Oracle® Data Mining

Concepts

12c Release 1 (12.1)

E17692-15

January 2014

Oracle Data Mining Concepts, 12c Release 1 (12.1)

E17692-15

Primary Author: Kathy L. Taylor

Contributor: The Oracle Database 12*c* documentation is dedicated to Mark Townsend, who was an inspiration to all who worked on this release.

Contents

Preface

This manual describes Oracle Data Mining, a comprehensive, state-of-the-art data mining capability within Oracle Database. This manual presents the concepts that underlie the procedural information that is presented in *Oracle Data Mining User's Guide*.

The preface contains these topics:

- Audience
- Related Documentation
- Documentation Accessibility
- Conventions

Audience

Oracle Data Mining Concepts is intended for anyone who wants to learn about Oracle Data Mining.

Related Documentation

Oracle Data Mining, a component of Oracle Advanced Analytics, is documented on the Data Warehousing and Business Intelligence page of the Oracle Database online documentation library:

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=dwbitab`

The following manuals document Oracle Data Mining:

- *Oracle Data Mining Concepts* (this manual)
- *Oracle Data Mining User's Guide*
- *Oracle Data Mining API Guide* (a virtual book)

Note: The virtual book combines key passages from the Data Mining manuals with related reference documentation in *Oracle Database PL/SQL Packages and Types Reference*, *Oracle Database SQL Language Reference*, and *Oracle Database Reference*.

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

Oracle Data Mining Resources on the Oracle Technology Network

The Oracle Data Mining page on the Oracle Technology Network (OTN) provides a wealth of information, including white papers, demonstrations, blogs, discussion forums, and Oracle By Example tutorials:

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datmin`

Oracle Data Miner, the graphical user interface to Oracle Data Mining, is an extension to Oracle SQL Developer. Instructions for downloading SQL Developer and installing the Data Miner repository are available on the Oracle Technology Network:

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datminGUI`

Application Development and Database Administration Documentation

For documentation to assist you in developing database applications and in administering Oracle Database, refer to the following:

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

Documentation Accessibility

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Conventions

The following text conventions are used in this document:

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

Changes in This Release for Oracle Data Mining Concepts

This preface lists changes in *Oracle Data Mining Concepts*.

Changes in Oracle Data Mining 12*c* Release 1 (12.1)

The following are changes in *Oracle Data Mining Concepts* for Oracle Database 12*c* Release 1 (12.1).

New Features

The following features are new in this release:

- New clustering algorithm: Expectation Maximization

  In addition to enhanced *k*-Means and O-Cluster, Oracle Data Mining now supports Expectation Maximization, a probabilistic clustering algorithm that creates a density model of the data. The density model allows for an improved approach to combining data originating in different domains (for example, sales transactions and customer demographics, or structured data and text or other unstructured data).

  Because of the probabilistic nature of Expectation Maximization, its cluster assignment probabilities may be more reliable than those produced by *k*-Means or O-Cluster. Also, the Expectation Maximization algorithm automatically determines the optimal number of clusters needed to model the data.

  See Chapter 12, "Expectation Maximization".

- New feature extraction algorithm: Singular Value Decomposition with Principal Component Analysis

  In addition to Non-Negative Matrix Factorization, Oracle Data Mining now supports Singular Value Decomposition and Principal Component Analysis, two powerful feature extraction methods that use orthogonal linear projections to capture the underlying variance of the data. Principal Component Analysis is implemented as a special scoring method for the Singular Value Decomposition algorithm.

  Singular Value Decomposition and Principal Component Analysis scale well to very large data sizes (both rows and attributes), and they have a powerful data compression capability. With the introduction of these new methods, Oracle Data Mining extends its feature extraction capabilities to new contexts involving time series, unstructured data, and very large numerical data sets (for example, data from sensors such as Radio Frequency Identification).

See Chapter 19, "Singular Value Decomposition".

- Generalized Linear Models enhanced to support feature selection and creation

  Generalized Linear Models provide great transparency, which may be achieved at the expense of accuracy. With the introduction of a feature selection and creation capability, Generalized Linear Models can maintain a high degree of accuracy without sacrificing transparency (the ability to explain the predictions made by the model).

  Feature selection is the process of selecting the most meaningful attributes. Feature creation is the process of combining attributes into features. With feature selection, Generalized Linear Models can be created with fewer predictors, leading to smaller models and faster scoring. With feature creation, Generalized Linear Models use non-linear terms (up to cubic terms), leading to more powerful models and increased transparency.

  See Chapter 13, "Generalized Linear Models".

- Significant enhancements in text mining

  This enhancement greatly simplifies the data mining process (model build, deployment and scoring) when unstructured text data is present in the input:

  See "Text Data" on page 3-6. (See *Oracle Data Mining User's Guide* for details.)

- Prediction details expanded

  The `PREDICTION_DETAILS` function now supports all predictive algorithms and returns more details about the predictors. New functions, `CLUSTER_DETAILS` and `FEATURE_DETAILS`, are introduced.

  See "In-Database Scoring" on page 3-6 for information about the Data Mining SQL functions. (See *Oracle Database SQL Language Reference* for details.)

- Dynamic scoring

  The Data Mining SQL functions now support an analytic clause for scoring data dynamically without a pre-defined model.

  See "In-Database Scoring" on page 3-6 for information about the Data Mining SQL functions. (See *Oracle Database SQL Language Reference* for details.)

Desupported Features

The following features are no longer supported by Oracle. See *Oracle Database Upgrade Guide* for a complete list of desupported features in this release.

- Oracle Data Mining Java API
- Adaptive Bayes Network (ABN) algorithm

Other Changes

The following are additional changes in *Oracle Data Mining Concepts* for 12c Release 1 (12.1):

- The single-chapter product overview that was previously in Part I has been divided into two chapters:
  - Chapter 2, "Introduction to Oracle Data Mining"
  - Chapter 3, "Oracle Data Mining Basics"
- The chapter on Predictive Analytics that was previously in Part I has been removed.

This chapter was based on examples that were generated by Oracle Spreadsheet Add-In for Predictive Analytics. The Spreadsheet Add-In is still available for download on the Oracle Technology Network.

`http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/odm-pred-analytics-addin-092973.html`

Part I

Introductions

Part I presents an introduction to Oracle Data Mining. The first chapter is a general, high-level overview for those who are new to data mining technology.

Part I contains these chapters:

1

What Is Data Mining?

This chapter provides a high-level orientation to data mining technology. This chapter contains these topics:

- What Is Data Mining?
- What Can Data Mining Do and Not Do?
- The Data Mining Process

Note: Information about data mining is widely available. No matter what your level of expertise, you will be able to find helpful books and articles on data mining. For example:
`http://en.wikipedia.org/wiki/Data_mining`

What Is Data Mining?

Data mining is a technique that discovers previously unknown relationships in data.

Data mining is the practice of automatically searching large stores of data to discover patterns and trends that go beyond simple analysis. Data mining uses sophisticated mathematical algorithms to segment the data and to predict the likelihood of future events based on past events. Data mining is also known as Knowledge Discovery in Data (KDD).

The key properties of data mining are:

- Automatic discovery of patterns
- Prediction of likely outcomes
- Creation of actionable information
- Focus on large data sets and databases

Data mining can answer questions that cannot be addressed through simple query and reporting techniques.

Automatic Discovery

Data mining is performed by a model that uses an algorithm to act on a set of data. Data mining models can be used to mine the data on which they are built, but most types of models are generalizable to new data. The process of applying a model to new data is known as **scoring**.

Prediction

Many forms of data mining are predictive. For example, a model might predict income based on education and other demographic factors. Predictions have an associated probability (How likely is this prediction to be true?). Prediction probabilities are also known as **confidence** (How confident can I be of this prediction?).

Some forms of predictive data mining generate **rules**, which are conditions that imply a given outcome. For example, a rule might specify that a person who has a bachelor's degree and lives in a certain neighborhood is likely to have an income greater than the regional average. Rules have an associated **support** (What percentage of the population satisfies the rule?).

Grouping

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

Actionable Information

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

Data Mining and Statistics

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

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

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

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

Data Mining and OLAP

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

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

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

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

Data Mining and Data Warehousing

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

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

What Can Data Mining Do and Not Do?

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

You might already be aware of important patterns as a result of working with your data over time. Data mining can confirm or qualify such empirical observations in addition to finding new patterns that may not be immediately discernible through simple observation.

It is important to remember that the predictive relationships discovered through data mining are not *causal* relationships. For example, data mining might determine that males with incomes between $50,000 and $65,000 who subscribe to certain magazines are likely to buy a given product. You can use this information to help you develop a marketing strategy. However, you should not assume that the population identified through data mining will buy the product *because* they belong to this population.

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

Asking the Right Questions

Data mining does not automatically discover information without guidance. The patterns you find through data mining will be very different depending on how you formulate the problem.

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

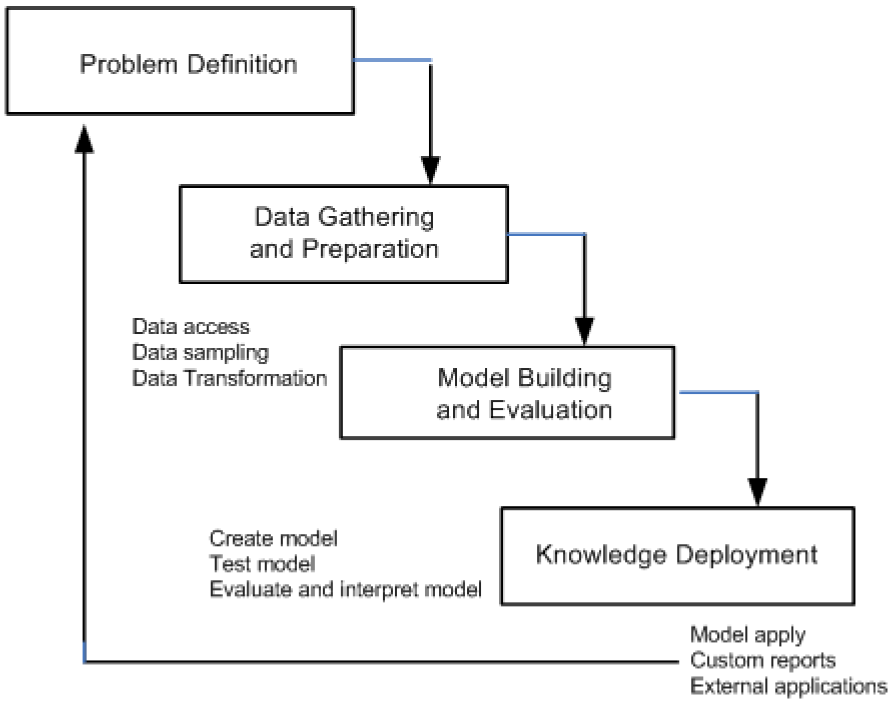
Understanding Your Data

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

The Data Mining Process

Figure 1–1 illustrates the phases, and the iterative nature, of a data mining project. The process flow shows that a data mining project does not stop when a particular solution is deployed. The results of data mining trigger new business questions, which in turn can be used to develop more focused models.

Figure 1–1 The Data Mining Process



Problem Definition

This initial phase of a data mining project focuses on understanding the project objectives and requirements. Once you have specified the problem from a business perspective, you can formulate it as a data mining problem and develop a preliminary implementation plan.

For example, your business problem might be: "How can I sell more of my product to customers?" You might translate this into a data mining problem such as: "Which customers are most likely to purchase the product?" A model that predicts who is most likely to purchase the product must be built on data that describes the customers who have purchased the product in the past. Before building the model, you must assemble the data that is likely to contain relationships between customers who have purchased the product and customers who have not purchased the product. Customer attributes might include age, number of children, years of residence, owners/renters, and so on.

Data Gathering and Preparation

The data understanding phase involves data collection and exploration. As you take a closer look at the data, you can determine how well it addresses the business problem. You might decide to remove some of the data or add additional data. This is also the time to identify data quality problems and to scan for patterns in the data.

The data preparation phase covers all the tasks involved in creating the table or view that you will use to build the model. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks may include column selection and the creation of views, as well as data cleansing and transformation. For example, you might transform a `DATE_OF_BIRTH` column to `AGE`; you might insert the median income in cases where the `INCOME` column is null.

Additionally you might add new computed attributes in an effort to tease information closer to the surface of the data. For example, rather than using the purchase amount, you might create a new attribute: "Number of Times Amount Purchase Exceeds $500 in a 12 month time period." Customers who frequently make large purchases may also be related to customers who respond or don't respond to an offer.

Thoughtful data preparation can significantly improve the information that is discovered through data mining.

Note: Oracle Data Mining supports Automatic Data Preparation (ADP), which greatly simplifies the process of data preparation. (See "Data Preparation" on page 3-5.)

Model Building and Evaluation

In this phase, you select and apply various modeling techniques and calibrate the parameters to optimal values. If the algorithm requires data transformations, you will need to step back to the previous phase to implement them.

In preliminary model building, it often makes sense to work with a reduced set of data since the final data set might contain thousands or millions of rows.

At this stage of the project, it is time to evaluate how well the model satisfies the originally-stated business goal (phase 1). If the model is supposed to predict customers who are likely to purchase a product, does it sufficiently differentiate between the two classes? Is there sufficient lift? Are the trade-offs shown in the confusion matrix acceptable? Would the model be improved by adding text data? Should transactional data such as purchases (market-basket data) be included? Should costs associated with false positives or false negatives be incorporated into the model?

Knowledge Deployment

Knowledge deployment is the use of data mining within a target environment. In the deployment phase, insight and actionable information can be derived from data.

Deployment can involve scoring (the application of models to new data), the extraction of model details (for example the rules of a decision tree), or the integration of data mining models within applications, data warehouse infrastructure, or query and reporting tools.

Because Oracle Data Mining builds and applies data mining models inside Oracle Database, the results are immediately available. BI reporting tools and dashboards can easily display the results of data mining. Additionally, Oracle Data Mining supports scoring in real time: Data can be mined and the results returned within a single database transaction. For example, a sales representative could run a model that predicts the likelihood of fraud within the context of an online sales transaction.

2 Introduction to Oracle Data Mining

This chapter introduces Oracle Data Mining and describes its place within Oracle Database. This chapter contains these topics:

- About Oracle Data Mining
- Data Mining in the Database Kernel
- Data Mining in Oracle Exadata
- Interfaces to Oracle Data Mining
- Overview of Database Analytics

About Oracle Data Mining

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

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

Oracle Data Mining is one of the two components of the **Oracle Advanced Analytics Option** of Oracle Database Enterprise Edition. The other component is Oracle R Enterprise, which integrates R, the open-source statistical environment, with Oracle Database. Together, Oracle Data Mining and Oracle R Enterprise constitute a comprehensive advanced analytics platform for big data analytics.

Data Mining in the Database Kernel

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

Data mining within Oracle Database offers many advantages:

- **No Data Movement**. Some data mining products require that the data be exported from a corporate database and converted to a specialized format for mining. With

Oracle Data Mining, no data movement or conversion is needed. This makes the entire mining process less complex, time-consuming, and error-prone, and it allows for the mining of very large data sets.

- **Security**. Your data is protected by the extensive security mechanisms of Oracle Database. Moreover, specific database privileges are needed for different data mining activities. Only users with the appropriate privileges can define, manipulate, or apply mining model objects.
- **Data Preparation and Administration**. Most data must be cleansed, filtered, normalized, sampled, and transformed in various ways before it can be mined. Up to 80% of the effort in a data mining project is often devoted to data preparation. Oracle Data Mining can automatically manage key steps in the data preparation process. Additionally, Oracle Database provides extensive administrative tools for preparing and managing data.
- **Ease of Data Refresh**. Mining processes within Oracle Database have ready access to refreshed data. Oracle Data Mining can easily deliver mining results based on current data, thereby maximizing its timeliness and relevance.
- **Oracle Database Analytics**. Oracle Database offers many features for advanced analytics and business intelligence. Oracle Data Mining can easily be integrated with other analytical features of the database, such as statistical analysis and OLAP. (See "Overview of Database Analytics" on page 2-5.)
- **Oracle Technology Stack**. You can take advantage of all aspects of Oracle's technology stack to integrate data mining within a larger framework for business intelligence or scientific inquiry.
- **Domain Environment**. Data mining models have to be built, tested, validated, managed, and deployed in their appropriate application domain environments. Data mining results may need to be post-processed as part of domain specific computations (for example, calculating estimated risks and response probabilities) and then stored into permanent repositories or data warehouses. With Oracle Data Mining, the pre- and post-mining activities can all be accomplished within the same environment.
- **Application Programming Interfaces**. The PL/SQL API and SQL language operators provide direct access to Oracle Data Mining functionality in Oracle Database.

Data Mining in Oracle Exadata

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

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

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

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

Interfaces to Oracle Data Mining

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

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

PL/SQL API

The Oracle Data Mining PL/SQL API is implemented in the `DBMS_DATA_MINING` PL/SQL package, which contains routines for building, testing, and maintaining data mining models. A batch apply operation is also included in this package.

Example 2–1 shows part of a simple PL/SQL script for creating an SVM classification model called SVMC_SH_Clas_sample. The model build uses weights, specified in a weights table, and settings, specified in a settings table. The weights influence the weighting of target classes. The settings override default behavior. The model uses Automatic Data Preparation (`prep_auto_on` setting). The model is trained on the data in mining_data_build_v.

Example 2–1 Creating a Classification Model

```
-----------------------  CREATE AND POPULATE A CLASS WEIGHTS TABLE  ------------
CREATE TABLE svmc_sh_sample_class_wt (
  target_value NUMBER,
  class_weight NUMBER);
INSERT INTO svmc_sh_sample_class_wt VALUES (0,0.35);
INSERT INTO svmc_sh_sample_class_wt VALUES (1,0.65);
COMMIT;
-----------------------  CREATE AND POPULATE A SETTINGS TABLE ------------------
CREATE TABLE svmc_sh_sample_settings (
  setting_name  VARCHAR2(30),
  setting_value VARCHAR2(4000));
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.svms_kernel_function, dbms_data_mining.svms_linear);
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.clas_weights_table_name, 'svmc_sh_sample_class_wt');
INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.prep_auto, dbms_data_mining.prep_auto_on);
END;
/
------------------------  CREATE THE MODEL -------------------------------------
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'SVMC_SH_Clas_sample',
    mining_function     => dbms_data_mining.classification,
    data_table_name     => 'mining_data_build_v',
    case_id_column_name => 'cust_id',
    target_column_name  => 'affinity_card',
    settings_table_name => 'svmc_sh_sample_settings');
END;
/
```

SQL Functions

The Data Mining SQL functions perform prediction, clustering, and feature extraction. The functions score data by applying a mining model object or by executing an analytic clause that performs dynamic scoring.

Example 2–2 shows a query that applies the classification model svmc_sh_clas_sample to the data in the view mining_data_apply_v. The query returns the average age of customers who are likely to use an affinity card. The results are broken out by gender.

Example 2–2 The PREDICTION Function

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

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

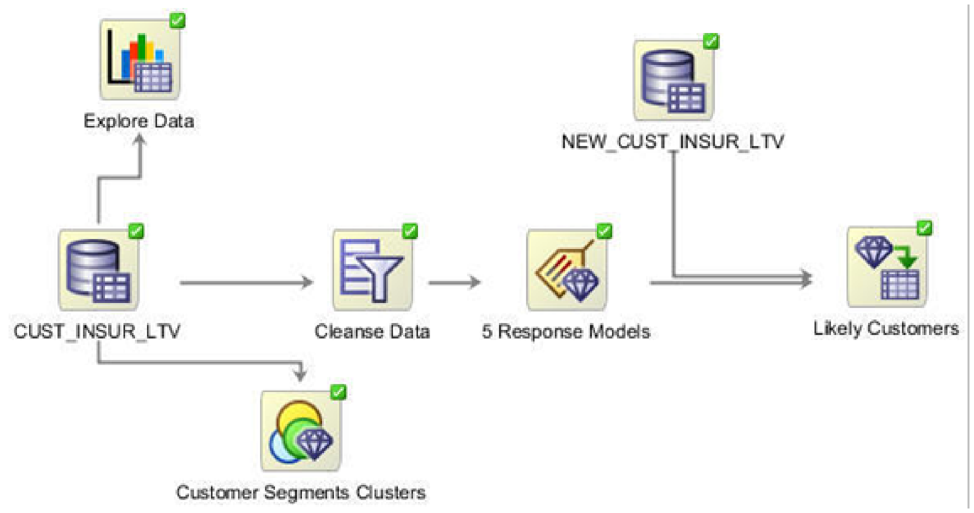
See Also: "In-Database Scoring" on page 3-6 for more information about scoring, including an example of dynamic scoring

Oracle Data Miner

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

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

Figure 2–1 An Oracle Data Miner Workflow



For information about Oracle Data Miner, including installation instructions, visit the following page on the Oracle Technology Network:

http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datminGUI

Predictive Analytics

Predictive Analytics is a technology that captures data mining processes in simple routines. Sometimes called "one-click data mining," predictive analytics simplifies and automates the data mining process.

Predictive analytics uses data mining technology, but knowledge of data mining is not needed to use predictive analytics. You can use predictive analytics simply by specifying an operation to perform on your data. You do not need to create or use mining models or understand the mining functions and algorithms summarized in Chapter 3.

Oracle Data Mining predictive analytics operations are described in Table 2–1.

Table 2–1 Oracle Predictive Analytics Operations

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

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

Overview of Database Analytics

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

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

Example 2–3 SQL Query Combining Oracle Data Mining and Oracle Text

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

Some of the native analytics supported by Oracle Database are described in Table 2–2.

Table 2–2 Oracle Database Native Analytics

Analytical Feature	Description	Documented In...
Complex data transformations	Data transformation is a key aspect of analytical applications and ETL (extract, transform, and load). You can use SQL expressions to implement data transformations, or you can use the `DBMS_DATA_MINING_TRANSFORM` package. `DBMS_DATA_MINING_TRANSFORM` is a flexible data transformation package that includes a variety of missing value and outlier treatments, as well as binning and normalization capabilities.	*Oracle Database PL/SQL Packages and Types Reference*
Statistical functions	Oracle Database provides a long list of SQL statistical functions with support for: hypothesis testing (such as t-test, F-test), correlation computation (such as pearson correlation), cross-tab statistics, and descriptive statistics (such as median and mode). The `DBMS_STAT_FUNCS` package adds distribution fitting procedures and a summary procedure that returns descriptive statistics for a column.	*Oracle Database SQL Language Reference* and *Oracle Database PL/SQL Packages and Types Reference*
Window and analytic SQL functions	Oracle Database supports analytic and windowing functions for computing cumulative, moving, and centered aggregates. With windowing aggregate functions, you can calculate moving and cumulative versions of `SUM`, `AVERAGE`, `COUNT`, `MAX`, `MIN`, and many more functions.	*Oracle Database Data Warehousing Guide*
Linear algebra	The `UTL_NLA` package exposes a subset of the popular `BLAS` and `LAPACK` (Version 3.0) libraries for operations on vectors and matrices represented as VARRAYs. This package includes procedures to solve systems of linear equations, invert matrices, and compute eigenvalues and eigenvectors.	*Oracle Database PL/SQL Packages and Types Reference*
OLAP	Oracle OLAP supports multidimensional analysis and can be used to improve performance of multidimensional queries. Oracle OLAP provides functionality previously found only in specialized OLAP databases. Moving beyond drill-downs and roll-ups, Oracle OLAP also supports time-series analysis, modeling, and forecasting.	*Oracle OLAP User's Guide*
Spatial analytics	Oracle Spatial provides advanced spatial features to support high-end GIS and LBS solutions. Oracle Spatial's analysis and mining capabilities include functions for binning, detection of regional patterns, spatial correlation, colocation mining, and spatial clustering. Oracle Spatial also includes support for topology and network data models and analytics. The topology data model of Oracle Spatial allows one to work with data about nodes, edges, and faces in a topology. It includes network analysis functions for computing shortest path, minimum cost spanning tree, nearest-neighbors analysis, traveling salesman problem, among others.	*Oracle Spatial and Graph Developer's Guide*
Text Mining	Oracle Text uses standard SQL to index, search, and analyze text and documents stored in the Oracle database, in files, and on the web. Oracle Text also supports automatic classification and clustering of document collections. Many of the analytical features of Oracle Text are layered on top of Oracle Data Mining functionality.	*Oracle Text Application Developer's Guide*

3

Oracle Data Mining Basics

This chapter introduces the basics you will need to start using Oracle Data Mining. This chapter contains these topics:

Mining Functions

A basic understanding of data mining functions and algorithms is required for using Oracle Data Mining. This section introduces the concept of data mining functions. Algorithms are introduced in "Algorithms" on page 3-3.

Each data mining **function** specifies a class of problems that can be modeled and solved. Data mining functions fall generally into two categories: **supervised** and **unsupervised**. Notions of supervised and unsupervised learning are derived from the science of machine learning, which has been called a sub-area of artificial intelligence.

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

Supervised Data Mining

Supervised learning is also known as directed learning. The learning process is directed by a previously known dependent attribute or target. Directed data mining attempts to explain the behavior of the target as a function of a set of independent attributes or predictors.

Supervised learning generally results in predictive models. This is in contrast to unsupervised learning where the goal is pattern detection.

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

Supervised Learning: Testing

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

The process of applying the model to test data helps to determine whether the model, built on one chosen sample, is generalizable to other data. In particular, it helps to avoid the phenomenon of overfitting, which can occur when the logic of the model fits the build data too well and therefore has little predictive power.

Supervised Learning: Scoring

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

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

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

Oracle Data Mining supports the supervised data mining functions described in Table 3–1.

Table 3–1 Oracle Data Mining Supervised Functions

Function	Description	Sample Problem
Attribute Importance	Identifies the attributes that are most important in predicting a target attribute	Given customer response to an affinity card program, find the most significant predictors
Classification	Assigns items to discrete classes and predicts the class to which an item belongs	Given demographic data about a set of customers, predict customer response to an affinity card program
Regression	Approximates and forecasts continuous values	Given demographic and purchasing data about a set of customers, predict customers' age

Unsupervised Data Mining

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

Unsupervised learning can be used for **descriptive** purposes. It can also be used to make predictions.

Unsupervised Learning: Scoring

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

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

Oracle Data Mining supports the unsupervised functions described in Table 3–2.

Table 3–2 Oracle Data Mining Unsupervised Functions

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

See Also:

- Part II for details about the mining functions supported by Oracle Data Mining
- "In-Database Scoring" on page 3-6 for more information about scoring

Algorithms

An algorithm is a mathematical procedure for solving a specific kind of problem. Oracle Data Mining supports at least one algorithm for each data mining function. For some functions, you can choose among several algorithms. For example, Oracle Data Mining supports four classification algorithms.

Each data mining model is produced by a specific algorithm. Some data mining problems can best be solved by using more than one algorithm. This necessitates the development of more than one model. For example, you might first use a feature extraction model to create an optimized set of predictors, then a classification model to make a prediction on the results.

Oracle Data Mining Supervised Algorithms

Oracle Data Mining supports the supervised data mining algorithms described in Table 3–3. The algorithm abbreviations are used throughout this manual.

Table 3–3 Oracle Data Mining Algorithms for Supervised Functions

Algorithm	Function	Description
Decision Tree	Classification	Decision trees extract predictive information in the form of human-understandable rules. The rules are if-then-else expressions; they explain the decisions that lead to the prediction.
Generalized Linear Models	Classification and Regression	Generalized Linear Models (GLM) implement logistic regression for classification of binary targets and linear regression for continuous targets. GLM classification supports confidence bounds for prediction probabilities. GLM regression supports confidence bounds for predictions.
Minimum Description Length	Attribute Importance	Minimum Description Length (MDL) is an information theoretic model selection principle. MDL assumes that the simplest, most compact representation of data is the best and most probable explanation of the data.
Naive Bayes	Classification	Naive Bayes makes predictions using Bayes' Theorem, which derives the probability of a prediction from the underlying evidence, as observed in the data.
Support Vector Machines	Classification and Regression	Distinct versions of Support Vector Machines (SVM) use different kernel functions to handle different types of data sets. Linear and Gaussian (nonlinear) kernels are supported. SVM classification attempts to separate the target classes with the widest possible margin. SVM regression tries to find a continuous function such that the maximum number of data points lie within an epsilon-wide tube around it.

Oracle Data Mining Unsupervised Algorithms

Oracle Data Mining supports the unsupervised data mining algorithms described in Table 3–4. The algorithm abbreviations are used throughout this manual.

Table 3–4 Oracle Data Mining Algorithms for Unsupervised Functions

Algorithm	Function	Description
Apriori	Association	Apriori performs market basket analysis by identifying co-occurring items (frequent itemsets) within a set. Apriori finds rules with support greater than a specified minimum support and confidence greater than a specified minimum confidence.
Expectation Maximization	Clustering	Expectation Maximization (EM) is a density estimation algorithm that performs probabilistic clustering. In density estimation, the goal is to construct a density function that captures how a given population is distributed. The density estimate is based on observed data that represents a sample of the population. Oracle Data Mining supports probabilistic clustering and data frequency estimates and other applications of Expectation Maximization.
k-Means	Clustering	*k*-Means is a distance-based clustering algorithm that partitions the data into a predetermined number of clusters. Each cluster has a centroid (center of gravity). Cases (individuals within the population) that are in a cluster are close to the centroid. Oracle Data Mining supports an enhanced version of *k*-Means. It goes beyond the classical implementation by defining a hierarchical parent-child relationship of clusters.
Non-Negative Matrix Factorization	Feature Extraction	Non-Negative Matrix Factorization (NMF) generates new attributes using linear combinations of the original attributes. The coefficients of the linear combinations are non-negative. During model apply, an NMF model maps the original data into the new set of attributes (features) discovered by the model.

Table 3–4 (Cont.) Oracle Data Mining Algorithms for Unsupervised Functions

Algorithm	Function	Description
One Class Support Vector Machines	Anomaly Detection	One-class SVM builds a profile of one class. When the model is applied, it identifies cases that are somehow different from that profile. This allows for the detection of rare cases that are not necessarily related to each other.
Orthogonal Partitioning Clustering	Clustering	Orthogonal Partitioning Clustering (o-cluster) creates a hierarchical, grid-based clustering model. The algorithm creates clusters that define dense areas in the attribute space. A sensitivity parameter defines the baseline density level.
Singular Value Decomposition and Principal Component Analysis	Feature Extraction	Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) are orthogonal linear transformations that are optimal at capturing the underlying variance of the data. This property is extremely useful for reducing the dimensionality of high-dimensional data and for supporting meaningful data visualization. In addition to dimensionality reduction, SVD and PCA have a number of other important applications, such as data de-noising (smoothing), data compression, matrix inversion, and solving a system of linear equations.

See Also: Part III for details about the algorithms supported by Oracle Data Mining

Data Preparation

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

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

Oracle Data Mining Simplifies Data Preparation

Oracle Data Mining offers several features that significantly simplify the process of data preparation:

- **Embedded data preparation** — The transformations used in training the model are embedded in the model and automatically executed whenever the model is applied to new data. If you specify transformations for the model, you only have to specify them once.
- **Automatic Data Preparation (ADP)** — Oracle Data Mining supports an automated data preparation mode. When ADP is active, Oracle Data Mining automatically performs the data transformations required by the algorithm. The transformation instructions are embedded in the model along with any user-specified transformation instructions.
- **Automatic management of missing values and sparse data** — Oracle Data Mining uses consistent methodology across mining algorithms to handle sparsity and missing values.
- **Transparency** — Oracle Data Mining provides model details, which are a view of the attributes that are internal to the model. This insight into the inner details of the model is possible because of reverse transformations, which map the transformed attribute values to a form that can be interpreted by a user. Where possible, attribute values are reversed to the original column values. Reverse transformations are also applied to the target of a supervised model, thus the results of scoring are in the same units as the units of the original target.

- **Tools for custom data preparation —** Oracle Data Mining provides many common transformation routines in the `DBMS_DATA_MINING_TRANSFORM` PL/SQL package. You can use these routines, or develop your own routines in SQL, or both. The SQL language is well suited for implementing transformations in the database. You can use custom transformation instructions along with ADP or instead of ADP.

Case Data

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

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

Nested Data

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

Text Data

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

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

See Also:

- Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*
- Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*
- Chapter 7, "Mining Unstructured Text" *Oracle Data Mining User's Guide*

In-Database Scoring

Scoring is the application of a data mining algorithm to new data. In traditional data mining, models are built using specialized software on a remote system and deployed to another system for scoring. This is a cumbersome, error-prone process open to security violations and difficulties in data synchronization.

With Oracle Data Mining, scoring is easy and secure. The scoring engine and the data both reside within the database. Scoring is an extension to the SQL language, so the results of mining can easily be incorporated into applications and reporting systems.

Parallel Execution and Ease of Administration

In-database scoring provides performance advantages. All Oracle Data Mining scoring routines support parallel execution, which significantly reduces the time required for executing complex queries and scoring large data sets. For information about parallel execution, refer to *Oracle Database VLDB and Partitioning Guide*.

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

SQL Functions for Model Apply and Dynamic Scoring

In Oracle Data Mining, scoring is performed by SQL language functions. The functions perform prediction, clustering, and feature extraction. The functions can be invoked in two different ways: By applying a mining model object (Example 3–1), or by executing an analytic clause that computes the mining analysis dynamically and applies it to the data (Example 3–2). Dynamic scoring, which eliminates the need for a model, can supplement, or even replace, the more traditional data mining methodology described in "The Data Mining Process" on page 1-4.

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

Example 3–1 Applying a Mining Model to Score Data

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

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

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

Example 3–2 Executing an Analytic Function to Score Data

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

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

See Also:

- "Data Mining Functions" in *Oracle Database SQL Language Reference*
- Chapter 6, "Scoring and Deployment" in *Oracle Data Mining User's Guide*

Part II

Mining Functions

Part II provides basic conceptual information about the mining functions supported by Oracle Data Mining. Mining functions represent a class of mining problems that can be solved using data mining algorithms. Oracle Data Mining algorithms are described in Part III.

Part II contains these chapters:

Note on Terminology: The term *mining function* has no relationship to a *SQL language function*.

See *Oracle Database SQL Language Reference* for information about the Data Mining SQL language functions.

4 Regression

This chapter describes regression, the supervised mining function for predicting a continuous, numerical target. For an overview of supervised data mining, see Chapter 3.

This chapter contains these topics:

- About Regression
- Testing a Regression Model
- Regression Algorithms

About Regression

Regression is a data mining function that predicts numeric values along a continuum. Profit, sales, mortgage rates, house values, square footage, temperature, or distance could all be predicted using regression techniques. For example, a regression model could be used to predict the value of a house based on location, number of rooms, lot size, and other factors.

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

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

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

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

How Does Regression Work?

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

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

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

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

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

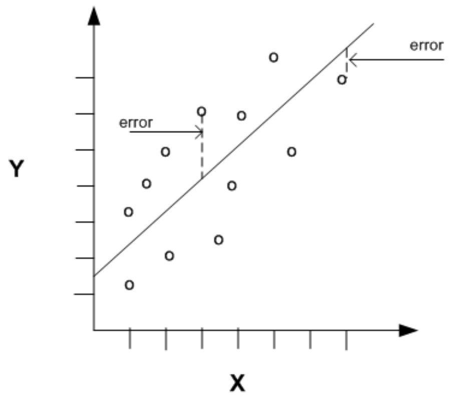
There are different families of regression functions and different ways of measuring the error.

Linear Regression

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

Regression with a single predictor is the easiest to visualize. Simple linear regression with a single predictor is shown in Figure 4–1.

Figure 4–1 Linear Regression With a Single Predictor



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

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

The regression parameters in simple linear regression are:

- The **slope** of the line (θ_2) — the angle between a data point and the regression line
- The ***y* intercept** (θ_1) — the point where **x** crosses the y axis (**x** = 0)

Multivariate Linear Regression

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

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

Regression Coefficients

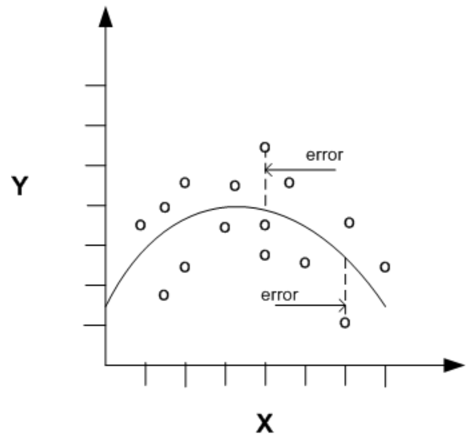
In multivariate linear regression, the regression parameters are often referred to as coefficients. When you build a multivariate linear regression model, the algorithm computes a coefficient for each of the predictors used by the model. The coefficient is a measure of the impact of the predictor **x**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

Nonlinear Regression

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

Nonlinear regression models define ***y*** as a function of **x** using an equation that is more complicated than the linear regression equation. In Figure 4–2, **x** and ***y*** have a nonlinear relationship.

Figure 4–2 Nonlinear Regression With a SIngle Predictor



Multivariate Nonlinear Regression

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

Confidence Bounds

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

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

Testing a Regression Model

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

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

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

Regression Statistics

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

Root Mean Squared Error

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

This SQL expression calculates the RMSE.

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

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

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

Mean Absolute Error

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

This SQL expression calculates the MAE.

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

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

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

Regression Algorithms

Oracle Data Mining supports two algorithms for regression. Both algorithms are particularly suited for mining data sets that have very high dimensionality (many attributes), including transactional and unstructured data.

- **Generalized Linear Models (GLM)**

  GLM is a popular statistical technique for linear modeling. Oracle Data Mining implements GLM for regression and for binary classification. GLM provides extensive coefficient statistics and model statistics, as well as row diagnostics. GLM also supports confidence bounds. See Chapter 13, "Generalized Linear Models".

- **Support Vector Machines (SVM)**

  SVM is a powerful, state-of-the-art algorithm for linear and nonlinear regression. Oracle Data Mining implements SVM for regression, classification, and anomaly detection. SVM regression supports two kernels: the Gaussian kernel for nonlinear regression, and the linear kernel for linear regression. SVM also supports active learning. See Chapter 20, "Support Vector Machines".

Note: Oracle Data Mining uses linear kernel SVM as the default regression algorithm.

5 Classification

This chapter describes classification, the supervised mining function for predicting a categorical target. For an overview of supervised data mining, see Chapter 3.

This chapter contains these topics:

About Classification

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

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

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

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

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

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

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

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

Testing a Classification Model

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

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

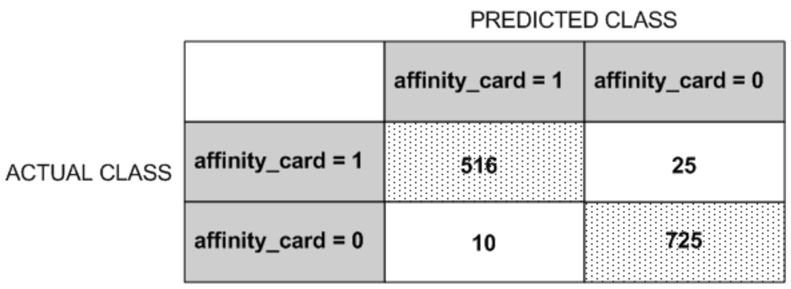
Test metrics are used to assess how accurately the model predicts the known values. If the model performs well and meets the business requirements, it can then be applied to new data to predict the future.

Confusion Matrix

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

Figure 5–1 shows a confusion matrix for a binary classification model. The rows present the number of actual classifications in the test data. The columns present the number of predicted classifications made by the model.

Figure 5–1 Confusion Matrix for a Binary Classification Model



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

In this example, the model correctly predicted the positive class for `affinity_card` 516 times and incorrectly predicted it 25 times. The model correctly predicted the negative class for `affinity_card` 725 times and incorrectly predicted it 10 times. The following can be computed from this confusion matrix:

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

Lift

Lift measures the degree to which the predictions of a classification model are better than randomly-generated predictions. Lift applies to binary classification only, and it requires the designation of a positive class. (See "Positive and Negative Classes" on page 5-5.) If the model itself does not have a binary target, you can compute lift by designating one class as positive and combining all the other classes together as one negative class.

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

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

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

Lift Statistics

Oracle Data Mining computes the following lift statistics:

- **Probability threshold** for a quantile *n* is the minimum probability for the positive target to be included in this quantile or any preceding quantiles (quantiles *n*-1, *n*-2,..., 1). If a cost matrix is used, a cost threshold is reported instead. The cost threshold is the maximum cost for the positive target to be included in this quantile or any of the preceding quantiles. (See "Costs" on page 5-5.)
- **Cumulative gain** is the ratio of the cumulative number of positive targets to the total number of positive targets.
- **Target density** of a quantile is the number of true positive instances in that quantile divided by the total number of instances in the quantile.
- **Cumulative target density** for quantile *n* is the target density computed over the first *n* quantiles.
- **Quantile lift** is the ratio of target density for the quantile to the target density over all the test data.
- **Cumulative percentage of records** for a quantile is the percentage of all cases represented by the first *n* quantiles, starting at the end that is most confidently positive, up to and including the given quantile.
- **Cumulative number of targets** for quantile *n* is the number of true positive instances in the first *n* quantiles.
- **Cumulative number of nontargets** is the number of actually negative instances in the first *n* quantiles.
- **Cumulative lift** for a quantile is the ratio of the cumulative target density to the target density over all the test data.

Receiver Operating Characteristic (ROC)

ROC is another metric for comparing predicted and actual target values in a classification model. ROC, like lift, applies to binary classification and requires the designation of a positive class. (See "Positive and Negative Classes" on page 5-5.)

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

ROC measures the impact of changes in the **probability threshold**. The probability threshold is the decision point used by the model for classification. The default probability threshold for binary classification is .5. When the probability of a prediction is 50% or more, the model predicts that class. When the probability is less than 50%, the other class is predicted. (In multiclass classification, the predicted class is the one predicted with the highest probability.)

The ROC Curve

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

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

Area Under the Curve

The area under the ROC curve (AUC) measures the discriminating ability of a binary classification model. The larger the AUC, the higher the likelihood that an actual positive case 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).

ROC and Model Bias

Changes in the probability threshold affect the predictions made by the model. For instance, if the threshold for predicting the positive class is changed from .5 to.6, fewer positive predictions will be made. This will affect the distribution of values in the confusion matrix: the number of true and false positives and true and false negatives will all be different.

The ROC curve for a model represents all the possible combinations of values in its confusion matrix. You can use ROC to find the probability thresholds that yield the highest overall accuracy or the highest per-class accuracy. For example, if it is important to you to accurately predict the positive class, but you don't care about prediction errors for the negative class, you could lower the threshold for the positive class. This would bias the model in favor of the positive class.

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

See Also: "Costs" on page 5-5

ROC Statistics

Oracle Data Mining computes the following ROC statistics:

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

- **True negatives:** Negative cases in the test data with predicted probabilities strictly less than the probability threshold (correctly predicted).
- **True positives:** Positive cases in the test data with predicted probabilities greater than or equal to the probability threshold (correctly predicted).
- **False negatives:** Positive cases in the test data with predicted probabilities strictly less than the probability threshold (incorrectly predicted).
- **False positives:** Negative cases in the test data with predicted probabilities greater than or equal to the probability threshold (incorrectly predicted).
- **True positive fraction**: Hit rate. (true positives/(true positives + false negatives))
- **False positive fraction:** False alarm rate. (false positives**/**(false positives + true negatives))

Biasing a Classification Model

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

Costs

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

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

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

Costs Versus Accuracy

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

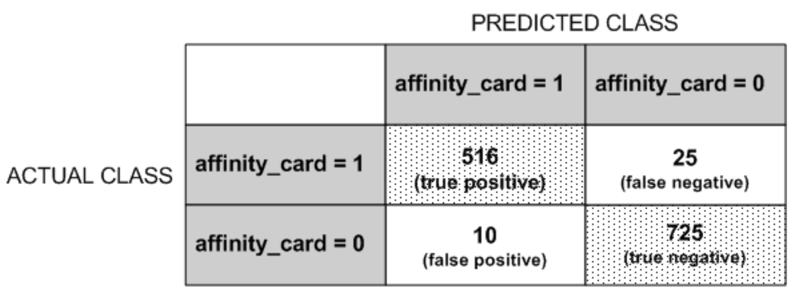
A confusion matrix is used to measure accuracy, the ratio of correct predictions to the total number of predictions. A cost matrix is used to specify the relative importance of accuracy for different predictions. In most business applications, it is important to consider costs in addition to accuracy when evaluating model quality. (See "Confusion Matrix" on page 5-2.)

Positive and Negative Classes

The positive class is the class that you care the most about. Designation of a positive class is required for computing lift and ROC. (See "Lift" on page 5-3 and "Receiver Operating Characteristic (ROC)" on page 5-4).

In the confusion matrix in Figure 5–2, the value `1` is designated as the positive class. This means that the creator of the model has determined that it is more important to accurately predict customers who will increase spending with an affinity card (`affinity_card`=1) than to accurately predict non-responders (`affinity_card`=0). If you give affinity cards to some customers who are not likely to use them, there is little loss to the company since the cost of the cards is low. However, if you overlook the customers who are likely to respond, you miss the opportunity to increase your revenue.

Figure 5–2 Positive and Negative Predictions



		PREDICTED CLASS	
		affinity_card = 1	affinity_card = 0
ACTUAL CLASS	affinity_card = 1	516 (true positive)	25 (false negative)
	affinity_card = 0	10 (false positive)	725 (true negative)

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

- False positive rate — 10/(10 + 725) =.01
- True positive rate — 516/(516 + 25) =.95

Assigning Costs and Benefits

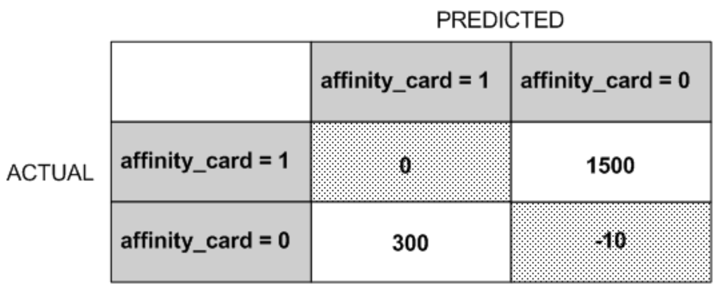
In a cost matrix, positive numbers (costs) can be used to influence negative outcomes. Since negative costs are interpreted as benefits, negative numbers (benefits) can be used to influence positive outcomes.

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

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

Figure 5–3 shows how you would represent these costs and benefits in a cost matrix.

Figure 5–3 Cost Matrix



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

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

Priors and Class Weights

With Bayesian models, you can specify prior probabilities to offset differences in distribution between the build data and the real population (scoring data). With other forms of classification, you may be able to specify class weights, which have the same biasing effect as priors.

In many problems, one target value dominates in frequency. For example, the positive responses for a telephone marketing campaign 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 of the rare class 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 frequent 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 the quality of a classification model.

To correct for unrealistic distributions in the training data, you can specify priors for the model build process. Other approaches to compensating for data distribution issues include stratified sampling and anomaly detection. (See Chapter 6.)

Classification Algorithms

Oracle Data Mining provides the following algorithms for classification:

- **Decision Tree**

  Decision trees automatically generate rules, which are conditional statements that reveal the logic used to build the tree. See Chapter 11, "Decision Tree".

- **Naive Bayes**

  Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data. See Chapter 16, "Naive Bayes".

- **Generalized Linear Models (GLM)**

  GLM is a popular statistical technique for linear modeling. Oracle Data Mining implements GLM for binary classification and for regression. GLM provides extensive coefficient statistics and model statistics, as well as row diagnostics. GLM also supports confidence bounds. See Chapter 13, "Generalized Linear Models".

- **Support Vector Machines (SVM)**

  SVM is a powerful, state-of-the-art algorithm based on linear and nonlinear regression. Oracle Data Mining implements SVM for binary and multiclass classification. See Chapter 20, "Support Vector Machines".

Note: Oracle Data Mining uses Naive Bayes as the default classification algorithm.

6 Anomaly Detection

This chapter describes anomaly detection, an unsupervised mining function for detecting rare cases in the data. For an overview of unsupervised data mining, see Chapter 3.

This chapter contains these topics:

- About Anomaly Detection
- Anomaly Detection Algorithm

Reference:

Campos, M.M., Milenova, B.L., Yarmus, J.S., "Creation and Deployment of Data Mining-Based Intrusion Detection Systems in Oracle Database 10*g*"

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datmin`

About Anomaly Detection

The goal of anomaly detection is to identify cases that are unusual within data that is seemingly homogeneous. Anomaly detection is an important tool for detecting fraud, network intrusion, and other rare events that may have great significance but are hard to find.

Anomaly detection can be used to solve problems like the following:

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

  The law enforcement data is all of one class. There are no counter-examples.

- An insurance agency processes millions of insurance claims, knowing that a very small number are fraudulent. How can the fraudulent claims be identified?

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

One-Class Classification

Anomaly detection is a form of classification. See "About Classification" on page 5-1 for an overview of the classification mining function.

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

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

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

Note: Solving a one-class classification problem can be difficult. The accuracy of one-class classifiers cannot usually match the accuracy of standard classifiers built with meaningful counterexamples.

The goal of anomaly detection is to provide some useful information where no information was previously attainable. However, if there are enough of the "rare" cases so that stratified sampling could produce a training set with enough counterexamples for a standard classification model, then that would generally be a better solution.

Anomaly Detection for Single-Class Data

In single-class data, all the cases have the same classification. Counter-examples, instances of another class, may be hard to specify or expensive to collect. For instance, in text document classification, it may be easy to classify a document under a given topic. However, the universe of documents outside of this topic may be very large and diverse. Thus it may not be feasible to specify other types of documents as counter-examples.

Anomaly detection could be used to find unusual instances of a particular type of document.

Anomaly Detection for Finding Outliers

Outliers are cases that are unusual because they fall outside the distribution that is considered normal for the data. For example, census data might show a median household income of $70,000 and a mean household income of $80,000, but one or two households might have an income of $200,000. These cases would probably be identified as outliers.

The distance from the center of a normal distribution indicates how typical a given point is with respect to the distribution of the data. Each case can be ranked according to the probability that it is either typical or atypical.

The presence of outliers can have a deleterious effect on many forms of data mining. Anomaly detection can be used to identify outliers before mining the data.

Anomaly Detection Algorithm

Oracle Data Mining supports One-Class Support Vector Machines (SVM) for anomaly detection. When used for anomaly detection, SVM classification does not use a target.

See Also: "One-Class SVM" on page 20-5

7 Clustering

This chapter describes clustering, the unsupervised mining function for discovering natural groupings in the data. For an overview of unsupervised data mining, see Chapter 3.

This chapter contains these topics:

- About Clustering
- Evaluating a Clustering Model
- Clustering Algorithms

About Clustering

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

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

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

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

Clustering can also be used for anomaly detection. Once the data has been segmented into clusters, you might find that some cases do not fit well into any clusters. These cases are anomalies or outliers.

How are Clusters Computed?

There are several different approaches to the computation of clusters. Oracle Data Mining supports the following methods:

- **Density-based** — This type of clustering finds the underlying distribution of the data and estimates how areas of high density in the data correspond to peaks in the distribution. High-density areas are interpreted as clusters. Density-based cluster estimation is probabilistic.

- **Distance-based** — This type of clustering uses a distance metric to determine similarity between data objects. The distance metric measures the distance between actual cases in the cluster and the prototypical case for the cluster. The prototypical case is known as the **centroid**.
- **Grid-based** — This type of clustering divides the input space into hyper-rectangular cells and identifies adjacent high-density cells to form clusters.

Scoring New Data

Although clustering is an unsupervised mining function, Oracle Data Mining supports the scoring operation for clustering. New data is scored probabilistically.

Hierarchical Clustering

The clustering algorithms supported by Oracle Data Mining perform hierarchical clustering. The leaf clusters are the final clusters generated by the algorithm. Clusters higher up in the hierarchy are intermediate clusters.

Rules

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

Support and Confidence

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

Evaluating a Clustering Model

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

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

Clustering Algorithms

Oracle Data Mining supports these clustering algorithms:

- **Expectation Maximization**

  Expectation Maximization is a probabilistic, density-estimation clustering algorithm. See Chapter 12, "Expectation Maximization".
- ***k*-Means**

  k-Means is a distance-based clustering algorithm. Oracle Data Mining supports an enhanced version of *k*-Means. See Chapter 14, "k-Means".
- **Orthogonal Partitioning Clustering (O-Cluster)**

O-Cluster is a proprietary, grid-based clustering algorithm. See Chapter 18, "O-Cluster".

Reference:

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

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datmin`

The main characteristics of the two algorithms are compared in Table 7–1.

Table 7–1 Clustering Algorithms Compared

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

Note: Oracle Data Mining uses *k*-Means as the default clustering algorithm.

8 Association

This chapter describes association, the unsupervised mining function for discovering association rules. For an overview of unsupervised data mining, see Chapter 3.

This chapter contains these topics:

About Association

Association is a data mining function that discovers the probability of the co-occurrence of items in a collection. The relationships between co-occurring items are expressed as **association rules**.

Association Rules

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

Association rules are ranked by these metrics:

Support — How often do these items occur together in the data?
Confidence — How likely are these items to occur together in the data?

Market-Basket Analysis

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

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

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

Association Rules and eCommerce

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

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

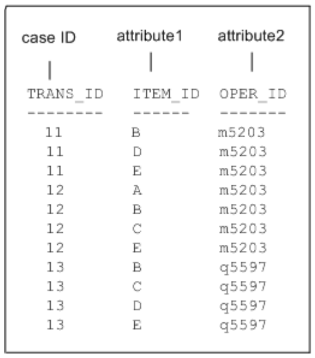
See Also: "Confidence" on page 10-6

Transactional Data

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

Transactional data, also known as **market-basket data**, is said to be in **multi-record case** format because a set of records (rows) constitute a case. For example, in Figure 8–1, case 11 is made up of three rows while cases 12 and 13 are each made up of four rows.

Figure 8–1 Transactional Data


case ID
attribute1
attribute2

```
TRANS_ID  ITEM_ID  OPER_ID
--------  ------  -------
  11       B       m5203
  11       D       m5203
  11       E       m5203
  12       A       m5203
  12       B       m5203
  12       C       m5203
  12       E       m5203
  13       B       q5597
  13       C       q5597
  13       D       q5597
  13       E       q5597
```

Non transactional data is said to be in **single-record case** format because a single record (row) constitutes a case. In Oracle Data Mining, association models can be built using either transactional or non transactional data. If the data is non transactional, it must be transformed to a nested column before association mining activities can be performed.

See Also:

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

"Data Preparation for Apriori" on page 10-2

Association Algorithm

Oracle Data Mining uses the Apriori algorithm to calculate association rules for items in frequent itemsets.

See Also: Chapter 10, "Apriori"

9 Feature Selection and Extraction

This chapter describes feature selection, attribute importance, and feature extraction. Oracle Data Mining supports attribute importance as a supervised mining function and feature extraction as an unsupervised mining function. For an overview of supervised and unsupervised data mining, see Chapter 3.

This chapter contains these topics:

- Finding the Best Attributes
- About Feature Selection and Attribute Importance
- About Feature Extraction

Finding the Best Attributes

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

For example, you might collect a great deal of data about a given population because you want to predict the likelihood of a certain illness within this group. Some of this information, perhaps much of it, will have little or no effect on susceptibility to the illness. Attributes such as the number of cars per household may have no effect whatsoever.

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

Data sets with many attributes may contain groups of attributes that are correlated. These attributes may actually be measuring the same underlying feature. Their presence together in the build data can skew the logic of the algorithm and affect the accuracy of the model.

Wide data (many attributes) generally presents processing challenges for data mining algorithms. Model attributes are the dimensions of the processing space used by the algorithm. The higher the dimensionality of the processing space, the higher the computation cost involved in algorithmic processing.

To minimize the effects of noise, correlation, and high dimensionality, some form of dimension reduction is sometimes a desirable preprocessing step for data mining. Feature selection and extraction are two approaches to dimension reduction.

- **Feature selection** — Selecting the most relevant attributes
- **Feature extraction** — Combining attributes into a new reduced set of features

About Feature Selection and Attribute Importance

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

Oracle Data Mining supports the **attribute importance** mining function, which ranks attributes according to their importance in predicting a target. Attribute importance does not actually perform feature selection since all the predictors are retained in the model. In true feature selection, the attributes that are ranked below a given threshold of importance are removed from the model.

Feature selection is useful as a preprocessing step to improve computational efficiency in predictive modeling. Oracle Data Mining implements feature selection for optimization within the Decision Tree algorithm and within Naive Bayes when ADP is enabled. GLM models can be configured to perform feature selection as a preprocessing step.

Attribute Importance and Scoring

Oracle Data Mining does not support the scoring operation for attribute importance. The results of attribute importance are the attributes of the build data ranked according to their predictive influence. The ranking and the measure of importance can be used in selecting training data for classification models.

About Feature Extraction

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

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

Models built on extracted features may be of higher quality, because the data is described by fewer, more meaningful attributes.

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

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

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.

Feature Extraction and Scoring

Oracle Data Mining supports the scoring operation for feature extraction. As an unsupervised mining function, feature extraction does not involve a target. When applied, a feature extraction model transforms the input into a set of features.

Algorithms for Attribute Importance and Feature Extraction

Oracle Data Mining supports the **Minimum Description Length** algorithm for attribute importance. See Chapter 15, "Minimum Description Length".

Oracle Data Mining supports these feature extraction algorithms:

- **Non-Negative Matrix Factorization** (NMF). See Chapter 17, "Non-Negative Matrix Factorization".
- **Singular Value Decomposition** (SVD) and **Prediction Component Analysis** (PCA). See Chapter 19, "Singular Value Decomposition".

Note: Oracle Data Mining uses NMF as the default feature extraction algorithm.

Part III

Algorithms

Part III provides basic conceptual information about the algorithms supported by Oracle Data Mining. There is at least one algorithm for each of the mining functions described in Part II.

Part III contains these chapters:

10 Apriori

This chapter describes Apriori, the algorithm used by Oracle Data Mining for calculating association rules. See Chapter 8 for information about association rules.

This chapter contains these topics:

About Apriori

An association mining problem can be decomposed into two subproblems:

- Find all combinations of items in a set of transactions that occur with a specified minimum frequency. These combinations are called **frequent itemsets**.
- Calculate rules that express the probable co-occurrence of items within frequent itemsets. (See "Example: Calculating Rules from Frequent Itemsets" on page 10-4.)

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

Association rule mining is not recommended for finding associations involving rare events in problem domains with a large number of items. Apriori 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 itemsets, especially in cases with a large number of items. This could increase the execution time significantly. Classification or anomaly detection may be more suitable for discovering rare events when the data has a high number of attributes.

The build process for Apriori supports parallel execution. For information about parallel execution, refer to *Oracle Database VLDB and Partitioning Guide*.

Association Rules and Frequent Itemsets

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

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

Antecedent and Consequent

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

Oracle Data Mining supports association rules that have one or more items in the antecedent and a single item in the consequent.

Confidence

Rules have an associated confidence, which is the conditional probability that the consequent will occur given the occurrence of the antecedent. The minimum confidence for rules can be specified by the user.

Data Preparation for Apriori

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

Note: Unstructured text should not be used in association models.

Native Transactional Data and Star Schemas

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

See Also:

"Transactional Data" on page 8-2

"Using Nested Data" in *Oracle Data Mining User's Guide*

Items and Collections

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

Sparse Data

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

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

Oracle Data Mining assumes sparsity in transactional data. The Apriori algorithm is optimized for processing sparse data.

See Also: "Handling Missing Values" in *Oracle Data Mining User's Guide*

Note: Apriori is not affected by Automatic Data Preparation.

Calculating Association Rules

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

Itemsets

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

Example 10–1 Sample Transactional Data

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

Table 10–1 shows the itemsets derived from the transactions shown in Example 10–1, assuming that maximum number of items in an itemset is set to 3.

Table 10–1 Itemsets

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

Frequent Itemsets

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

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

Table 10–2 shows the itemsets from Table 10–1 that are frequent itemsets with support > 66%.

Table 10–2 Frequent Itemsets

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

See Also: Chapter 10, "Apriori" for information about the calculation of association rules

Example: Calculating Rules from Frequent Itemsets

Table 10–3 and Table 10–4 show the itemsets and frequent itemsets that were calculated in Chapter 8. The frequent itemsets are the itemsets that occur with a minimum support of 67%; at least 2 of the 3 transactions must include the itemset.

Table 10–3 Itemsets

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

Table 10–4 Frequent Itemsets with Minimum Support 67%

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

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

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

As shown in Table 10–3:

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

Table 10–5, shows the rules that could be derived from the frequent itemsets in Table 10–4.

Table 10–5 Frequent Itemsets and Rules

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

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

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

Evaluating Association Rules

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

Support

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

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

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

Support can be expressed in probability notation as follows.

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

See Also: "Frequent Itemsets" on page 10-3

Confidence

The confidence of a rule indicates the probability of both the antecedent and the consequent appearing in the same transaction. Confidence is the conditional probability of the consequent given the antecedent. For example, cereal might appear in 50 transactions; 40 of the 50 might also include milk. The rule confidence would be:

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

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

Confidence can be expressed in probability notation as follows.

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

See Also: "Confidence" on page 10-2 and "Example: Calculating Rules from Frequent Itemsets" on page 10-4

Lift

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

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

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

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

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

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

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

which is 0.83 – an improvement of less than 1.

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

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

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

11 Decision Tree

This chapter describes Decision Tree, one of the classification algorithms supported by Oracle Data Mining. See Chapter 5 for information about classification.

This chapter contains these topics:

About Decision Tree

The Decision Tree algorithm, like Naive Bayes, is based on conditional probabilities. Unlike Naive Bayes, decision trees generate **rules**. A rule is a conditional statement that can be understood by humans and used within a database to identify a set of records.

In some applications of data mining, the reason for predicting one outcome or another may not be important in evaluating the overall quality of a model. In others, the ability to explain the reason for a decision can be crucial. For example, a Marketing professional would need complete descriptions of customer segments in order to launch a successful marketing campaign. The Decision Tree algorithm is ideal for this type of application.

Decision tree rules can be used to validate models. If the rules make sense to a subject matter expert, this validates the model.

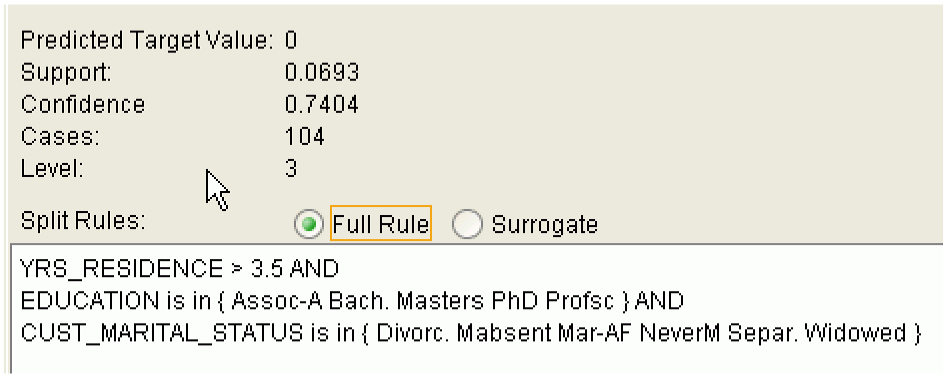
Decision Tree Rules

Oracle Data Mining supports several algorithms that provide rules. In addition to decision trees, clustering algorithms (described in Chapter 7) provide rules that describe the conditions shared by the members of a cluster, and association rules (described in Chapter 8) provide rules that describe associations between attributes.

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

Figure 11–1 shows a rule generated by a Decision Tree model. This rule comes from a decision tree that predicts the probability that customers will increase spending if given a loyalty card. A target value of 0 means not likely to increase spending; 1 means likely to increase spending.

Figure 11–1 Sample Decision Tree Rule



```
Predicted Target Value: 0
Support:                0.0693
Confidence              0.7404
Cases:                  104
Level:                  3

Split Rules:            (●) Full Rule  ( ) Surrogate
YRS_RESIDENCE > 3.5 AND
EDUCATION is in { Assoc-A Bach. Masters PhD Profsc } AND
CUST_MARITAL_STATUS is in { Divorc. Mabsent Mar-AF NeverM Separ. Widowed }
```

The rule shown in Figure 11–1 represents the conditional statement:

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

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

Confidence and Support

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

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

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

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

Advantages of Decision Trees

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

The algorithm is fast, both at build time and apply time. The build process for Decision Tree supports parallel execution. (Scoring supports parallel execution irrespective of the algorithm.) For information about parallel execution, refer to *Oracle Database VLDB and Partitioning Guide*.

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

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 XML specification is available at `http://www.dmg.org`.

Growing a Decision Tree

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

Figure 11–2 Sample Decision Tree

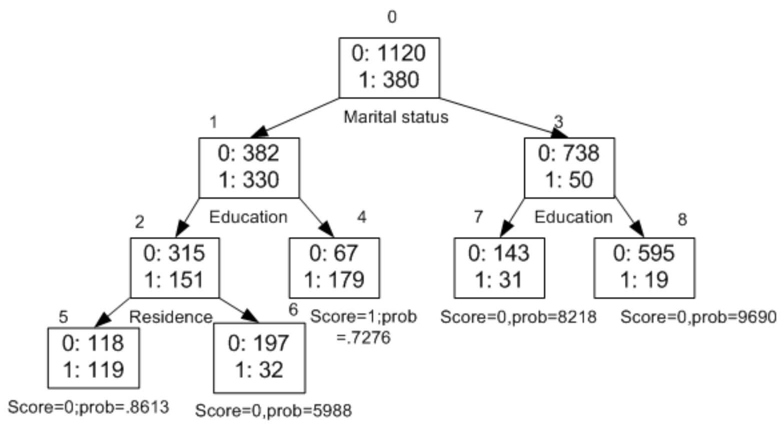


Figure 11–2 is a decision tree with nine nodes (and nine corresponding rules). The target attribute is binary: 1 if the customer will increase spending, 0 if the customer will not increase spending. The first split in the tree is based on the `CUST_MARITAL_STATUS` attribute. The root of the tree (node 0) is split into nodes 1 and 3. Married customers are in node 1; single customers are in node 3.

The rule associated with node 1 is:

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

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

Splitting

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

Homogeneity metrics asses the quality of alternative split conditions and select the one that results in the most homogeneous child nodes. Homogeneity is also called **purity**;

it refers to the degree to which the resulting child nodes are made up of cases with the same target value. The objective is to maximize the purity in the child nodes. For example, if the target can be either yes or no (will or will not increase spending), the objective is to produce nodes where most of the cases will increase spending or most of the cases will not increase spending.

Cost Matrix

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

See "Costs" on page 5-5 and "Priors and Class Weights" on page 5-7.

Preventing Over-Fitting

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

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

Tuning the Decision Tree Algorithm

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

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

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

See Also: "Algorithm Settings: Decision Tree" in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for Decision Tree

The Decision Tree algorithm manages its own data preparation internally. It does not require pretreatment of the data. Decision Tree is not affected by Automatic Data Preparation.

See Also:

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

12 Expectation Maximization

This chapter describes Expectation Maximization clustering as it is implemented by Oracle Data Mining. See Chapter 7 for information about clustering.

This chapter contains these topics:

About Expectation Maximization

Expectation Maximization (EM) estimation of mixture models is a popular probability density estimation technique that is used in a variety of applications. Oracle Data Mining uses EM to implement a distribution-based clustering algorithm (EM-clustering).

Expectation Step and Maximization Step

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

Probability Density Estimation

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

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

Algorithm Enhancements

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

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

Limitations of Standard Expectation Maximization:

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

Scalability

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

For information about parallel execution, refer to *Oracle Database VLDB and Partitioning Guide*.

High Dimensionality

The Oracle Data Mining implementation of Expectation Maximization can efficiently process high-dimensional data with thousands of attributes. This is achieved through a two-fold process:

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

examples of high-dimensional, numerical data are: text, recommendations, gene expressions, and market basket data.

Number of Components

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

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

Parameter Initialization

Choosing appropriate initial parameter values can have a significant effect on the quality of the solution. EM is not guaranteed to converge to the global maximum of the likelihood function but may instead converge to a local maximum. Therefore different initial parameter values can lead to different model parameters and different model quality.

In Oracle Data Mining, the EM model is grown incrementally. As new components are added, their parameters are initialized to areas with poor distribution fit.

From Components to Clusters

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

Ideally, high density areas of arbitrary shape should be interpreted as single clusters. To accomplish this, the Oracle Data Mining implementation of EM builds a component hierarchy that is based on the overlap of the individual components' distributions. Oracle Data Mining EM uses agglomerative hierarchical clustering. Component distribution overlap is measured using the Bhattacharyya distance function. The number of high-level clusters is determined automatically by choosing an appropriate cutoff level in the hierarchy.

The Oracle Data Mining implementation of EM produces an assignment of the model components to high-level clusters. The high-level clusters are additionally described by statistics like means, variances, modes, histograms, rules. The algorithm can be configured to either produce clustering assignments at the component level or at the cluster level.

Configuring the Algorithm

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

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

- Whether or not independent non-nested column attributes are included in the model. The choice is system-determined by default.
- Whether to use Bernoulli or Gaussian distribution for numerical attributes. By default, the algorithm chooses the most appropriate distribution, and individual attributes may use different distributions. When the distribution is user-specified, it is used for all numerical attributes.
- Whether the convergence criterion is based on a held-aside data set or on Bayesian Information Criterion (BIC). The convergence criterion is system-determined by default.
- The percentage improvement in the value of the log likelihood function that is required to add a new component to the model. The default percentage is 0.001.
- Whether to define clusters as individual components or groups of components. Clusters are associated to groups of components by default.
- The maximum number of components in the model. By default, the algorithm determines the number of components based on improvements in the likelihood function or based on regularization (BIC), up to the specified maximum.
- Whether the linkage function for the agglomerative clustering step uses the nearest distance within the branch (single linkage), the average distance within the branch (average linkage), or the maximum distance within the branch (complete linkage). By default the algorithm uses single linkage.

See Also: "Global Settings" and "Algorithm Settings: Expectation Maximization" in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for Expectation Maximization

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

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

See Also: Chapters 3 and 4 in *Oracle Data Mining User's Guide* for information about data transformations, missing value treatment, and Automatic Data Preparation

13 Generalized Linear Models

This chapter describes Generalized Linear Models (GLM), a statistical technique for linear modeling. Oracle Data Mining supports GLM for regression and binary classification. See Chapter 4 and Chapter 5 for information about regression and classification.

This chapter contains these topics:

About Generalized Linear Models

Generalized Linear Models (GLM) include and extend the class of linear models described in "Linear Regression" on page 4-2.

Linear models make a set of restrictive assumptions, most importantly, that the target (dependent variable *y*) is normally distributed conditioned on the value of predictors with a constant variance regardless of the predicted response value. The advantage of linear models and their restrictions include computational simplicity, an interpretable model form, and the ability to compute certain diagnostic information about the quality of the fit.

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

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

Oracle Data Mining includes two of the most popular members of the GLM family of models with their most popular link and variance functions:

- **Linear regression** with the identity link and variance function equal to the constant 1 (constant variance over the range of response values). See "Linear Regression" on page 13-7.
- **Logistic regression** with the logit link and binomial variance functions. See "Logistic Regression" on page 13-9.

GLM in Oracle Data Mining

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

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

Interpretability and Transparency

Oracle Data Mining GLM models are easy to interpret. Each model build generates many statistics and diagnostics. Transparency is also a key feature: model details describe key characteristics of the coefficients, and global details provide high-level statistics. (See "Tuning and Diagnostics for GLM".)

Wide Data

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

Confidence Bounds

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

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

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

Ridge Regression

The best regression models are those in which the predictors correlate highly with the target, but there is very little correlation between the predictors themselves.

Multicollinearity is the term used to describe multivariate regression with correlated predictors.

Ridge regression is a technique that compensates for multicollinearity. Oracle Data Mining supports ridge regression for both regression and classification mining functions. The algorithm automatically uses ridge if it detects singularity (exact multicollinearity) in the data.

Information about singularity is returned in the global model details. See "Global Model Statistics for Linear Regression" on page 13-8 and "Global Model Statistics for Logistic Regression" on page 13-10.

Configuring Ridge Regression

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

Configuration settings for ridge are documented in *Oracle Database PL/SQL Packages and Types Reference*. The configuration choices are summarized as follows:

- Whether or not to override the automatic choice made by the algorithm regarding ridge regression
- The value of the ridge parameter, used only if you specifically enable ridge regression.
- Whether or not to produce Variance Inflation Factor (VIF) statistics when ridge is being used for linear regression.

Ridge and Confidence Bounds

Confidence bounds are not supported by models built with ridge regression. See "Confidence Bounds" on page 13-2.

Ridge and Variance Inflation Factor for Linear Regression

GLM produces Variance Inflation Factor (VIF) statistics for linear regression models, unless they were built with ridge. You can explicitly request VIF with ridge by specifying the `GLMS_VIF_FOR_RIDGE` setting. The algorithm will produce VIF with ridge only if enough system resources are available.

Ridge and Data Preparation

When ridge regression is enabled, different data preparation is likely to produce different results in terms of model coefficients and diagnostics. Oracle recommends that you enable Automatic Data Preparation for GLM models, especially when ridge regression is being used. See "Data Preparation for GLM" on page 13-6.

Scalable Feature Selection

Oracle Data Mining supports a highly scalable and automated version of feature selection and generation for GLM. This capability can enhance the performance of the algorithm and improve accuracy and interpretability. Feature selection and generation are available for both linear regression and binary logistic regression.

Feature Selection

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

Configuring Feature Selection

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

- The feature selection criteria can be AIC, SBIC, RIC, or α-investing. When the feature selection criteria is α-investing, feature acceptance can be either strict or relaxed.
- The maximum number of features can be specified.
- Features can be pruned in the final model. Pruning is based on t-statistics for linear regression or wald statistics for logistic regression.
- The exploded values for a categorical attribute can be added to the model all at once or one at a time (block size).

Feature Selection and Ridge Regression

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

Note: If you configure the model to use both feature selection and ridge regression, an error is raised.

Feature Generation

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

Configuring Feature Generation

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

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

The following options for feature selection also affect feature generation:

- Strict or relaxed feature acceptance (only when the feature selection criteria is α-investing)
- Maximum number of features
- Model pruning

Note: The blocksize setting is not appropriate for feature generation. If you set the blocksize when feature generation is enabled, an error is raised.

See Also: "Algorithm Settings: Generalized Linear Models" in *Oracle Database PL/SQL Packages and Types Reference*

Tuning and Diagnostics for GLM

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

Build Settings

You can use build settings to specify:

- The degree of certainty that the true coefficient lies within the confidence bounds computed by the model. The default confidence is.95.
- A column that contains a weighting factor for the rows.
- A table to contain row-level diagnostics.

Additional build settings are available to:

- Control the use of ridge regression, as described in "Ridge Regression" on page 13-2.
- Specify the handling of missing values in the training data, as described in "Data Preparation for GLM" on page 13-6.
- Specify the target value to be used as a reference in a logistic regression model, as described in "Logistic Regression" on page 13-9.

> **See Also:** "Algorithm Settings: Generalized Linear Models" in *Oracle Database PL/SQL Packages and Types Reference*

Diagnostics

GLM models generate many metrics to help you evaluate the quality of the model.

Coefficient Statistics

The same set of statistics is returned for both linear and logistic regression, but statistics that do not apply to the mining function are returned as NULL. The coefficient statistics are described in "Coefficient Statistics for Linear Regression" on page 13-7 and "Coefficient Statistics for Logistic Regression" on page 13-9.

Coefficient statistics are returned by the `GET_MODEL_DETAILS_GLM` function in `DBMS_DATA_MINING`.

Global Model Statistics

Separate high-level statistics describing the model as a whole, are returned for linear and logistic regression. When ridge regression is enabled, fewer global details are returned (See "Ridge Regression" on page 13-2). The global model statistics are described in "Global Model Statistics for Linear Regression" on page 13-8 and "Global Model Statistics for Logistic Regression" on page 13-10.

Global statistics are returned by the `GET_MODEL_DETAILS_GLOBAL` function in `DBMS_DATA_MINING`.

Row Diagnostics

You can configure GLM models to generate per-row statistics by specifying the name of a diagnostics table in the build setting `GLMS_DIAGNOSTICS_TABLE_NAME`. The row diagnostics are described in "Row Diagnostics for Linear Regression" on page 13-8 and

"Row Diagnostics for Logistic Regression" on page 13-10.

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

Data Preparation for GLM

Automatic Data Preparation (ADP) implements suitable data transformations for both linear and logistic regression.

Note: Oracle recommends that you use Automatic Data Preparation with GLM.

See Also: *Oracle Data Mining User's Guide*

Data Preparation for Linear Regression

When ADP is enabled, the build data are standardized using a widely used correlation transformation (Netter, et. al, 1990). The data are first centered by subtracting the attribute means from the attribute values for each observation. Then the data are scaled by dividing each attribute value in an observation by the square root of the sum of squares per attribute across all observations. This transformation is applied to both numeric and categorical attributes.

Prior to standardization, categorical attributes are exploded into N-1 columns where N is the attribute cardinality. The most frequent value (mode) is omitted during the explosion transformation. In the case of highest frequency ties, the attribute values are sorted alpha-numerically in ascending order, and the first value on the list is omitted during the explosion. This explosion transformation occurs whether or not ADP is enabled.

In the case of high cardinality categorical attributes, the described transformations (explosion followed by standardization) can increase the build data size because the resulting data representation is dense. To reduce memory, disk space, and processing requirements, an alternative approach needs to be used. For large datasets where the estimated internal dense representation would require more than 1Gb of disk space, categorical attributes are not standardized. Under these circumstances, the VIF statistic should be used with caution.

Reference: Neter, J., Wasserman, W., and Kutner, M.H., "Applied Statistical Models", Richard D. Irwin, Inc., Burr Ridge, IL, 1990.

See Also:

- "Ridge and Data Preparation" on page 13-3
- Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

Data Preparation for Logistic Regression

Categorical attributes are exploded into *N*-1 columns where *N* is the attribute cardinality. The most frequent value (mode) is omitted during the explosion transformation. In the case of highest frequency ties, the attribute values are sorted alpha-numerically in ascending order and the first value on the list is omitted during the explosion. This explosion transformation occurs whether or not ADP is enabled.

When ADP is enabled, numerical attributes are standardized by scaling the attribute values by a measure of attribute variability. This measure of variability is computed as the standard deviation per attribute with respect to the origin (not the mean) (Marquardt, 1980).

Reference: Marquardt, D.W., "A Critique of Some Ridge Regression Methods: Comment", Journal of the American Statistical Association, Vol. 75, No. 369 , 1980, pp. 87-91.

Missing Values

When building or applying a model, Oracle Data Mining automatically replaces missing values of numerical attributes with the mean and missing values of categorical attributes with the mode.

You can configure a GLM model to override the default treatment of missing values. With the `ODMS_MISSING_VALUE_TREATMENT` setting, you can cause the algorithm to delete rows in the training data that have missing values instead of replacing them with the mean or the mode. However, when the model is applied, Oracle Data Mining will perform the usual mean/mode missing value replacement. As a result, statistics generated from scoring may not match the statistics generated from building the model.

If you want to delete rows with missing values in the scoring the model, you must perform the transformation explicitly. To make build and apply statistics match, you must remove the rows with NULLs from the scoring data before performing the apply operation. You can do this by creating a view.

```
CREATE VIEW viewname AS SELECT * from tablename
     WHERE column_name1 is NOT NULL
     AND   column_name2 is NOT NULL
     AND   column_name3 is NOT NULL .....
```

Note: In Oracle Data Mining, missing values in nested data indicate sparsity, not values missing at random.

The value `ODMS_MISSING_VALUE_DELETE_ROW` is only valid for tables without nested columns. If this value is used with nested data, an exception is raised.

Linear Regression

Linear regression is the GLM regression algorithm supported by Oracle Data Mining. The algorithm assumes no target transformation and constant variance over the range of target values.

Coefficient Statistics for Linear Regression

GLM regression models generate the following coefficient statistics:

- Linear coefficient estimate
- Standard error of the coefficient estimate
- t-value of the coefficient estimate

- Probability of the t-value
- Variance Inflation Factor (VIF)
- Standardized estimate of the coefficient
- Lower and upper confidence bounds of the coefficient

Global Model Statistics for Linear Regression

GLM regression models generate the following statistics that describe the model as a whole:

- Model degrees of freedom
- Model sum of squares
- Model mean square
- Model *F* statistic
- Model *F* value probability
- Error degrees of freedom
- Error sum of squares
- Error mean square
- Corrected total degrees of freedom
- Corrected total sum of squares
- Root mean square error
- Dependent mean
- Coefficient of variation
- R-Square
- Adjusted R-Square
- Akaike's information criterion
- Schwarz's Baysian information criterion
- Estimated mean square error of the prediction
- Hocking Sp statistic
- JP statistic (the final prediction error)
- Number of parameters (the number of coefficients, including the intercept)
- Number of rows
- Whether or not the model converged
- Whether or not a covariance matrix was computed

Row Diagnostics for Linear Regression

For linear regression, the diagnostics table has the columns described in Table 13–1. All the columns are `NUMBER`, except the `CASE_ID` column, which preserves the type from the training data.

Table 13–1 Diagnostics Table for GLM Regression Models

Column	Description
CASE_ID	Value of the case ID column
TARGET_VALUE	Value of the target column
PREDICTED_VALUE	Value predicted by the model for the target
HAT	Value of the diagonal element of the hat matrix
RESIDUAL	Measure of error
STD_ERR_RESIDUAL	Standard error of the residual
STUDENTIZED_RESIDUAL	Studentized residual
PRED_RES	Predicted residual
COOKS_D	Cook's D influence statistic

Logistic Regression

Binary logistic regression is the GLM classification algorithm supported by Oracle Data Mining. The algorithm uses the logit link function and the binomial variance function.

Reference Class

You can use the build setting `GLMS_REFERENCE_CLASS_NAME` to specify the target value to be used as a reference in a binary logistic regression model. Probabilities will be produced for the other (non-reference) class. By default, the algorithm chooses the value with the highest prevalence. If there are ties, the attributes are sorted alpha-numerically in ascending order.

Class Weights

You can use the build setting `CLAS_WEIGHTS_TABLE_NAME` to specify the name of a class weights table. Class weights influence the weighting of target classes during the model build.

Coefficient Statistics for Logistic Regression

GLM classification models generate the following coefficient statistics:

- Name of the predictor
- Coefficient estimate
- Standard error of the coefficient estimate
- Wald chi-square value of the coefficient estimate
- Probability of the Wald chi-square value
- Standardized estimate of the coefficient
- Lower and upper confidence bounds of the coefficient
- Exponentiated coefficient
- Exponentiated coefficient for the upper and lower confidence bounds of the coefficient

Global Model Statistics for Logistic Regression

GLM classification models generate the following statistics that describe the model as a whole:

- Akaike's criterion for the fit of the intercept only model
- Akaike's criterion for the fit of the intercept and the covariates (predictors) model
- Schwarz's criterion for the fit of the intercept only model
- Schwarz's criterion for the fit of the intercept and the covariates (predictors) model
- -2 log likelihood of the intercept only model
- -2 log likelihood of the model
- Likelihood ratio degrees of freedom
- Likelihood ratio chi-square probability value
- Pseudo R-square Cox an Snell
- Pseudo R-square Nagelkerke
- Dependent mean
- Percent of correct predictions
- Percent of incorrect predictions
- Percent of ties (probability for two cases is the same)
- Number of parameters (the number of coefficients, including the intercept)
- Number of rows
- Whether or not the model converged
- Whether or not a covariance matrix was computed.

Row Diagnostics for Logistic Regression

For logistic regression, the diagnostics table has the columns described in Table 13–2. All the columns are `NUMBER`, except the `CASE_ID` and `TARGET_VALUE` columns, which preserve the type from the training data.

Table 13–2 Row Diagnostics Table for Logistic Regression

Column	Description
`CASE_ID`	Value of the case ID column
`TARGET_VALUE`	Value of the target value
`TARGET_VALUE_PROB`	Probability associated with the target value
`HAT`	Value of the diagonal element of the hat matrix
`WORKING_RESIDUAL`	Residual with respect to the adjusted dependent variable
`PEARSON_RESIDUAL`	The raw residual scaled by the estimated standard deviation of the target
`DEVIANCE_RESIDUAL`	Contribution to the overall goodness of fit of the model
`C`	Confidence interval displacement diagnostic
`CBAR`	Confidence interval displacement diagnostic
`DIFDEV`	Change in the deviance due to deleting an individual observation

Table 13–2 (Cont.) Row Diagnostics Table for Logistic Regression

Column	**Description**
`DIFCHISQ`	Change in the Pearson chi-square

14

k-Means

This chapter describes the enhanced k-Means clustering algorithm supported by Oracle Data Mining. See Chapter 7 for information about clustering.

This chapter contains these topics:

About *k*-Means

The *k*-Means algorithm is a distance-based clustering algorithm that partitions the data into a specified number of clusters.

Distance-based algorithms rely on a distance function to measure the similarity between cases. Cases are assigned to the nearest cluster according to the distance function used.

Oracle Data Mining Enhanced *k*-Means

Oracle Data Mining implements an enhanced version of the *k*-Means algorithm with the following features:

- **Distance function** — The algorithm supports Euclidean, Cosine, and Fast Cosine distance functions. The default is Euclidean.
- **Hierarchical model build** —The algorithm builds a model in a top-down hierarchical manner, 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 centroids of the inner nodes in the hierarchy are updated to reflect changes as the tree evolves. The whole tree is returned.
- **Tree growth** — The algorithm uses a specified split criterion to grow the tree one node at a time until a specified maximum number of clusters is reached, or until the number of distinct cases is reached. The split criterion may be the variance or the cluster size. By default the split criterion is the variance.
- **Cluster properties** — For each cluster, the algorithm returns the centroid, a histogram for each attribute, and a rule describing the hyperbox that encloses the majority of the data assigned to the cluster. The centroid reports the mode for categorical attributes and the mean and variance for numerical attributes.

This approach to *k*-Means avoids the need for building multiple *k*-Means models and provides clustering results that are consistently superior to the traditional *k*-Means.

Centroid

The **centroid** represents the most typical case in a cluster. For example, in a data set of customer ages and incomes, the centroid of each cluster would be a customer of average age and average income in that cluster. The centroid is a prototype. It does not necessarily describe any given case assigned to the cluster.

The attribute values for the centroid are the mean of the numerical attributes and the mode of the categorical attributes.

Scoring

The clusters discovered by *k*-Means are used to generate a Bayesian probability model that can be used to score new data.

Tuning the *k*-Means Algorithm

The Oracle Data Mining enhanced *k*-Means algorithm supports several build-time settings. All the settings have default values. There is no reason to override the defaults unless you want to influence the behavior of the algorithm in some specific way.

You can configure *k*-Means by specifying any of the following:

- Number of clusters
- Growth factor for memory allocated to hold clusters
- Convergence tolerance
- Distance Function. The default distance function is Euclidean.
- Split criterion. The default criterion is the variance.
- Number of iterations for building the cluster tree.
- The fraction of attribute values that must be non-null in order for an attribute to be included in the rule description for a cluster. Setting the parameter value too high in data with missing values can result in very short or even empty rules.
- Number of histogram bins. The bin boundaries for each attribute are computed globally on the entire training data set. The binning method is equi-width. All attributes have the same number of bins with the exception of attributes with a single value that have only one bin.

See Also: "Algorithm Settings: *k*-Means" in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for *k*-Means

Normalization is typically required by the *k*-Means algorithm. Automatic Data Preparation performs outlier-sensitive normalization for *k*-Means. If you do not use ADP, you should normalize numeric attributes before creating or applying the model.

When there are missing values in columns with simple data types (not nested), *k*-Means interprets them as missing at random. The algorithm replaces missing categorical values with the mode and missing numerical values with the mean.

When there are missing values in nested columns, *k*-Means interprets them as sparse. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors.

See Also:

"Linear Normalization" in *Oracle Database PL/SQL Packages and Types Reference*

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

15 Minimum Description Length

This chapter describes Minimum Description Length, the supervised technique used by Oracle Data Mining for calculating attribute importance. See "About Feature Selection and Attribute Importance" on page 9-2 for information about attribute importance.

This chapter contains these topics:

About MDL

Minimum Description Length (MDL) is an information theoretic model selection principle. It is an important concept in information theory (the study of the quantification of information) and in learning theory (the study of the capacity for generalization based on empirical data).

MDL assumes that the simplest, most compact representation of the data is the best and most probable explanation of the data. The MDL principle is used to build Oracle Data Mining attribute importance models.

The build process for attribute importance supports parallel execution. For information about parallel execution, refer to *Oracle Database VLDB and Partitioning Guide*.

Compression and Entropy

Data compression is the process of encoding information using fewer **bits** than the original representation would use. The MDL Principle is based on the notion that the shortest description of the data is the most probable. In typical instantiations of this principle, a model is used to compress the data by reducing the uncertainty (entropy) as discussed below. The description of the data includes a description of the model and the data as described by the model.

Entropy is a measure of uncertainty. It quantifies the uncertainty in a random variable as the information required to specify its value. **Information** in this sense is defined as the number of yes/no questions known as **bits** (encoded as 0 or 1) that must be answered for a complete specification. Thus, the information depends upon the number of values that variable can assume.

For example, if the variable represents the sex of an individual, then the number of possible values is two: female and male. If the variable represents the salary of individuals expressed in whole dollar amounts, it may have values in the range

$0-$10B, or billions of unique values. Clearly it will take more information to specify an exact salary than to specify an individual's sex.

Values of a Random Variable: Statistical Distribution

Information (the number of bits) depends on the statistical distribution of the values of the variable as well as the number of values of the variable. If we are judicious in the choice of Yes/No questions, the amount of information for salary specification may not be as much as it first appears. Most people do not have billion dollar salaries. If most people have salaries in the range $32000-$64000, then most of the time, we would require only 15 questions to discover their salary, rather than the 30 required, if every salary from $0-$1000000000 were equally likely. In the former example, if the persons were known to be pregnant, then their sex is known to be female. There is no uncertainty, no Yes/No questions need be asked. The entropy is 0.

Values of a Random Variable: Significant Predictors

Suppose that for some random variable there is a predictor that when its values are known reduces the uncertainty of the random variable. For example, knowing whether a person is pregnant or not, reduces the uncertainty of the random variable sex-of-individual. This predictor seems like a valuable feature to include in a model. How about name? Imagine that if you knew the name of the person, you would also know the person's sex. If so, the name predictor would seemingly reduce the uncertainty to zero. However, if names are unique, then what was gained? Is the person named Sally? Is the person named George?... We would have as many Yes/No predictors in the name model as there are people. Therefore, specifying the name model would require as many bits as specifying the sex of each person.

Total Entropy

For a random variable, X, the **total entropy** is defined as minus the Probability(X) multiplied by the log to the base 2 of the Probability(X). This can be shown to be the variable's most efficient encoding.

Model Size

MDL takes into consideration the size of the model as well as the reduction in uncertainty due to using the model. Both model size and entropy are measured in bits. For our purposes, both numeric and categorical predictors are binned. Thus the size of each single predictor model is the number of predictor bins. The uncertainty is reduced to the within-bin target distribution.

Model Selection

MDL considers each attribute as a simple predictive model of the target class. **Model selection** refers to the process of comparing and ranking the single-predictor models.

MDL uses a communication model for solving the model selection problem. In the communication model there is a sender, a receiver, and data to be transmitted.

These single predictor models are compared and ranked with respect to the MDL metric, which is the relative compression in bits. MDL penalizes model complexity to avoid over-fit. It is a principled approach that takes into account the complexity of the predictors (as models) to make the comparisons fair.

The MDL Metric

Attribute importance uses a two-part code as the metric for transmitting each unit of data. The first part (preamble) transmits the model. The parameters of the model are the target probabilities associated with each value of the prediction.

For a target with *j* values and a predictor with *k* values, n_i (i= 1,..., k) rows per value, there are C_i, the combination of *j*-1 things taken n_i-1 at a time possible conditional probabilities. The size of the preamble in bits can be shown to be 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.

Data Preparation for MDL

Automatic Data Preparation performs supervised binning for MDL. Supervised binning uses decision trees to create the optimal bin boundaries. Both categorical and numerical attributes are binned.

MDL handles missing values naturally as missing at random. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors. Missing values in nested columns are interpreted as sparse. Missing values in columns with simple data types are interpreted as missing at random.

If you choose to manage your own data preparation, keep in mind that MDL usually benefits from binning. However, the discriminating power of an attribute importance model can be significantly reduced when there are outliers in the data and external equal-width binning is used. This technique can cause most of the data to concentrate in a few bins (a single bin in extreme cases). In this case, quantile binning is a better solution.

See Also:

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

16 Naive Bayes

This chapter describes Naive Bayes, one of the classification algorithms supported by Oracle Data Mining. See Chapter 5 for information about classification.

This chapter contains these topics:

About Naive Bayes

The Naive Bayes algorithm is based on conditional probabilities. It uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. If `B` represents the dependent event and `A` represents the prior event, Bayes' theorem can be stated as follows.

Bayes' Theorem: Prob(B given A) = Prob(A and B)/Prob(A)

To calculate the probability of `B` given `A`, the algorithm counts the number of cases where `A` and `B` occur together and divides it by the number of cases where `A` occurs alone.

Example 16–1 Use Bayes' Theorem to Predict an Increase in Spending

Suppose you want to determine the likelihood that a customer under 21 will increase spending. In this case, the prior condition (`A`) would be "under 21," and the dependent condition (`B`) would be "increase spending."

If there are 100 customers in the training data and 25 of them are customers under 21 who have increased spending, then:

Prob(A and B) = 25%

If 75 of the 100 customers are under 21, then:

Prob(A) = 75%

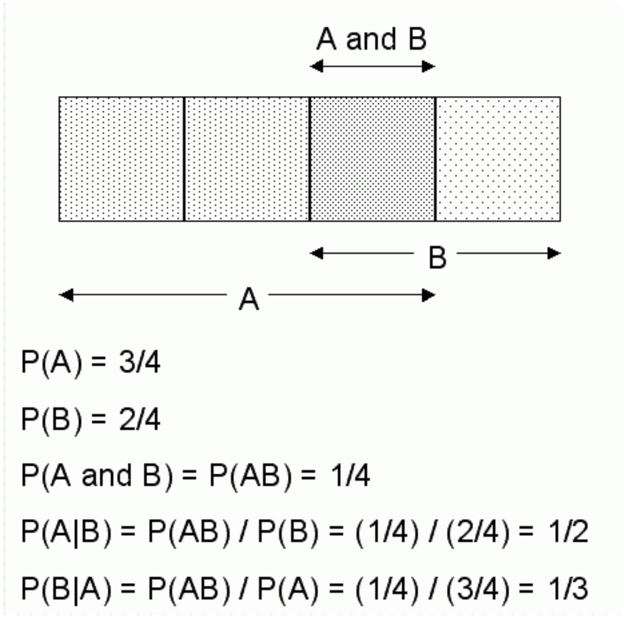
Bayes' Theorem would predict that 33% of customers under 21 are likely to increase spending (25/75).

The cases where both conditions occur together are referred to as **pairwise**. In Example 16–1, 25% of all cases are pairwise.

The cases where only the prior event occurs are referred to as **singleton**. In Example 16–1, 75% of all cases are singleton.

A visual representation of the conditional relationships used in Bayes' Theorem is shown in Figure 16–1.

Figure 16–1 Conditional Probabilities in Bayes' Theorem


A and B
B
A
P(A) = 3/4
P(B) = 2/4
P(A and B) = P(AB) = 1/4
P(A|B) = P(AB) / P(B) = (1/4) / (2/4) = 1/2
P(B|A) = P(AB) / P(A) = (1/4) / (3/4) = 1/3

For purposes of illustration, Example 16–1 and Figure 16–1 show a dependent event based on a single independent event. In reality, the Naive Bayes algorithm must usually take many independent events into account. In Example 16–1, factors such as income, education, gender, and store location might be considered in addition to age.

Naive Bayes makes the assumption that each predictor is conditionally independent of the others. For a given target value, the distribution of each predictor is independent of the other predictors. In practice, this assumption of independence, even when violated, does not degrade the model's predictive accuracy significantly, and makes the difference between a fast, computationally feasible algorithm and an intractable one.

Sometimes the distribution of a given predictor is clearly not representative of the larger population. For example, there might be only a few customers under 21 in the training data, but in fact there are many customers in this age group in the wider customer base. To compensate for this, you can specify **prior probabilities** when training the model. See "Priors and Class Weights" on page 5-7.

Advantages of Naive Bayes

The Naive Bayes algorithm affords fast, highly scalable model building and scoring. It scales linearly with the number of predictors and rows.

The build process for Naive Bayes supports parallel execution. (Scoring supports parallel execution irrespective of the algorithm.) For information about parallel execution, refer to *Oracle Database VLDB and Partitioning Guide*.

Naive Bayes can be used for both binary and multiclass classification problems.

Tuning a Naive Bayes Model

Naive Bayes calculates a probability by dividing the percentage of pairwise occurrences by the percentage of singleton occurrences. If these percentages are very small for a given predictor, they probably will not contribute to the effectiveness of the model. Occurrences below a certain threshold can usually be ignored.

Two build settings are available for adjusting the probability thresholds. You can specify:

- the minimum percentage of pairwise occurrences required for including a predictor in the model
- the minimum percentage of singleton occurrences required for including a predictor in the model

The default thresholds work well for most models, so you will not generally need to adjust these settings.

See Also: "Algorithm Settings: Naive Bayes" in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for Naive Bayes

Automatic Data Preparation performs supervised binning for Naive Bayes. Supervised binning uses decision trees to create the optimal bin boundaries. Both categorical and numerical attributes are binned.

Naive Bayes handles missing values naturally as missing at random. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors. Missing values in nested columns are interpreted as sparse. Missing values in columns with simple data types are interpreted as missing at random.

If you choose to manage your own data preparation, keep in mind that Naive Bayes usually requires binning. Naive Bayes relies on counting techniques to calculate probabilities. Columns should be binned to reduce the cardinality as appropriate. Numerical data can be binned into ranges of values (for example, low, medium, and high), and categorical data can be binned into meta-classes (for example, regions instead of cities). Equi-width binning is not recommended, since outliers will cause most of the data to concentrate in a few bins, sometimes a single bin. As a result, the discriminating power of the algorithms will be significantly reduced

See Also:

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

17 Non-Negative Matrix Factorization

This chapter describes Non-Negative Matrix Factorization (NMF), the unsupervised algorithm used by Oracle Data Mining for feature extraction. See Chapter 9 for information about feature extraction.

This chapter contains these topics:

- About NMF
- Tuning the NMF Algorithm
- Data Preparation for NMF

Note: 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).

About NMF

Non-Negative Matrix Factorization is a state of the art feature extraction algorithm. NMF is useful when there are many attributes and the attributes are ambiguous or have weak predictability. By combining attributes, NMF can produce meaningful patterns, topics, or themes.

Each feature created by NMF is a linear combination of the original attribute set. Each feature has a set of coefficients, which are a measure of the weight of each attribute on the feature. There is a separate coefficient for each numerical attribute and for each distinct value of each categorical attribute. The coefficients are all non-negative.

Matrix Factorization

Non-Negative Matrix Factorization uses techniques from multivariate analysis and linear algebra. It decomposes the data as a matrix *M* into the product of two lower ranking matrices *W* and *H*. The sub-matrix *W* contains the NMF basis; the sub-matrix *H* contains the associated coefficients (weights).

The algorithm iteratively modifies of the values of *W* and *H* so that their product approaches *M*. The technique preserves much of the structure of the original data and guarantees that both basis and weights are non-negative. The algorithm terminates when the approximation error converges or a specified number of iterations is reached.

The NMF algorithm must be initialized with a seed to indicate the starting point for the iterations. Because of the high dimensionality of the processing space and the fact that there is no global minimization algorithm, the appropriate initialization can be critical in obtaining meaningful results. Oracle Data Mining uses a random seed that initializes the values of W and H based on a uniform distribution. This approach works well in most cases.

Scoring with NMF

NMF can be used as a pre-processing step for dimensionality reduction in classification, regression, clustering, and other mining tasks. Scoring an NMF model produces data projections in the new feature space. The magnitude of a projection indicates how strongly a record maps to a feature.

The SQL scoring functions for feature extraction support NMF models. When the functions are invoked with the analytical syntax, the functions build and apply a transient NMF model. The feature extraction functions are: `FEATURE_DETAILS`, `FEATURE_ID`, `FEATURE_SET`, and `FEATURE_VALUE`.

See Also: "Scoring and Deployment" in *Oracle Data Mining User's Guide*

Text Mining with NMF

NMF is especially well-suited for text mining. In a text document, the same word can occur in different places with different meanings. For example, "hike" can be applied to the outdoors or to interest rates. By combining attributes, NMF introduces context, which is essential for explanatory power:

"hike" + "mountain" -> "outdoor sports"
"hike" + "interest" -> "interest rates"

See Also: "Mining Unstructured Text" in *Oracle Data Mining User's Guide*

Tuning the NMF Algorithm

Oracle Data Mining supports five configurable parameters for NMF. All of them have default values which will be appropriate for most applications of the algorithm. The NMF settings are:

- Number of features. By default, the number of features is determined by the algorithm.
- Convergence tolerance. The default is .05.
- Number of iterations. The default is 50.
- Random seed. The default is -1.
- Non-negative scoring. You can specify whether negative numbers should be allowed in scoring results. By default they are allowed.

See Also: "Algorithm Settings: Non-Negative Matrix Factorization in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for NMF

Automatic Data Preparation normalizes numerical attributes for NMF.

When there are missing values in columns with simple data types (not nested), NMF interprets them as missing at random. The algorithm replaces missing categorical values with the mode and missing numerical values with the mean.

When there are missing values in nested columns, NMF interprets them as sparse. The algorithm replaces sparse numerical data with zeros and sparse categorical data with zero vectors.

If you choose to manage your own data preparation, keep in mind that outliers can significantly impact NMF. Use a clipping transformation before binning or normalizing. NMF typically benefits from normalization. However, outliers with min-max normalization cause poor matrix factorization. To improve the matrix factorization, you need to decrease the error tolerance. This in turn leads to longer build times.

See Also:

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

18

O-Cluster

This chapter describes Orthogonal Partitioning Clustering (O-Cluster), an Oracle-proprietary clustering algorithm. See Chapter 7 for information about clustering.

This chapter contains these topics:

- About O-Cluster
- Tuning the O-Cluster Algorithm
- Data Preparation for O-Cluster

Reference:

Campos, M.M., Milenova, B.L., "Clustering Large Databases with Numeric and Nominal Values Using Orthogonal Projections", Oracle Data Mining Technologies, Oracle Corporation.

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datmin`

About O-Cluster

O-Cluster is a fast, scalable grid-based clustering algorithm well-suited for mining large, high-dimensional data sets. The algorithm can produce high quality clusters without relying on user-defined parameters.

The objective of O-Cluster is to identify areas of high density in the data and separate the dense areas into clusters. It uses axis-parallel uni-dimensional (orthogonal) data projections to identify the areas of density. The algorithm looks for splitting points that result in distinct clusters that do not overlap and are balanced in size.

O-Cluster operates recursively by creating a binary tree hierarchy. The number of leaf clusters is determined automatically. The algorithm can be configured to limit the maximum number of clusters.

Partitioning Strategy

Partitioning strategy refers to the process of discovering areas of density in the attribute histograms. The process differs for numerical and categorical data. When both are present in the data, the algorithm performs the searches separately and then compares the results.

In choosing a partition, the algorithm balances two objectives: finding well separated clusters, and creating clusters that are balanced in size. The following paragraphs detail how partitions for numerical and categorical attributes are identified.

Partitioning Numerical Attributes

To find the best valid cutting plane, O-Cluster searches the attribute histograms for bins of low density (valleys) between bins of high density (peaks). O-Cluster attempts to find a pair of peaks with a valley between them where the difference between the peak and valley histogram counts is statistically significant.

A **sensitivity** level parameter specifies the lowest density that may be considered a peak. Sensitivity is an optional parameter for numeric data. It may be used to filter the splitting point candidates.

Partitioning Categorical Attributes

Categorical values do not have an intrinsic order associated with them. Therefore it is impossible to apply the notion of histogram peaks and valleys that is used to partition numerical values.

Instead the counts of individual values form a histogram. Bins with large counts are interpreted as regions with high density. The clustering objective is to separate these high-density areas and effectively decrease the entropy (randomness) of the data.

O-Cluster identifies the histogram with highest entropy along the individual projections. Entropy is measured as the number of bins above **sensitivity** level. O-Cluster places the two largest bins into separate partitions, thereby creating a splitting predicate. The remainder of the bins are assigned randomly to the two resulting partitions.

Active Sampling

The O-Cluster algorithm operates on a data buffer of a limited size. It uses an active sampling mechanism to handle data sets that do not fit into memory.

After processing an initial random sample, O-Cluster identifies cases that are of no further interest. Such cases belong to *frozen* partitions where further splitting is highly unlikely. These cases are replaced with examples from *ambiguous* regions where further information (additional cases) is needed to find good splitting planes and continue partitioning. A partition is considered ambiguous if a valid split can only be found at a lower confidence level.

Cases associated with frozen partitions are marked for deletion from the buffer. They are replaced with cases belonging to ambiguous partitions. The histograms of the ambiguous partitions are updated and splitting points are reevaluated.

Process Flow

The O-Cluster algorithm evaluates possible splitting points for all projections in a partition, selects the best one, and splits the data into two new partitions. The algorithm proceeds by searching for good cutting planes inside the newly created partitions. Thus O-Cluster creates a binary tree structure that divides the input space into rectangular regions with no overlaps or gaps.

The main processing stages are:

1. Load the buffer. Assign all cases from the initial buffer to a single active root partition.
2. Compute histograms along the orthogonal uni-dimensional projections for each active partition.
3. Find the best splitting points for active partitions.

4. Flag ambiguous and frozen partitions.
5. When a valid separator exists, split the active partition into two new active partitions and start over at step 2.
6. Reload the buffer after all recursive partitioning on the current buffer is completed. Continue loading the buffer until either the buffer is filled again, or the end of the data set is reached, or until the number of cases is equal to the data buffer size.

Note: O-Cluster requires at most one pass through the data

Scoring

The clusters discovered by O-Cluster are used to generate a Bayesian probability model that can be used to score new data. The generated probability model is a mixture model where the mixture components are represented by a product of independent normal distributions for numerical attributes and multinomial distributions for categorical attributes.

Tuning the O-Cluster Algorithm

The O-Cluster algorithm supports two build-time settings. Both settings have default values. There is no reason to override the defaults unless you want to influence the behavior of the algorithm in some specific way.

You can configure O-Cluster by specifying any of the following:

- **Buffer size** — Size of the processing buffer. (See Active Sampling.)
- **Sensitivity factor** — A fraction that specifies the peak density required for separating a new cluster. (See Partitioning Strategy.)

See Also: "Algorithm Settings: O-Cluster" in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for O-Cluster

Automatic Data Preparation bins numerical attributes for O-Cluster. It uses a specialized form of equi-width binning that computes the number of bins per attribute automatically. Numerical columns with all nulls or a single value are removed. O-Cluster handles missing values naturally as missing at random.

Note: O-Cluster does not support nested columns, sparse data, or unstructured text.

See Also:

Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*

Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

User-Specified Data Preparation for O-Cluster

Keep the following in mind if you choose to prepare the data for O-Cluster:

- O-Cluster does not necessarily use all the input data when it builds a model. It reads the data in batches (the default batch size is 50000). It will only read another batch if it believes, based on statistical tests, that there may still exist clusters that it has not yet uncovered.

- Binary attributes should be declared as categorical.
- Automatic equi-width binning is highly recommended. The bin identifiers are expected to be positive consecutive integers starting at 1. See *Oracle Database PL/SQL Packages and Types Reference* for an example.
- The presence of outliers can significantly impact clustering algorithms. Use a clipping transformation before binning or normalizing. Outliers with equi-width binning can prevent O-Cluster from detecting clusters. As a result, the whole population appears to fall within a single cluster.

19

Singular Value Decomposition

This chapter describes Singular Value Decomposition, an unsupervised algorithm used by Oracle Data Mining for feature extraction. See Chapter 9 for information about feature extraction.

This chapter contains these topics:

About Singular Value Decomposition

Singular Value Decomposition (SVD) and the closely-related Principal Component Analysis (PCA) are well established feature extraction methods that have a wide range of applications. Oracle Data Mining implements SVD as a feature extraction algorithm and PCA as a special scoring method for SVD models.

SVD and PCA are orthogonal linear transformations that are optimal at capturing the underlying variance of the data. This property is very useful for reducing the dimensionality of high-dimensional data and for supporting meaningful data visualization.

SVD and PCA have a number of important applications in addition to dimensionality reduction. These include matrix inversion, data compression, and the imputation of unknown data values.

Matrix Manipulation

SVD is a factorization method that decomposes a rectangular matrix **X** into the product of three matrices:

$$\mathbf{X} = \mathbf{USV'}$$

The **U** matrix consists of a set of 'left' orthonormal bases
The **S** matrix is a diagonal matrix
The **V** matrix consists of set of 'right' orthonormal bases

The values in **S** are called singular values. They are non-negative, and their magnitudes indicate the importance of the corresponding bases (components). The singular values reflect the amount of data variance captured by the bases. The first basis (the one with largest singular value) lies in the direction of the greatest data variance. The second basis captures the orthogonal direction with the second greatest variance, and so on.

SVD essentially performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance in the data. This is a useful procedure under the assumption that the observed data has a high signal-to-noise ratio and that a large variance corresponds to interesting data content while a lower variance corresponds to noise.

SVD makes the assumption that the underlying data is Gaussian distributed and can be well described in terms of means and covariances.

Low Rank Decomposition

To reduce dimensionality, SVD keeps lower-order bases (the ones with the largest singular values) and ignores higher-order bases (the ones with the smallest singular values). The rationale behind this strategy is that the low-order bases retain the characteristics of the data that contribute most to its variance and are likely to capture the most important aspects of the data.

Given a data set **X** (*n*x*m*), where *n* is the number of rows and *m* is the number of attributes, a low-rank SVD will use only *k* components ($k \leq \mathbf{min}(m, n)$). In typical implementations of SVD, the value of *k* requires a visual inspection of the ranked singular values associated with the individual components. In Oracle Data Mining, SVD automatically estimates the cutoff point, which corresponds to a significant drop in the explained variance.

SVD produces two sets of orthonormal bases (**U** and **V**). Either of these bases could be used as a new coordinate system. In Oracle Data Mining SVD, **V** is the new coordinate system, and **U** represents the projection of **X** in this coordinate system. The algorithm computes the projection of new data as follows:

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{V}_k\mathbf{S}_k^{-1}$$

where **X** (*n*x*k*) is the projected data in the reduced data space, defined by the first *k* components, and $\mathbf{V}_k$ and $\mathbf{S}_k$ define the reduced component set.

Scalability

In Oracle Data Mining, SVD can process data sets with millions of rows and thousands of attributes. Oracle Data Mining automatically recommends an appropriate number of features, based on the data, for dimensionality reduction.

SVD has linear scalability with the number of rows and cubic scalability with the number of attributes when a full decomposition is computed. A low-rank decomposition is typically linear with the number of rows and linear with the number of columns. The scalability with the reduced rank depends on how the rank compares to the number of rows and columns. It can be linear when the rank is significantly smaller or cubic when it is on the same scale.

Configuring the Algorithm

Several options are available for configuring the SVD algorithm. Among them are settings to control model size and performance, and whether to score with SVD projections or PCA projections.

Model Size

The **U** matrix in SVD has as many rows as the number of rows in the build data. To avoid creating a large model, the **U** matrix is persisted only when an algorithm-specific setting is enabled. By default the **U** matrix is not persisted.

Performance

SVD can use approximate computations to improve performance. Approximation may be appropriate for data sets with many columns. An approximate low-rank decomposition provides good solutions at a reasonable computational cost. The quality of the approximation is dependent on the characteristics of the data. For data sets with more than 2500 attributes (the maximum number of features allowed) only approximate decomposition is possible.

PCA scoring

SVD models can be configured to perform PCA projections. PCA is closely related to SVD. PCA computes a set of orthonormal bases (principal components) that are ranked by their corresponding explained variance. The main difference between SVD and PCA is that the PCA projection is not scaled by the singular values. The PCA projection to the new coordinate system is given by:

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{V}_k$$

where $\tilde{\mathbf{X}}$ (*n*x*k*) is the projected data in the reduced data space, defined by the first *k* components, and $\mathbf{V}_k$ defines the reduced component set.

> **See Also:** DBMS_DATA_MINING, "Algorithm Constants and Settings: Singular Value Decomposition" in *Oracle Database PL/SQL Packages and Types Reference*

Data Preparation for SVD

Oracle Data Mining implements SVD for numerical data only. SVD is not well suited for processing categorical data.

Automatic Data Preparation performs normalization for SVD. Additional data preparation for outlier treatment may be beneficial since SVD can be sensitive to outliers.

Missing value treatment is not needed, because Oracle Data Mining algorithms handle missing values automatically. SVD replaces random missing values with the mean. For sparse data (missing values in nested columns), SVD replaces missing values with zeros.

> **See Also:**
>
> Chapter 3, "Preparing the Data" in *Oracle Data Mining User's Guide*
>
> Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*

20 Support Vector Machines

This chapter describes Support Vector Machines, a powerful algorithm based on statistical learning theory. Oracle Data Mining implements Support Vector Machines for classification, regression, and anomaly detection. See Chapter 4 — Chapter 6 for information about these mining functions.

This chapter contains these topics:

Reference:

Milenova, B.L., Yarmus, J.S., Campos, M.M., "Support Vector Machines in Oracle Database 10*g*: Removing the Barriers to Widespread Adoption of Support Vector Machines", Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

`http://www.oracle.com/pls/topic/lookup?ctx=db121&id=datmin`

About Support Vector Machines

Support Vector Machine (SVM) is a powerful, state-of-the-art algorithm with strong theoretical foundations based on the Vapnik-Chervonenkis theory. SVM has strong **regularization** properties. Regularization refers to the generalization of the model to new data.

Advantages of SVM

SVM models have similar functional form to neural networks and radial basis functions, both popular data mining techniques. However, neither of these algorithms has the well-founded theoretical approach to regularization that forms the basis of SVM. The quality of generalization and ease of training of SVM is far beyond the capacities of these more traditional methods.

SVM can model complex, real-world problems such as text and image classification, hand-writing recognition, and bioinformatics and biosequence analysis.

SVM performs well on data sets that have many attributes, even if there are very few cases on which to train the model. There is no upper limit on the number of attributes; the only constraints are those imposed by hardware. Traditional neural nets do not perform well under these circumstances.

Advantages of SVM in Oracle Data Mining

Oracle Data Mining has its own proprietary implementation of SVM, which exploits the many benefits of the algorithm while compensating for some of the limitations inherent in the SVM framework. Oracle Data Mining SVM provides the scalability and usability that are needed in a production quality data mining system.

Usability

Usability is a major enhancement, because SVM has often been viewed as a tool for experts. The algorithm typically requires data preparation, tuning, and optimization. Oracle Data Mining minimizes these requirements. You do not need to be an expert to build a quality SVM model in Oracle Data Mining. For example:

- Data preparation is not required in most cases. (See "Data Preparation for SVM" on page 20-4.)
- Default tuning parameters are generally adequate. (See "Tuning an SVM Model" on page 20-3.)

Scalability

When dealing with very large data sets, sampling is often required. However, sampling is not required with Oracle Data Mining SVM, because the algorithm itself uses stratified sampling to reduce the size of the training data as needed.

Oracle Data Mining SVM is highly optimized. It builds a model incrementally by optimizing small working sets toward a global solution. The model is trained until convergence on the current working set, then the model adapts to the new data. The process continues iteratively until the convergence conditions are met. The Gaussian kernel uses caching techniques to manage the working sets. See "Kernel-Based Learning" on page 20-2.

Oracle Data Mining SVM supports **active learning**, an optimization method that builds a smaller, more compact model while reducing the time and memory resources required for training the model. See "Active Learning" on page 20-3.

Kernel-Based Learning

SVM is a kernel-based algorithm. A **kernel** is a function that transforms the input data to a high-dimensional space where the problem is solved. Kernel functions can be linear or nonlinear.

Oracle Data Mining supports linear and Gaussian (nonlinear) kernels.

In Oracle Data Mining, the **linear kernel** function reduces to a linear equation on the original attributes in the training data. A linear kernel works well when there are many attributes in the training data.

The **Gaussian kernel** transforms each case in the training data to a point in an *n*-dimensional space, where *n* is the number of cases. The algorithm attempts to separate the points into subsets with homogeneous target values. The Gaussian kernel uses nonlinear separators, but within the kernel space it constructs a linear equation.

Active Learning

Active learning is an optimization method for controlling model growth and reducing model build time. Without active learning, SVM models grow as the size of the build data set increases, which effectively limits SVM models to small and medium size training sets (less than 100,000 cases). Active learning provides a way to overcome this restriction. With active learning, SVM models can be built on very large training sets.

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 a standard (exact) SVM model.

Active learning provides a significant improvement in both linear and Gaussian SVM models, whether for classification, regression, or anomaly detection. However, active learning is especially advantageous for the Gaussian kernel, because nonlinear models can otherwise grow to be very large and can place considerable demands on memory and other system resources.

Tuning an SVM Model

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

The build settings described in Table 20–1 are available for configuring SVM models. Settings pertain to regression, classification, and anomaly detection unless otherwise specified.

Table 20–1 Build Settings for Support Vector Machines

Setting Name	Configures....	Description
`SVMS_KERNEL_FUNCTION`	Kernel	Linear or Gaussian. The algorithm automatically uses the kernel function that is most appropriate to the data. SVM uses the linear kernel when there are many attributes (more than 100) in the training data, otherwise it uses the Gaussian kernel. See "Kernel-Based Learning" on page 20-2. The number of attributes does not correspond to the number of columns in the training data. SVM explodes categorical attributes to binary, numeric attributes. In addition, Oracle Data Mining interprets each row in a nested column as a separate attribute. See "Data Preparation for SVM" on page 20-4.
`SVMS_STD_DEV`	Standard deviation for Gaussian kernel	Controls the spread of the Gaussian kernel function. SVM uses a data-driven approach to find a standard deviation value that is on the same scale as distances between typical cases.
`SVMS_KERNEL_CACHE_SIZE`	Cache size for Gaussian kernel	Amount of memory allocated to the Gaussian kernel cache maintained in memory to improve model build time. The default cache size is 50 MB.
`SVMS_ACTIVE_LEARNING`	Active learning	Whether or not to use active learning. This setting is especially important for nonlinear (Gaussian) SVM models. By default, active learning is enabled. See "Active Learning" on page 20-3.
`SVMS_COMPLEXITY_FACTOR`	Complexity factor	Regularization setting that balances the complexity of the model against model robustness to achieve good generalization on new data. SVM uses a data-driven approach to finding the complexity factor.

Table 20–1 (Cont.) Build Settings for Support Vector Machines

Setting Name	Configures....	Description
`SVMS_CONVERGENCE_TOLERANCE`	Convergence tolerance	The criterion for completing the model training process. The default is 0.001.
`SVMS_EPSILON`	Epsilon factor for regression	Regularization setting for regression, similar to complexity factor. Epsilon specifies the allowable residuals, or noise, in the data.
`SVMS_OUTLIER_RATE`	Outliers for anomaly detection	The expected outlier rate in anomaly detection. The default rate is 0.1.

See Also: *Oracle Database PL/SQL Packages and Types Reference* for details about SVM settings

Data Preparation for SVM

The SVM algorithm operates natively on numeric attributes. The algorithm automatically "explodes" categorical data into a set of binary attributes, one per category value. For example, a character column for marital status with values `married` or `single` would be transformed to two numeric attributes: `married` and `single`. The new attributes could have the value 1 (true) or 0 (false).

When there are missing values in columns with simple data types (not nested), SVM interprets them as missing at random. The algorithm automatically replaces missing categorical values with the mode and missing numerical values with the mean.

When there are missing values in nested columns, SVM interprets them as sparse. The algorithm automatically replaces sparse numerical data with zeros and sparse categorical data with zero vectors.

Normalization

SVM requires the normalization of numeric input. Normalization places the values of numeric attributes on the same scale and prevents attributes with a large original scale from biasing the solution. Normalization also minimizes the likelihood of overflows and underflows. Furthermore, normalization brings the numerical attributes to the same scale (0,1) as the exploded categorical data.

SVM and Automatic Data Preparation

The SVM algorithm automatically handles missing value treatment and the transformation of categorical data, but normalization and outlier detection must be handled by ADP or prepared manually. ADP performs min-max normalization for SVM.

Note: Oracle recommends that you use Automatic Data Preparation with SVM. The transformations performed by ADP are appropriate for most models. Refer to Chapter 4, "Transforming the Data" in *Oracle Data Mining User's Guide*.

SVM Classification

SVM classification is based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. SVM finds the vectors ("support vectors") that define the separators giving the widest separation of classes.

SVM classification supports both binary and multiclass targets.

Class Weights

In SVM classification, weights are a biasing mechanism for specifying the relative importance of target values (classes).

SVM models are automatically initialized to achieve the best average prediction across all classes. However, if the training data does not represent a realistic distribution, you can bias the model to compensate for class values that are under-represented. If you increase the weight for a class, the percent of correct predictions for that class should increase.

See Also: "Priors and Class Weights" on page 5-7

One-Class SVM

Oracle Data Mining uses SVM as the one-class classifier for anomaly detection. When SVM is used for anomaly detection, it has the classification mining function but no target.

One-class SVM models, when applied, produce a prediction and a probability for each case in the scoring data. If the prediction is 1, the case is considered typical. If the prediction is 0, the case is considered anomalous. This behavior reflects the fact that the model is trained with normal data.

You can specify the percentage of the data that you expect to be anomalous with the `SVMS_OUTLIER_RATE` build setting. If you have some knowledge that the number of "suspicious" cases is a certain percentage of your population, then you can set the outlier rate to that percentage. The model will identify approximately that many "rare" cases when applied to the general population. The default is 10%, which is probably high for many anomaly detection problems.

SVM Regression

SVM uses an epsilon-insensitive loss function to solve regression problems.

SVM regression tries to find a continuous function such that the maximum number of data points lie within the epsilon-wide insensitivity tube. Predictions falling within epsilon distance of the true target value are not interpreted as errors.

The epsilon factor is a regularization setting for SVM regression. It balances the margin of error with model robustness to achieve the best generalization to new data. See Table 20–1 for descriptions of build settings for SVM.

Glossary

active learning

A feature of the **Support Vector Machines** algorithm that provides a way to deal with large training data sets.

ADP

See **Automatic Data Preparation**.

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 Oracle Data Mining API supports the following algorithms: **MDL**, **Apriori**, **Decision Tree**, **k-Means**, **Naive Bayes**, **GLM**, **O-Cluster**, **Support Vector Machines**, **Expectation Maximization**, and **Singular Value Decomposition**.

algorithm settings

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

anomaly detection

The detection of outliers or atypical cases. Oracle Data Mining implements Anomaly Detection as one-class SVM.

apply

The data mining operation that scores data. Scoring is the process of applying a model to new data to predict results.

Apriori

The algorithm that uses frequent itemsets to calculate associations.

association

A machine learning technique that identifies relationships among items.

association rules

A 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 itemset A implies the presence of itemset B with certain support and confidence. The support of the rule is the ratio of the number of transactions where the itemsets A and B are present to the total number of transactions. The confidence of the rule is the ratio of the number of transactions where the itemsets A and B are present to the number of transactions where itemset A is present. Oracle Data Mining uses the Apriori algorithm for association models.

attribute

An attribute is a predictor in a predictive model or an item of descriptive information in a descriptive model. **Data attributes** are the columns of data that are used to build a model. Data attributes undergo transformations so that they can be used as categoricals or numericals by the model. Categoricals and numericals are **model attributes**. See also **target**.

attribute importance

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

Automatic Data Preparation

Mining models can be created with Automatic Data Preparation (ADP), which transforms the build data according to the requirements of the algorithm and embeds the transformation instructions in the model. The embedded transformations are executed whenever the model is applied to new data.

binning

See **discretization**.

build data

Data used to build (train) a model. Also called *training data*.

case

All the data collected about a specific transaction or related set of values. A data set is a collection of cases. Cases are also called *records* or *examples*. In the simplest situation, a case corresponds to a row in a table.

case table

A table or view in single-record case format. All the data for each case is contained in a single row. The case table may include a case ID column that holds a unique identifier for each row. Mining data must be presented as a case table.

categorical attribute

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

centroid

See **cluster centroid**.

classification

A mining function for predicting categorical target values for new records using a model built from records with known target values. Oracle Data Mining supports the following algorithms for classification: Naive Bayes, Decision Tree, Generalized Linear Models, 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 training 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. Oracle Data Mining supports three algorithms for clustering, **k-Means**, **Orthogonal Partitioning Clustering**, and **Expectation Maximization**.

confusion matrix

Measures the correctness of predictions made by a model from a test task. The row indexes of a confusion matrix correspond to *actual values* observed and provided in the test data. The column indexes correspond to *predicted values* produced by applying the model to the test data. For any pair of actual/predicted indexes, the value indicates the number of records classified in that pairing.

When predicted value equals actual value, the model produces correct predictions. All other entries indicate errors.

cost matrix

An *n* by *n* table that defines the cost associated with a prediction versus the actual value. A cost matrix is typically used in classification models, where *n* is the number of distinct values in the target, and the columns and rows are labeled with target values. The rows are the actual values; the columns are the predicted values.

counterexample

Negative instance of a target. Counterexamples are required for classification models, except for **one-class Support Vector Machines**.

data mining

Data mining is the practice of automatically searching large stores of data to discover patterns and trends that go beyond simple analysis. Data mining uses sophisticated mathematical algorithms to segment the data and evaluate the probability of future events. Data mining is also known as *Knowledge Discovery in Data* (KDD).

A data mining **model** implements a data mining algorithm to solve a given type of problem for a given set of data.

data mining algorithm

A specific technique or procedure for producing a data mining model. An algorithm uses a specific data representation and a specific **mining function**.

The algorithms supported by Oracle Data Mining are **Naive Bayes**, **Support Vector Machines**, **Generalized Linear Model**, and **Decision Tree** for classification; **Support**

Vector Machines and **Generalized Linear Model** for regression; **k-Means**, **O-Cluster** and Expectation Maximization for clustering; **Minimum Description Length** for attribute importance; **Non-Negative Matrix Factorization** and **Singular Value Decomposition** for feature extraction; **Apriori** for associations, and **one-class Support Vector Machines** for anomaly detection.

data mining server

The component of 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.

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 may describe consumer buying patterns. See also **mining model**.

discretization

Discretization (binning) groups related values together under a single value (or bin). This reduces the number of distinct values in a column. Fewer bins result in models that build faster. Many Oracle Data Mining algorithms (for example NB) may benefit from input data that is *discretized* prior to model building, testing, computing lift, and applying (scoring). Different algorithms may require different types of binning. Oracle Data Mining supports **supervised binning**, **top N frequency binning** for categorical attributes and **equi-width binning** and **quantile binning** for numerical attributes.

distance-based (clustering algorithm)

Distance-based algorithms rely on a distance metric (function) to measure the similarity between data points. Data points are assigned to the nearest cluster according to the distance metric used.

Decision Tree

A decision tree is a representation of a classification system or supervised model. The tree is structured as a sequence of questions; the answers to the questions trace a path down the tree to a leaf, which yields the prediction.

Decision trees are a way of representing a series of questions that lead to a class or value. The top node of a decision tree is called the root node; terminal nodes are called leaf nodes. Decision trees are grown through an iterative splitting of data into discrete groups, where the goal is to maximize the distance between groups at each split.

An important characteristic of the decision tree models is that they are transparent; that is, there are rules that explain the classification.

See also **rule**.

equi-width binning

Equi-width binning determines bins for numerical attributes by dividing the range of values into a specified number of bins of equal size.

Expectation Maximization

Expectation Maximization is a probabilistic clustering algorithm that creates a density model of the data. The density model allows for an improved approach to combining data originating in different domains (for example, sales transactions and customer demographics, or structured data and text or other unstructured data).

explode

For a **categorical attribute**, replace a multi-value categorical column with several binary categorical columns. To explode the attribute, create a new binary column for each distinct value that the attribute takes on. In the new columns, 1 indicates that the value of the attribute takes on the value of the column; 0, that it does not. For example, suppose that a categorical attribute takes on the values {1, 2, 3}. To explode this attribute, create three new columns, `col_1`, `col_2`, and `col_3`. If the attribute takes on the value 1, the value in `col_1` is 1; the values in the other two columns is 0.

feature

A combination of attributes in the data that is of special interest and that captures important characteristics of the data. See **feature extraction**.

See also **text feature**.

feature extraction

Creates a new set of features by decomposing the original data. Feature extraction lets you describe the data with a number of features that is usually far smaller than the number of original attributes. See also **Non-Negative Matrix Factorization** and **Singular Value Decomposition**.

Generalized Linear Model

A statistical technique for linear modeling. Generalized linear models (GLM) include and extend the class of simple linear models. Oracle Data Mining supports logistic regression for GLM classification and linear regression for GLM regression.

GLM

See **Generalized Linear Model**.

k-Means

A distance-based clustering algorithm that partitions the data into a predetermined number of clusters (provided there are enough distinct cases). Distance-based algorithms rely on a distance metric (function) to measure the similarity between data points. Data points are assigned to the nearest cluster according to the distance metric used. Oracle Data Mining 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.

linear regression

The **GLM** regression algorithm supported by Oracle Data Mining.

logistic regression

The **GLM** classification algorithm supported by Oracle Data Mining.

MDL

See **Minimum Description Length**.

min-max normalization

Normalizes numerical attributes using this transformation:

```
x_new = (x_old-min) / (max-min)
```

Minimum Description Length

Given a sample of data and an effective enumeration of the appropriate alternative theories to explain the data, the best theory is the one that minimizes the sum of

- The length, in bits, of the description of the theory
- The length, in bits, of the data when encoded with the help of the theory

The Minimum Description Length principle is used to select the attributes that most influence target value discrimination in **attribute importance**.

mining function

A major subdomain of data mining that shares common high level characteristics. The Oracle Data Mining API supports the following mining functions: **classification**, **regression**, **attribute importance**, **feature extraction**, **clustering**, and **anomaly detection**.

mining model

A first-class schema object that specifies a data mining **model** in Oracle Database.

missing value

A data value that is missing at random. The value could be missing because it is unavailable, unknown, or because it was lost. Oracle Data Mining interprets missing values in columns with simple data types (not nested) as missing at random. Oracle Data Mining interprets missing values in nested columns as sparsity.

Data mining algorithms vary in the way they treat missing values. There are several typical ways to treat them: ignore them, omit any records containing missing values, replace missing values with the mode or mean, or infer missing values from existing values. See also **sparse data**.

model

A model uses an algorithm to implement a given mining function. A model can be a **supervised model** or an **unsupervised model**. A model can be used for direct inspection, for example, to examine the rules produced from an association model, or to score data (predict an outcome). In Oracle Database, data mining models are implemented as **mining model** schema objects.

multi-record case

Each case in the data table is stored in multiple rows. Also known as **transactional data**. See also **single-record case**.

Naive Bayes

An algorithm for classification that is based on Bayes's theorem. Naive Bayes makes the assumption that each attribute is conditionally independent of the others: given a particular value of the target, the distribution of each predictor is independent of the other predictors.

nested data

Oracle Data Mining supports **transactional data** in nested columns of name/value pairs. Multidimensional data that expresses a one-to-many relationship can be loaded into a nested column and mined along with single-record case data in a **case table**.

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 Machines

The version of **Support Vector Machines** used to solve **anomaly detection** problems. The algorithm performs classification without a target.

Orthogonal Partitioning Clustering

An Oracle proprietary clustering algorithm that creates a hierarchical grid-based clustering model, that is, it creates axis-parallel (orthogonal) partitions in the input attribute space. The algorithm operates recursively. The resulting hierarchical structure represents an irregular grid that tessellates the attribute space into clusters.

outlier

A data value that does not come from the typical population of data; in other words, extreme values. In a normal distribution, outliers are typically at least three standard deviations from the mean.

positive target value

In binary classification problems, you may designate one of the two classes (target values) as positive, the other as negative. When Oracle Data Mining 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 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. **Automatic Data Preparation** greatly simplifies model development and deployment by automatically preparing the data for the algorithm.

Principal Component Analysis

Principal Component Analysis is implemented as a special scoring method for the **Singular Value Decomposition** algorithm.

prior probabilities

The set of prior probabilities specifies the distribution of examples of the various classes in the original source data. Also referred to as *priors*, these could be different from the distribution observed in the data set provided for model build.

priors

See **prior probabilities**.

quantile binning

A numerical attribute is divided into bins such that each bin contains approximately the same number of cases.

random sample

A sample in which every element of the data set has an equal chance of being selected.

recode

Literally "change or rearrange the code." Recoding can be useful in preparing data according to the requirements of a given business problem, for example:

- Missing values treatment: Missing values may be indicated by something other than `NULL`, such as "0000" or "9999" or "NA" or some other string. One way to treat the missing value is to recode, for example, "0000" to `NULL`. Then the Oracle Data Mining 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. Oracle Data Mining supports **linear regression** (GLM) and **Support Vector Machines** algorithms for regression.

rule

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

sample

See **random sample**.

scale normalization

Normalize numerical attributes using this transformation:

```
x_new = (x_old - 0) / (max(abs(max),abs(min)))
```

schema

A collection of objects in an Oracle database, including logical structures such as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. A schema is associated with a specific database user.

score

Scoring data means applying a data mining model to data to generate predictions.

settings

See **algorithm settings**.

single-record case

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

Singular Value Decomposition

A feature extraction algorithm that uses orthogonal linear projections to capture the underlying variance of the data. Singular Value Decomposition scales well to very large data sizes (both rows and attributes), and has a powerful data compression capability.

See **Singular Value Decomposition**.

sparse data

Data for which only a small fraction of the attributes are non-zero or non-null in any given case. Market basket data and unstructured text data are typically sparse. Oracle Data Mining interprets **nested data** as sparse. See also **missing value**.

split

Divide a data set into several disjoint subsets. For example, in a classification problem, a data set is often divided in to a training data set and a test data set.

stratified sample

Divide the data set into disjoint subsets (strata) and then take a random sample from each of the subsets. This technique is used when the distribution of target values is skewed greatly. For example, response to a marketing campaign may have a positive target value 1% of the time or less. A stratified sample provides the data mining algorithms with enough positive examples to learn the factors that differentiate positive from negative target values. See also **random sample**.

supervised binning

A form of intelligent binning wherein bin boundaries are derived from important characteristics of the data. Supervised binning builds a single-predictor decision tree to find the interesting bin boundaries with respect to a target. Supervised binning can be used for numerical or categorical attributes.

supervised learning

See **supervised model**.

supervised model

A data mining model that is built using a known dependent variable, also referred to as the target. Classification and regression techniques are examples of supervised mining. See **unsupervised model**. Also referred to as **predictive model**.

Support Vector Machines

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 Machines can be used for classification, regression, and anomaly detection.

SVM

See **Support Vector Machines**.

target

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

text feature

A combination of words that captures important attributes of a document or class of documents. Text features are usually keywords, frequencies of words, or other document-derived features. A document typically contains a large number of words and a much smaller number of features.

text 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 Oracle Data Mining 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.

training data

See **build data**.

transactional data

The data for one case is contained in several rows. An example is market basket data, in which a case represents one basket that contains multiple items. Oracle Data Mining supports transactional data in nested columns of attribute name/value pairs. See also **nested data**, **multi-record case**, and **single-record case**.

transformation

A function applied to data resulting in a new representation of the data. For example, discretization and normalization are transformations on data.

trimming

A technique for minimizing the impact of outliers. Trimming removes values in the tails of a distribution in the sense that trimmed values are ignored in further computations. Trimming is achieved by setting the tails to `NULL`.

unstructured data

Images, audio, video, geospatial mapping data, and documents or text data are collectively known as unstructured data. Oracle Data Mining 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**.

winsorizing

A technique for minimizing the impact of outliers. Winsorizing involves setting the tail values of an particular attribute to some specified value. For example, for a 90% Winsorization, the bottom 5% of values are set equal to the minimum value in the 6th percentile, while the upper 5% are set equal to the maximum value in the 95th percentile.

z-score normalization

Normalize numerical attributes using this transformation:

```
x_new = (x_old-mean) / standard_deviation
```

Index