm to solve a given type of problem for a given set of data.

Oracle Machine Learning for SQL algorithm

A specific technique or procedure for producing an Oracle Machine Learning for SQL model. An algorithm uses a specific data representation and a specific machine learning function.

The algorithms supported by Oracle Machine Learning for SQL are Naive Bayes, Support Vector Machines, Generalized Linear Model, Decision Tree, and XGBoost for classification; Support Vector Machines Generalized Linear Model, and XGBoost for regression; k-Means, O-Cluster and Expectation Maximization for clustering; Minimum Description Length for attribute importance; Non-Negative Matrix Factorization and Singular Value Decomposition for feature extraction; Apriori for associations, and one-class Support Vector Machines and Multivariate State Estimation Technique - Sequential Probability Ratio Test for anomaly detection.

machine learning server

The component of Oracle Database that implements the machine learning engine and persistent metadata repository. You must connect to a machine learning server before performing machine learning tasks.

data set

In general, a collection of data. A data set is a collection of cases.

descriptive model

A descriptive model helps in understanding underlying processes or behavior. For example, an association model may describe consumer buying patterns. See also machine learning model.

discretization

Discretization (binning) groups related values together under a single value (or bin). This reduces the number of distinct values in a column. Fewer bins result in models that build faster. Many Oracle Machine Learning for SQL algorithms (for example NB) may benefit from input data that is *discretized* prior to model building, testing, computing lift, and applying (scoring). Different algorithms may require different types of binning. Oracle Machine Learning for SQL supports supervised binning, top N frequency binning for categorical attributes and equi-width binning and quantile binning for numerical attributes.

distance-based (clustering algorithm)

Distance-based algorithms rely on a distance metric (function) to measure the similarity between data points. Data points are assigned to the nearest cluster according to the distance metric used.

Decision Tree

A decision tree is a representation of a classification system or supervised model. The tree is structured as a sequence of questions; the answers to the questions trace a path down the tree to a leaf, which yields the prediction.

Decision trees are a way of representing a series of questions that lead to a class or value. The top node of a decision tree is called the root node; terminal nodes are called leaf nodes. Decision trees are grown through an iterative splitting of data into discrete groups, where the goal is to maximize the distance between groups at each split.

An important characteristic of the Decision Tree models is that they are transparent; that is, there are rules that explain the classification.

See also rule .

equi-width binning

Equi-width binning determines bins for numerical attributes by dividing the range of values into a specified number of bins of equal size.

Expectation Maximization

Expectation Maximization is a probabilistic clustering algorithm that creates a density model of the data. The density model allows for an improved approach to combining data originating in different domains (for example, sales transactions and customer demographics, or structured data and text or other unstructured data).

exponential smoothing

Exponential Smoothing algorithms are widely used for forecasting and can be extended to damped trends and time series.

explode

For a categorical attribute, replace a multi-value categorical column with several binary categorical columns. To explode the attribute, create a new binary column for each distinct value that the attribute takes on. In the new columns, 1 indicates that the value of the attribute takes on the value of the column; 0, that it does not. For example, suppose that a categorical attribute takes on the values {1, 2, 3}. To explode this

attribute, create three new columns, `col_1`, `col_2`, and `col_3`. If the attribute takes on the value 1, the value in `col_1` is 1; the values in the other two columns is 0.

feature

A combination of attributes in the data that is of special interest and that captures important characteristics of the data. See feature extraction.

See also text feature.

feature extraction

Creates a new set of features by decomposing the original data. Feature extraction lets you describe the data with a number of features that is usually far smaller than the number of original attributes. See also Non-Negative Matrix Factorization and Singular Value Decomposition.

Generalized Linear Model

A statistical technique for linear modeling. Generalized Linear Model (GLM) models include and extend the class of simple linear models. Oracle Machine Learning for SQL supports logistic regression for GLM classification and linear regression for GLM regression.

GLM

See Generalized Linear Model.

***k*-Means**

A distance-based clustering algorithm that partitions the data into a predetermined number of clusters (provided there are enough distinct cases). Distance-based algorithms rely on a distance metric (function) to measure the similarity between data points. Data points are assigned to the nearest cluster according to the distance metric used. Oracle Machine Learning for SQL provides an enhanced version of *k*-Means.

lift

A measure of how much better prediction results are using a model than could be obtained by chance. For example, suppose that 2% of the customers mailed a catalog make a purchase; suppose also that when you use a model to select catalog recipients, 10% make a purchase. Then the lift for the model is 10/2 or 5. Lift may also be used as a measure to compare different machine learning models. Since lift is computed using a data table with actual outcomes, lift compares how well a model performs with respect to this data on predicted outcomes. Lift indicates how well

the model improved the predictions over a random selection given actual results. Lift allows a user to infer how a model performs on new data.

lineage

The sequence of transformations performed on a data set during the data preparation phase of the model build process.

linear regression

The GLM regression algorithm supported by Oracle Machine Learning for SQL.

logistic regression

The GLM classification algorithm supported by Oracle Machine Learning for SQL.

MDL

See Minimum Description Length.

min-max normalization

Normalizes numerical attributes using this transformation:

```
x_new = (x_old-min) / (max-min)
```

Minimum Description Length

Given a sample of data and an effective enumeration of the appropriate alternative theories to explain the data, the best theory is the one that minimizes the sum of

- The length, in bits, of the description of the theory
- The length, in bits, of the data when encoded with the help of the theory

The Minimum Description Length principle is used to select the attributes that most influence target value discrimination in attribute importance.

machine learning function

A major subdomain of Oracle Machine Learning for SQL that shares common high level characteristics. The Oracle Machine Learning for SQL API supports the following machine learning functions: classification , regression, attribute importance, feature extraction, clustering, and anomaly detection.

machine learning model

A first-class schema object that specifies a machine learning model in Oracle Database.

missing value

A data value that is missing at random. The value could be missing because it is unavailable, unknown, or because it was lost. Oracle Machine Learning for SQL interprets missing values in columns with simple data types (not nested) as missing at random. Oracle Machine Learning for SQL interprets missing values in nested columns as sparsity.

Machine learning algorithms vary in the way they treat missing values. There are several typical ways to treat them: ignore them, omit any records containing missing values, replace missing values with the mode or mean, or infer missing values from existing values. See also sparse data.

model

A model uses an algorithm to implement a given machine learning function. A model can be a supervised model or an unsupervised model. A model can be used for direct inspection, for example, to examine the rules produced from an association model, or to score data (predict an outcome). In Oracle Database, machine learning models are implemented as machine learning model schema objects.

multi-record case

Each case in the data table is stored in multiple rows. Also known as transactional data. See also single-record case.

Naive Bayes

An algorithm for classification that is based on Bayes's theorem. Naive Bayes makes the assumption that each attribute is conditionally independent of the others: given a particular value of the target, the distribution of each predictor is independent of the other predictors.

nested data

Oracle Machine Learning for SQL supports transactional data in nested columns of name/value pairs. Multidimensional data that expresses a one-to-many relationship can be loaded into a nested column and mined along with single-record case data in a case table.

Neural Network

Neural Network is a machine learning algorithm that mimics the biological human brain neural network to recognize relationships in a data set that depend on large number of unknown inputs.

NMF

See Non-Negative Matrix Factorization.

Non-Negative Matrix Factorization

A feature extraction algorithm that decomposes multivariate data by creating a user-defined number of features, which results in a reduced representation of the original data.

normalization

Normalization consists of transforming numerical values into a specific range, such as [–1.0,1.0] or [0.0,1.0] such that `x_new = (x_old-shift)/scale`. Normalization applies only to numerical attributes. Oracle Machine Learning for SQL provides transformations that perform min-max normalization, scale normalization, and z-score normalization.

numerical attribute

An attribute whose values are numbers. The numeric value can be either an integer or a real number. Numerical attribute values can be manipulated as continuous values. See also categorical attribute.

O-Cluster

See Orthogonal Partitioning Clustering.

one-class Support Vector Machine

The version of Support Vector Machines used to solve anomaly detection problems. The algorithm performs classification without a target.

Orthogonal Partitioning Clustering

An Oracle proprietary clustering algorithm that creates a hierarchical grid-based clustering model, that is, it creates axis-parallel (orthogonal) partitions in the input attribute space. The algorithm operates recursively. The resulting hierarchical structure represents an irregular grid that tessellates the attribute space into clusters.

outlier

A data value that does not come from the typical population of data or extreme values. In a normal distribution, outliers are typically at least three standard deviations from the mean.

positive target value

In binary classification problems, you may designate one of the two classes (target values) as positive, the other as negative. When Oracle Machine Learning for SQL computes a model's lift, it calculates the density of positive target values among a set of test instances for which the model predicts positive values with a given degree of confidence.

predictive model

A predictive model is an equation or set of rules that makes it possible to predict an unseen or unmeasured value (the dependent variable or output) from other, known values (independent variables or input). The form of the equation or rules is suggested by machine learning data collected from the process under study. Some training or estimation technique is used to estimate the parameters of the equation or rules. A predictive model is a supervised model.

predictor

An attribute used as input to a supervised algorithm to build a model.

prepared data

Data that is suitable for model building using a specified algorithm. Data preparation often accounts for much of the time spent in a machine learning project. Automatic Data Preparation greatly simplifies model development and deployment by automatically preparing the data for the algorithm.

Principal Component Analysis

Principal Component Analysis is implemented as a special scoring method for the Singular Value Decomposition algorithm.

prior probabilities

The set of prior probabilities specifies the distribution of examples of the various classes in the original source data. Also referred to as *priors*, these could be different from the distribution observed in the data set provided for model build.

priors

See prior probabilities.

quantile binning

A numerical attribute is divided into bins such that each bin contains approximately the same number of cases.

random sample

A sample in which every element of the data set has an equal chance of being selected.

recode

Literally "change or rearrange the code." Recoding can be useful in preparing data according to the requirements of a given business problem, for example:

- Missing values treatment: Missing values may be indicated by something other than `NULL`, such as "0000" or "9999" or "NA" or some other string. One way to treat the missing value is to recode, for example, "0000" to `NULL`. Then the Oracle Machine Learning for SQL algorithms and the database recognize the value as missing.
- Change data type of variable: For example, change "Y" or "Yes" to 1 and "N" or "No" to 0.
- Establish a cutoff value: For example, recode all incomes less than $20,000 to the same value.
- Group items: For example, group individual US states into regions. The "New England region" might consist of ME, VT, NH, MA, CT, and RI; to implement this, recode the five states to, say, NE (for New England).

record

See case.

regression

A machine learning function for predicting continuous target values for new records using a model built from records with known target values. Oracle Machine Learning for SQL supports linear regression (GLM) and Support Vector Machines algorithms for regression.

rule

An expression of the general form *if X, then Y*. An output of certain algorithms, such as clustering, association, and Decision Tree. The predicate *X* may be a compound predicate.

sample

See random sample.

scale normalization

Normalize numerical attributes using this transformation:

```
x_new = (x_old - 0) / (max(abs(max),abs(min)))
```

schema

A collection of objects in an Oracle database, including logical structures such as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. A schema is associated with a specific database user.

score

Scoring data means applying a machine learning model to data to generate predictions.

settings

See algorithm settings.

single-record case

Each case in the data table is stored in one row. Contrast with multi-record case.

Singular Value Decomposition

A feature extraction algorithm that uses orthogonal linear projections to capture the underlying variance of the data. Singular Value Decomposition scales well to very large data sizes (both rows and attributes), and has a powerful data compression capability.

See Singular Value Decomposition.

sparse data

Data for which only a small fraction of the attributes are non-zero or non-null in any given case. Market basket data and unstructured text data are typically sparse. Oracle Machine Learning for SQL interprets nested data as sparse. See also missing value.

split

Divide a data set into several disjoint subsets. For example, in a classification problem, a data set is often divided in to a training data set and a test data set.

stratified sample

Divide the data set into disjoint subsets (strata) and then take a random sample from each of the subsets. This technique is used when the distribution of target values is skewed greatly. For example, response to a marketing campaign may have a positive target value 1% of the time or less. A stratified sample provides the machine learning algorithms with enough positive examples to learn the factors that differentiate positive from negative target values. See also random sample.

supervised binning

A form of intelligent binning wherein bin boundaries are derived from important characteristics of the data. Supervised binning builds a single-predictor decision tree to find the interesting bin boundaries with respect to a target. Supervised binning can be used for numerical or categorical attributes.

supervised learning

See supervised model.

supervised model

A data mining model that is built using a known dependent variable, also referred to as the target. Classification and regression techniques are examples of supervised mining. See unsupervised model. Also referred to as predictive model.

Support Vector Machine

An algorithm that uses machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. Support Vector Machine can make predictions with sparse data, that is, in domains that have a large number of predictor columns and relatively few rows, as is the case with bioinformatics data. Support Vector Machine can be used for classification, regression, and anomaly detection.

SVM

See Support Vector Machines.

target

In supervised learning, the identified attribute that is to be predicted. Sometimes called *target value* or *target attribute*. See also attribute.

text feature

A combination of words that captures important attributes of a document or class of documents. Text features are usually keywords, frequencies of words, or other document-derived features. A document typically contains a large number of words and a much smaller number of features.

text analysis

Conventional machine learning done using text features. Text features are usually keywords, frequencies of words, or other document-derived features. Once you derive text features, you mine them just as you would any other data. Both Oracle Machine Learning for SQL and Oracle Text support text analysis.

time series

Time Series is a machine learning function that forecasts target value based solely on a known history of target values. It is a specialized form of Regression, known in the literature as auto-regressive modeling. Time Series supports exponential smoothing.

top N frequency binning

This type of binning bins categorical attributes. The bin definition for each attribute is computed based on the occurrence frequency of values that are computed from the data. The user specifies a particular number of bins, say N. Each of the bins bin_1,..., bin_N corresponds to the values with top frequencies. The bin bin_N+1 corresponds to all remaining values.

training data

See build data.

transactional data

The data for one case is contained in several rows. An example is market basket data, in which a case represents one basket that contains multiple items. Oracle Machine Learning for SQL supports transactional data in nested columns of attribute name/value pairs. See also nested data, multi-record case, and single-record case.

transformation

A function applied to data resulting in a new representation of the data. For example, discretization and normalization are transformations on data.

trimming

A technique for minimizing the impact of outliers. Trimming removes values in the tails of a distribution in the sense that trimmed values are ignored in further computations. Trimming is achieved by setting the tails to `NULL`.

unstructured data

Images, audio, video, geospatial mapping data, and documents or text data are collectively known as unstructured data. Oracle Machine Learning for SQL supports the analysis of unstructured text data.

unsupervised learning

See unsupervised model.

unsupervised model

A machine learning model built without the guidance (supervision) of a known, correct result. In supervised learning, this correct result is provided in the target attribute. Unsupervised learning has no such target attribute. Clustering and association are examples of unsupervised machine learning functions. See supervised model.

winsorizing

A technique for minimizing the impact of outliers. Winsorizing involves setting the tail values of an particular attribute to some specified value. For example, for a 90% Winsorization, the bottom 5% of values are set equal to the minimum value in the 6th percentile, while the upper 5% are set equal to the maximum value in the 95th percentile.

z-score normalization

Normalize numerical attributes using this transformation:

x_new = (x_old-mean) / standard_deviation

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W

X