Oracle® Data Mining

Concepts

10*g* Release 2 (10.2)

B14339-01

June 2005

Oracle Data Mining Concepts, 10*g* Release 2 (10.2)

B14339-01

Primary Authors: Margaret Taft, Ramkumar Krishnan, Mark Hornick, Denis Muhkin, George Tang, Shiby Thomas, Peter Stengard

Contributing Authors: Charlie Berger, Marcos Campos, Boriana Milenova, Sunil Venkayla

Contributor: Robert Haberstroh

Contents

Preface

This manual discusses the basic concepts underlying Oracle Data Mining (ODM). Details of programming with the Java and PL/SQL interfaces are discussed in the *Oracle Data Mining Application Developer's Guide*.

Intended Audience

This manual is intended for anyone planning to use any of the ODM interfaces to do data mining in an Oracle Database. ODM has a graphical user interface (Oracle Data Miner) and programmatic interfaces for Java and PL/SQL.

Documentation Accessibility

Related Documents

The documentation set for Oracle Data Mining is part of the *Oracle Database Documentation Library* 10*g* Release 2 (10.2). The ODM documentation set consists of the following documents, available online:

- *Oracle Data Mining Administrator's Guide*
- *Oracle Data Mining Concepts* (this document)
- *Oracle Data Mining Application Developer's Guide*
- For the Javadoc documentation for the JDM-compliant Java interface to Oracle Data Mining, see *Oracle Data Mining Java API Reference*.
- For information about the PL/SQL interface for Oracle Data Mining, see the descriptions of the packages DBMS_DATA_MINING, DBMS_DATA_MINING_TRANSFORM, and DBMS_PREDICTIVE_ANALYTICS in the *Oracle Database PL/SQL Packages and Types Reference*
- For information about the Data Mining built-in functions, see the description of the Data Mining Functions in the *Oracle Database SQL Reference*.

Last-minute information about ODM is provided in the platform-specific release notes or README files.

For information about Oracle Data Miner, the graphical user interface for ODM, see the online help included with Oracle Data Miner. Oracle Data Miner is distributed on Oracle Technology Network at

`http://www.oracle.com/technology/index.html`

For information about the data mining process in general, independent of both industry and tool, a good source is the CRISP-DM project (Cross-Industry Standard Process for Data Mining) at

`http://www.crisp-dm.org`

For more information about the database underlying Oracle Data Mining, see:

- *Oracle Database 2 Day DBA*
- *Oracle Administrator's Guide,*
- *Oracle Database 10g Installation Guide* for your platform.

For information about developing applications to interact with the Oracle Database, see

- *Oracle Application Developer's Guide — Fundamentals*

For information about upgrading from Oracle Data Mining release 10.1, see

- *Oracle Database Upgrade Guide*
- *Oracle Data Mining Administrator's 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.

Convention	Meaning
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.

1

Introduction to Oracle Data Mining

This chapter includes the following topics:

What is Data Mining?

Too much data and not enough information — this is a problem facing many businesses and industries. Most businesses have an enormous amount of data, with a great deal of information hiding within it, but "hiding" is usually exactly what it is doing: So much data exists that it overwhelms traditional methods of data analysis.

Data mining provides a way to get at the information buried in the data. Data mining creates models to find hidden patterns in large, complex collections of data, patterns that sometimes elude traditional statistical approaches to analysis because of the large number of attributes, the complexity of patterns, or the difficulty in performing the analysis.

What Is Data Mining in the Database?

Data mining projects usually require a significant amount of data collection and data processing before and after model building. Data tables are created by combining many different types and sources of information. Real-world data is often dirty, that is, includes wrong or missing values; data must often be cleaned before it can be used. Data is filtered, normalized, sampled, transformed in various ways, and eventually used as input to data mining algorithms. Up to 80% of the effort in a data mining project is often devoted to data preparation. When the data is stored as a table in a database, data preparation can be performed using database facilities.

Data mining models have to be built, tested, validated, managed, and deployed in their appropriate application domain environments. The data mining results may need to be post-processed as part of domain specific computations (for example, calculating estimated risks, expected utilities, and response probabilities) and then stored into permanent databases or data warehouses.

Making the entire data mining process work in a reproducible and reliable way is challenging; it may involve automation and transfers across servers, data repositories, applications, and tools. For example, some data mining tools require that data be exported from the corporate database and converted to the data mining tool's format;

data mining results must be imported into the database. Removing or reducing these obstacles can enable data mining to be utilized more frequently to extract more valuable information and, in many cases, to make a significant impact on the bottom-line of an enterprise. Data mining in the database makes the data movement required by tools that do not operate in the database unnecessary and make it much easier to mine up-to-date data. Also, the less data movement, the less time the entire data mining process takes.

Data movement can make data insecure. If data never leaves the database, database security protects the data.

In summary, data mining in the database provides the following benefits:

- Less data movement
- More data security
- Up-to-date data

What Is Oracle Data Mining?

Oracle Data Mining (ODM) embeds data mining within the Oracle database. ODM algorithms operate natively on relational tables or views, thus eliminating the need to extract and transfer data into standalone tools or specialized analytic servers. ODM's integrated architecture results in a simpler, more reliable, and more efficient data management and analysis environment. Data mining tasks can run asynchronously and independently of any specific user interface as part of standard database processing pipelines and applications. Data analysts can mine the data in the database, build models and methodologies, and then turn those results and methodologies into full-fledged application components ready to be deployed in production environments. The benefits of the integration with the database cannot be emphasized enough when it comes to deploying models and scoring data in a production environment. ODM allows a user to take advantage of all aspects of Oracle's technology stack as part of an application. Also, fewer "moving parts" results in a simpler, more reliable, more powerful advanced business intelligence application.

ODM provides single-user multi-session access to models. ODM programs can run either asynchronously or synchronously in the Java interface. ODM programs using the PL/SQL interface run synchronously; to run PL/SQL asynchronously requires using the Oracle Scheduler. For a brief description of the ODM interfaces, see "Java and PL/SQL Interfaces" on page 5-2.

Data Mining Functions

Data mining functions can be divided into two categories: *supervised* (directed) and *unsupervised* (undirected).

Supervised functions are used to predict a value; they require the specification of a target (known outcome). Targets are either binary attributes indicating yes/no decisions (buy/don't buy, churn or don't churn, etc.) or multi-class targets indicating a preferred alternative (color of sweater, likely salary range, etc.). Naive Bayes for classification is a supervised mining algorithm.

Unsupervised functions are used to find the intrinsic structure, relations, or affinities in data. Unsupervised mining does not use a target. Clustering algorithms can be used to find naturally occurring groups in data.

Data mining can also be classified as *predictive* or *descriptive*. Predictive data mining constructs one or more models; these models are used to predict outcomes for new data sets. Predictive data mining functions are classification and regression. Naive Bayes is one algorithm used for predictive data mining. Descriptive data mining describes a data set in a concise way and presents interesting characteristics of the data. Descriptive data mining functions are clustering, association models, and feature extraction. *k*-Means clustering is an algorithm used for descriptive data mining.

Different algorithms serve different purposes; each algorithm has advantages and disadvantages. A given algorithm can be used to solve different kinds of problems. For example, *k*-Means clustering is unsupervised data mining; however, if you use *k*-Means clustering to assign new records to a cluster, it performs predictive data mining. Similarly, decision tree classification is supervised data mining; however, the decision tree rules can be used for descriptive purposes.

Oracle Data Mining supports the following data mining functions:

- Supervised data mining:
  - Classification: Grouping items into discrete classes and predicting which class an item belongs to
  - Regression: Approximating and forecasting continuous values
  - Attribute Importance: Identifying the attributes that are most important in predicting results
  - Anomaly Detection: Identifying items that do not satisfy the characteristics of "normal" data (outliers)
- Unsupervised data mining:
  - Clustering: Finding natural groupings in the data
  - Association models: Analyzing "market baskets"
  - Feature extraction: Creating new attributes (features) as a combination of the original attributes

Oracle Data Mining permits mining of one or more columns of text data.

Oracle Data Mining also supports specialized sequence search and alignment algorithms (BLAST) used to detect similarities between nucleotide and amino acid sequences.

New Features

The following features are new in Oracle Data Mining 10*g*, Release 2 (compared with ODM 10*g*, Release 1):

- New algorithms and improvements to existing algorithms:
  - Decision Tree algorithm, a fast, scalable means of extracting predictive and descriptive information from a database table with respect to a user-supplied target; Decision Tree provides human-understandable rules
  - One-Class Support Vector Machine algorithm for anomaly detection
  - Active learning, an enhancement to the Support Vector Machine algorithm that provides a way to deal with large build data sets
- Completely revised Java interface consistent with the Java Data Mining (JDM) standard for data mining (JSR-73)

  For information about JDM, see the Java Help for the JSR-73 Specification, available on the Oracle Technology Network at http://www.oracle.com/technology/products/bi/odm/JSR-73/index.html

- SQL Data Mining Functions for applying a model to new data: `CLUSTER_ID`, `CLUSTER_PROBABILITY`, `CLUSTER_SET`, `FEATURE_ID`,`FEATURE_SET`, `FEATURE_VALUE`,`PREDICTION`, `PREDICTION_COST`, `PREDICTION_DETAILS`, `PREDICTION_PROBABILITY`, and `PREDICTION_SET`
- O-cluster algorithm added to PL/SQL interface
- Oracle Data Miner, a graphical user interface for ODM; Oracle Data Miner is distributed on Oracle Technology Network
- The PL/SQL package `DBMS_PREDICTIVE_ANALYTICS` that automates the later stages of data mining; includes the following procedures:
  - `EXPLAIN` to rank attributes in order of influence in explaining a target column
  - `PREDICT` to predict the value of a target attribute (categorical or numerical)
- Oracle Spreadsheet Add-In for Predictive Analytics enables Microsoft Excel users to mine their Oracle Database or Excel data using the automated methodologies provided by DBMS_PREDICTIVE_ANALYTICS; the Add-In is distributed on Oracle Technology Network

2 Data for Oracle Data Mining

This chapter describes data requirements and how the data should be prepared before it is mined using either of the Oracle Data Mining (ODM) interfaces. The data preparation required depends on the type of model that you plan to build and the characteristics of the data. For example, data with attributes that take on a small number of values, that is, that have low cardinality, may not require binning.

In general, users must prepare data before invoking ODM algorithms.

The following topics are addressed:

- Data, Cases, and Attributes
- Data Requirements
- Data Preparation

Data, Cases, and Attributes

Data used by ODM consists of tables or views stored in an Oracle database. Both ordinary tables and nested tables can be used as input data. The data used in a data mining operation is often called a *data set*.

Data has a physical organization and a logical interpretation. Column names refer to physical organization; attribute names, described in the next paragraph, refer to the logical interpretation of the data.

The rows of a data table are often called *cases*, *records*, or *examples*. The columns of the data tables are called *attributes* or *fields*; each attribute in a record holds a cell of information. Attribute names are constant from record to record for unnested tables; the values in the attributes can vary from record to record. For example, each record may have an attribute labeled "annual income." The value in the annual income attribute can vary from one record to another.

ODM distinguishes two types of attributes: *categorical* and *numerical*. Categorical attributes are those that define their values as belonging to a small number of discrete categories or classes; there is no implicit order associated with the values. If there are only two possible values, for example, yes and no, or male and female, the attribute is said to be *binary.* If there are more than two possible values, for example, small, medium, large, extra large, the attribute is said to be *multiclass*.

Numerical attributes are numbers that take on a large number of values that have an order, for example, annual income. For numerical attributes, the differences between values are also ordered. Annual income could theoretically be any value from zero to infinity, though in practice annual income occupies a bounded range and takes on a finite number of values.

You can often transform numerical attributes to categorical attributes. For example, annual income could be divided into three categories: low, medium, high. Conversely, you can explode categorical values to transform them into numerical values.

Classification and Regression algorithms require a target attribute. A supervised model can predict a single target attribute. The target attribute for all classification algorithms can be numerical or categorical. The ODM regression algorithm supports only numerical target attributes.

Certain ODM algorithms support unstructured text attributes. Although unstructured data includes images, audio, video, geospatial mapping data, and documents or text, ODM supports mining text data only. An input table can contain one or more text columns.

Data Requirements

ODM supports several types of input data, depending on data table format, column data type, and attribute type.

ODM Data Table Format

ODM data must reside in a single table or view in an Oracle database. The table or view must be a standard relational table, where each case is represented by one row in the table, with each attribute represented by a column in the table. The columns must be of one of the types supported by ODM.

Column Data Types Supported by ODM

ODM does not support all the data types that Oracle supports. Each attribute (column) in a data set used by ODM must have one of the following data types:

- INTEGER
- NUMBER
- FLOAT
- VARCHAR2
- CHAR
- DM_NESTED_NUMERICALS (nested column)
- DM_NESTED_CATEGORICALS (nested column)

The supported attribute data types have a default attribute type (categorical or numerical). For details, see *Oracle Data Mining Application Developer's Guide*.

Nested Columns in ODM

Nested table columns can be used for capturing in a single table or view data that is distributed over many tables (for example, a star schema). Nested columns allow you to capture one-to-many relationships (for example, one customers can buy many products). Nested tables are required if the data has more than 1000 attributes; nested tables are useful if the data is sparse, or if the data is already persisted in a transactional format and must be passed to the data mining interface through an object view.

Note: The Decision Tree algorithm, described in "Decision Tree Algorithm" on page 3-2, does not support nested columns.

The fixed collection types `DM_NESTED_NUMERICALS` and `DM_NESTED_CATEGORICALS` are used to define columns that represent collections of numerical attributes and categorical attributes, respectively.

For a given case identifier, attribute names must be unique across all the collections and individual columns. The fixed collection types enforce this requirement.

Missing Values

Data tables often contain missing values.

Missing Values and NULL Values in ODM

Certain algorithms assume that a `NULL` value indicates a missing value; others assume that a `NULL` value indicates sparse data, as described in "Sparse Data" on page 2-3.

Missing Value Handling

ODM is robust in handling missing values and does not require users to treat missing values in any special way. ODM will ignore missing values but will use non-missing data in a case.

If an algorithm assumes that NULL values indicate sparse data, then you should treat any values that are true missing values.

Sparse Data

Data is said to be *sparse* if only a small fraction (no more than 20%, often 3% or less) of the attributes are non-zero or non-null for any given case. Sparse data occurs, for example, in market basket problems. In a grocery store, there might be 10,000 products in the store, and the average size of a basket (the collection of distinct items that a customer purchases in a typical transaction) is on average 50 products. In this example, a transaction (case or record) has on average 50 out of 10,000 attributes that are not null. This implies that the fraction of non-zero attributes in the table (or the density) is approximately 50/10,000, or 0.5%. This density is typical for market basket and text mining problems.

Association models are designed to process sparse data; indeed, if the data is not sparse, the algorithm may require a large amount of temporary space or may not be able to build a model.

Sparse data is represented in a table in such a way that avoids the specification of the most common value to save storage. In such a specification of sparse data, a missing value is implicitly interpreted as the most common value.

Different algorithms make different assumptions about what indicates sparse data. For Support Vector Machine, *k*-Means, association, and Non-Negative Matrix Factorization, `NULL` values indicate sparse data; for all other algorithms, `NULL` values indicate missing values. See the description of each algorithm for information about how it interprets `NULL` values.

Outliers and Oracle Data Mining

An *outlier* is a value that is far outside the normal range in a data set, typically a value that is several standard deviations from the mean. The presence of outliers can have a significant impact on certain kinds of ODM models. Naive Bayes, Adaptive Bayes Network, Support Vector Machine, Attribute Importance, either clustering algorithm, and Non-Negative Matrix Factorization are sensitive to outliers.

For example, the presence of outliers, when external equal-width binning is used, makes most of the data concentrate in a few bins (a single bin in extreme cases). As a result, the ability of the model to detect differences in numerical attributes may be significantly lessened. For example, a numerical attribute such as income may have all the data belonging to a single bin except for one entry (the outlier) that belongs to a different bin.

For outlier treatments, see "Winsorizing and Trimming" on page 2-4.

Data Preparation

Data is said to be *prepared* when certain data transformations required by a data mining algorithm are performed by the user before the algorithm is invoked. For most algorithms, data must be prepared before the algorithm is invoked.

Different algorithms have different requirements for data preparation; recommended data preparation is discussed with each algorithm in Chapter 3 and Chapter 4.

Data preparation can take many forms, such as joining two or more tables so that all required data is in a single table or view, transforming numerical attributes by applying numerical functions to them, recoding attributes, treating missing values, treating outliers, omitting selected columns for a training data set, and so forth.

ODM includes transformations that perform the following data-mining-specific transformations:

- Winsorizing and trimming to treat outliers
- Discretization to reduce the number of distinct values for an attribute
- Normalization to convert attribute values to a common range

Winsorizing and Trimming

Certain algorithms are sensitive to outliers. Winsorizing and trimming transformations are used to deal with outliers.

Winsorizing involves setting the tail values of an 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.

Trimming removes the tails in the sense that trimmed values are ignored in further computations. This is achieved by setting the tails to `NULL`. This process is sometimes called clipping.

Binning (Discretization)

Some ODM algorithms may benefit from *binning* (*discretizing*) both numeric and categorical data. Naive Bayes, Adaptive Bayes Network, Clustering, Attribute Importance, and Association Rules algorithms may benefit from binning.

Binning means grouping related values together, thus reducing the number of distinct values for an attribute. Having fewer distinct values typically leads to a more compact model and one that builds faster. Binning must be performed carefully. Proper binning can improve model accuracy; improper binning can lead to loss in accuracy.

Methods for Computing Bin Boundaries

ODM utilities provide three methods for computing bin boundaries from the data:

- **Top N most frequent items:** This technique is used to bin categorical values. 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.
- **Equi-Width Binning:** This technique is used to bin numerical values. For numerical attributes, ODM finds the minimum (*min*) and maximum (*max*) values for every attribute in the data. Then ODM divides the [*min, max*] range into *N* equal bins of size *d=(max-min)/N.* Thus bin 1 is [*min, min+d*), bin 2 is [*min+d, min+2d*), and bin *N* is [*min+(N-1)*d,max*]. The number of bins can either be specified by the user or calculated by the transformation.
- Quantile Binning: This technique is used to bin numerical values. The definition for each relevant attribute is computed based on the minimum values for each quantile, where quantiles are computed from the data using `NTILE` function. Bins bin_1,..., bin_N span the following ranges: bin_1 spans [min_1,min_2]; bin_2,..., bin_i,..., bin_N-1 span (min_i, min_(i+1)] and bin_N spans (min_N, max_N]. Bins with equal left and right boundaries are collapsed.

Normalization

Normalization converts individual numerical attributes so that each attribute's values lie in the same range. Values are converted to be in the range 0.0 to 1.0 or the range -1.0 to 1.0. Normalization ensures that attributes do not receive artificial weighting caused by differences in the ranges that they span. Some algorithms, such as *k*-Means, Support Vector Machine, and Non-Negative Matrix Factorization, benefit from normalization.

3 Supervised Data Mining

This chapter describes supervised models; supervised models are sometimes referred to as predictive models. These models predict a target value. The Java and PL/SQL Oracle Data Mining interfaces support the following supervised functions:

- Classification
- Regression
- Attribute Importance
- Anomaly Detection

This chapter also describes

- Testing Supervised Models

Classification

Classification of a collection consists of dividing the items that make up the collection into categories or classes. In the context of data mining, classification is done using a model that is built on historical data. The goal of predictive classification is to accurately predict the target class for each record in new data, that is, data that is not in the historical data.

A classification task begins with *build data* (also known as *training data*) for which the target values (or class assignments) are known. Different classification algorithms use different techniques for finding relations between the predictor attributes' values and the target attribute's values in the build data. These relations are summarized in a model; the model can then be applied to new cases with unknown target values to predict target values. A classification model can also be applied to data that was held aside from the training data to compare the predictions to the known target values; such data is also known as *test data* or *evaluation data*. The comparison technique is called *testing a model*, which measures the model's predictive accuracy. The application of a classification model to new data is called *applying the model*, and the data is called *apply data* or *scoring data*. Applying a model to data is often called *scoring the data*.

Classification is used in customer segmentation, business modeling, credit analysis, and many other applications. For example, a credit card company may wish to predict which customers are likely to default on their payments. Customers are divided into two classes: those who default and those who do not default. Each customer corresponds to a case; data for each case might consist of a number of attributes that describe the customer's spending habits, income, demographic attributes, etc. These are the predictor attributes. The target attribute indicates whether or not the customer has defaulted. The build data is used to build a model that predicts whether new customers are likely to default.

Classification problems can have either binary and multiclass targets. Binary targets are those that take on only two values, for example, *good credit risk* and *poor credit risk*. Multiclass targets have more than two values, for example, the product purchased (comb or hair brush or hair pin). Multiclass target values are not assumed to exist in an ordered relation to each other, for example, hair brush is not assumed to be greater or less than comb.

Classification problems may require the specification of Costs, described on page 3-7 and Priors, described on page 3-7.

Algorithms for Classification

ODM provides the following algorithms for classification:

- Decision Tree Algorithm
- Naive Bayes Algorithm
- Adaptive Bayes Network Algorithm
- Support Vector Machine Algorithm

Table 3–1 compares several important features of the classification algorithms.

Table 3–1 Classification Algorithm Comparison

Feature	Naive Bayes	Adaptive Bayes Network	Support Vector Machine	Decision Tree
Speed	Very fast	Fast	Fast with active learning	Fast
Accuracy	Good in many domains	Good in many domains	Significant	Good in many domains
Transparency	No rules (black box)	Rules for Single Feature Build only	No rules (black box)	Rules
Missing value interpretation	Missing value	Missing value	Sparse data	Missing value

Decision Tree Algorithm

Decision tree rules provide model transparency so that a business user, marketing analyst, or business analyst can understand the basis of the model's predictions, and therefore, be comfortable acting on them and explaining them to others.

In addition to transparency, the Decision Tree algorithm provides speed and scalability. The build algorithm scales linearly with the number of predictor attributes and on the order of `nlog(n)` with the number of rows, `n`. Scoring is very fast. Both build and apply are parallelized.

The Decision Tree algorithm builds models for binary and multi-class targets. It produces accurate and interpretable models with relatively little user intervention required. The Decision Tree algorithm is implemented in such a way as to handle data in the typical data table formats, to have reasonable defaults for splitting and termination criteria, to perform automatic pruning, and to perform automatic handling of missing values. However, it does not distinguish sparse data from missing data. (See "Sparse Data" on page 2-3 for more information.) Users can specify costs and priors.

Decision Tree does not support nested tables.

Decision Tree Models can be converted to XML.

Decision Tree Rules A Decision Tree model always produces rules. Decision tree rules are in the form "IF predictive information THEN target," as in "IF income is greater than $70K and household size is greater than 3 THEN the probability of Churn is 0.075."

XML for Decision Tree Models You can generate XML representing a decision tree model; the generated XML satisfies the definition specified in the Data Mining Group Predictive Model Markup Language (PMML) version 2.1 specification. The specification is available at `http://www.dmg.org`.

Naive Bayes Algorithm

The Naive Bayes algorithm (NB) can be used for both binary and multiclass classification problems.

NB builds and scores models extremely rapidly; it scales linearly in the number of predictors and rows.

NB makes predictions using Bayes' Theorem, which derives the probability of a prediction from the underlying evidence. Bayes' Theorem states that the probability of event A occurring given that event B has occurred (P(A|B)) is proportional to the probability of event B occurring given that event A has occurred multiplied by the probability of event A occurring ((P(B|A)P(A)).

Naive Bayes makes the assumption that each attribute is conditionally independent of the others, that is, given a particular value of the target, 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.

Adaptive Bayes Network Algorithm

Adaptive Bayes Network (ABN) is an Oracle proprietary algorithm that provides a fast, scalable, non-parametric means of extracting predictive information from data with respect to a target attribute. (Non-parametric statistical techniques avoid assuming that the population is characterized by a family of simple distributional models, such as standard linear regression, where different members of the family are differentiated by a small set of parameters.)

ABN, in Single Feature Build mode, can describe the model in the form of human-understandable rules. The rules produced by ABN are one of its main advantages over Naive Bayes. ABN rules provide model transparency so that a business user, marketer, or business analyst can understand the basis of the model's predictions and therefore, be comfortable acting on them and explaining them to others.

In addition to rules, ABN provides performance and scalability, which are derived via various user parameters controlling the trade-off of accuracy and build time.

ABN predicts binary as well as multiclass targets.

ABN can use costs and priors for both building and scoring (see "Costs" on page 3-7 and "Priors" on page 3-7).

ABN Model Types An ABN model is an (adaptive conditional independence model that uses the minimum description length principle to construct and prune an array of conditionally independent network features. Each network feature consists of one or more conditional probability expressions. The collection of network features forms a product model that provides estimates of the target class probabilities. There can be one or more network features. The number and depth of the network features in the model determine the model mode. There are three model modes for ABN:

- Pruned Naive Bayes (Naive Bayes Build)
- Simplified decision tree (Single Feature Build)
- Boosted (Multi Feature Build)

Users can select the ABN model type. Rules are available only for Single Feature Build.

Each network feature consists of one or more attributes included in a conditional probability expression. An array of single attribute network features is an MDL-pruned Naive Bayes model. A single multi-attribute network feature model is equivalent to a simplified C4.5 decision tree; such a model is simplified in the sense that numerical attributes are binned and treated as categorical. Furthermore, a single predictor is used to split all nodes at a given tree depth. The splits are *k*-way, where *k* is the number of unique (binned) values of the splitting predictor. Finally, a collection of multi-attribute network features forms a product model (*boosted mode*). All three types provide estimates of the target class probabilities.

ABN Rules Rules can be extracted from an Adaptive Bayes Network model as compound predicates. Rules form a human-interpretable depiction of the model and include statistics indicating the number of the relevant training data instances in support of the rule. A record apply instance specifies a pathway in a network feature taking the form of a compound predicate.

Note: Rules are generated for the single feature build model type only.

For example, suppose the feature consists of two training attributes: Age {20-40, 40-60, 60-80} and Income {<=50K, >50K}. A record instance consisting of a person age 25 and income $42K is expressed as

IF AGE IN (20-40) and INCOME IN (<=50K)

Suppose that the associated target (for example, response to a promotion) probabilities are {0.8 (no), 0.2 (yes)}. Then we have a detailed rule of the form

IF AGE IN (20-40) and INCOME IN (<=50K) THEN Probability = {0.8, 0.2}

In addition to the probability distribution, there are the associated training data counts, e.g. {400, 100}.

Suppose there is a cost matrix specifying that it is 6 times more costly to predict a *no* incorrectly than it is to predict a *yes* incorrectly. Then the cost of predicting *yes* for this instance is 0.8 * 1 = 0.8 (because the model is wrong in this prediction 80% of the time) and the cost of predicting *no* is 0.2 * 6 = 1.2. Thus, the minimum cost (best) prediction is *yes*. Without the cost matrix, the decision is reversed. Implicitly, all errors are equal and we have: 0.8 * 1 = 0.8 for *yes* and 0.2 * 1 = 0.2 for *no*.

The order of the predicates in the generated rules implies relative importance.

When you apply an ABN model for which rules were generated, with a single feature, you get the same result that you would get if you wrote an external program that applied the rules.

Support Vector Machine Algorithm

Support Vector Machine (SVM) is a state-of-the-art classification and regression algorithm. SVM is an algorithm with strong regularization properties, that is, the optimization procedure maximizes predictive accuracy while automatically avoiding over-fitting of the training data. Neural networks and radial basis functions, both popular data mining techniques, have the same functional form as SVM models; however, neither of these algorithms has the well-founded theoretical approach to regularization that forms the basis of SVM.

SVM projects the input data into a kernel space. Then it builds a linear model in this kernel space. A classification SVM model attempts to separate the target classes with the widest possible margin. A regression SVM model tries to find a continuous function such that maximum number of data points lie within an epsilon-wide tube around it. Different types of kernels and different kernel parameter choices can produce a variety of decision boundaries (classification) or function approximators (regression). The ODM SVM implementation supports two types of kernels: linear and Gaussian. ODM also provides automatic parameter estimation on the basis of the characteristics of the data.

SVM performs well with real-world applications such as classifying text, recognizing hand-written characters, classifying images, as well as bioinformatics and biosequence analysis. The introduction of SVM in the early 1990s led to an explosion of applications and deepening theoretical analysis that established SVM along with neural networks as one of the standard tools for machine learning and data mining.

There is no upper limit on the number of attributes and target cardinality for SVMs; the only constraints are those imposed by hardware.

SVM is the preferred algorithm for sparse data.

The following new features have been added to the SVM algorithm in ODM 10*g* Release 2:

- SVM can also be used to identify novel or anomalous patterns using one-class SVM. For more information, see "Anomaly Detection" on page 3-9.
- SVM supports active learning. For more information, see "Active Learning" on page 3-5.
- SVM automatically creates stratified samples for large training sets (see "Sampling for Classification" on page 3-6) and automatically chooses a kernel type for model build (see "Automatic Kernel Selection" on page 3-6).

Active Learning SVM models grow as the size of the training data set increases. This property limits SVM models to small and medium size training sets (less than 100,000 cases). Active learning provides a way to deal with large training sets.

The termination criteria for active learning is usually an upper bound on the number of support vectors; when the upper bound is attained, the build stops. Alternatively, stopping criteria are qualitative, such as no significant improvement in model accuracy on a held-aside sample.

Active learning forces the SVM algorithm to restrict learning to the most informative training examples and not to attempt to use the entire body of data. In most cases, the resulting models have predictive accuracy comparable to that of the standard (exact) SVM model.

Active learning can be applied to all SVM models (classification, regression, and one-class).

Active learning is on by default. It can be turned off.

Sampling for Classification For classification, SVM automatically performs stratified sampling during model build. The algorithm scans the entire build data set and selects a sample that is balanced across target values.

Automatic Kernel Selection SVM automatically determines the appropriate kernel type based on build data characteristics. This selection can be overridden by explicitly specifying a kernel type.

Data Preparation and Settings Choice for Support Vector Machines You can influence both the Support Vector Machine (SVM) model quality (accuracy) and performance (build time) through two basic mechanisms: data preparation and model settings. Significant performance degradation can be caused by a poor choice of settings or inappropriate data preparation. Poor settings choices can also lead to inaccurate models.

For detailed information about data preparation for SVM models, see the *Oracle Data Mining Application Developer's Guide*.

SVM has built-in mechanisms that attempt to choose appropriate settings automatically based on the data provided. You may need to override the system-determined settings for some domains.

Data Preparation for Classification

This section summarizes data preparation that may be required by classification algorithms.

Outliers

Outliers affect classification algorithms as follows:

- Naive Bayes and Adaptive Bayes Network: The presence of outliers, when external equal-width binning is used, makes most of the data concentrate in a few bins (a single bin in extreme cases). As a result, the discriminating power of these algorithms may be significantly reduced. In this case, quantile binning helps to overcome these problems.
- Support Vector Machine: The presence of outliers can significantly impact models. Use a clipping transformation to avoid the problems caused by outliers.
- Decision Tree: The presence of outliers does not impact decision tree models.

NULL Values

The meaning of `NULL` values and how to treat them depends on the algorithm as follows:

- Support Vector Machine: `NULL` values indicate sparse data. Missing values are not automatically handled. If the data is not sparse and the values are indeed missing at random, it is necessary to perform missing data imputation (that is, perform some kind of missing values treatment) and substitute a non-NULL value for the `NULL` value. One simple approach is to use the mean for numerical attributes and the mode for categorical attributes. If you do not treat missing values, the algorithm will not handle the data correctly.
- For all other classification algorithms, `NULL` values indicate missing values:
  - Decision Tree, Naive Bayes, and Adaptive Bayes Network: Missing values are handled automatically.

Normalization

Support Vector Machine may benefit from normalization.

Costs

In a classification problem, it may be important to specify the costs involved in making an incorrect decision. Doing so can be useful when the costs of different misclassifications vary significantly.

For example, suppose the problem is to predict whether a user will respond to a promotional mailing. The target has two categories: YES (the customer responds) and NO (the customer does not respond). Suppose a positive response to the promotion generates $500 and that it costs $5 to do the mailing. If the model predicts YES and the actual value is YES, the cost of misclassification is $0. If the model predicts YES and the actual value is NO, the cost of misclassification is $5. If the model predicts NO and the actual value is YES, the cost of misclassification is $500. If the model predicts NO and the actual value is NO, the cost is $0. In this case, you would probably want to avoid cases where the model predicts NO and the actual value is YES

Exactly how costs are specified depends on the classification algorithm used:

- NB and ABN use a cost matrix
- SVM uses weights

The cost of misclassification is summarized in a cost matrix. The rows of the matrix represent actual values and the columns, predicted values. A cell in the matrix represents the misclassification cost that occurs when the model predicts the class indicated by the column when the class is really the one specified by the row.

Weights for an SVM model are automatically initialized to achieve the best average prediction across all target values. If you change a weight value, the percentage of correct predictions changes in the same way; for example, if you increase a weight value, the percent of correct predictions increases for the associated class.

Classification algorithms apply the cost information to the predicted probabilities during scoring to estimate the least expensive prediction. If a cost matrix is specified for scoring, the output of the scoring is the minimum cost for the prediction. If no cost matrix is supplied, the output is the most likely prediction.

You must be careful how you assign costs. You are making a trade-off between false-positives (falsely accusing someone of fraud) and false negatives (letting a crime go unpunished). Your costs should reflect this trade-off. Perhaps you are willing to let some crimes go unpunished so that you don't falsely accuse millions of committing fraud; for example, you must be sure that you are right before you accuse someone (say 99%, rather than just 50% sure). Predicting on probability means you are indifferent to the type of error you make. If you are concerned about the type of error, a cost matrix or carefully adjusted weights are warranted.

Priors

In building a classification model, describing the distribution in the real population can be useful when the training data does not accurately reflect the real population. The real population is described by providing the prior distribution, often referred to as the priors, to the build operation.

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

One solution to this problem involves creating a source table for the build operation that contains approximately equal numbers of each target value. However, the algorithm will take the observed distribution as realistic, and will build a model that will predict each of the target values in equal numbers unless it is instructed otherwise. Supplying the actual distribution of target values, the priors, to the Build process can result in a more effective model.

Note that the model should be tested against data that has the actual distribution of target values. For example, 98% negative and 2% positive for the marketing campaign.

Regression

Regression models are similar to classification models. The difference between regression and classification is that regression deals with numerical or continuous target attributes, whereas classification deals with discrete or categorical target attributes. In other words, if the target attribute contains continuous (floating-point) values or integer values that have inherent order, a regression technique can be used. If the target attribute contains categorical values, that is, string or integer values where order has no significance, a classification technique is called for. Note that a continuous target can be turned into a discrete target by binning; this turns a regression problem into a problem that can be solved using classification algorithms.

Algorithm for Regression

Support Vector Machine (SVM) builds both classification and regression models. For more information about SVM, see "Support Vector Machine Algorithm" on page 3-5.

ODM SVM provides improved data-driven estimation of epsilon and the complexity factor for SVM regression models.

Active learning (see "Active Learning" on page 3-5) can be used for regression models.

One-class SVM (see "Anomaly Detection" on page 3-9) cannot be used for regression problems.

Attribute Importance

Attribute Importance (AI) provides an automated solution for improving the speed and possibly the accuracy of classification models built on data tables with a large number of attributes.

The time required to build ODM classification models increases with the number of attributes. Attribute Importance identifies a proper subset of the attributes that are most relevant to predicting the target. Model building can proceed using the selected attributes only.

Using fewer attributes does not necessarily result in lost predictive accuracy. Using too many attributes (especially those that are "noise") can affect the model and degrade its performance and accuracy. Mining using the smallest number of attributes can save significant computing time and may build better models.

The programming interfaces for Attribute Importance permit the user to specify a number or percentage of attributes to use; alternatively the user can specify a cutoff point.

Data Preparation for Attribute Importance

The presence of outliers, when external equal-width binning is used, makes most of the data concentrate in a few bins (a single bin in extreme cases). As a result, the discriminating power of an attribute importance model may be significantly reduced. In this case, quantile binning helps to overcome these problems.

`NULL` values are treated as missing values and not as indicators of sparse data.

Algorithm for Attribute Importance

ODM uses the Minimum Descriptor Length algorithm for attribute importance.

Minimum Description Length Algorithm

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. The MDL principle is used to build ODM Attribute Importance models.

MDL considers each attribute as a simple predictive model of the target class. These single predictor models are compared and ranked with respect to the MDL metric (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.

With MDL, the model selection problem is treated as a communication problem. There is a sender, a receiver, and data to be transmitted. For classification models, the data to be transmitted is a model and the sequence of target class values in the training data.

Attribute importance uses a two-part code to transmit the 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 time possible conditional probabilities. The size of the preamble in bits can be shown to be 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 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.

Anomaly Detection

Anomaly detection consist of identifying novel or anomalous patterns. Identifying such patterns can be useful in problems of fraud detection (insurance, tax, credit card, etc.) and computer network intrusion detection. 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.

An anomaly detection model discriminates between the known examples of the positive class and the unknown negative set of counterexamples. An anomaly detection model identifies items that do not fit in the distribution.

Anomaly detection is a mining function in the Oracle Data Miner interface. In the ODM Java and PL/SQL interfaces, an anomaly detection model is a classification model. See "Specify the One-Class SVM Algorithm" on page 3-10 for more information.

Algorithm for Anomaly Detection

Standard binary supervised classification algorithms, such as Naive Bayes, require the presence of both positive examples and negative examples (counterexamples) of a target class. One-class SVM classification requires only the presence of examples of a single target class. The model learns to discriminate between the known examples of the positive class and the unknown negative set of counterexamples. In other words, one-class SVM detects anomalies. One-class SVM was initially used to estimate the support of a distribution. The goal is to estimate a function that will be positive if an example belongs to a set and negative if the example belongs to the complement of the set. The model computes a binary function that identifies regions in the input space where the majority of the positive data lives.

One-class SVM models are useful in situations such as:

- Outlier detection
- Cases where it is difficult to provide counterexamples

In outlier detection, you separate the typical examples in a distribution from the atypical (outlier) examples. The distance from the separating plane indicates how typical a given point is with respect to the distribution of the training data. Outliers can be ranked on the basis of the probability of them being typical or atypical cases. In a similar way, you can use one-class SVM to detect other kinds of anomalies.

In some classes of problems, it is difficult or impossible to provide a useful and representative set of counterexamples. Examples may be easy to identify, but counterexamples are either hard to specify or expensive to collect. For example, in text document classification, it is easy to classify a document under a given topic. However, the universe of documents not belonging to this topic can be very large and it may not be feasible to provide counterexamples.

The accuracy of one-class SVM models cannot usually match the accuracy of standard SVM classifiers built with meaningful counterexamples.

SVM is the only supervised classification algorithm in ODM that can operate in one-class mode.

One-class SVM cannot be used for regression problems.

Specify the One-Class SVM Algorithm

To build a one-class SVM model in either of the ODM programmatic interfaces, select classification as the mining function, SVM as the algorithm, and pass a `NULL` or empty string as the target column name.

To build a one-class SVM model in Oracle Data Miner, select anomaly detection.

Testing Supervised Models

Supervised models are tested to evaluate the accuracy of their predictions. A model is tested by applying it to data with known values of the target and comparing the model's 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.

Model testing results in the calculation of test metrics. The exact test metrics calculated depend on the type of model. Testing a classification model results in a confusion matrix; testing a regression model results in error estimates. Optionally, lift and receiver operating characteristics (ROC) can be calculated for classification models.

The rest of this section describes these test metrics, as follows:

- Confusion Matrix
- Lift
- Receiver Operating Characteristics
- Test Metrics for Regression Models

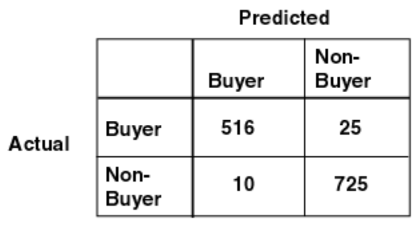
Confusion Matrix

ODM supports the calculation of a confusion matrix to asses the accuracy of a classification model. The simplest example of a confusion matrix is one for a binary classification problem. For a binary problem, the confusion matrix is a two-dimensional square matrix. (In general, the confusion matrix is an *n*-dimensional square matrix, where *n* is the number of distinct target values.) The row indexes of a confusion matrix correspond to *actual values* observed and used for model testing; 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. Figure 3–1 contains an example of a confusion matrix. For example, a value of 25 for an actual value index of "buyer" and a predicted value index of "nonbuyer" indicates that the model incorrectly classified a "buyer" as a "nonbuyer" 25 times. A value of 516 for an actual/predicted value index of "buyer" indicates that the model correctly classified a "buyer" 516 times.

The predictions were correct 516 + 725 = 1241 times, and incorrect 25 + 10 = 35 times. The sum of the values in the matrix is equal to the number of scored records in the input data table. The number of scored records is the sum of correct and incorrect predictions, which is 1241 + 35 = 1276. The error rate is 35/1276 = 0.0274; the accuracy rate is 1241/1276 = 0.9725.

A confusion matrix provides a quick understanding of model accuracy and the types of errors the model makes when scoring records. It is the result of a test task for classification models.

Figure 3–1 Confusion Matrix



		Predicted	
		Buyer	Non-Buyer
Actual	Buyer	516	25
	Non-Buyer	10	725

Lift

ODM supports computing lift for a classification model. Lift can be computed for binary (two values) target fields. Lift can also be computed for multiclass targets by designating a preferred positive class and combining all other target class values, effectively turning a multiclass target into a binary target. Given a designated positive target value (that is, the value of most interest for prediction, such as "buyer," or "has disease"), test cases are sorted according to how confidently they are predicted to be positive cases. Positive cases with highest confidence come first, followed by positive cases with lower confidence. Negative cases with lowest confidence come next, followed by negative cases with highest confidence. Based on that ordering, they are partitioned into quantiles, and the following statistics are calculated by the programming interfaces:

- 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 for a given quantile 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 target density for the quantile to the target density over all the test data.
- Cumulative percentage of records for a given quantile is the percentage of all test 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 (defined as above).
- Cumulative number of nontargets is the number of actually negative instances in the first *n* quantiles (defined as above).
- Cumulative lift for a given quantile is the ratio of the cumulative target density to the target density over all the test data.

Cumulative targets can be computed from the quantities that are available in the `LiftResultElement` using the following formula:

*targets_cumulative = lift_cumulative * percentage_records_cumulative*

Oracle Data Miner calculates different statistics for lift. See the online help for Oracle Data miner for more information.

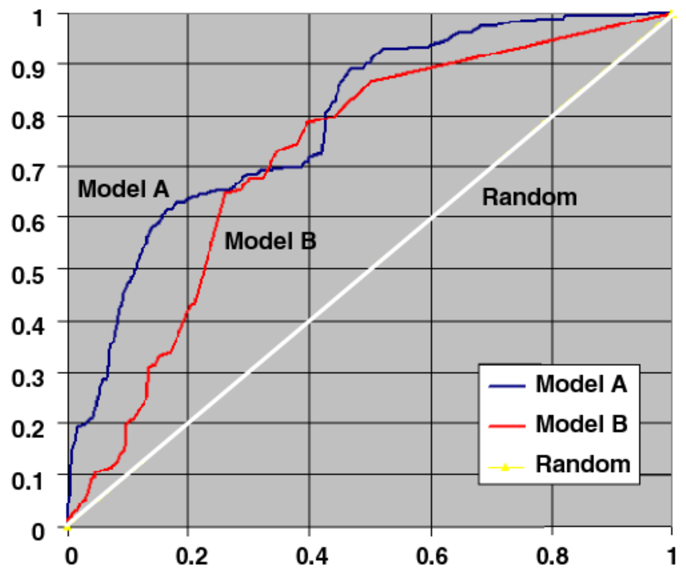
Receiver Operating Characteristics

Another useful method for evaluating classification models is Receiver Operating Characteristics (ROC) analysis. ROC curves are similar to lift charts in that they provide a means of comparison between individual models and determine thresholds which yield a high proportion of positive hits. ROC was originally used in signal detection theory to gauge the true hit versus false alarm ratio when sending signals over a noisy channel.

The horizontal axis of an ROC graph measures the false positive rate as a percentage. The vertical axis shows the true positive rate. The top left hand corner is the optimal location in an ROC curve, indicating high TP (true-positive) rate versus low FP (false-positive) rate. 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 will be assigned 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).

In the example graph in Figure 3–2, Model A clearly has a higher AUC for the entire data set. However, if the user decides that a false positive rate of 40% is acceptable, Model B is better suited, since it achieves a better error true positive rate at that false positive rate.

Figure 3–2 Receiver Operating Characteristics Curves



Besides model selection the ROC also helps to determine a threshold value to achieve an acceptable trade-off between hit (true positives) rate and false alarm (false positives) rate. By selecting a point on the curve for a given model a given trade-off is achieved. This threshold can then be used as a post-processing parameter for achieving the desired performance with respect to the error rates. ODM models by default use a threshold of 0.5.

The Oracle Data Mining ROC computation calculates the following statistics:

- Probability threshold: The minimum predicted positive class probability resulting in a positive class prediction. Different threshold values result in different hit rates and 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).

- Hit rate ("True Positives" in Oracle Data Miner): (true positives/(true positives + false negatives))
- False alarm rate ("False Positives" in Oracle Data Miner): (false positives/(false positives + true negatives))

Test Metrics for Regression Models

Regression test results provide the following measures of model accuracy:

- Root mean square
- Mean absolute error

These two statistics are the metrics most commonly used to test regression models. For more information about these metrics, see the *Oracle Data Mining Application Developer's Guide*.

4 Unsupervised Data Mining

This chapter describes unsupervised models. These models do not predict a target value, but focus on the intrinsic structure, relations, and interconnectedness of the data. Unsupervised models are sometimes called descriptive models. Oracle Data Mining supports the following unsupervised functions:

- Clustering
- Association
- Feature Extraction

Clustering

Clustering is useful for exploring data. If there are many cases and no obvious natural groupings, clustering data mining algorithms can be used to find natural groupings.

Clustering analysis identifies clusters embedded in the data. A cluster is a collection of data objects that are similar in some sense to one another. A good clustering method produces high-quality clusters to ensure that the inter-cluster similarity is low and the intra-cluster similarity is high; in other words, members of a cluster are more like each other than they are like members of a different cluster.

Clustering can also serve as a useful data-preprocessing step to identify homogeneous groups on which to build supervised models. Clustering models are different from supervised models in that the outcome of the process is not guided by a known result, that is, there is no target attribute. Supervised models predict values for a target attribute, and an error rate between the target and predicted values can be calculated to guide model building. Clustering models, on the other hand, are built using optimization criteria that favor high intra-cluster and low inter-cluster similarity. The model can then be used to assign cluster identifiers to data points.

In ODM a cluster is characterized by its centroid, attribute histograms, and the cluster's place in the model's hierarchical tree. ODM performs hierarchical clustering using an enhanced version of the *k*-means algorithm and the orthogonal partitioning clustering algorithm, an Oracle proprietary algorithm. The clusters discovered by these algorithms are then used to create rules that capture the main characteristics of the data assigned to each cluster. The rules represent the bounding boxes that envelop the data in the clusters discovered by the clustering algorithm. The antecedent of each rule describes the clustering bounding box. The consequent encodes the cluster ID for the cluster described by the rule. For example, for a data set with two attributes: AGE and HEIGHT, the following rule represents most of the data assigned to cluster 10:

If AGE >= 25 and AGE <= 40 and   HEIGHT >= 5.0ft and HEIGHT <= 5.5ft then CLUSTER = 10

The clusters are also used to generate a Bayesian probability model which is used during scoring for assigning data points to clusters.

Algorithms for Clustering

Oracle Data Mining supports the following algorithms for clustering:

- Enhanced k-Means Algorithm
- Orthogonal Partitioning Clustering (O-Cluster) Algorithm

For a brief comparison of the two clustering algorithms, see "K-Means and O-Cluster Comparison" on page 4-3.

Enhanced *k*-Means Algorithm

The *k*-Means algorithm is 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. The distance metric is either Euclidean, Cosine, or Fast Cosine distance. Data points are assigned to the nearest cluster according to the distance metric used.

ODM implements an enhanced version of the *k*-means algorithm with the following features:

- The algorithm builds models in a hierarchical manner. The algorithm builds a model top down using binary splits and refinement of all nodes at the end. In this sense, the algorithm is similar to the bisecting *k*-means algorithm. The centroid of the inner nodes in the hierarchy are updated to reflect changes as the tree evolves. The whole tree is returned.
- The algorithm grows the tree one node at a time (unbalanced approach). Based on a user setting available in either of the programming interfaces, the node with the largest variance is split to increase the size of the tree until the desired number of clusters is reached.
- The algorithm provides probabilistic scoring and assignment of data to clusters.
- The algorithm returns, for each cluster, a centroid (cluster prototype), histograms (one 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 or 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.

Data for *k*-Means The ODM implementation of *k*-Means supports both categorical and numerical data.

Data Preparation for k-Means For numerical attributes, data normalization is recommended.

For the *k*-Means algorithm, `NULL` values indicate sparse data. Missing values are not automatically handled. If the data is not sparse and the values are indeed missing at random, you should perform missing data imputation (that is, perform some kind of missing values treatment) and substitute a non-`NULL` value for the `NULL` value. One simple way to treat missing values is to use the mean for numerical attributes and the mode for categorical attributes. If you do not treat missing values, the algorithm will not handle the data correctly.

Scoring (Applying Models) The clusters discovered by enhanced *k*-Means are used to generate a Bayesian probability model that is then used during scoring (model apply) for assigning data points to clusters. The *k*-means algorithm can be interpreted as a mixture model where the mixture components are spherical multivariate normal distributions with the same variance for all components.

Orthogonal Partitioning Clustering (O-Cluster) Algorithm

The O-Cluster algorithm supports Orthogonal Partitioning Clustering; O-Cluster 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. The resulting clusters define dense areas in the attribute space. The clusters are described by intervals along the attribute axes and the corresponding centroids and histograms. A parameter called *sensitivity* defines a baseline density level. Only areas with peak density above this baseline level can be identified as clusters.

The *k*-means algorithm tessellates the space even when natural clusters may not exist. For example, if there is a region of uniform density, *k*-Means tessellates it into *n* clusters (where *n* is specified by the user). O-Cluster separates areas of high density by placing cutting planes through areas of low density. O-Cluster needs multi-modal histograms (peaks and valleys). If an area has projections with uniform or monotonically changing density, O-Cluster does not partition it.

O-Cluster Data Use 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 will only read another batch if it believes, based on statistical tests, that there may still exist clusters that it has not yet uncovered.

Because O-Cluster may stop the model build before it reads all of the data, it is highly recommended that the data be randomized.

Binning for O-Cluster The use of ODM’s equi-width binning transformation with automated estimation of the required number of bins is highly recommended.

O-Cluster Attribute Type Binary attributes should be declared as categorical.

O-Cluster Scoring The clusters discovered by O-Cluster are used to generate a Bayesian probability model that is then used during scoring (model apply) for assigning data points to clusters. 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.

Outliers and Clustering

The presence of outliers can significantly impact either type of clustering model. Use a clipping transformation before you bin or normalize the table to avoid the problems caused by outliers.

K-Means and O-Cluster Comparison

The main characteristics of the enhanced *k*-means and O-Cluster algorithms are summarized in Table 4–1.

Table 4–1 Clustering Algorithms Compared

Feature	Enhanced *k*-Means	O-Cluster
Clustering methodolgy	Distance-based	Grid-based
Number of cases	Handles data sets of any size	More appropriate for data sets that have more than 500 cases. Handles large tables via active sampling
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
Number of clusters	User-specified	Automatically determined
Hierarchical clustering	Yes	Yes
Probablistic cluster assignment	Yes	Yes
Recommended data preparation	Normalization	Equi-width binning after clipping

Association

An Association model is often used for market basket analysis, which attempts to discover relationships or correlations in a set of items. Market basket analysis is widely used in data analysis for direct marketing, catalog design, and other business decision-making processes. A typical association rule of this kind asserts that, for example, "70% of the people who buy spaghetti, wine, and sauce also buy garlic bread."

Association models capture the co-occurrence of items or events in large volumes of customer transaction data. Because of progress in bar-code technology, it is now possible for retail organizations to collect and store massive amounts of sales data. Association models were initially defined for such sales data, even though they are applicable in several other applications. Finding association rules is valuable for cross-marketing and mail-order promotions, but there are other applications as well: catalog design, add-on sales, store layout, customer segmentation, web page personalization, and target marketing.

Traditionally, association models are used to discover business trends by analyzing customer transactions. However, they can also be used effectively to predict Web page accesses for personalization. For example, assume that after mining the Web access log, Company X discovered an association rule "A and B implies C," with 80% confidence, where A, B, and C are Web page accesses. If a user has visited pages A and B, there is an 80% chance that he/she will visit page C in the same session. Page C may or may not have a direct link from A or B. This information can be used to create a dynamic link to page C from pages A or B so that the user can "click-through" to page C directly. This kind of information is particularly valuable for a Web server supporting an e-commerce site to link the different product pages dynamically, based on the customer interaction.

There are several properties of association models that can be calculated. ODM calculates the following two properties related to rules:

- **Support:** Support of a rule is a measure of how frequently the items involved in it occur together. Using probability notation, support (A implies B) = P(A, B).

- **Confidence:** Confidence of a rule is the conditional probability of B given A; confidence (A implies B) = P (B given A).

These statistical measures can be used to rank the rules and hence the predictions.

ODM uses the Apriori algorithm for association models.

Data for Association Models

Association models are designed to use *sparse data*. Sparse data is data for which only a small fraction of the attributes are non-zero or non-null in any given row. Examples of sparse data include market basket and text mining data. For example, in a market basket problem, there might be 1,000 products in the company's catalog, and the average size of a basket (the collection of items that a customer purchases in a typical transaction) might be 20 products. In this example, a transaction/case/record has on average 20 out of 1000 attributes that are not null. This implies that the fraction of non-zero attributes on the table (or the density) is 20/1000, or 2%. This density is typical for market basket and text processing problems. Data that has a significantly higher density can require extremely large amounts of temporary space to build associations.

Association models treat NULL values as sparse data. The algorithm does not handle missing values. If the data is not sparse and the NULL values are indeed missing at random, you should perform missing data imputation (that is, treat the missing values) and substitute non-null values for the NULL value.

The presence of outliers, when external equal-width binning is used, makes most of the data concentrate in a few bins (a single bin in extreme cases). As a result, the ability of the model to detect differences in numerical attributes may be significantly lessened. For example, a numerical attribute such as income may have all the data belonging to a single bin except for one entry (the outlier) that belongs to a different bin. As a result, there won't be any rules reflecting different levels of income. All rules containing income will only reflect the range in the single bin; this range is basically the income range for the whole population. In cases like this, use a clipping transformation to handle outliers.

Difficult Cases for Associations

The Apriori algorithm works by iteratively enumerating item sets of increasing lengths subject to the minimum support threshold. Since state-of-the-art algorithms for associations work by iterative enumeration, association rules algorithms do *not* handle the following cases efficiently:

- Finding associations involving rare events
- Finding associations in data sets that are dense and that have a large number of attributes.

Finding Associations Involving Rare Events

Association mining discovers patterns with frequency above the minimum support threshold. Therefore, in order to find associations involving rare events, the algorithm must run with very low minimum support values. However, doing so could potentially explode the number of enumerated item sets, especially in cases with large number of items. That could increase the execution time significantly.

Therefore, association rule mining is not recommended for finding associations involving rare events in problem domains with a large number of items.

One option is to use classification models in such problem domains.

Algorithm for Associations

Associations are calculated using the Apriori algorithm. The association mining problem can be decomposed into two subproblems:

- Find all combinations of items, called frequent itemsets, whose support is greater than the specified minimum support.
- Use the frequent itemsets to generate the desired rules. ODM Association supports single consequent rules only (for example, "ABC implies D").

The number of frequent itemsets is controlled by the minimum support parameters. The number of rules generated is controlled by the number of frequent itemsets and the confidence parameter. If the confidence parameter is set too high, there may be frequent itemsets in the association model but no rules.

Feature Extraction

Feature Extraction creates a set of features based on the original data. A *feature* is a combination of attributes that is of special interest and captures important characteristics of the data. It becomes a new attribute. Typically, there are far fewer features than there are original attributes.

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 supervised learning.

For example, 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.

Algorithm for Feature Extraction

ODM uses the Non-Negative Matrix nFactorization (NMF) algorithm for feature extraction. Non-Negative Matrix Factorization (NMF) is described in the 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). Non-Negative Matrix Factorization is 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. NMF decomposes a data matrix V into the product of two lower rank matrices W and H so that V is approximately equal to W times H. NMF uses an iterative procedure to modify the initial values of W and H so that the product approaches V. The procedure terminates when the approximation error converges or the specified number of iterations is reached. Each feature is a linear combination of the original attribute set; the coefficients of these 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.

NMF for Text Mining

Text mining involves extracting information from unstructured data. Typically, text data is high-dimensional and sparse. Unsupervised algorithms like Principal Components Analysis (PCA), Singular Value Decomposition (SVD), and NMF involve factoring the document-term matrix based on different constraints. One widely used approach for text mining is latent semantic analysis. NMF focuses on reducing dimensionality. By comparing the vectors for two adjoining segments of text in a high-dimensional semantic space, NMF provides a characterization of the degree of semantic relatedness between the segments. NMF is less complex than PCA and can be applied to sparse data. NMF-based latent semantic analysis is an attractive alternative to SVD approaches due to the additive non-negative nature of the solution and the reduced computational complexity and resource requirements.

Data Preparation for NMF

The presence of outliers can significantly impact NMF models. Use a clipping transformation before you bin or normalize the table to avoid the problems caused by outliers for these algorithms.

`NULL` values are treated as missing and not as indicators of sparse data.

NMF may benefit from normalization.

5 Data Mining Process

This chapter describes the data mining process in general and how it is supported by Oracle Data Mining. Data mining requires data preparation, model building, model testing and computing lift for a model, model applying (scoring), and model deployment. The Oracle database and Oracle Data Mining provide facilities for performing all of the data mining steps. This chapter discusses the following topics:

- How Is Data Mining Done?
- How Does Oracle Data Mining Support Data Mining?

How Is Data Mining Done?

CRISP-DM is a widely accepted methodology for data mining projects. For details, see `htttp://www.crisp-dm.org`. The steps in the process are:

1. **Business Understanding:** Understand the project objectives and requirements from a business perspective, and then convert this knowledge into a data mining problem definition and a preliminary plan designed to achieve the objectives.
2. **Data Understanding:** Start by collecting data, then get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses about hidden information.
3. **Data Preparation:** Includes all activities required to construct the final data set (data that will be fed into the modeling tool) from the initial raw data. Tasks include table, case, and attribute selection as well as transformation and cleaning of data for modeling tools.
4. **Modeling:** Select and apply a variety of modelling techniques, and calibrate tool parameters to optimal values. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements on the form of data. Therefore, stepping back to the data preparation phase is often needed.
5. **Evaluation:** Thoroughly evaluate the model, and review the steps executed to construct the model, to be certain it properly achieves the business objectives. Determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results is reached.
6. **Deployment:** Organize and present the results of data mining. Deployment can be as simple as generating a report or as complex as implementing a repeatable data mining process.

Data mining is iterative. A data mining process continues after a solution is deployed. The lessons learned during the process can trigger new business questions. Changing data can require new models. Subsequent data mining processes benefit from the experiences of previous ones.

Oracle Data Mining (ODM) supports the last three steps of CRISP-DM process. The first step, Business Understanding, is unique to your business. The remaining steps are supported by a combination of ODM and the Oracle database, especially in the context of an Oracle data warehouse. The facilities of the Oracle database can be very useful during data understanding and data preparation.

How Does Oracle Data Mining Support Data Mining?

ODM integrates data mining with the Oracle database and exposes data mining through the following interfaces:

- Java interface: Java Data Mining (JSR-73) compliant interface that allows users to embed data mining in Java applications.
- PL/SQL interface: The packages DBMS_DATA_MINING and DBMS_DATA_MINING_TRANSFORM allow users to embed data mining in PL/SQL applications.
- Automated data mining: The DBMS_PREDICTIVE_ANALYTICS PL/SQL package, described briefly in "Automated Data Mining" on page 5-2, automates the entire data mining process from data preprocessing through model building to scoring data.
- Data mining SQL functions: The SQL Data Mining functions (CLUSTER_ID, CLUSTER_PROBABILITY, CLUSTER_SET, FEATURE_ID, FEATURE_SET, FEATURE_VALUE, PREDICTION, PREDICTION_COST, PREDICTION_DETAILS, PREDICTION_PROBABILITY, and PREDICTION_SET) support deployment of models within the context of existing applications, improve scoring performance, and enable pipelining of results involving data mining predictions. For more information, see "Data Mining Functions" on page 5-3.
- Graphical interfaces: Oracle Data Miner and Oracle Spreadsheet Add-In for Predictive Analytics are graphical interfaces that solve data mining problems. See "Graphical Interfaces" on page 5-4 for a brief overview.

The end result of data mining is a model. Often this model is deployed so that its results can be embedded in an application. ODM provides facilities for deployment described in "Model Deployment" on page 5-4.

Java and PL/SQL Interfaces

The Java and PL/SQL programmatic interfaces provide the facilities to do basic data preparation (binning, normalization, winsorizing, clipping, and missing values treatment.) The two interfaces also provide calls that build, test, and apply the models described in Chapter 3 and Chapter 4.

The ODM Java interface and the ODM PL/SQL interface have the same capabilities. Models produced by either interface are interoperable, for example, a model can be built using one interface and applied using the other interface.

Automated Data Mining

The PL/SQL package `DBMS_PREDICTIVE_ANALYTICS` automates the data mining process from data preprocessing through model building to scoring new data. This automation provides a simple and intuitive interface. The package provides an important tool that simplifies data mining for users who are not data mining experts.

`DBMS_PREDICTIVE_ANALYTICS` provides the following functionality:

- `EXPLAIN` - Rank attributes in order of influence in explaining a target column
- `PREDICT` - Predict the value of an attribute

For detailed information about `DBMS_PREDICTIVE_ANALYTICS`, see the *Oracle Database PL/SQL Packages and Types Reference*.

The Oracle Spreadsheet Add-In for Predictive Analytics provides a graphical user interface to `DBMS_PREDICTIVE_ANALYTICS`; the Add-In is briefly described in "Graphical Interfaces" on page 5-4.

Data Mining Functions

The data mining functions are SQL functions that apply existing ODM models; they also return information about existing ODM models. The functions are as follows:

- `CLUSTER_ID`: Returns the cluster identifier of the predicted cluster with the highest probability for a specified set of predictors.
- `CLUSTER_PROBABILITY`: Returns a measure of the degree of confidence of membership of an input row in a cluster associated with the specified model.
- `CLUSTER_SET`: Returns a varray of objects containing all possible clusters and the probabilities for the returned clusters that a given row belongs to subject to certain filtering criteria.
- `FEATURE_ID`: Returns the identifier of the feature (in a feature extraction model) with the highest coefficient value.
- `FEATURE_SET`: Returns a varray of objects containing all possible features and the feature values in a feature extraction model subject to certain filtering criteria.
- `FEATURE_VALUE`: Returns the value of a given feature in a feature extraction model.
- `PREDICTION`: Returns the best prediction for a classification or regression model given a set of predictors.
- `PREDICTION_COST`: Returns the cost for a prediction made using an ODM classification model. Applies to decision tree models only if a cost matrix was specified during build.
- `PREDICTION_DETAILS`: Returns an XML string containing model-specific information about the scoring by an ODM decision tree or NB model of an input row.
- `PREDICTION_PROBABLILITY`: Returns the probability for a prediction made using an ODM classification model.
- `PREDICTION_SET`: Returns a varray of objects that contains all classes, the probability for each class, the costs for a decision tree model in a classification prediction.

The data mining functions have many benefits, the most important of which are the following:

- The functions make deployment of models within the context of existing applications straightforward since existing SQL statements can be easily enhanced with them.
- The functions greatly improve scoring (model apply) performance.

- The functions enable pipelining of results involving data mining predictions; this permits, among other things, the ability to return a few results quickly to an end user.

For more information about the SQL data mining functions, see the *Oracle Database SQL Reference*.

Graphical Interfaces

ODM has two graphical interfaces, both of which are available as downloads from Oracle Technology Network:

- Oracle Data Miner is a user interface to ODM that helps data analysts and application developers build advanced business intelligence applications based on ODM. ODM Java Code Generator is an extension to Oracle JDeveloper that exports models created using Oracle Data Miner to Java code.
- Oracle Spreadsheet Add-In for Predictive Analytics enables Microsoft Excel users to mine data in Oracle tables or Excel spreadsheets using the features of the `DBMS_PREDICTIVE_ANALYTICS` PL/SQL package.

Model Deployment

It is common to build models on one system and then deploy the models to a production system. The ODM Scoring Engine, described in Chapter 7, supports common deployment scenarios.

ODM supports data mining model export and import in native format between Oracle databases or schemas to provide a way to move models.

Model export/import is supported at different levels, as follows:

- Database export/import. When a DBA exports a full database using the `expdp` utility, all the existing data mining models in the database will be exported. When a DBA imports a database dump using the `impdp` utility, all the data mining models in the dump will be restored.
- Schema export/import. When a user or DBA exports a schema using `expdp`, all the data mining models in the schema will be exported. When the user or DBA imports the schema dump using `impdp`, all the models in the dump will be imported.
- Selected model export/import. Both ODM interfaces include calls that export or import specific models, for example, the PL/SQL interface includes `DBMS_DATA_MINING.export_model()` and `DBMS_DATA_MINING.import_model()`.

ODM model export and model import are based on the Oracle DBMS Data Pump. When you export a model, the tables that constitute the model and the associated metadata are written to a dump file set that consists of one or more files. When you import a model, the tables and metadata are retrieved from the file and restored in the new database.

For detailed information about model export/import, see *Oracle Data Mining Administrator's Guide*.

6 Text Mining Using Oracle Data Mining

This chapter describes support for text mining. Oracle provides support for text mining with two products:

- Oracle Data Mining (ODM)
- Oracle Text

The support for text data in ODM is different from that provided by Oracle Text. Oracle Text is dedicated to text document processing. ODM allows the combination of text (unstructured) columns and non-text (categorical and numerical) columns of data as input for clustering, classification, and feature extraction.

Oracle Text is described in the *Oracle Text Reference* and the *Oracle Text Application Developer's Guide*.

Text data is the only unstructured data type supported by ODM.

Table 6–1 summarizes how DBMS_DATA_MINING, the ODM Java interface, and Oracle Text support text mining.

Oracle Data Mining Application Developer's Guide provides information that helps you develop text mining applications using the PL/SQL and Java ODM interfaces. Oracle Data Mining Administrator's Guide contains descriptions of the sample text mining programs included with ODM.

This chapter discusses the following topics:

- What is Text Mining?
- ODM Support for Text Mining
- Oracle Support for Text Mining

What is Text Mining?

Text mining is conventional data mining 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.

Some of the applications for text mining include:

- Create and manage taxonomies
- Classify or categorize documents
- Integrate search capabilities with classification and clustering of documents returned from a search
- Extract topics automatically

- Extract features for subsequent mining

Document Classification

Document classification, also known as document categorization, is the process of assigning documents to categories (for example, themes or subjects). A particular document may fit into two or more different categories. This type of classification can often be represented as a multi target classification problem where a supervised model is built for each category.

Combining Text and Structured Data

In some classes of problems, text is combined with structured data. For example, patient records or other clinical records usually contain both structured data (temperature, blood pressure, etc.) and unstructured data (physician's notes). In such a case, you can use ODM to perform mining on the structured data, the unstructured data, or both the structured and unstructured data combined.

ODM supports mining one or more columns of text data. A column of text data must have data type `CLOB`, `BLOB`, `BFILE`, `LONG`, `VARCHAR2`, `XMLType`, `CHAR`, `RAW`, or `LONG RAW`. Before text columns can be used in mining, the features of the text columns must be extracted into a nested table. Before you can extract features, you must create a text index for the columns containing text using Oracle Text.

The sample programs distributed with ODM include examples of text mining. For information about the sample programs, see the *Oracle Data Mining Administrator's Guide*.

ODM Support for Text Mining

ODM provides infrastructure for developing data mining applications suitable for addressing a variety of business problems involving text. Among these, the following specific technologies provide key elements for addressing problems that require text mining:

- Classification
- Clustering
- Feature extraction
- Association
- Regression
- Anomaly Detection

The technologies that are most used in text mining are classification, clustering, and feature extraction.

Classification and Text Mining

A large number of document classification applications fall into one of the following:

- Assigning multiple labels to a document. ODM does not support this case.
- Assigning a document to one of many labels. For example, automatically assigning a mail message to a folder and spam filtering. This application requires multi-class classification.

The Support Vector Machine (SVM) algorithm provides powerful classifiers that have been used successfully in document classification applications. SVM can deal with thousands of features and is easy to train with small or large amounts of data. SVM is known to work well with text data. For more information about SVM, see Chapter 3.

Clustering and Text Mining

Clustering is used frequently in text mining; the main applications of clustering in text mining are

- Taxonomy generation
- Topic extraction
- Grouping the hits returned by a search engine

Clustering can also be used to group textual information with other indications from business databases to provide novel insights.

The current release of ODM supports clustering text features using both the PL/SQL and Java interfaces.

The *k*-Means clustering algorithm, described in "Enhanced k-Means Algorithm" on page 4-2, supports mining text columns.

Feature Extraction and Text Mining

There are two kinds of text mining problems for which feature extraction is useful:

- Extract features from actual text. Oracle Text is designed to solve this kind of problem. ODM also supports feature extraction from text. Most text mining is focused on this problem.
- Extract semantic features or higher-level features from the basic features uncovered when features are extracted from actual text. Statistical techniques, such as single value decomposition (SVD) and non-negative matrix factorization (NMF), are important in solving this kind of problem. Higher-order features can greatly improve the quality of information retrieval, classification, and clustering tasks.

NMF has been found to provide superior text retrieval when compared to SVD and other traditional decomposition methods. NMF takes as input a term-document matrix and generates a set of topics that represent weighted sets of co-occurring terms. The discovered topics form a basis that provides an efficient representation of the original documents. For more information about NMF, see "Algorithm for Feature Extraction" on page 4-6.

Association and Text Mining

Association models can be used to uncover the semantic meaning of words. For example, suppose that the word *sheep* co-occurs with words like *sleep*, *fence*, *chew*, *grass*, *meadow*, *farmer*, and *shear*. An association model would include rules connecting sheep with these concepts. Inspection of the rules would provide context for *sheep* in the document collection. Such associations can improve information retrieval engines. For more information about association models, see "Association" on page 4-4.

Regression and Text Mining

Regression is most often used in problems that combine text with other types of data. For more information about regression, see "Regression" on page 3-8.

Anomaly Detection and Text Mining

One-Class Support Vector Machine can be used to detect or identify novel or anomalous patterns. For more information, see "Anomaly Detection" on page 3-9.

Oracle Support for Text Mining

Table 6–1 summarizes how the ODM (both the Java and PL/SQL interfaces) and Oracle Text support text mining functions.

Table 6–1 Text Mining Comparison

Feature	ODM	Oracle Text
Association	Text data only or text and non-text data	No support
Clustering	*k*-Means algorithm supports text only or text and non-text data	*k*-means algorithm supports text only
Attribute importance	No support for text data	No support
Regression	Support Vector Machine (SVM) algorithm supports text data only or text and non-text data	No support
Classification	SVM supports text only or text and non-text data	SVM and decision trees support text only
	Support for assigning documents to one of many labels	Support for assigning documents to one of many labels and also for assigning documents to multiple labels at the same time
One-Class SVM	One-Class SVM supports text only or text and non-text data.	No support
Feature extraction (basic features)	The Java API handles the process supports the feature extraction process that transforms a text column to a nested table. The PL/SQL API requires the use of Oracle Text procedures to perform extraction. ODM allows the same degree of control as Oracle Text	Feature extraction is done internally; the results are not exposed
Feature extraction (higher order features)	Non-negative matrix factorization (NMF) supports either text or text and non-text data	No support
Record apply	No support for record apply	Supports record apply for classification
Support for text columns	Features extracted from a column of type CLOB, BLOB, BFILE. LONG, VARCHAR2, XMLType, CHAR, RAW, LONG RAW using an appropriate transformation	Supports table columns of type CLOB, BLOB, BFILE. LONG, VARCHAR2, XMLType, CHAR, RAW, LONG RAW

7

Oracle Data Mining Scoring Engine

This chapter describes one way to deploy the results of data mining. Some data-mining enabled applications have models that are developed on one system and then deployed to other (production) systems. Production applications often need only to apply models built elsewhere. The Oracle Data Mining Scoring Engine supports scoring data (applying models) using models created by other instances of Oracle Data Mining.

The Scoring Engine allows customers to limit the Oracle Data Mining (ODM) functionality available within their scoring applications to ensure that compute-intensive operations such as model building are not performed on production systems where data scoring is performed.

This chapter discusses the following topics:

- ODM Scoring Engine Features
- ODM Scoring Engine Installation
- Scoring in Data Mining Applications
- Moving Data Mining Models
- Using the Oracle Data Mining Scoring Engine

ODM Scoring Engine Features

The ODM Scoring Engine supports operations for preparing data as required from the build process, importing a model, and applying a model to data. All transformation functionality is included in the Scoring Engine. All functionality provided in the Scoring Engine behaves exactly as the same functionality in the full system.

You cannot build models using the Scoring Engine.

ODM Scoring Engine Installation

To install the ODM Scoring Engine, select during installation the “Data Mining Scoring Engine” option, a custom install option for Oracle Data Mining.

Scoring in Data Mining Applications

A single model can be used to score large volumes of data, often in multiple geographically distributed application settings. Data analysis and model building might be performed by a small group of data mining experts using data from a centralized data warehouse. However, the model can be used by a much larger number of applications working with data at geographically dispersed sites using local data. Local data may consist of millions of records representing customers; therefore, it can make sense to move the model to where the data is.

In real-time applications such as call centers, models are often built in one environment and used in another. There may be one machine dedicated to model building, using large volumes of data to produce models on a daily basis. Several other machines may be dedicated to real-time scoring, receiving new models to support, for example, the call center application. Call center representatives collect information from callers; the collected information is then used to obtain predictions or recommendations for that particular caller in real time. Scoring in real time often requires that the model is moved to where the data is.

Moving Data Mining Models

Oracle Data Mining supports data movement from one schema or database instance to another with native export and import.

Naive model export and import is based on Oracle Data Pump technology.

Native export is supported at three different levels, as follows:

- When a full database is exported using the Oracle Data Pump Export Utility (`expdp)`, all data mining models in the database are exported.
- When a schema is exported using the Oracle Data Pump Export Utility (`expdp)`, all data mining models in the schema are exported.
- An ODM user can export one or more specific models in a schema using the PL/SQL `DBMS_DATA_MINING.export_model` procedure or the Java task `javax.datamining.task.ExportTask`.

Native import is also supported in all scenarios. Using the dump file set produced by the Oracle Data Pump Export Utility (`expdp)`, an ODM user can run the Oracle Data Pump Import Utility (`impdp)` to import all data mining models contained in the dump file set.

ODM users can import a specific model from the dump file set using the PL/SQL `DBMS_DATA_MINING.import_model` procedure or the Java `javax.datamining.task.ImportTask`.

You can also perform native export and import using the ODM Java interface; for an example, see the `dmexpimpdemo.java` sample program.

For more information about native export and import, see the *Oracle Data Mining Administrator's Guide*, the DBMS_DATA_MINING chapter in the *Oracle Database PL/SQL Packages and Types Reference*, and the *Oracle Data Mining Application Developer's Guide*.

Using the Oracle Data Mining Scoring Engine

Suppose that the application builds a model on one system named BUILDSYS and applies the model on a different system named SCORESYS.

BUILDSYS must have ODM installed. ODM supports all data mining activities (building models, testing models, applying models, etc.). SCORESYS has the ODM Scoring Engine installed. It will not be possible to build models on SCORESYS, but it will be possible to apply the model. Note that SCORESYS could have a full ODM product installation, but model building may interfere with scoring performance.

The following processing takes place:

1. Build models on BUILDYSYS. Select the appropriate model to deploy.
2. Export the model using the ODM PL/SQL or Java interface.
3. Copy the dump file generated by model export to SCORESYS.
4. Import the model on SCORESYS using the ODM PL/SQL or Java interface.
5. Apply the model to data on SCORESYS.

8 Sequence Similarity Search and Alignment (BLAST)

This chapter describes Oracle Data Mining support for certain problems in the life sciences. In addition to data mining functions that produce supervised and unsupervised models, ODM supports the sequence similarity search and alignment algorithm Basic Local Alignment Search Tool (BLAST). In life sciences, vast quantities of data including nucleotide and amino acid sequences are stored, typically in a database. These sequence data help biologists determine the chemical structure, biological function, and evolutionary history of organisms. A key feature of managing the exponential growth in sequence data sources is the availability of fast, sensitive, and statistically rigorous techniques for detecting similarities between these sequences.

As the amount of nucleotide and amino acid sequence data continues to grow, the data becomes increasingly useful in the analysis of newly sequenced genes and proteins because of the greater chance of finding such sequence similarities.

This chapter discusses the following topics:

- Bioinformatics Sequence Search and Alignment
- BLAST in the Oracle Database
- Oracle Data Mining Sequence Search and Alignment Capabilities

For detailed information about Oracle's implementation of BLAST, see *Oracle Data Mining Application Developer's Guide*.

Bioinformatics Sequence Search and Alignment

Sequence alignment is one of the most common bioinformatics tasks. It is present in almost any research and development activity across the many industries in the area of life sciences including academia, biotech, services, software, pharmaceutical companies, and hospitals.

The National Center for Biotechnology Information (NCBI) developed one of the commonly used versions of the Basic Local Alignment Search Tool (BLAST).

Of all the sequence alignment algorithms, the one that is most widely used is BLAST. It is typically used to compare one query nucleotide or protein sequence against a database of sequences, and uncover similarities and sequence matches. Its success and popularity comes from its combination of speed, sensitivity, and statistical assessment of the results.

BLAST is a heuristic method to find the high-scoring locally optimal alignments between a query sequence and a database. The BLAST algorithm and family of programs rely on the statistics of gapped and un-gapped sequence alignments. The statistics allow the probability of obtaining an alignment with a particular score to be estimated. BLAST is unlikely to be as sensitive for all protein searches as a full dynamic programming algorithm. However, the underlying statistics provide a direct estimate of the significance of any match found.

The inclusion of BLAST in ODM positions the Oracle database as a platform for bioinformatics.

For more information about BLAST, see the NCBI BLAST Home Page at `http://www.ncbi.nlm.nih.gov/BLAST`.

BLAST in the Oracle Database

Implementing BLAST in the database provides the following benefits:

- You can include BLAST in complex queries, thereby enabling complex analytical pipelines that include BLAST searches.
- You can subselect portions of the database using SQL, thereby restricting searches.
- Since sequence data is already stored in the database, it is not necessary to export the sequence data and pre-process them to create BLAST data sets and then import the results back into the database.

Oracle Data Mining Sequence Search and Alignment Capabilities

Sequence search and alignment, with capabilities similar to those of NCBI BLAST 2.0, has been implemented in the database using table functions. This implementation enables users to perform queries against data that is held directly inside an Oracle database. As the algorithms are implemented as table functions, parallel computation is intrinsically supported.

The five core variants of BLAST have been implemented:

- BLASTN compares a nucleotide query sequence against a nucleotide database.
- BLASTP compares a protein query sequence against a protein sequence database.
- BLASTX compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database.
- TBLASTN compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands).
- TBLASTX compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

The BLAST table functions are implemented in the database, and can be invoked by SQL. Using SQL, it is possible to pre-process the sequences as well as perform any required post-processing. This additional processing capability means it is possible to combine BLAST searches with queries that involve images, date functions, literature search, etc. Using these complex queries make it possible to perform BLAST searches on a required subset of data, potentially resulting in highly performant queries. This functionality is expected to provide considerable value.

The performance of BLAST searches can be improved if the data set is transformed into a compressed binary format and used in the searches. BLASTN_COMPRESS() compresses nucleotide sequence data. The BLAST queries can use this compressed format for searches.

BLAST queries can be invoked directly using the SQL interface or through an application. The query below shows an example of a SQL-invoked BLAST search where a protein sequence is compared with the protein database SwissProt, and sequences are filtered so that only human sequences that were deposited after 1 January 1990 are searched against. The column of numbers at the end of the query reflects the parameters chosen.

```
select t_seq_id, alignment_length, q_seq_start, q_seq_end
       q_frame, t_seq_start, t_seq_end, t_frame, score, expect
  from TABLE(
       BLASTP_ALIGN (
         (select sequence from query_db),
         CURSOR(SELECT seq_id, seq_data
                FROM swissprot
                WHERE organism = 'Homo sapiens (Human)' AND
                      creation_date > '01-Jan-90'),
         1,
         -1,
         0,
         0,
         'BLOSUM62',
         10,
         0,
         0,
         0,
         0,
         0)
       );
```

The results of a SQL query can be displayed either through an application or a SQL*Plus interface. When the SQL*Plus interface is used, the user can decide how the results will be displayed. The following shows an example of the format of the output that could be displayed by the SQL query shown above.

```
T_SEQ_ID  ALIGNMENT_LENGTH  Q_SEQ_START  Q_SEQ_END  Q_FRAME  T_SEQ_START  T_SEQ_END T_FRAME  SCORE  EXPECT
----------
P31946         50            0          50         0         13          63         0      205    5.1694E-18

Q04917         50            0          50         0         12          62         0      198    3.3507E-17

P31947         50            0          50         0         12          62         0      169    7.7247E-14

P27348         50            0          50         0         12          62         0      198    3.3507E-17

P58107         21            30         51         0        792         813         0      94     6.34857645
```

The first row of information represents some of the attributes returned by the query: the target sequence ID, the length of the alignment, the position where the alignment starts on the query sequence, the position where the alignment ends on the query sequence, which open reading frame was used for the query sequence, among others.

The next five rows represent sequence alignments that were returned; for example, the protein with the highest alignment to query sequence has the accession number "P31946", the alignment length was 50 amino acids, the alignment started at the first base of the amino acid query, and ended with the 50th base of the amino acid query.

Glossary

ABN

See **adaptive bayes network**.

active learning

A feature of the **support vector machine** algorithm that provides a way to deal with large build sets.

adaptive bayes network

An Oracle proprietary classification algorithm. ABN provides a fast, scalable, non-parametric means of extracting predictive information from data with respect to a target attribute. It operates in three modes: pruned naive bayes, single feature, and multi-feature. Single feature mode produces rules.

aggregation

The process of consolidating data values into a smaller number of values. For example, sales data could be collected on a daily basis and then be totalled to the week level.

algorithm

A sequence of steps for solving a problem. See **data mining algorithm**. The ODM programmatic interfaces support the following algorithms: **ABN**, **MDL**, **apriori**, **decision tree**, **k-means**, **naive bayes**, **non-negative matrix factorization**, **O-cluster**, and **support vector machine**.

algorithm settings

The settings that specify algorithm-specific behavior for model building.

anomaly detection

The detection of outliers or atypical cases. To build an anomaly detection model using the ODM programmatic interfaces, specify classification as the mining function, SVM as the algorithm, and pass a `NULL` or empty string as the target column name.

apply

The data mining operation that scores data, that is, uses the model with new data to predict results.

apply settings

In JDM, an object used to specify the kind of output desired from applying a model to data. This output may include predicted values, associated probabilities, key values, and other data.

apriori

Uses frequent itemsets to calculate associations.

association

A machine learning technique that identifies relationships among items.

association rules

A mining 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 item set A implies the presence of item set B with certain support and confidence. The support of the rule is the ratio of the number of transactions where the item sets 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 item sets A and B are present to the number of transactions where item set A is present. ODM uses the Apriori algorithm for association models.

attribute

An attribute corresponds to a column in a database table. An attribute has a name and a data type. Each attribute in a record holds an item of information. Attribute names are constant from record to record for unnested tables. Attributes are also called *variables*, *features*, *data fields*, or *table columns*. See also **target**.

attribute importance

A mining function providing a measure of the importance of an attribute in predicting a specified target. The measure of different attributes of a build data table enables users to select the attributes that are found to be most relevant to a mining model. A smaller set of attributes results in a faster model build; the resulting model could be more accurate. ODM uses the **minimum description length** to discover important attributes. Sometimes referred to as *feature selection* or *key fields*.

binning

See **discretization**.

build settings

In JDM, an object that captures the high-level specifications used to build a model. Build settings must specify a mining function; they may specify an algorithm. ODM supports the following mining functions: classification, regression, association, attribute importance, and clustering. **anomaly detection** is support through the classification mining function.

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.

categorical attribute

An attribute whose values correspond to discrete categories. For example, *state* is a categorical attribute with discrete values (CA, NY, MA, etc.). Categorical attributes are either non-ordered (nominal) like state, gender, etc., or ordered (ordinal) such as high, medium, or low temperatures.

category

In the Java interface, corresponds to a distinct value of a categorical attribute. Categories may have character or numeric values.

centroid

See **cluster centroid**.

classification

A mining function for predicting categorical target values for new records using a model built from records with known target values. ODM supports the following algorithms for classification: Naive Bayes, Adaptive Bayes Networks, Decision Tree, and Support Vector Machines.

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 build data assigned to a cluster. A cluster centroid is often referred to as "the centroid."

clustering

A mining 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. ODM supports two algorithms for clustering,**k-means** and **orthogonal partitioning clustering**.

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.

connection

In JDM, an object used to connect to the data mining server in an Oracle database to perform data mining tasks.

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 machine**s.

data mining

The process of discovering hidden, previously unknown, and usable information from a large amount of data. This information is represented in a compact form, often referred to as a *model*.

data mining algorithm

A specific technique or procedure for producing a data mining model. An algorithm uses a specific model representation and may support one or more **mining function**s. The algorithms in the ODM programming interfaces are **naive bayes**, **adaptive bayes network**, **support vector machine**, and **decision tree** for classification; **support vector machine** for regression; **k-means** and **O-cluster** for clustering; **minimum description length** for attribute importance; **non-negative matrix factorization** for feature extraction; **apriori** for associations, and **one-class support vector machine** for anomaly detection.

data mining server

The component of the Oracle database that implements the data mining engine and persistent metadata repository. You must connect to a data mining server before performing data mining tasks. The data mining server is the ODM implementation of the JDM Data Mining Engine (DME).

data set

In general, a collection of data. A data set is a collection of **case**s.

descriptive model

A descriptive model helps in understanding underlying processes or behavior. For example, an association model describes consumer behavior. See also **mining model**.

discretization

Discretization 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 ODM algorithms (NB, ABN, etc.) 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 Data Mining includes transformations that perform **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**.

DMS

See **data mining server**.

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.

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 **network feature** and **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**.

JDM

See **Java Data Mining**.

Java Data Mining

A Pure Java API that facilitates development of data mining-enabled applications. Java Data Mining (JDM) supports common data mining operations, as well as the creation, persistence, access, and maintenance of meta data supporting mining activities. JDM is described in the Java Community Process Specification JSR-73. The Java interface to Oracle Data Mining is a compliant subset of JDM.

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. ODM 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 data mining 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 will perform on new data.

lineage

The sequence of transformations performed on a data set during the data preparation phase of the model build process.

MDL

See **minimum description length**.

min-max normalization

Normalize each attribute using the 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

This principle is used to select the attributes that most influence target value discrimination in **attribute importance**.

mining activity

In the Oracle Data Miner user interface, a mining activity is a step-by-step guide to model build or model apply. The steps indicate the order in which operations must be applied and appropriate defaults for the operations. There are two basic mining activities: *build activity* to build and test a model and *apply activity* to apply a model to new data (score new data using the model).

mining function

A major subdomain of data mining that shares common high level characteristics. The ODM programming interfaces support the following mining functions: **classification**, **regression**, **attribute importance**, **feature extraction**, and **clustering**. In both programming interfaces, anomaly detection is supported as classification.

mining model

An important function of data mining is the production of a model. A model can be a **supervised model** or an **unsupervised model**. Technically, a mining model is the result of building a model from mining settings. The representation of the model is specific to the algorithm specified by the user or selected by the DMS. A model can be used for direct inspection, e.g., to examine the rules produced from an ABN model or association models, or to score data.

mining object

Mining tasks, models, settings, and their components.

mining result

The end product(s) of a mining task. For example, a build task produces a mining model; a test task produces a test result.

mining task

See **task**.

missing value

A data value that is missing because it was not measured (that is, has a null value), not answered, was unknown, or was lost. Data mining algorithms vary in the way they treat missing values. There are several typical ways to treat them: ignore then, omit any records containing missing values, replace missing values with the mode or mean, or infer missing values from existing values.

model

In JDM, the object resulting from the application of an algorithm to data, as specified in a **build settings** object. See also **mining model**.

multi-record case

Each case in the data is stored as multiple records in a table with columns `sequenceID`, `attribute_name`, and `value`. Also known as **transactional format**. 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 table

An unordered set of data elements, all of the same data type. It has a single column, and the type of that column is a built-in type or an object type. If an object type, the table can also be viewed as a multicolumn table, with a column for each attribute of the object type. Nested tables can contain other nested tables.

network feature

A network feature is a tree-like multi-attribute structure. From the standpoint of the network, features are conditionally independent components. Features contain at least one attribute (the root attribute). Network features are used in the Adaptive Bayes Network algorithm.

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 Data Mining 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 the **support vector machine** model used to solve **anomaly detection** problems. The ODM programmatic interfaces implement the one-class algorithm as classification.

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; in other words, extreme values. In a normal distribution, outliers are typically at least 3 standard deviations from the mean.

physical data set

In JDM, an object that identifies data to be used as input to data mining. The data referenced by a *physical data set* object can be used in model build, model apply (scoring), lift computation, statistical analysis, and similar operations. The physical data specification includes information about the format of the data and the roles that the data columns play. In the ODM Java interface, the physical data set must be a table or view within the database instance specified in the connection object.

positive target value

In binary classification problems, you may designate one of the two classes (target values) as positive, the other as negative. When ODM 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 mining 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 model or 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 data mining project. ODM includes transformations to perform common data preparation functions (binning, normalization, etc.)

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 many instances in data mining. Here are some examples:

- 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 ODM 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 data mining function for predicting continuous target values for new records using a model built from records with known target values. ODM provides the Support Vector Machine algorithm for regression.

rule

An expression of the general form *if X, then Y*. An output of certain algorithms, such as clustering, association, decision tree, and ABN. The predicate *X* may be a compound predicate.

sample

See **random sample**.

scale normalization

Normalize each attribute using the transformation x_new = (x-0)/ max(abs(max), abs(min)).

schema

Database schema, that is, a collection of database objects, including logical structures such as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links.

score

Scoring data means applying a data mining model to data to generate predictions. See **apply**.

scoring engine

An instance of Oracle Data Mining that can be used to apply or score models, but cannot be used to build models.

settings

See **algorithm settings** and **build settings**.

single-record case

Each case in the data is stored as one record (row) in a table. Contrast with **multi-record case**.

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 text mining data are often sparse.

split

Divide a data set into several disjoint subsets. For example, in a classification problem, a data set is often divided in to a build 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 data mining algorithms with enough positive examples to learn the factors that differentiate positive from negative target values. See also **random sample**.

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 machines 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 machine**.

table

The basic unit of data storage in an Oracle database. Table data is stored in rows and columns.

target

In supervised learning, the identified attribute that is to be predicted. Sometimes called *target value* or *target attribute*. See also **attribute**.

test metrics

In JDM, an object results from the testing of a supervised model with test data. The test metrics computed depend on the mining function. For classification, the accuracy, confusion-matrix, lift, and receiver-operating characteristics can be computed to access the model. Similarly for regression, R-squared and RMS errors can be computed. Test metrics can also be computed using the PL/SQL interface.

task

In JDM, an object that represents all the information needed to perform a mining operation, such as build, test, apply, import, and export. A task is executed using the execute method of a connection object.

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 mining

Conventional data mining 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 ODM and Oracle Text support text mining.

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.

transactional format

Each case in the data is stored as multiple records in a table with columns `sequenceID`, `attribute_name`, and `value`. Also known as **multi-record case**. Compare with **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 used for dealing with outliers. Trimming removes values in the tails of a distribution in the sense that trimmed values are ignored in further computations. This 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. ODM supports the mining of unstructured text data.

unsupervised learning

See **unsupervised model**.

unsupervised model

A data mining 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 mining functions. See **supervised model**.

view

A view takes the output of a query and treats it as a table. Therefore, a view can be thought of as a stored query or a virtual table. You can use views in most places where a table can be used.

winsorizing

A way of dealing with 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 the mean and standard deviation computed from the data. Normalize each attribute using the transformation x_new = (x-mean)/standard deviation.

Index