

Oracle® Data Mining

User's Guide

12c Release 2 (12.2)

E49657-12

May 2017

Oracle Data Mining User's Guide, 12c Release 2 (12.2)

E49657-12

Copyright © 2005, 2016, Oracle and/or its affiliates. All rights reserved.

Primary Author: Sarika Surampudi

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

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

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

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

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

Contents

Preface	xi
Audience	xi
Documentation Accessibility	xi
Related Documentation	xi
Oracle Data Mining Resources on the Oracle Technology Network.....	xii
Application Development and Database Administration Documentation.....	xii
Conventions.....	xiii
Changes in This Release for Oracle Data Mining User's Guide.....	xv
Oracle Data Mining User's Guide is New in This Release.....	xv
Changes in Oracle Data Mining 12c Release 2 (12.2).....	xv
New Features in 12c Release 2.....	xv
Changes in Oracle Data Mining 12c Release 1 (12.1).....	xvii
New Features	xvii
Desupported Features	xviii
Other Changes	xviii
1 Data Mining With SQL	
1.1 Highlights of the Data Mining API.....	1-1
1.2 Example: Targeting Likely Candidates for a Sales Promotion	1-2
1.3 Example: Analyzing Preferred Customers	1-3
1.4 Example: Segmenting Customer Data.....	1-5
1.5 Example : Building an ESA Model with a Wiki Dataset.....	1-6
2 About the Data Mining API	
2.1 About Mining Models.....	2-1
2.2 Data Mining Data Dictionary Views.....	2-2
2.2.1 ALL_MINING_MODELS.....	2-2
2.2.2 ALL_MINING_MODEL_ATTRIBUTES	2-3
2.2.3 ALL_MINING_MODEL_PARTITIONS	2-4
2.2.4 ALL_MINING_MODEL_SETTINGS.....	2-5

2.2.5	ALL_MINING_MODEL_VIEWS	2-5
2.2.6	ALL_MINING_MODEL_XFORMS	2-6
2.3	Data Mining PL/SQL Packages	2-7
2.3.1	DBMS_DATA_MINING.....	2-7
2.3.2	DBMS_DATA_MINING_TRANSFORM.....	2-7
2.3.3	DBMS_PREDICTIVE_ANALYTICS	2-9
2.4	Data Mining SQL Scoring Functions	2-9

3 Preparing the Data

3.1	Data Requirements	3-1
3.1.1	Column Data Types	3-2
3.1.2	Data Sets for Classification and Regression	3-2
3.1.3	Scoring Requirements	3-2
3.2	About Attributes	3-3
3.2.1	Data Attributes and Model Attributes	3-3
3.2.2	Target Attribute	3-4
3.2.3	Numericals, Categoricals, and Unstructured Text	3-4
3.2.4	Model Signature	3-5
3.2.5	Scoping of Model Attribute Name.....	3-5
3.2.6	Model Details	3-6
3.3	Using Nested Data.....	3-6
3.3.1	Nested Object Types	3-7
3.3.2	Example: Transforming Transactional Data for Mining.....	3-8
3.4	Using Market Basket Data	3-9
3.4.1	Example: Creating a Nested Column for Market Basket Analysis	3-10
3.5	Using Retail Analysis Data.....	3-11
3.5.1	Example: Calculating Aggregates.....	3-11
3.6	Handling Missing Values	3-12
3.6.1	Examples: Missing Values or Sparse Data?	3-12
3.6.2	Missing Value Treatment in Oracle Data Mining.....	3-13
3.6.3	Changing the Missing Value Treatment	3-14

4 Transforming the Data

4.1	About Transformations.....	4-1
4.2	Preparing the Case Table.....	4-2
4.2.1	Creating Nested Columns.....	4-2
4.2.2	Converting Column Data Types	4-2
4.2.3	Text Transformation.....	4-2
4.2.4	About Business and Domain-Sensitive Transformations.....	4-3
4.3	Understanding Automatic Data Preparation	4-3
4.3.1	Binning.....	4-3
4.3.2	Normalization.....	4-4
4.3.3	Outlier Treatment.....	4-4

4.3.4	How ADP Transforms the Data	4-4
4.4	Embedding Transformations in a Model	4-5
4.4.1	Specifying Transformation Instructions for an Attribute.....	4-5
4.4.2	Building a Transformation List	4-7
4.4.3	Transformation Lists and Automatic Data Preparation.....	4-9
4.4.4	Oracle Data Mining Transformation Routines.....	4-9
4.5	Understanding Reverse Transformations	4-11

5 Creating a Model

5.1	Before Creating a Model	5-1
5.2	The CREATE_MODEL Procedure.....	5-2
5.2.1	Choosing the Mining Function.....	5-2
5.2.2	Choosing the Algorithm.....	5-3
5.2.3	Supplying Transformations	5-4
5.2.4	About Partitioned Model	5-5
5.3	Specifying Model Settings	5-7
5.3.1	Specifying Costs.....	5-9
5.3.2	Specifying Prior Probabilities	5-10
5.3.3	Specifying Class Weights	5-10
5.3.4	Model Settings in the Data Dictionary	5-11
5.3.5	Specifying Mining Model Settings for R Model	5-12
5.4	Model Detail Views	5-19
5.4.1	Model Detail Views for Association Rules	5-20
5.4.2	Model Detail View for Frequent Itemsets.....	5-26
5.4.3	Model Detail View for Transactional Itemsets.....	5-26
5.4.4	Model Detail View for Transactional Rule	5-27
5.4.5	Model Detail Views for Classification Algorithms.....	5-28
5.4.6	Model Detail Views for Decision Tree	5-29
5.4.7	Model Detail Views for Generalized Linear Model	5-31
5.4.8	Model Detail Views for Naive Bayes.....	5-37
5.4.9	Model Detail View for Support Vector Machine	5-38
5.4.10	Model Detail Views for Clustering Algorithms.....	5-39
5.4.11	Model Detail Views for Expectation Maximization	5-42
5.4.12	Model Detail Views for k-Means	5-45
5.4.13	Model Detail Views for O-Cluster	5-46
5.4.14	Model Detail Views for Explicit Semantic Analysis.....	5-47
5.4.15	Model Detail Views for Non-Negative Matrix Factorization	5-48
5.4.16	Model Detail Views for Singular Value Decomposition	5-50
5.4.17	Model Detail View for Minimum Description Length	5-52
5.4.18	Model Detail View for Binning	5-53
5.4.19	Model Detail Views for Global Information.....	5-53
5.4.20	Model Detail View for Normalization and Missing Value Handling	5-54

6	Scoring and Deployment	
6.1	About Scoring and Deployment.....	6-1
6.2	Using the Data Mining SQL Functions.....	6-2
6.2.1	Choosing the Predictors	6-2
6.2.2	Single-Record Scoring.....	6-3
6.3	Prediction Details.....	6-4
6.3.1	Cluster Details.....	6-4
6.3.2	Feature Details	6-5
6.3.3	Prediction Details	6-5
6.3.4	GROUPING Hint.....	6-7
6.4	Real-Time Scoring.....	6-7
6.5	Dynamic Scoring.....	6-8
6.6	Cost-Sensitive Decision Making	6-9
6.7	DBMS_DATA_MINING.Apply	6-11
7	Mining Unstructured Text	
7.1	About Unstructured Text	7-1
7.2	About Text Mining and Oracle Text	7-1
7.3	Data Preparation for Text Features	7-2
7.4	Creating a Model that Includes Text Mining.....	7-2
7.5	Creating a Text Policy	7-4
7.6	Configuring a Text Attribute	7-5
8	Administrative Tasks for Oracle Data Mining	
8.1	Installing and Configuring a Database for Data Mining	8-1
8.1.1	About Installation.....	8-1
8.1.2	Enabling or Disabling a Database Option	8-2
8.1.3	Database Tuning Considerations for Data Mining	8-2
8.2	Upgrading or Downgrading Oracle Data Mining	8-3
8.2.1	Pre-Upgrade Steps.....	8-3
8.2.2	Upgrading Oracle Data Mining	8-3
8.2.3	Post Upgrade Steps	8-6
8.2.4	Downgrading Oracle Data Mining.....	8-7
8.3	Exporting and Importing Mining Models.....	8-7
8.3.1	About Oracle Data Pump.....	8-7
8.3.2	Options for Exporting and Importing Mining Models.....	8-8
8.3.3	Directory Objects for EXPORT_MODEL and IMPORT_MODEL.....	8-9
8.3.4	Using EXPORT_MODEL and IMPORT_MODEL	8-9
8.3.5	Importing From PMML.....	8-11
8.4	Controlling Access to Mining Models and Data.....	8-11
8.4.1	Creating a Data Mining User	8-11
8.4.2	System Privileges for Data Mining	8-13

8.4.3	Object Privileges for Mining Models.....	8-14
8.5	Auditing and Adding Comments to Mining Models.....	8-15
8.5.1	Adding a Comment to a Mining Model.....	8-15
8.5.2	Auditing Mining Models.....	8-16

A The Data Mining Sample Programs

A.1	About the Data Mining Sample Programs.....	A-1
A.2	Installing the Data Mining Sample Programs	A-2
A.3	The Data Mining Sample Data.....	A-3

Index

List of Tables

2-1	Data Dictionary Views for Oracle Data Mining.....	2-2
2-2	Data Mining PL/SQL Packages.....	2-7
2-3	DBMS_DATA_MINING_TRANSFORM Transformation Methods.....	2-8
2-4	Data Mining SQL Functions.....	2-9
3-1	Target Data Types.....	3-4
3-2	Grocery Store Data.....	3-11
3-3	Missing Value Treatment by Algorithm.....	3-13
4-1	Oracle Data Mining Algorithms With ADP.....	4-4
4-2	Fields in a Transformation Record for an Attribute.....	4-6
4-3	Binning Methods in DBMS_DATA_MINING_TRANSFORM.....	4-10
4-4	Normalization Methods in DBMS_DATA_MINING_TRANSFORM.....	4-10
4-5	Outlier Treatment Methods in DBMS_DATA_MINING_TRANSFORM.....	4-11
5-1	Preparation for Creating a Mining Model.....	5-1
5-2	Mining Model Functions.....	5-2
5-3	Data Mining Algorithms.....	5-3
5-4	Settings Table Required Columns.....	5-8
5-5	General Model Settings.....	5-8
5-6	Algorithm-Specific Model Settings.....	5-8
5-7	Cost Matrix Table Required Columns.....	5-9
5-8	Priors Table Required Columns.....	5-10
5-9	Class Weights Table Required Columns.....	5-10
5-10	ALL_MINING_MODEL_SETTINGS.....	5-11
5-11	Rule View Columns for Transactional Inputs.....	5-21
5-12	Frequent Itemsets View.....	5-26
5-13	Transactional Itemsets View.....	5-27
5-14	Transactional Rule View.....	5-27
5-15	Target Map View.....	5-28
5-16	Scoring Cost View.....	5-29
5-17	Split Information View.....	5-29
5-18	Node Statistics View.....	5-30
5-19	Node Description View.....	5-31
5-20	Cost Matrix View.....	5-31
5-21	Model View for Linear and Logistic Regression Models.....	5-32
5-22	Row Diagnostic View for Linear Regression.....	5-34
5-23	Row Diagnostic View for Logistic Regression.....	5-35
5-24	Prior View for Naive Bayes.....	5-37
5-25	Result View for Naive Bayes.....	5-37
5-26	Linear Coefficient View for Support Vector Machine.....	5-38
5-27	Cluster Description View for Clustering Algorithm.....	5-39
5-28	Attribute View for Clustering Algorithm.....	5-40
5-29	Histogram View for Clustering Algorithm.....	5-40
5-30	Rule View for Clustering Algorithm.....	5-41
5-31	Component View.....	5-42
5-32	Frequency Component View.....	5-43
5-33	2-Dimensional Attribute Ranking for Expectation Maximization.....	5-43
5-34	Kullback-Leibler Divergence for Expectation Maximization.....	5-44
5-35	Projection table for Expectation Maximization.....	5-44
5-36	Cluster Description for k-Means.....	5-45
5-37	Scoring View for k-Means.....	5-45
5-38	Description View.....	5-46
5-39	Histogram Component View.....	5-47

5-40	Attribute View for Explicit Semantic Analysis.....	5-48
5-41	Feature View for Explicit Semantic Analysis.....	5-48
5-42	Encoding H Matrix View for Non-Negative Matrix Factorization.....	5-49
5-43	Inverse H Matrix View for Non-Negative Matrix Factorization.....	5-49
5-44	S Matrix View.....	5-50
5-45	Right-singular Vectors of Singular Value Decomposition.....	5-51
5-46	Left-singular Vectors of Singular Value Decomposition or Projection Data in Principal Components.....	5-52
5-47	Attribute Importance View for Minimum Description Length.....	5-52
5-48	Model Details View for Binning.....	5-53
5-49	Global Statistics View.....	5-53
5-50	Alert View.....	5-54
5-51	Computed Settings View.....	5-54
5-52	Normalization and Missing Value Handling View.....	5-55
6-1	Sample Cost Matrix.....	6-10
6-2	APPLY Output Table.....	6-12
7-1	Text Feature View for Extracted Text Features.....	7-2
7-2	Column Data Types That May Contain Unstructured Text.....	7-2
7-3	Model Settings for Text.....	7-3
7-4	CTX_DDL.CREATE_POLICY Procedure Parameters.....	7-4
7-5	Attribute-Specific Text Transformation Instructions.....	7-5
8-1	Export and Import Options for Oracle Data Mining.....	8-8
8-2	System Privileges for Data Mining.....	8-13
8-3	Object Privileges for Mining Models.....	8-14
A-1	System Privileges Granted by dmshgrants.sql to the Data Mining User.....	A-3
A-2	The Data Mining Sample Data.....	A-4

Preface

This guide explains how to use the programmatic interfaces to Oracle Data Mining and how to use features of Oracle Database to administer Oracle Data Mining. This guide presents the tools and procedures for implementing the concepts that are presented in *Oracle Data Mining Concepts*.

This preface contains these topics:

- [Audience](#) (page xi)
- [Documentation Accessibility](#) (page xi)
- [Related Documentation](#) (page xi)
- [Conventions](#) (page xiii)

Audience

This guide is intended for application developers and database administrators who are familiar with SQL programming and Oracle Database administration and who have a basic understanding of data mining concepts.

Documentation Accessibility

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

Access to Oracle Support

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

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=db122&id=dwbitab>

The following manuals document Oracle Data Mining:

- *Oracle Data Mining Concepts*
- *Oracle Data Mining User's Guide* (page 1-1) (this guide)
- *Oracle Data Mining API Guide*

Note:

The virtual book combines key passages from the two 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* (PL/SQL packages)
 - DBMS_DATA_MINING
 - DBMS_DATA_MINING_TRANSFORM
 - DBMS_PREDICTIVE_ANALYTICS
- *Oracle Database Reference* (data dictionary views for ALL_, USER_, and DBA_)
 - ALL_MINING_MODELS
 - ALL_MINING_MODEL_ATTRIBUTES
 - ALL_MINING_MODEL_SETTINGS
- *Oracle Database SQL Language Reference* (Data Mining functions)
 - CLUSTER_DETAILS, CLUSTER_DISTANCE, CLUSTER_ID, CLUSTER_PROBABILITY, CLUSTER_SET
 - FEATURE_DETAILS, FEATURE_ID, FEATURE_SET, FEATURE_VALUE
 - PREDICTION, PREDICTION_BOUNDS, PREDICTION_COST, PREDICTION_DETAILS, PREDICTION_PROBABILITY, PREDICTION_SET

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=db122&id=datmin>

You can download Oracle Data Miner, the graphical user interface to Oracle Data Mining, from this site:

<http://www.oracle.com/pls/topic/lookup?ctx=db122&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*

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.
<i>italic</i>	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 User's Guide

Changes in this release for *Oracle Data Mining User's Guide*.

Oracle Data Mining User's Guide is New in This Release

- This guide is new in release 12c. *Oracle Data Mining User's Guide* replaces two manuals that were provided in previous releases: *Oracle Data Mining Administrator's Guide* and *Oracle Data Mining Application Developer's Guide*.
- Information about database administration for Oracle Data Mining is now consolidated in [Administrative Tasks for Oracle Data Mining](#) (page 8-1). The remaining chapters of this guide are devoted to application development.
- Information about the Data Mining sample programs is now in [The Data Mining Sample Programs](#) (page A-1).

Changes in Oracle Data Mining 12c Release 2 (12.2)

The following changes are documented in *Oracle Data Mining User's Guide* for 12c Release 2 (12.2).

New Features in 12c Release 2

The following features are new in this release:

Partitioned Models

- Data Mining SQL function
A new Data Mining SQL function `ORA_DM_PARTITION_NAME` is included for partitioned models. The function returns the partition names for a partitioned model.
See [Data Mining SQL Scoring Functions](#) (page 2-9).
Provided new scoring functions
See [Partitioned Model scoring](#) (page 5-7).
See [GROUPING Hint](#) (page 6-7).
- About partitioned model
Description of Partitioned model is added

See [About Partitioned Model](#) (page 5-5).

- DDL in partitioned model
Explained the newly added Add and Drop partition for maintenance operations.
See [DDL Implementation for Partitioned Model](#) (page 5-6).

Model Views

- Added new Model Detail Views. Model Detail Views are preferred over GET* functions.
See [Model Detail Views](#) (page 5-19).
New Data Dictionary Views. See [Data Mining Data Dictionary Views](#) (page 2-2).

Explicit Semantic Analysis

- Newly added FEATURE_COMPARE SQL function
See [Data Mining SQL Scoring Functions](#) (page 2-9).
- FEATURE_COMPARE SQL function
Provides an example of the new SQL function FEATURE_COMPARE using ESA algorithm.
See [Example : Building an ESA model with a Wiki dataset](#) (page 1-6).

Association Rules Aggregates

- Using retail analysis data
Added enhancements to Association Rules and an example to show the concept of aggregates.
See [Using Retail Analysis Data](#) (page 3-11).
See [Model Detail Views](#) (page 5-19).

R Extensibility

- Mining model settings for R
New mining model settings are included for R, to define the characteristics of R models. The mining model settings can be used with generic settings that are independent of algorithms, to specify R model build, score and view.
- DBMS_DATA_MINING for R
The DBMS_DATA_MINING subprograms that are independent of algorithms, can operate on R model for mining functions such as Classification, Clustering, Feature Extraction, and Regression.
See [DBMS_DATA_MINING](#) (page 2-7).
See [Specifying Mining Model Settings for R Model](#) (page 5-12).

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

The following changes are documented in *Oracle Data Mining User's Guide* for 12c Release 1 (12.1).

New Features

The following features are new in this release:

- Expanded prediction details
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 [Prediction Details](#) (page 6-4).
- Dynamic scoring
The Data Mining SQL functions now support an analytic clause for scoring data dynamically without a pre-defined model.
See [Dynamic Scoring](#) (page 6-8).
- 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.
 - Manual pre-processing of text data is no longer needed.
 - No text index must be created.
 - Additional data types are supported: CLOB, BLOB, BFILE.
 - Character data can be specified as either categorical values or text.See [Mining Unstructured Text](#) (page 7-1).
- New clustering algorithm: Expectation Maximization
See the following:
 - [Table 4-1](#) (page 4-4)
 - [Example 1-6](#) (page 1-5)
 - [Example 1-7](#) (page 1-5)
 - [Example 1-8](#) (page 1-6)
 - [Example 6-6](#) (page 6-4)
 - [About the Data Mining Sample Programs](#) (page A-1)
- New feature extraction algorithm: Singular Value Decomposition with Principal Component Analysis
See the following:
 - [Table 4-1](#) (page 4-4)

- [Example 6-7](#) (page 6-5)
- [About the Data Mining Sample Programs](#) (page A-1)
- Generalized Linear Models are enhanced to support feature selection and creation.
See [The Data Mining Sample Programs](#) (page A-1).

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 new features in this release:

- A new SQL function, `CLUSTER_DISTANCE`, is introduced. `CLUSTER_DISTANCE` returns the raw distance between each row and the cluster centroid.
See [Scoring and Deployment](#) (page 6-1).
- New support for native double data types, `BINARY_DOUBLE` and `BINARY_FLOAT`, improves the performance of the SQL scoring functions.
See [Preparing the Data](#) (page 3-1).
- Decision Tree algorithm now supports nested data.
See [Preparing the Data](#) (page 3-1).

Data Mining With SQL

Learn how to solve business problems using the Oracle Data Mining application programming interface (API).

- [Highlights of the Data Mining API](#) (page 1-1)
- [Example: Targeting Likely Candidates for a Sales Promotion](#) (page 1-2)
- [Example: Analyzing Preferred Customers](#) (page 1-3)
- [Example: Segmenting Customer Data](#) (page 1-5)
- [Example : Building an ESA Model with a Wiki Dataset](#) (page 1-6)

1.1 Highlights of the Data Mining API

Learn about the advantages of Data Mining application programming interface (API).

Data mining is a valuable technology in many application domains. It has become increasingly indispensable in the private sector as a tool for optimizing operations and maintaining a competitive edge. Data mining also has critical applications in the public sector and in scientific research. However, the complexities of data mining application development and the complexities inherent in managing and securing large stores of data can limit the adoption of data mining technology.

Oracle Data Mining is uniquely suited to addressing these challenges. The data mining engine is implemented in the Database kernel, and the robust administrative features of Oracle Database are available for managing and securing the data. While supporting a full range of data mining algorithms and procedures, the API also has features that simplify the development of data mining applications.

The Oracle Data Mining API consists of extensions to Oracle SQL, the native language of the Database. The API offers the following advantages:

- Scoring in the context of SQL queries. Scoring can be performed dynamically or by applying data mining models.
- Automatic Data Preparation (ADP) and embedded transformations.
- Model transparency. Algorithm-specific queries return details about the attributes that were used to create the model.
- Scoring transparency. Details about the prediction, clustering, or feature extraction operation can be returned with the score.
- Simple routines for predictive analytics.
- A workflow-based graphical user interface (GUI) within Oracle SQL Developer. You can download SQL Developer free of charge from the following site:

<http://www.oracle.com/pls/topic/lookup?ctx=db122&id=datminGUI>

Note:

A set of sample data mining programs ship with Oracle Database. The examples in this manual are taken from these samples.

Related Topics:

[The Data Mining Sample Programs](#) (page A-1)

Describes the data mining sample programs that ship with Oracle Database.

Oracle Data Mining Concepts

1.2 Example: Targeting Likely Candidates for a Sales Promotion

This example targets customers in Brazil for a special promotion that offers coupons and an affinity card.

The query uses data on marital status, education, and income to predict the customers who are most likely to take advantage of the incentives. The query applies a decision tree model called `dt_sh_clas_sample` to score the customer data.

Example 1-1 Predict Best Candidates for an Affinity Card

```
SELECT cust_id
FROM mining_data_apply_v
WHERE
    PREDICTION(dt_sh_clas_sample
                USING cust_marital_status, education, cust_income_level ) = 1
AND country_name IN 'Brazil';

CUST_ID
-----
100404
100607
101113
```

The same query, but with a bias to favor false positives over false negatives, is shown here.

```
SELECT cust_id
FROM mining_data_apply_v
WHERE
    PREDICTION(dt_sh_clas_sample COST MODEL
                USING cust_marital_status, education, cust_income_level ) = 1
AND country_name IN 'Brazil';

CUST_ID
-----
100139
100163
100275
100404
100607
101113
```

```
101170
101463
```

The `COST MODEL` keywords cause the cost matrix associated with the model to be used in making the prediction. The cost matrix, stored in a table called `dt_sh_sample_costs`, specifies that a false negative is eight times more costly than a false positive. Overlooking a likely candidate for the promotion is far more costly than including an unlikely candidate.

```
SELECT * FROM dt_sh_sample_cost;
```

ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	COST
0	0	0
0	1	1
1	0	8
1	1	0

1.3 Example: Analyzing Preferred Customers

The examples in this section reveal information about customers who use affinity cards or are likely to use affinity cards.

Example 1-2 Find Demographic Information About Preferred Customers

This query returns the gender, age, and length of residence of typical affinity card holders. The anomaly detection model, `SVMO_SH_Clas_sample`, returns 1 for typical cases and 0 for anomalies. The demographics are predicted for typical customers only; outliers are not included in the sample.

```
SELECT cust_gender, round(avg(age)) age,
       round(avg(yrs_residence)) yrs_residence,
       count(*) cnt
FROM mining_data_one_class_v
WHERE PREDICTION(SVMO_SH_Clas_sample using *) = 1
GROUP BY cust_gender
ORDER BY cust_gender;
```

CUST_GENDER	AGE	YRS_RESIDENCE	CNT
F	40	4	36
M	45	5	304

Example 1-3 Dynamically Identify Customers Who Resemble Preferred Customers

This query identifies customers who do not currently have an affinity card, but who share many of the characteristics of affinity card holders. The `PREDICTION` and `PREDICTION_PROBABILITY` functions use an `OVER` clause instead of a predefined model to classify the customers. The predictions and probabilities are computed dynamically.

```
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
.
.
.
.
102580      .52
102269      .52
102533      .51
101604      .51
101656      .51

```

226 rows selected.

Example 1-4 Predict the Likelihood that a New Customer Becomes a Preferred Customer

This query computes the probability of a first-time customer becoming a preferred customer (an affinity card holder). This query can be executed in real time at the point of sale.

The new customer is a 44-year-old American executive who has a bachelors degree and earns more than \$300,000/year. He is married, lives in a household of 3, and has lived in the same residence for the past 6 years. The probability of this customer becoming a typical affinity card holder is only 5.8%.

```

SELECT PREDICTION_PROBABILITY(SVMO_SH_Clas_sample, 1 USING
                                44 AS age,
                                6 AS yrs_residence,
                                'Bach.' AS education,
                                'Married' AS cust_marital_status,
                                'Exec.' AS occupation,
                                'United States of America' AS country_name,
                                'M' AS cust_gender,
                                'L: 300,000 and above' AS cust_income_level,
                                '3' AS household_size
                                ) prob_typical

FROM DUAL;

PROB_TYPICAL
-----
5.8

```

Example 1-5 Use Predictive Analytics to Find Top Predictors

The DBMS_PREDICTIVE_ANALYTICS PL/SQL package contains routines that perform simple data mining operations without a predefined model. In this example, the EXPLAIN routine computes the top predictors for affinity card ownership. The results show that household size, marital status, and age are the top three predictors.

```

BEGIN
  DBMS_PREDICTIVE_ANALYTICS.EXPLAIN(
    data_table_name      => 'mining_data_test_v',
    explain_column_name => 'affinity_card',
    result_table_name   => 'cust_explain_result');
END;

```

```

/

SELECT * FROM cust_explain_result
  WHERE rank < 4;

ATTRIBUTE_NAME          ATTRIBUTE_SUBNAME    EXPLANATORY_VALUE    RANK
-----
HOUSEHOLD_SIZE          .209628541          1
CUST_MARITAL_STATUS     .199794636          2
AGE                      .111683067          3

```

1.4 Example: Segmenting Customer Data

The examples in this section use an Expectation Maximization clustering model to segment the customer data based on common characteristics.

Example 1-6 Compute Customer Segments

This query computes natural groupings of customers and returns the number of customers in each group.

```

SELECT CLUSTER_ID(em_sh_clus_sample USING *) AS clus, COUNT(*) AS cnt
  FROM mining_data_apply_v
 GROUP BY CLUSTER_ID(em_sh_clus_sample USING *)
 ORDER BY cnt DESC;

```

CLUS	CNT
9	311
3	294
7	215
12	201
17	123
16	114
14	86
19	64
15	56
18	36

Example 1-7 Find the Customers Who Are Most Likely To Be in the Largest Segment

The query in [Example 1-6](#) (page 1-5) shows that segment 9 has the most members. The following query lists the five customers who are most likely to be in segment 9.

```

SELECT cust_id
FROM (SELECT cust_id, RANK() over (ORDER BY prob DESC, cust_id) rnk_clus2
      FROM (SELECT cust_id,
                  ROUND(CLUSTER_PROBABILITY(em_sh_clus_sample, 9 USING *),3) prob
            FROM mining_data_apply_v))
WHERE rnk_clus2 <= 5
ORDER BY rnk_clus2;

```

CUST_ID
100002
100012
100016
100019
100021

Example 1-8 Find Key Characteristics of the Most Representative Customer in the Largest Cluster

The query in [Example 1-7](#) (page 1-5) lists customer 100002 first in the list of likely customers for segment 9. The following query returns the five characteristics that are most significant in determining the assignment of customer 100002 to segments with probability > 20% (only segment 9 for this customer).

```
SELECT S.cluster_id, probability prob,
       CLUSTER_DETAILS(em_sh_clus_sample, S.cluster_id, 5 using T.*) det
FROM
  (SELECT v.*, CLUSTER_SET(em_sh_clus_sample, NULL, 0.2 USING *) pset
   FROM mining_data_apply_v v
   WHERE cust_id = 100002) T,
TABLE(T.pset) S
ORDER BY 2 desc;
```

CLUSTER_ID	PROB	DET
9	1.0000	<Details algorithm="Expectation Maximization" cluster="9"> <Attribute name="YRS_RESIDENCE" actualValue="4" weight="1" rank="1"/> <Attribute name="EDUCATION" actualValue="Bach." weight="0" rank="2"/> <Attribute name="AFFINITY_CARD" actualValue="0" weight="0" rank="3"/> <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight="0" rank="4"/> <Attribute name="Y_BOX_GAMES" actualValue="0" weight="0" rank="5"/> </Details>

1.5 Example : Building an ESA Model with a Wiki Dataset

The examples shows FEATURE_COMPARE function with Explicit Semantic Analysis (ESA) model, which compares a similar set of texts and then a dissimilar set of texts.

The example shows an ESA model built against a 2005 Wiki dataset rendering over 200,000 features. The documents are mined as text and the document titles are given as the feature IDs.

Similar Texts

```
SELECT 1-FEATURE_COMPARE(esa_wiki_mod USING 'There are several PGA tour golfers from
South Africa' text AND USING 'Nick Price won the 2002 Mastercard Colonial Open'
text) similarity FROM DUAL;
```

```
SIMILARITY
-----
      .258
```

The output metric shows distance calculation. Therefore, smaller number represent more similar texts. So, 1 minus the distance in the queries result in similarity.

Dissimilar Texts

```
SELECT 1-FEATURE_COMPARE(esa_wiki_mod USING 'There are several PGA tour golfers from
South Africa' text AND USING 'John Elway played quarterback for the Denver Broncos'
text) similarity FROM DUAL;
```

```
SIMILARITY
-----
      .007
```

About the Data Mining API

Overview of the Oracle Data Mining application programming interface (API) components.

- [About Mining Models](#) (page 2-1)
- [Data Mining Data Dictionary Views](#) (page 2-2)
- [Data Mining PL/SQL Packages](#) (page 2-7)
- [Data Mining SQL Scoring Functions](#) (page 2-9)

2.1 About Mining Models

Mining models are database schema objects that perform data mining.

As with all schema objects, access to mining models is controlled by database privileges. Models can be exported and imported. They support comments, and they can be tracked in the Database auditing system.

Mining models are created by the `CREATE_MODEL` procedure in the `DBMS_DATA_MINING` PL/SQL package. Models are created for a specific mining function, and they use a specific algorithm to perform that function. **Mining function** is a data mining term that refers to a class of mining problems to be solved. Examples of mining functions are: regression, classification, attribute importance, clustering, anomaly detection, and feature extraction. Oracle Data Mining supports one or more algorithms for each mining function.

Note:

Most types of mining models can be used to score data. However, it is possible to score data without applying a model. Dynamic scoring and predictive analytics return scoring results without a user-supplied model. They create and apply transient models that are not visible to you.

Related Topics:

[Dynamic Scoring](#) (page 6-8)

[DBMS_PREDICTIVE_ANALYTICS](#) (page 2-9)

Understand the routines of `DBMS_PREDICTIVE_ANALYTICS` package.

[Creating a Model](#) (page 5-1)

Explains how to create data mining models and query model details.

[Administrative Tasks for Oracle Data Mining](#) (page 8-1)

Explains how to perform administrative tasks related to Oracle Data Mining.

2.2 Data Mining Data Dictionary Views

Lists Oracle Data Mining data dictionary views.

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

Table 2-1 Data Dictionary Views for Oracle Data Mining

View Name	Description
ALL_MINING_MODELS	Provides information about all accessible mining models
ALL_MINING_MODEL_ATTRIBUTES	Provides information about the attributes of all accessible mining models
ALL_MINING_MODEL_PARTITIONS	Provides information about the partitions of all accessible partitioned mining models
ALL_MINING_MODEL_SETTINGS	Provides information about the configuration settings for all accessible mining models
ALL_MINING_MODEL_VIEWS	Provides information about the model views for all accessible mining models
ALL_MINING_MODEL_TRANSFORMS	Provides the user-specified transformations embedded in all accessible mining models.

2.2.1 ALL_MINING_MODELS

Describes an example of ALL_MINING_MODELS and shows a sample query.

The following example describes ALL_MINING_MODELS and shows a sample query.

Example 2-1 ALL_MINING_MODELS

```
describe ALL_MINING_MODELS
Name                                     Null?   Type
-----
OWNER                                     NOT NULL VARCHAR2(128)
MODEL_NAME                               NOT NULL VARCHAR2(128)
MINING_FUNCTION                           VARCHAR2(30)
ALGORITHM                                 VARCHAR2(30)
CREATION_DATE                             NOT NULL DATE
BUILD_DURATION                             NUMBER
MODEL_SIZE                                 NUMBER
PARTITIONED                               VARCHAR2(3)
COMMENTS                                   VARCHAR2(4000)
```

The following query returns the models accessible to you that use the Support Vector Machine algorithm.

```
SELECT mining_function, model_name
       FROM all_mining_models
       WHERE algorithm = 'SUPPORT_VECTOR_MACHINES'
       ORDER BY mining_function, model_name;
```

```
MINING_FUNCTION      MODEL_NAME
-----
CLASSIFICATION      PART2_CLAS_SAMPLE
CLASSIFICATION      PART_CLAS_SAMPLE
```

CLASSIFICATION	SVCM_SH_CLAS_SAMPLE
CLASSIFICATION	SVMO_SH_CLAS_SAMPLE
CLASSIFICATION	T_SVM_CLAS_SAMPLE
REGRESSION	SVMR_SH_REGR_SAMPLE

Related Topics:

[Creating a Model](#) (page 5-1)

Explains how to create data mining models and query model details.

Oracle Database Reference

2.2.2 ALL_MINING_MODEL_ATTRIBUTES

Describes an example of ALL_MINING_MODEL_ATTRIBUTES and shows a sample query.

The following example describes ALL_MINING_MODEL_ATTRIBUTES and shows a sample query. Attributes are the predictors or conditions that are used to create models and score data.

Example 2-2 ALL_MINING_MODEL_ATTRIBUTES

```
describe ALL_MINING_MODEL_ATTRIBUTES
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(128)
MODEL_NAME	NOT NULL	VARCHAR2(128)
ATTRIBUTE_NAME	NOT NULL	VARCHAR2(128)
ATTRIBUTE_TYPE		VARCHAR2(11)
DATA_TYPE		VARCHAR2(106)
DATA_LENGTH		NUMBER
DATA_PRECISION		NUMBER
DATA_SCALE		NUMBER
USAGE_TYPE		VARCHAR2(8)
TARGET		VARCHAR2(3)
ATTRIBUTE_SPEC		VARCHAR2(4000)

The following query returns the attributes of an SVM classification model named T_SVM_CLAS_SAMPLE. The model has both categorical and numerical attributes and includes one attribute that is unstructured text.

```
SELECT attribute_name, attribute_type, target
FROM all_mining_model_attributes
WHERE model_name = 'T_SVM_CLAS_SAMPLE'
ORDER BY attribute_name;
```

ATTRIBUTE_NAME	ATTRIBUTE_TYPE	TAR
AFFINITY_CARD	CATEGORICAL	YES
AGE	NUMERICAL	NO
BOOKKEEPING_APPLICATION	NUMERICAL	NO
BULK_PACK_DISKETTES	NUMERICAL	NO
COMMENTS	TEXT	NO
COUNTRY_NAME	CATEGORICAL	NO
CUST_GENDER	CATEGORICAL	NO
CUST_INCOME_LEVEL	CATEGORICAL	NO
CUST_MARITAL_STATUS	CATEGORICAL	NO
EDUCATION	CATEGORICAL	NO
FLAT_PANEL_MONITOR	NUMERICAL	NO
HOME_THEATER_PACKAGE	NUMERICAL	NO

HOUSEHOLD_SIZE	CATEGORICAL	NO
OCCUPATION	CATEGORICAL	NO
OS_DOC_SET_KANJI	NUMERICAL	NO
PRINTER_SUPPLIES	NUMERICAL	NO
YRS_RESIDENCE	NUMERICAL	NO
Y_BOX_GAMES	NUMERICAL	NO

Related Topics:

[About the Data Mining API](#) (page 2-1)

Overview of the Oracle Data Mining application programming interface (API) components.

Oracle Database Reference

2.2.3 ALL_MINING_MODEL_PARTITIONS

Describes an example of ALL_MINING_MODEL_PARTITIONS and shows a sample query.

The following example describes ALL_MINING_MODEL_PARTITIONS and shows a sample query.

Example 2-3 ALL_MINING_MODEL_PARTITIONS

```
describe ALL_MINING_MODEL_PARTITIONS
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(128)
MODEL_NAME	NOT NULL	VARCHAR2(128)
PARTITION_NAME		VARCHAR2(128)
POSITION		NUMBER
COLUMN_NAME	NOT NULL	VARCHAR2(128)
COLUMN_VALUE		VARCHAR2(4000)

The following query returns the partition names and partition key values for two partitioned models. Model PART2_CLAS_SAMPLE has a two column partition key with system-generated partition names.

```
SELECT model_name, partition_name, position, column_name, column_value
FROM all_mining_model_partitions
ORDER BY model_name, partition_name, position;
```

MODEL_NAME	PARTITION_	POSITION	COLUMN_NAME	COLUMN_VALUE
PART2_CLAS_SAMPLE	DM\$\$_P0	1	CUST_GENDER	F
PART2_CLAS_SAMPLE	DM\$\$_P0	2	CUST_INCOME_LEVEL	HIGH
PART2_CLAS_SAMPLE	DM\$\$_P1	1	CUST_GENDER	F
PART2_CLAS_SAMPLE	DM\$\$_P1	2	CUST_INCOME_LEVEL	LOW
PART2_CLAS_SAMPLE	DM\$\$_P2	1	CUST_GENDER	F
PART2_CLAS_SAMPLE	DM\$\$_P2	2	CUST_INCOME_LEVEL	MEDIUM
PART2_CLAS_SAMPLE	DM\$\$_P3	1	CUST_GENDER	M
PART2_CLAS_SAMPLE	DM\$\$_P3	2	CUST_INCOME_LEVEL	HIGH
PART2_CLAS_SAMPLE	DM\$\$_P4	1	CUST_GENDER	M
PART2_CLAS_SAMPLE	DM\$\$_P4	2	CUST_INCOME_LEVEL	LOW
PART2_CLAS_SAMPLE	DM\$\$_P5	1	CUST_GENDER	M
PART2_CLAS_SAMPLE	DM\$\$_P5	2	CUST_INCOME_LEVEL	MEDIUM
PART_CLAS_SAMPLE	F	1	CUST_GENDER	F
PART_CLAS_SAMPLE	M	1	CUST_GENDER	M
PART_CLAS_SAMPLE	U	1	CUST_GENDER	U

Related Topics:*Oracle Database Reference***2.2.4 ALL_MINING_MODEL_SETTINGS**

Describes an example of ALL_MINING_MODEL_SETTINGS and shows a sample query.

The following example describes ALL_MINING_MODEL_SETTINGS and shows a sample query. Settings influence model behavior. Settings may be specific to an algorithm or to a mining function, or they may be general.

Example 2-4 ALL_MINING_MODEL_SETTINGS

```
describe ALL_MINING_MODEL_SETTINGS
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(128)
MODEL_NAME	NOT NULL	VARCHAR2(128)
SETTING_NAME	NOT NULL	VARCHAR2(30)
SETTING_VALUE		VARCHAR2(4000)
SETTING_TYPE		VARCHAR2(7)

The following query returns the settings for a model named SVD_SH_SAMPLE. The model uses the Singular Value Decomposition algorithm for feature extraction.

```
SELECT setting_name, setting_value, setting_type
FROM all_mining_model_settings
WHERE model_name = 'SVD_SH_SAMPLE'
ORDER BY setting_name;
```

SETTING_NAME	SETTING_VALUE	SETTING
ALGO_NAME	ALGO_SINGULAR_VALUE_DECOMP	INPUT
ODMS_MISSING_VALUE_TREATMENT	ODMS_MISSING_VALUE_AUTO	DEFAULT
ODMS_SAMPLING	ODMS_SAMPLING_DISABLE	DEFAULT
PREP_AUTO	OFF	INPUT
SVDS_SCORING_MODE	SVDS_SCORING_SVD	DEFAULT
SVDS_U_MATRIX_OUTPUT	SVDS_U_MATRIX_ENABLE	INPUT

Related Topics:

[Specifying Model Settings](#) (page 5-7)

Understand how to configure data mining models at build time.

*Oracle Database Reference***2.2.5 ALL_MINING_MODEL_VIEWS**

Describes an example of ALL_MINING_MODEL_VIEWS and shows a sample query.

The following example describes ALL_MINING_MODEL_VIEWS and shows a sample query. Model views provide details on the models.

Example 2-5 ALL_MINING_MODEL_VIEWS

```
describe ALL_MINING_MODEL_VIEWS
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(128)
MODEL_NAME	NOT NULL	VARCHAR2(128)

```

VIEW_NAME                NOT NULL VARCHAR2(128)
VIEW_TYPE                VARCHAR2(128)

```

The following query returns the model views for a model SVD_SH_SAMPLE. The model uses the Singular Value Decomposition algorithm for feature extraction.

```

SELECT view_name, view_type
   FROM all_mining_model_views
  WHERE model_name = 'SVD_SH_SAMPLE'
 ORDER BY view_name;

```

```

VIEW_NAME                VIEW_TYPE
-----
DM$VESVD_SH_SAMPLE      Singular Value Decomposition S Matrix
DM$VGSVD_SH_SAMPLE      Global Name-Value Pairs
DM$VNSVD_SH_SAMPLE      Normalization and Missing Value Handling
DM$VSSVD_SH_SAMPLE      Computed Settings
DM$VUSVD_SH_SAMPLE      Singular Value Decomposition U Matrix
DM$VVSVD_SH_SAMPLE      Singular Value Decomposition V Matrix
DM$VWSVD_SH_SAMPLE      Model Build Alerts

```

Related Topics:

Oracle Database Reference

2.2.6 ALL_MINING_MODEL_XFORMS

Describes an example of ALL_MINING_MODEL_XFORMS and provides a sample query.

The following example describes ALL_MINING_MODEL_XFORMS and provides a sample query.

Example 2-6 ALL_MINING_MODEL_XFORMS

```
describe ALL_MINING_MODEL_XFORMS
```

```

Name                               Null?   Type
-----
OWNER                               NOT NULL VARCHAR2(128)
MODEL_NAME                          NOT NULL VARCHAR2(128)
ATTRIBUTE_NAME                       VARCHAR2(128)
ATTRIBUTE_SUBNAME                    VARCHAR2(4000)
ATTRIBUTE_SPEC                       VARCHAR2(4000)
EXPRESSION                           CLOB
REVERSE                             VARCHAR2(3)

```

The following query returns the embedded transformations for a model PART2_CLAS_SAMPLE.

```

SELECT attribute_name, expression
   FROM all_mining_model_xforms
  WHERE model_name = 'PART2_CLAS_SAMPLE'
 ORDER BY attribute_name;

ATTRIBUTE_NAME
-----
EXPRESSION
-----
CUST_INCOME_LEVEL
CASE CUST_INCOME_LEVEL WHEN 'A: Below 30,000' THEN 'LOW'
  WHEN 'L: 300,000 and above' THEN 'HIGH'
  ELSE 'MEDIUM' END

```

Related Topics:*Oracle Database Reference*

2.3 Data Mining PL/SQL Packages

The PL/SQL interface to Oracle Data Mining is implemented in three packages.

The following table displays the PL/SQL packages.

Table 2-2 Data Mining PL/SQL Packages

Package Name	Description
DBMS_DATA_MINING	Routines for creating and managing mining models
DBMS_DATA_MINING_TRANSFORM	Routines for transforming the data for mining
DBMS_PREDICTIVE_ANALYTICS	Routines that perform predictive analytics

Related Topics:

DBMS_DATA_MINING

DBMS_DATA_MINING_TRANSFORM

DBMS_PREDICTIVE_ANALYTICS

2.3.1 DBMS_DATA_MINING

Understand the routines of DBMS_DATA_MINING package.

The DBMS_DATA_MINING package contains routines for creating mining models, for performing operations on mining models, and for querying mining models. The package includes routines for:

- Creating, dropping, and performing other DDL operations on mining models
- Obtaining detailed information about model attributes, rules, and other information internal to the model (model details)
- Computing test metrics for classification models
- Specifying costs for classification models
- Exporting and importing models
- Building models using Oracle's native algorithms as well as algorithms written in R

Related Topics:*Oracle Database PL/SQL Packages and Types Reference*

2.3.2 DBMS_DATA_MINING_TRANSFORM

Understand the routines of DBMS_DATA_MINING_TRANSFORM package.

The `DBMS_DATA_MINING_TRANSFORM` package contains routines that perform data transformations such as binning, normalization, and outlier treatment. The package includes routines for:

- Specifying transformations in a format that can be embedded in a mining model.
- Specifying transformations as relational views (external to mining model objects).
- Specifying distinct properties for columns in the build data. For example, you can specify that the column must be interpreted as unstructured text, or that the column must be excluded from Automatic Data Preparation.

Related Topics:

[Transforming the Data](#) (page 4-1)

Understand how to transform data for building a model or for scoring.

Oracle Database PL/SQL Packages and Types Reference

2.3.2.1 Transformation Methods in `DBMS_DATA_MINING_TRANSFORM`

Summarizes the methods for transforming data in `DBMS_DATA_MINING_TRANSFORM` package.

Table 2-3 `DBMS_DATA_MINING_TRANSFORM` Transformation Methods

Transformation Method	Description
XFORM interface	CREATE, INSERT, and XFORM routines specify transformations in external views
STACK interface	CREATE, INSERT, and XFORM routines specify transformations for embedding in a model
SET_TRANSFORM	Specifies transformations for embedding in a model

The statements in the following example create an Support Vector Machine (SVM) Classification model called `T_SVM_Clas_sample` with an embedded transformation that causes the comments attribute to be treated as unstructured text data.

Example 2-7 Sample Embedded Transformation

```

DECLARE
  xformlist dbms_data_mining_transform.TRANSFORM_LIST;
BEGIN
  dbms_data_mining_transform.SET_TRANSFORM(
    xformlist, 'comments', null, 'comments', null, 'TEXT');
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'T_SVM_Clas_sample',
    mining_function     => dbms_data_mining.classification,
    data_table_name    => 'mining_build_text',
    case_id_column_name => 'cust_id',
    target_column_name => 'affinity_card',
    settings_table_name => 't_svmc_sample_settings',
    xform_list => xformlist);
END;
/

```


2.3.3 DBMS_PREDICTIVE_ANALYTICS

Understand the routines of DBMS_PREDICTIVE_ANALYTICS package.

The DBMS_PREDICTIVE_ANALYTICS package contains routines that perform an automated form of data mining known as predictive analytics. With predictive analytics, you do not need to be aware of model building or scoring. All mining activities are handled internally by the procedure. The DBMS_PREDICTIVE_ANALYTICS package includes these routines:

- **EXPLAIN** ranks attributes in order of influence in explaining a target column.
- **PREDICT** predicts the value of a target column based on values in the input data.
- **PROFILE** generates rules that describe the cases from the input data.

The EXPLAIN statement in the following example lists attributes in the view mining_data_build_v in order of their importance in predicting affinity_card.

Example 2-8 Sample EXPLAIN Statement

```
BEGIN
  DBMS_PREDICTIVE_ANALYTICS.EXPLAIN(
    data_table_name      => 'mining_data_build_v',
    explain_column_name  => 'affinity_card',
    result_table_name    => 'explain_results');
END;
/
```

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

2.4 Data Mining SQL Scoring Functions

Understand the different data mining SQL scoring functions.

The Data Mining SQL language functions use Oracle Data Mining to score data. The functions can apply a mining model schema object to the data, or they can dynamically mine the data by executing an analytic clause. SQL functions are available for all the data mining algorithms that support the scoring operation. All Data Mining SQL functions, as listed in the following table can operate on R Mining Model with the corresponding mining function. However, the functions are not limited to the ones listed here.

Table 2-4 Data Mining SQL Functions

Function	Description
CLUSTER_ID	Returns the ID of the predicted cluster
CLUSTER_DETAIL S	Returns detailed information about the predicted cluster
CLUSTER_DISTAN CE	Returns the distance from the centroid of the predicted cluster
CLUSTER_PROBAB ILITY	Returns the probability of a case belonging to a given cluster

Table 2-4 (Cont.) Data Mining SQL Functions

Function	Description
CLUSTER_SET	Returns a list of all possible clusters to which a given case belongs along with the associated probability of inclusion
FEATURE_COMPAR E	Compares two similar and dissimilar set of texts from two different documents or keyword phrases or a combination of both
FEATURE_ID	Returns the ID of the feature with the highest coefficient value
FEATURE_DETAIL S	Returns detailed information about the predicted feature
FEATURE_SET	Returns a list of objects containing all possible features along with the associated coefficients
FEATURE_VALUE	Returns the value of the predicted feature
ORA_DM_PARTITI ON_NAME	Returns the partition names for a partitioned model
PREDICTION	Returns the best prediction for the target
PREDICTION_BOU NDS	(GLM only) Returns the upper and lower bounds of the interval wherein the predicted values (linear regression) or probabilities (logistic regression) lie.
PREDICTION_COS T	Returns a measure of the cost of incorrect predictions
PREDICTION_DET AILS	Returns detailed information about the prediction
PREDICTION_PRO BABILITY	Returns the probability of the prediction
PREDICTION_SET	Returns the results of a classification model, including the predictions and associated probabilities for each case

The following example shows a query that returns the results of the `CLUSTER_ID` function. The query applies the model `em_sh_clus_sample`, which finds groups of customers that share certain characteristics. The query returns the identifiers of the clusters and the number of customers in each cluster.

Example 2-9 CLUSTER_ID Function

```
-- -List the clusters into which the customers in this
-- -data set have been grouped.
--
SELECT CLUSTER_ID(em_sh_clus_sample USING *) AS clus, COUNT(*) AS cnt
  FROM mining_data_apply_v
 GROUP BY CLUSTER_ID(em_sh_clus_sample USING *)
 ORDER BY cnt DESC;

SQL> -- List the clusters into which the customers in this
SQL> -- data set have been grouped.
SQL> --
SQL> SELECT CLUSTER_ID(em_sh_clus_sample USING *) AS clus, COUNT(*) AS cnt
```

```
2 FROM mining_data_apply_v
3 GROUP BY CLUSTER_ID(em_sh_clus_sample USING *)
4 ORDER BY cnt DESC;
```

CLUS	CNT
9	311
3	294
7	215
12	201
17	123
16	114
14	86
19	64
15	56
18	36

Related Topics:

[Scoring and Deployment](#) (page 6-1)

Explains the scoring and deployment features of Oracle Data Mining.

Oracle Database SQL Language Reference

Preparing the Data

Learn how to create a table or view that can be used to build a model.

- [Data Requirements](#) (page 3-1)
- [About Attributes](#) (page 3-3)
- [Using Nested Data](#) (page 3-6)
- [Using Market Basket Data](#) (page 3-9)
- [Using Retail Analysis Data](#) (page 3-11)
- [Handling Missing Values](#) (page 3-12)

3.1 Data Requirements

Understand how data is stored and viewed for data mining.

Data mining activities require data that is defined within a single table or view. The information for each record must be stored in a separate row. The data records are commonly called **cases**. Each case can optionally be identified by a unique **case ID**. The table or view itself can be referred to as a **case table**.

The CUSTOMERS table in the SH schema is an example of a table that could be used for mining. All the information for each customer is contained in a single row. The case ID is the CUST_ID column. The rows listed in the following example are selected from SH.CUSTOMERS.

Note:

Oracle Data Mining requires single-record case data for all types of models except association models, which can be built on native transactional data.

Example 3-1 Sample Case Table

```
SQL> select cust_id, cust_gender, cust_year_of_birth,
           cust_main_phone_number from sh.customers where cust_id < 11;
```

CUST_ID	CUST_GENDER	CUST_YEAR_OF_BIRTH	CUST_MAIN_PHONE_NUMBER
1	M	1946	127-379-8954
2	F	1957	680-327-1419
3	M	1939	115-509-3391
4	M	1934	577-104-2792
5	M	1969	563-667-7731
6	F	1925	682-732-7260
7	F	1986	648-272-6181
8	F	1964	234-693-8728

9	F	1936	697-702-2618
10	F	1947	601-207-4099

Related Topics:

[Using Market Basket Data](#) (page 3-9)

3.1.1 Column Data Types

Understand the different types of column data in a case table.

The columns of the case table hold the attributes that describe each case. In [Example 3-1](#) (page 3-1), the attributes are: `CUST_GENDER`, `CUST_YEAR_OF_BIRTH`, and `CUST_MAIN_PHONE_NUMBER`. The attributes are the predictors in a supervised model or the descriptors in an unsupervised model. The case ID, `CUST_ID`, can be viewed as a special attribute; it is not a predictor or a descriptor.

Oracle Data Mining supports standard Oracle data types as well as the following collection types:

```
DM_NESTED_CATEGORICALS
DM_NESTED_NUMERICALS
DM_NESTED_BINARY_DOUBLES
DM_NESTED_BINARY_FLOATS
```

Related Topics:

[Using Nested Data](#) (page 3-6)

A join between the tables for one-to-many relationship is represented through nested columns.

[Mining Unstructured Text](#) (page 7-1)

Explains how to use Oracle Data Mining to mine unstructured text.

Oracle Database SQL Language Reference

3.1.2 Data Sets for Classification and Regression

Understand how data sets are used for training and testing the model.

You need two case tables to build and validate classification and regression models. One set of rows is used for training the model, another set of rows is used for testing the model. It is often convenient to derive the build data and test data from the same data set. For example, you could randomly select 60% of the rows for training the model; the remaining 40% could be used for testing the model.

Models that implement other mining functions, such as attribute importance, clustering, association, or feature extraction, do not use separate test data.

3.1.3 Scoring Requirements

Most data mining models can be applied to separate data in a process known as **scoring**. Oracle Data Mining supports the scoring operation for classification, regression, anomaly detection, clustering, and feature extraction.

The scoring process matches column names in the scoring data with the names of the columns that were used to build the model. The scoring process does not require all the columns to be present in the scoring data. If the data types do not match, Oracle Data Mining attempts to perform type coercion. For example, if a column called

PRODUCT_RATING is VARCHAR2 in the training data but NUMBER in the scoring data, Oracle Data Mining effectively applies a TO_CHAR () function to convert it.

The column in the test or scoring data must undergo the same transformations as the corresponding column in the build data. For example, if the AGE column in the build data was transformed from numbers to the values CHILD, ADULT, and SENIOR, then the AGE column in the scoring data must undergo the same transformation so that the model can properly evaluate it.

Note:

Oracle Data Mining can embed user-specified transformation instructions in the model and reapply them whenever the model is applied. When the transformation instructions are embedded in the model, you do not need to specify them for the test or scoring data sets.

Oracle Data Mining also supports Automatic Data Preparation (ADP). When ADP is enabled, the transformations required by the algorithm are performed automatically and embedded in the model along with any user-specified transformations.

See Also:

[Transforming the Data](#) (page 4-1) for more information on automatic and embedded data transformations

3.2 About Attributes

Attributes are the items of data that are used in data mining. In predictive models, attributes are the predictors that affect a given outcome. In descriptive models, attributes are the items of information being analyzed for natural groupings or associations. For example, a table of employee data that contains attributes such as job title, date of hire, salary, age, gender, and so on.

3.2.1 Data Attributes and Model Attributes

Data attributes are columns in the data set used to build, test, or score a model. **Model attributes** are the data representations used internally by the model.

Data attributes and model attributes can be the same. For example, a column called SIZE, with values S, M, and L, are attributes used by an algorithm to build a model. Internally, the model attribute SIZE is most likely be the same as the data attribute from which it was derived.

On the other hand, a nested column SALES_PROD, containing the sales figures for a group of products, does not correspond to a model attribute. The data attribute can be SALES_PROD, but each product with its corresponding sales figure (each row in the nested column) is a model attribute.

Transformations also cause a discrepancy between data attributes and model attributes. For example, a transformation can apply a calculation to two data attributes and store the result in a new attribute. The new attribute is a model attribute that has no corresponding data attribute. Other transformations such as binning, normalization, and outlier treatment, cause the model's representation of an attribute to be different from the data attribute in the case table.

Related Topics:

[Using Nested Data](#) (page 3-6)

A join between the tables for one-to-many relationship is represented through nested columns.

[Transforming the Data](#) (page 4-1)

Understand how to transform data for building a model or for scoring.

See Also:

3.2.2 Target Attribute

Understand what a **target** means in data mining and understand the different target data types.

The **target** of a supervised model is a special kind of attribute. The target column in the training data contains the historical values used to train the model. The target column in the test data contains the historical values to which the predictions are compared. The act of scoring produces a prediction for the target.

Clustering, Feature Extraction, Association, and Anomaly Detection models do not use a target.

Nested columns and columns of unstructured data (such as BFILE, CLOB, or BLOB) cannot be used as targets. Target attributes must have a simple data type.

Table 3-1 Target Data Types

Mining Function	Target Data Types
Classification	VARCHAR2, CHAR
	NUMBER, FLOAT
	BINARY_DOUBLE, BINARY_FLOAT
Regression	NUMBER, FLOAT
	BINARY_DOUBLE, BINARY_FLOAT

You can query the *_MINING_MODEL_ATTRIBUTES view to find the target for a given model.

Related Topics:

[ALL_MINING_MODEL_ATTRIBUTES](#) (page 2-3)

Describes an example of ALL_MINING_MODEL_ATTRIBUTES and shows a sample query.

3.2.3 Numericals, Categoricals, and Unstructured Text

Explains numeric, categorical, and unstructured text attributes.

Model attributes are numerical, categorical, or unstructured (text). Data attributes, which are columns in a case table, have Oracle data types, as described in "Column Data Types".

Numerical attributes can theoretically have an infinite number of values. The values have an implicit order, and the differences between them are also ordered. Oracle Data

Mining interprets NUMBER, FLOAT, BINARY_DOUBLE, BINARY_FLOAT, DM_NESTED_NUMERICALS, DM_NESTED_BINARY_DOUBLES, and DM_NESTED_BINARY_FLOATS as numerical.

Categorical attributes have values that identify a finite number of discrete categories or classes. There is no implicit order associated with the values. Some categoricals are binary: they have only two possible values, such as yes or no, or male or female. Other categoricals are multi-class: they have more than two values, such as small, medium, and large.

Oracle Data Mining interprets CHAR and VARCHAR2 as categorical by default, however these columns may also be identified as columns of unstructured data (text). Oracle Data Mining interprets columns of DM_NESTED_CATEGORICALS as categorical. Columns of CLOB, BLOB, and BFILE always contain unstructured data.

The target of a classification model is categorical. (If the target of a classification model is numeric, it is interpreted as categorical.) The target of a regression model is numerical. The target of an attribute importance model is either categorical or numerical.

Related Topics:

[Column Data Types](#) (page 3-2)

Understand the different types of column data in a case table.

[Mining Unstructured Text](#) (page 7-1)

Explains how to use Oracle Data Mining to mine unstructured text.

3.2.4 Model Signature

The model signature is the set of data attributes that are used to build a model. Some or all of the attributes in the signature must be present for scoring. The model accounts for any missing columns on a best-effort basis. If columns with the same names but different data types are present, the model attempts to convert the data type. If extra, unused columns are present, they are disregarded.

The model signature does not necessarily include all the columns in the build data. Algorithm-specific criteria can cause the model to ignore certain columns. Other columns can be eliminated by transformations. Only the data attributes actually used to build the model are included in the signature.

The target and case ID columns are not included in the signature.

3.2.5 Scoping of Model Attribute Name

The model attribute name consists of two parts: a column name, and a subcolumn name.

```
column_name[.subcolumn_name]
```

The `column_name` component is the name of the data attribute. It is present in all model attribute names. Nested attributes and text attributes also have a `subcolumn_name` component as shown in the following example.

Example 3-2 Model Attributes Derived from a Nested Column

The nested column SALESPROD has three rows.

```
SALESPROD(ATTRIBUTE_NAME, VALUE)
-----
((PROD1, 300),
```

```
(PROD2, 245),  
(PROD3, 679))
```

The name of the data attribute is SALESPROD. Its associated model attributes are:

```
SALESPROD.PROD1  
SALESPROD.PROD2  
SALESPROD.PROD3
```

3.2.6 Model Details

Model details reveal information about model attributes and their treatment by the algorithm. There is a separate GET_MODEL_DETAILS routine for each algorithm. Oracle recommends that users leverage the model detail views instead.

Transformation and reverse transformation expressions are associated with model attributes. Transformations are applied to the data attributes before the algorithmic processing that creates the model. Reverse transformations are applied to the model attributes after the model has been built, so that the model details are expressed in the form of the original data attributes, or as close to it as possible.

Reverse transformations support model transparency. They provide a view of the data that the algorithm is working with internally but in a format that is meaningful to a user.

Model Details for an Attribute Importance Model

The syntax of the GET_MODEL_DETAILS function for Attribute Importance models is shown as follows.

```
DBMS_DATA_MINING.GET_MODEL_DETAILS_AI (  
    model_name          VARCHAR2)  
RETURN DM_RANKED_ATTRIBUTES PIPELINED;
```

The function returns DM_RANKED_ATTRIBUTES, a virtual table. The columns are the model details. There is one row for each model attribute in the specified model. The columns are described as follows:

attribute_name	VARCHAR2(4000)
attribute_subname	VARCHAR2(4000)
importance_value	NUMBER
rank	NUMBER(38)

3.3 Using Nested Data

A join between the tables for one-to-many relationship is represented through nested columns.

Oracle Data Mining requires a case table in single-record case format, with each record in a separate row. What if some or all of your data is in multi-record case format, with each record in several rows? What if you want one attribute to represent a series or collection of values, such as a student's test scores or the products purchased by a customer?

This kind of one-to-many relationship is usually implemented as a join between tables. For example, you can join your customer table to a sales table and thus associate a list of products purchased with each customer.

Oracle Data Mining supports dimensioned data through nested columns. To include dimensioned data in your case table, create a view and cast the joined data to one of the Data Mining nested table types. Each row in the nested column consists of an

attribute name/value pair. Oracle Data Mining internally processes each nested row as a separate attribute.

Note:

O-Cluster is the only algorithm that does not support nested data.

Related Topics:

[Example: Creating a Nested Column for Market Basket Analysis](#) (page 3-10)

The example shows how to define a nested column for market basket analysis.

3.3.1 Nested Object Types

Nested tables are object data types that can be used in place of other data types.

Oracle Database supports user-defined data types that make it possible to model real-world entities as objects in the database. **Collection types** are object data types for modeling multi-valued attributes. Nested tables are collection types. Nested tables can be used anywhere that other data types can be used.

Oracle Data Mining supports the following nested object types:

```
DM_NESTED_BINARY_DOUBLES
DM_NESTED_BINARY_FLOATS
DM_NESTED_NUMERICALS
DM_NESTED_CATEGORICALS
```

Descriptions of the nested types are provided in this example.

Example 3-3 Oracle Data Mining Nested Data Types

```
describe dm_nested_binary_double
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               BINARY_DOUBLE

describe dm_nested_binary_doubles
DM_NESTED_BINARY_DOUBLES TABLE OF SYS.DM_NESTED_BINARY_DOUBLE
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               BINARY_DOUBLE

describe dm_nested_binary_float
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               BINARY_FLOAT

describe dm_nested_binary_floats
DM_NESTED_BINARY_FLOATS TABLE OF SYS.DM_NESTED_BINARY_FLOAT
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               BINARY_FLOAT
```

```

describe dm_nested_numerical
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               NUMBER

describe dm_nested_numericals
DM_NESTED_NUMERICALS TABLE OF SYS.DM_NESTED_NUMERICAL
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               NUMBER

describe dm_nested_categorical
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               VARCHAR2(4000)

describe dm_nested_categoricals
DM_NESTED_CATEGORICALS TABLE OF SYS.DM_NESTED_CATEGORICAL
Name                               Null?   Type
-----
ATTRIBUTE_NAME                     VARCHAR2(4000)
VALUE                               VARCHAR2(4000)

```

Related Topics:

Oracle Database Object-Relational Developer's Guide

3.3.2 Example: Transforming Transactional Data for Mining

[Example 3-4](#) (page 3-8) shows data from a view of a sales table. It includes sales for three of the many products sold in four regions. This data is not suitable for mining at the product level because sales for each case (product), is stored in several rows.

[Example 3-5](#) (page 3-9) shows how this data can be transformed for mining. The case ID column is PRODUCT. SALES_PER_REGION, a nested column of type DM_NESTED_NUMERICALS, is a data attribute. This table is suitable for mining at the product case level, because the information for each case is stored in a single row.

Oracle Data Mining treats each nested row as a separate model attribute, as shown in [Example 3-6](#) (page 3-9).

Note:

The presentation in this example is conceptual only. The data is not actually pivoted before being processed.

Example 3-4 Product Sales per Region in Multi-Record Case Format

PRODUCT	REGION	SALES
Prod1	NE	556432
Prod2	NE	670155
Prod3	NE	3111
.		
.		

```

Prod1      NW      90887
Prod2      NW     100999
Prod3      NW     750437
.
.
Prod1      SE      82153
Prod2      SE     57322
Prod3      SE     28938
.
.
Prod1      SW     3297551
Prod2      SW     4972019
Prod3      SW     884923
.
.

```

Example 3-5 Product Sales per Region in Single-Record Case Format

```

PRODUCT      SALES_PER_REGION
              (ATTRIBUTE_NAME, VALUE)
-----
Prod1      ('NE' ,      556432)
              ('NW' ,      90887)
              ('SE' ,      82153)
              ('SW' ,     3297551)
Prod2      ('NE' ,      670155)
              ('NW' ,     100999)
              ('SE' ,      57322)
              ('SW' ,     4972019)
Prod3      ('NE' ,       3111)
              ('NW' ,      750437)
              ('SE' ,      28938)
              ('SW' ,     884923)
.
.

```

Example 3-6 Model Attributes Derived From SALES_PER_REGION

```

PRODUCT      SALES_PER_REGION.NE      SALES_PER_REGION.NW      SALES_PER_REGION.SE      SALES_PER_REGION.SW
-----
Prod1              556432              90887              82153              3297551
Prod2              670155              100999              57322              4972019
Prod3              3111              750437              28938              884923
.
.

```

3.4 Using Market Basket Data

Market basket data identifies the items sold in a set of baskets or transactions. Oracle Data Mining provides the association mining function for market basket analysis.

Association models use the Apriori algorithm to generate association rules that describe how items tend to be purchased in groups. For example, an association rule can assert that people who buy peanut butter are 80% likely to also buy jelly.

Market basket data is usually **transactional**. In transactional data, a case is a transaction and the data for a transaction is stored in multiple rows. Oracle Data Mining association models can be built on transactional data or on single-record case data. The `ODMS_ITEM_ID_COLUMN_NAME` and `ODMS_ITEM_VALUE_COLUMN_NAME` settings specify whether the data for association rules is in transactional format.

Note:

Association models are the only type of model that can be built on native transactional data. For all other types of models, Oracle Data Mining requires that the data be presented in single-record case format.

The Apriori algorithm assumes that the data is transactional and that it has many missing values. Apriori interprets all missing values as sparse data, and it has its own native mechanisms for handling sparse data.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information on the `ODMS_ITEM_ID_COLUMN_NAME` and `ODMS_ITEM_VALUE_COLUMN_NAME` settings.

3.4.1 Example: Creating a Nested Column for Market Basket Analysis

The example shows how to define a nested column for market basket analysis.

Association models can be built on native transactional data or on nested data. The following example shows how to define a nested column for market basket analysis.

The following SQL statement transforms this data to a column of type `DM_NESTED_NUMERICALS` in a view called `SALES_TRANS_CUST_NESTED`. This view can be used as a case table for mining.

```
CREATE VIEW sales_trans_cust_nested AS
  SELECT trans_id,
         CAST(COLLECT(DM_NESTED_NUMERICAL(
           prod_name, 1))
           AS DM_NESTED_NUMERICALS) custprods
  FROM sales_trans_cust
  GROUP BY trans_id;
```

This query returns two rows from the transformed data.

```
SELECT * FROM sales_trans_cust_nested
  WHERE trans_id < 101000
     AND trans_id > 100997;

TRANS_ID  CUSTPRODS(ATTRIBUTE_NAME, VALUE)
-----  -----
100998    DM_NESTED_NUMERICALS
          (DM_NESTED_NUMERICAL('O/S Documentation Set - English', 1)
100999    DM_NESTED_NUMERICALS
          (DM_NESTED_NUMERICAL('CD-RW, High Speed Pack of 5', 1),
           DM_NESTED_NUMERICAL('External 8X CD-ROM', 1),
           DM_NESTED_NUMERICAL('SIMM- 16MB PCMCIAII card', 1))
```

Example 3-7 Convert to a Nested Column

The view `SALES_TRANS_CUST` provides a list of transaction IDs to identify each market basket and a list of the products in each basket.

```
describe sales_trans_cust
  Name                                     Null?   Type
-----
-----
```

TRANS_ID	NOT NULL NUMBER
PROD_NAME	NOT NULL VARCHAR2(50)
QUANTITY	NUMBER

Related Topics:

[Handling Missing Values](#) (page 3-12)

3.5 Using Retail Analysis Data

Retail analysis often makes use of Association Rules and Association models.

The Association Rules are enhanced to calculate aggregates along with rules or itemsets.

Related Topics:

Oracle Data Mining Concepts

3.5.1 Example: Calculating Aggregates

The following example shows the concept of Aggregates.

Calculating Aggregates for Grocery Store Data

Assume a grocery store has the following data:

Table 3-2 Grocery Store Data

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

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

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

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

For this rule, the profit for item A is $\$5.00 + \$3.00 = \$8.00$, for item B the profit is $\$3.20 + \$10.00 = \$13.20$, for consequent C, the profit is $\$12.00 + \$14.00 = \$26.00$, for the

antecedent itemset (A, B) is $\$8.00 + \$13.20 = \$21.20$. For the whole rule, the profit is $\$21.20 + \$26.00 = \$47.40$.

Related Topics:

Oracle Data Mining User's Guide

Oracle Database PL/SQL Packages and Types Reference

3.6 Handling Missing Values

Oracle Data Mining distinguishes between **sparse data** and data that contains **random missing values**. The latter means that some attribute values are unknown. Sparse data, on the other hand, contains values that are assumed to be known, although they are not represented in the data.

A typical example of sparse data is market basket data. Out of hundreds or thousands of available items, only a few are present in an individual case (the basket or transaction). All the item values are known, but they are not all included in the basket. Present values have a quantity, while the items that are not represented are sparse (with a known quantity of zero).

Oracle Data Mining interprets missing data as follows:

- Missing at random: Missing values in columns with a simple data type (not nested) are assumed to be missing at random.
- Sparse: Missing values in nested columns indicate sparsity.

3.6.1 Examples: Missing Values or Sparse Data?

The examples in this section illustrate how Oracle Data Mining identifies data as either sparse or missing at random.

3.6.1.1 Sparsity in a Sales Table

A sales table contains point-of-sale data for a group of products that are sold in several stores to different customers over a period of time. A particular customer buys only a few of the products. The products that the customer does not buy do not appear as rows in the sales table.

If you were to figure out the amount of money a customer has spent for each product, the unpurchased products have an inferred amount of zero. The value is not random or unknown; it is zero, even though no row appears in the table.

Note that the sales data is dimensioned (by product, stores, customers, and time) and are often represented as nested data for mining.

Since missing values in a nested column always indicate sparsity, you must ensure that this interpretation is appropriate for the data that you want to mine. For example, when trying to mine a multi-record case data set containing movie ratings from users of a large movie database, the missing ratings are unknown (missing at random), but Oracle Data Mining treats the data as sparse and infer a rating of zero for the missing value.

3.6.1.2 Missing Values in a Table of Customer Data

A table of customer data contains demographic data about customers. The case ID column is the customer ID. The attributes are age, education, profession, gender, house-hold size, and so on. Not all the data is available for each customer. Any

missing values are considered to be missing at random. For example, if the age of customer 1 and the profession of customer 2 are not present in the data, that information is simply unknown. It does not indicate sparsity.

Note that the customer data is not dimensioned. There is a one-to-one mapping between the case and each of its attributes. None of the attributes are nested.

3.6.2 Missing Value Treatment in Oracle Data Mining

Missing value treatment depends on the algorithm and on the nature of the data (categorical or numerical, sparse or missing at random). Missing value treatment is summarized in the following table.

Note:

Oracle Data Mining performs the same missing value treatment whether or not Automatic Data Preparation is being used.

Table 3-3 Missing Value Treatment by Algorithm

Missing Data	EM, GLM, NMF, k-Means, SVD, SVM	DT, MDL, NB, OC	Apriori
NUMERIC AL missing at random	The algorithm replaces missing numerical values with the mean. For Expectation Maximization (EM), the replacement only occurs in columns that are modeled with Gaussian distributions.	The algorithm handles missing values naturally as missing at random.	The algorithm interprets all missing data as sparse.
CATEGORI CAL missing at random	Genelized Linear Models (GLM), Non-Negative Matrix Factorization (NMF), <i>k</i> -Means, and Support Vector Machine (SVM) replaces missing categorical values with the mode. Singular Value Decomposition (SVD) does not support categorical data. EM does not replace missing categorical values. EM treats NULLs as a distinct value with its own frequency count.	The algorithm handles missing values naturally as missing random.	The algorithm interprets all missing data as sparse.

Table 3-3 (Cont.) Missing Value Treatment by Algorithm

Missing Data	EM, GLM, NMF, k-Means, SVD, SVM	DT, MDL, NB, OC	Apriori
NUMERIC AL sparse	The algorithm replaces sparse numerical data with zeros.	O-Cluster does not support nested data and therefore does not support sparse data. Decision Tree (DT), Minimum Description Length (MDL), and Naive Bayes (NB) and replace sparse numerical data with zeros.	The algorithm handles sparse data.
CATEGORI CAL sparse	All algorithms except SVD replace sparse categorical data with zero vectors. SVD does not support categorical data.	O-Cluster does not support nested data and therefore does not support sparse data. DT, MDL, and NB replace sparse categorical data with the special value DM\$SPARSE.	The algorithm handles sparse data.

3.6.3 Changing the Missing Value Treatment

Transform the missing data as sparse or missing at random.

If you want Oracle Data Mining to treat missing data as sparse instead of missing at random or missing at random instead of sparse, transform it before building the model.

If you want missing values to be treated as sparse, but Oracle Data Mining interprets them as missing at random, you can use a SQL function like `NVL` to replace the nulls with a value such as "NA". Oracle Data Mining does not perform missing value treatment when there is a specified value.

If you want missing nested attributes to be treated as missing at random, you can transform the nested rows into physical attributes in separate columns — as long as the case table stays within the 1000 column limitation imposed by the Database. Fill in all of the possible attribute names, and specify them as null. Alternatively, insert rows in the nested column for all the items that are not present and assign a value such as the mean or mode to each one.

Related Topics:

Oracle Database SQL Language Reference

Transforming the Data

Understand how to transform data for building a model or for scoring.

- [About Transformations](#) (page 4-1)
- [Preparing the Case Table](#) (page 4-2)
- [Understanding Automatic Data Preparation](#) (page 4-3)
- [Embedding Transformations in a Model](#) (page 4-5)
- [Understanding Reverse Transformations](#) (page 4-11)

4.1 About Transformations

Understand how you can transform data by using Automatic Data Preparation (ADP) and embedded data transformation.

A transformation is a SQL expression that modifies the data in one or more columns. Data must typically undergo certain transformations before it can be used to build a model. Many data mining algorithms have specific transformation requirements. Before data can be scored, it must be transformed in the same way that the training data was transformed.

Oracle Data Mining supports Automatic Data Preparation (ADP), which automatically implements the transformations required by the algorithm. The transformations are embedded in the model and automatically executed whenever the model is applied.

If additional transformations are required, you can specify them as SQL expressions and supply them as input when you create the model. These transformations are embedded in the model just as they are with ADP.

With automatic and embedded data transformation, most of the work of data preparation is handled for you. You can create a model and score multiple data sets in just a few steps:

1. Identify the columns to include in the case table.
2. Create nested columns if you want to include transactional data.
3. Write SQL expressions for any transformations not handled by ADP.
4. Create the model, supplying the SQL expressions (if specified) and identifying any columns that contain text data.
5. Ensure that some or all of the columns in the scoring data have the same name and type as the columns used to train the model.

Related Topics:

[Scoring Requirements](#) (page 3-2)

4.2 Preparing the Case Table

Understand why you have to prepare a case table.

The first step in preparing data for mining is the creation of a case table. If all the data resides in a single table and all the information for each case (record) is included in a single row (single-record case), this process is already taken care of. If the data resides in several tables, creating the data source involves the creation of a view. For the sake of simplicity, the term "case table" is used here to refer to either a table or a view.

Related Topics:

[Preparing the Data](#) (page 3-1)

Learn how to create a table or view that can be used to build a model.

4.2.1 Creating Nested Columns

Learn when to create nested columns.

When the data source includes transactional data (multi-record case), the transactions must be aggregated to the case level in nested columns. In transactional data, the information for each case is contained in multiple rows. An example is sales data in a star schema when mining at the product level. Sales is stored in many rows for a single product (the case) since the product is sold in many stores to many customers over a period of time.

See Also:

[Using Nested Data](#) (page 3-6) for information about converting transactional data to nested columns

4.2.2 Converting Column Data Types

You must convert the data type of a column if its type causes Oracle Data Mining to interpret it incorrectly. For example, zip codes identify different postal zones; they do not imply order. If the zip codes are stored in a numeric column, they are interpreted as a numeric attribute. You must convert the data type so that the column data can be used as a categorical attribute by the model. You can do this using the `TO_CHAR` function to convert the digits 1-9 and the `LPAD` function to retain the leading 0, if there is one.

```
LPAD(TO_CHAR(ZIPCODE),5,'0')
```

4.2.3 Text Transformation

You can use Oracle Data Mining to mine text. Columns of text in the case table can be mined once they have undergone the proper transformation.

The text column must be in a table, not a view. The transformation process uses several features of Oracle Text; it treats the text in each row of the table as a separate document. Each document is transformed to a set of text tokens known as **terms**, which have a numeric value and a text label. The text column is transformed to a nested column of `DM_NESTED_NUMERICALS`.

4.2.4 About Business and Domain-Sensitive Transformations

Understand why you need to transform data according to business problems.

Some transformations are dictated by the definition of the business problem. For example, you want to build a model to predict high-revenue customers. Since your revenue data for current customers is in dollars you need to define what "high-revenue" means. Using some formula that you have developed from past experience, you can recode the revenue attribute into ranges Low, Medium, and High before building the model.

Another common business transformation is the conversion of date information into elapsed time. For example, date of birth can be converted to age.

Domain knowledge can be very important in deciding how to prepare the data. For example, some algorithms produce unreliable results if the data contains values that fall far outside of the normal range. In some cases, these values represent errors or abnormalities. In others, they provide meaningful information.

Related Topics:

[Outlier Treatment](#) (page 4-4)

4.3 Understanding Automatic Data Preparation

Understand data transformation using Automatic Data Preparation (ADP).

Most algorithms require some form of data transformation. During the model build process, Oracle Data Mining can automatically perform the transformations required by the algorithm. You can choose to supplement the automatic transformations with additional transformations of your own, or you can choose to manage all the transformations yourself.

In calculating automatic transformations, Oracle Data Mining uses heuristics that address the common requirements of a given algorithm. This process results in reasonable model quality in most cases.

Binning, normalization, and outlier treatment are transformations that are commonly needed by data mining algorithms.

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

4.3.1 Binning

Binning, also called discretization, is a technique for reducing the cardinality of continuous and discrete data. Binning groups related values together in bins to reduce the number of distinct values.

Binning can improve resource utilization and model build response time dramatically without significant loss in model quality. Binning can improve model quality by strengthening the relationship between attributes.

Supervised binning is a form of intelligent binning in which important characteristics of the data are used to determine the bin boundaries. In supervised binning, the bin boundaries are identified by a single-predictor decision tree that takes into account the joint distribution with the target. Supervised binning can be used for both numerical and categorical attributes.

4.3.2 Normalization

Normalization is the most common technique for reducing the range of numerical data. Most normalization methods map the range of a single variable to another range (often 0,1).

4.3.3 Outlier Treatment

A value is considered an outlier if it deviates significantly from most other values in the column. The presence of outliers can have a skewing effect on the data and can interfere with the effectiveness of transformations such as normalization or binning.

Outlier treatment methods such as trimming or clipping can be implemented to minimize the effect of outliers.

Outliers represent problematic data, for example, a bad reading due to the abnormal condition of an instrument. However, in some cases, especially in the business arena, outliers are perfectly valid. For example, in census data, the earnings for some of the richest individuals can vary significantly from the general population. Do not treat this information as an outlier, since it is an important part of the data. You need domain knowledge to determine outlier handling.

4.3.4 How ADP Transforms the Data

The following table shows how ADP prepares the data for each algorithm.

Table 4-1 Oracle Data Mining Algorithms With ADP

Algorithm	Mining Function	Treatment by ADP
Apriori	Association Rules	ADP has no effect on association rules.
Decision Tree	Classification	ADP has no effect on Decision Tree. Data preparation is handled by the algorithm.
Expectation Maximization	Clustering	Single-column (not nested) numerical columns that are modeled with Gaussian distributions are normalized with outlier-sensitive normalization. ADP has no effect on the other types of columns.
GLM	Classification and Regression	Numerical attributes are normalized with outlier-sensitive normalization.
k-Means	Clustering	Numerical attributes are normalized with outlier-sensitive normalization.
MDL	Attribute Importance	All attributes are binned with supervised binning.
Naive Bayes	Classification	All attributes are binned with supervised binning.
NMF	Feature Extraction	Numerical attributes are normalized with outlier-sensitive normalization.
O-Cluster	Clustering	Numerical attributes are binned with a specialized form of equi-width binning, which computes the number of bins per attribute automatically. Numerical columns with all nulls or a single value are removed.

Table 4-1 (Cont.) Oracle Data Mining Algorithms With ADP

Algorithm	Mining Function	Treatment by ADP
SVD	Feature Extraction	Numerical attributes are normalized with outlier-sensitive normalization.
SVM	Classification, Anomaly Detection, and Regression	Numerical attributes are normalized with outlier-sensitive normalization.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference*
- Part III of *Oracle Data Mining Concepts* for more information about algorithm-specific data preparation

4.4 Embedding Transformations in a Model

You can specify your own transformations and embed them in a model by creating a transformation list and passing it to `DBMS_DATA_MINING.CREATE_MODEL`.

```
PROCEDURE create_model(
    model_name          IN VARCHAR2,
    mining_function     IN VARCHAR2,
    data_table_name     IN VARCHAR2,
    case_id_column_name IN VARCHAR2,
    target_column_name  IN VARCHAR2 DEFAULT NULL,
    settings_table_name IN VARCHAR2 DEFAULT NULL,
    data_schema_name    IN VARCHAR2 DEFAULT NULL,
    settings_schema_name IN VARCHAR2 DEFAULT NULL,
    xform_list          IN TRANSFORM_LIST DEFAULT NULL);
```

4.4.1 Specifying Transformation Instructions for an Attribute

Learn what is a transformation instruction for an attribute and learn about the fields in a transformation record.

A transformation list is defined as a table of transformation records. Each record (`transform_rec`) specifies the transformation instructions for an attribute.

```
TYPE transform_rec IS RECORD (
    attribute_name    VARCHAR2(30),
    attribute_subname VARCHAR2(4000),
    expression        EXPRESSION_REC,
    reverse_expression EXPRESSION_REC,
    attribute_spec    VARCHAR2(4000));
```

The fields in a transformation record are described in this table.

Table 4-2 Fields in a Transformation Record for an Attribute

Field	Description
attribute_name and attribute_subname	These fields identify the attribute, as described in "Scoping of Model Attribute Name"
expression	<p>A SQL expression for transforming the attribute. For example, this expression transforms the age attribute into two categories: child and adult:[0,19) for 'child' and [19,) for adult</p> <pre>CASE WHEN age < 19 THEN 'child' ELSE 'adult'</pre> <p>Expression and reverse expressions are stored in <code>expression_rec</code> objects. See "Expression Records" for details.</p>
reverse_expression	<p>A SQL expression for reversing the transformation. For example, this expression reverses the transformation of the age attribute:</p> <pre>DECODE(age, 'child', '(-Inf,19)', '[19,Inf)')</pre>
attribute_spec	<p>Specifies special treatment for the attribute. The <code>attribute_spec</code> field can be null or it can have one or more of these values:</p> <ul style="list-style-type: none"> • FORCE_IN — For GLM, forces the inclusion of the attribute in the model build when the <code>ftr_selection_enable</code> setting is enabled. (<code>ftr_selection_enable</code> is disabled by default.) If the model is not using GLM, this value has no effect. FORCE_IN cannot be specified for nested attributes or text. • NOPREP — When ADP is on, prevents automatic transformation of the attribute. If ADP is not on, this value has no effect. You can specify NOPREP for a nested attribute, but not for an individual subname (row) in the nested attribute. • TEXT — Indicates that the attribute contains unstructured text. ADP has no effect on this setting. TEXT may optionally include subsettings <code>POLICY_NAME</code>, <code>TOKEN_TYPE</code>, and <code>MAX_FEATURES</code>. See Example 4-1 (page 4-7) and Example 4-2 (page 4-7).

Related Topics:

[Scoping of Model Attribute Name](#) (page 3-5)

[Expression Records](#) (page 4-6)

4.4.1.1 Expression Records

The transformation expressions in a transformation record are `expression_rec` objects.

```
TYPE expression_rec IS RECORD (
  lstmt      DBMS_SQL.VARCHAR2A,
  lb         BINARY_INTEGER DEFAULT 1,
  ub         BINARY_INTEGER DEFAULT 0);
```



```
TYPE varchar2a IS TABLE OF VARCHAR2(32767)
INDEX BY BINARY_INTEGER;
```

The `lstmt` field stores a `VARCHAR2A`, which allows transformation expressions to be very long, as they can be broken up across multiple rows of `VARCHAR2`. Use the `DBMS_DATA_MINING_TRANSFORM.SET_EXPRESSION` procedure to create an `expression_rec`.

4.4.1.2 Attribute Specifications

Learn how to define the characteristics specific to an attribute through attribute specification.

The attribute specification in a transformation record defines characteristics that are specific to this attribute. If not null, the attribute specification can include values `FORCE_IN`, `NOPREP`, or `TEXT`, as described in [Table 4-2](#) (page 4-6).

Example 4-1 An Attribute Specification with Multiple Keywords

If more than one attribute specification keyword is applicable, you can provide them in a comma-delimited list. The following expression is the specification for an attribute in a GLM model. Assuming that the `ftr_selection_enable` setting is enabled, this expression forces the attribute to be included in the model. If ADP is on, automatic transformation of the attribute is not performed.

```
"FORCE_IN,NOPREP"
```

Example 4-2 A Text Attribute Specification

For text attributes, you can optionally specify subsettings `POLICY_NAME`, `TOKEN_TYPE`, and `MAX_FEATURES`. The subsettings provide configuration information that is specific to text transformation. In this example, the transformation instructions for the text content are defined in a text policy named `my_policy` with token type is `THEME`. The maximum number of extracted features is 3000.

```
"TEXT(POLICY_NAME:my_policy)(TOKEN_TYPE:THEME)(MAX_FEATURES:3000)"
```

Related Topics:

[Configuring a Text Attribute](#) (page 7-5)

Learn how to identify a column as a text attribute and provide transformation instructions for any text attribute.

4.4.2 Building a Transformation List

A transformation list is a collection of transformation records. When a new transformation record is added, it is appended to the top of the transformation list. You can use any of the following methods to build a transformation list:

- The `SET_TRANSFORM` procedure in `DBMS_DATA_MINING_TRANSFORM`
- The `STACK` interface in `DBMS_DATA_MINING_TRANSFORM`
- The `GET_MODEL_TRANSFORMATIONS` and `GET_TRANSFORM_LIST` functions in `DBMS_DATA_MINING`

4.4.2.1 SET_TRANSFORM

The `SET_TRANSFORM` procedure adds a single transformation record to a transformation list.

```

DBMS_DATA_MINING_TRANSFORM.SET_TRANSFORM (
    xform_list          IN OUT NOCOPY TRANSFORM_LIST,
    attribute_name      VARCHAR2,
    attribute_subname   VARCHAR2,
    expression          VARCHAR2,
    reverse_expression  VARCHAR2,
    attribute_spec      VARCHAR2 DEFAULT NULL);

```

SQL expressions that you specify with `SET_TRANSFORM` must fit within a `VARCHAR2`. To specify a longer expression, you can use the `SET_EXPRESSION` procedure, which builds an expression by appending rows to a `VARCHAR2` array.

4.4.2.2 The STACK Interface

The `STACK` interface creates transformation records from a table of transformation instructions and adds them to a transformation list.

The `STACK` interface specifies that all or some of the attributes of a given type must be transformed in the same way. For example, `STACK_BIN_CAT` appends binning instructions for categorical attributes to a transformation list. The `STACK` interface consists of three steps:

1. A `CREATE` procedure creates a transformation definition table. For example, `CREATE_BIN_CAT` creates a table to hold categorical binning instructions. The table has columns for storing the name of the attribute, the value of the attribute, and the bin assignment for the value.
2. An `INSERT` procedure computes the bin boundaries for one or more attributes and populates the definition table. For example, `INSERT_BIN_CAT_FREQ` performs frequency-based binning on some or all of the categorical attributes in the data source and populates a table created by `CREATE_BIN_CAT`.
3. A `STACK` procedure creates transformation records from the information in the definition table and appends the transformation records to a transformation list. For example, `STACK_BIN_CAT` creates transformation records for the information stored in a categorical binning definition table and appends the transformation records to a transformation list.

4.4.2.3 GET_MODEL_TRANSFORMATIONS and GET_TRANSFORM_LIST

Use the functions to create a new transformation list.

These two functions can be used to create a new transformation list from the transformations embedded in an existing model.

The `GET_MODEL_TRANSFORMATIONS` function returns a list of embedded transformations.

```

DBMS_DATA_MINING.GET_MODEL_TRANSFORMATIONS (
    model_name      IN VARCHAR2)
RETURN DM_TRANSFORMS PIPELINED;

```

`GET_MODEL_TRANSFORMATIONS` returns a table of `dm_transform` objects. Each `dm_transform` has these fields

```

attribute_name      VARCHAR2(4000)
attribute_subname   VARCHAR2(4000)
expression          CLOB
reverse_expression  CLOB

```

The components of a transformation list are `transform_rec`, not `dm_transform`. The fields of a `transform_rec` are described in [Table 4-2](#) (page 4-6). You can call `GET_MODEL_TRANSFORMATIONS` to convert a list of `dm_transform` objects to `transform_rec` objects and append each `transform_rec` to a transformation list.

```
DBMS_DATA_MINING.GET_TRANSFORM_LIST (
    xform_list          OUT NOCOPY TRANSFORM_LIST,
    model_xforms       IN  DM_TRANSFORMS);
```

See Also:

"DBMS_DATA_MINING_TRANSFORM Operational Notes",
 "SET_TRANSFORM Procedure", "CREATE_MODEL Procedure", and
 "GET_MODEL_TRANSFORMATIONS Function" in *Oracle Database PL/SQL
 Packages and Types Reference*

4.4.3 Transformation Lists and Automatic Data Preparation

If you enable ADP and you specify a transformation list, the transformation list is embedded with the automatic, system-generated transformations. The transformation list is executed before the automatic transformations.

If you enable ADP and do not specify a transformation list, only the automatic transformations are embedded in the model.

If ADP is disabled (the default) and you specify a transformation list, your custom transformations are embedded in the model. No automatic transformations are performed.

If ADP is disabled (the default) and you do not specify a transformation list, no transformations is embedded in the model. You have to transform the training, test, and scoring data sets yourself if necessary. You must take care to apply the same transformations to each data set.

4.4.4 Oracle Data Mining Transformation Routines

Learn about transformation routines.

Oracle Data Mining provides routines that implement various transformation techniques in the `DBMS_DATA_MINING_TRANSFORM` package.

Related Topics:

Oracle Database SQL Language Reference

4.4.4.1 Binning Routines

Explains Binning techniques in Oracle Data Mining.

A number of factors go into deciding a binning strategy. Having fewer values typically leads to a more compact model and one that builds faster, but it can also lead to some loss in accuracy.

Model quality can improve significantly with well-chosen bin boundaries. For example, an appropriate way to bin ages is to separate them into groups of interest, such as children 0-13, teenagers 13-19, youth 19-24, working adults 24-35, and so on.

The following table lists the binning techniques provided by Oracle Data Mining:

Table 4-3 Binning Methods in DBMS_DATA_MINING_TRANSFORM

Binning Method	Description
Top-N Most Frequent Items	You can use this technique to bin categorical attributes. You specify the number of bins. The value that occurs most frequently is labeled as the first bin, the value that appears with the next frequency is labeled as the second bin, and so on. All remaining values are in an additional bin.
Supervised Binning	Supervised binning is a form of intelligent binning, where 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. It can be used for numerical or categorical attributes.
Equi-Width Binning	You can use equi-width binning for numerical attributes. The range of values is computed by subtracting the minimum value from the maximum value, then the range of values is divided into equal intervals. You can specify the number of bins or it can be calculated automatically. Equi-width binning must usually be used with outlier treatment.
Quantile Binning	Quantile binning is a numerical binning technique. Quantiles are computed using the SQL analytic function <code>NTILE</code> . The bin boundaries are based on the minimum values for each quantile. Bins with equal left and right boundaries are collapsed, possibly resulting in fewer bins than requested.

Related Topics:

[Routines for Outlier Treatment](#) (page 4-11)

4.4.4.2 Normalization Routines

Learn about Normalization routines in Oracle Data Mining.

Most normalization methods map the range of a single attribute to another range, typically 0 to 1 or -1 to +1.

Normalization is very sensitive to outliers. Without outlier treatment, most values are mapped to a tiny range, resulting in a significant loss of information.

Table 4-4 Normalization Methods in DBMS_DATA_MINING_TRANSFORM

Transformation	Description
Min-Max Normalization	This technique computes the normalization of an attribute using the minimum and maximum values. The shift is the minimum value, and the scale is the difference between the maximum and minimum values.
Scale Normalization	This normalization technique also uses the minimum and maximum values. For scale normalization, $\text{shift} = 0$, and $\text{scale} = \max\{\text{abs}(\text{max}), \text{abs}(\text{min})\}$.
Z-Score Normalization	This technique computes the normalization of an attribute using the mean and the standard deviation. Shift is the mean, and scale is the standard deviation.

Related Topics:

[Routines for Outlier Treatment](#) (page 4-11)

4.4.4.3 Routines for Outlier Treatment

Outliers are extreme values, typically several standard deviations from the mean. To minimize the effect of outliers, you can Winsorize or trim the data.

Winsorizing involves setting the tail values of an attribute to some specified value. For example, for a 90% Winsorization, the bottom 5% of values are set equal to the minimum value in the 5th percentile, while the upper 5% of values are set equal to the maximum value in the 95th percentile.

Trimming sets the tail values to NULL. The algorithm treats them as missing values.

Outliers affect the different algorithms in different ways. In general, outliers cause distortion with equi-width binning and min-max normalization.

Table 4-5 Outlier Treatment Methods in DBMS_DATA_MINING_TRANSFORM

Transformation	Description
Trimming	This technique trims the outliers in numeric columns by sorting the non-null values, computing the tail values based on some fraction, and replacing the tail values with nulls.
Winsorizing	This technique trims the outliers in numeric columns by sorting the non-null values, computing the tail values based on some fraction, and replacing the tail values with some specified value.

4.5 Understanding Reverse Transformations

Understand why you need reverse transformations.

Reverse transformations ensure that information returned by the model is expressed in a format that is similar to or the same as the format of the data that was used to train the model. Internal transformations are reversed in the model details and in the results of scoring.

Some of the attributes used by the model correspond to columns in the build data. However, because of logic specific to the algorithm, nested data, and transformations, some attributes do not correspond to columns.

For example, a nested column in the training data is not interpreted as an attribute by the model. During the model build, Oracle Data Mining explodes nested columns, and each row (an attribute name/value pair) becomes an attribute.

Some algorithms, for example Support Vector Machines (SVM) and Generalized Linear Models (GLM), only operate on numeric attributes. Any non-numeric column in the build data is exploded into binary attributes, one for each distinct value in the column (SVM). GLM does not generate a new attribute for the most frequent value in the original column. These binary attributes are set to one only if the column value for the case is equal to the value associated with the binary attribute.

Algorithms that generate coefficients present challenges in regards to interpretability of results. Examples are SVM and Non-Negative Matrix Factorization (NMF). These algorithms produce coefficients that are used in combination with the transformed attributes. The coefficients are relevant to the data on the transformed scale, not the original data scale.

For all these reasons, the attributes listed in the model details do not resemble the columns of data used to train the model. However, attributes that undergo embedded transformations, whether initiated by Automatic Data Preparation (ADP) or by a user-specified transformation list, appear in the model details in their pre-transformed state, as close as possible to the original column values. Although the attributes are transformed when they are used by the model, they are visible in the model details in a form that can be interpreted by a user.

Related Topics:

[ALTER_REVERSE_EXPRESSION Procedure](#)

[GET_MODEL_TRANSFORMATIONS Function](#)

[Model Detail Views](#) (page 5-19)

The `GET_*` interfaces are replaced by model views, and Oracle recommends that users leverage the views instead.

Creating a Model

Explains how to create data mining models and query model details.

- [Before Creating a Model](#) (page 5-1)
- [The CREATE_MODEL Procedure](#) (page 5-2)
- [Specifying Model Settings](#) (page 5-7)
- [Model Detail Views](#) (page 5-19)

5.1 Before Creating a Model

Explains the preparation steps before creating a model.

Models are database schema objects that perform data mining. The DBMS_DATA_MINING PL/SQL package is the API for creating, configuring, evaluating, and querying mining models (model details).

Before you create a model, you must decide what you want the model to do. You must identify the training data and determine if transformations are required. You can specify model settings to influence the behavior of the model behavior. The preparation steps are summarized in the following table.

Table 5-1 *Preparation for Creating a Mining Model*

Preparation Step	Description
Choose the mining function	See " Choosing the Mining Function (page 5-2)"
Choose the algorithm	See " Choosing the Algorithm (page 5-3)"
Identify the build (training) data	See " Preparing the Data (page 3-1)"
For classification models, identify the test data	See " Data Sets for Classification and Regression (page 3-2)"
Determine your data transformation strategy	See " Transforming the Data (page 4-1)"
Create and populate a settings tables (if needed)	See " Specifying Model Settings (page 5-7)"

Related Topics:

[About Mining Models](#) (page 2-1)

Mining models are database schema objects that perform data mining.

[DBMS_DATA_MINING](#) (page 2-7)

Understand the routines of DBMS_DATA_MINING package.

5.2 The CREATE_MODEL Procedure

The CREATE_MODEL procedure in the DBMS_DATA_MINING package uses the specified data to create a mining model with the specified name and mining function. The model can be created with configuration settings and user-specified transformations.

```
PROCEDURE CREATE_MODEL(
    model_name           IN VARCHAR2,
    mining_function      IN VARCHAR2,
    data_table_name     IN VARCHAR2,
    case_id_column_name IN VARCHAR2,
    target_column_name  IN VARCHAR2 DEFAULT NULL,
    settings_table_name IN VARCHAR2 DEFAULT NULL,
    data_schema_name    IN VARCHAR2 DEFAULT NULL,
    settings_schema_name IN VARCHAR2 DEFAULT NULL,
    xform_list          IN TRANSFORM_LIST DEFAULT NULL);
```

5.2.1 Choosing the Mining Function

Explains about providing mining function to CREATE_MODEL.

The mining function is a required argument to the CREATE_MODEL procedure. A data mining function specifies a class of problems that can be modeled and solved.

Data mining functions implement either **supervised** or **unsupervised** learning. Supervised learning uses a set of independent attributes to predict the value of a dependent attribute or **target**. Unsupervised learning does not distinguish between dependent and independent attributes. Supervised functions are predictive. Unsupervised functions are descriptive.

Note:

In data mining terminology, a **function** is a general type of problem to be solved by a given approach to data mining. In SQL language terminology, a **function** is an operator that returns a value.

In Oracle Data Mining documentation, the term **function**, or **mining function** refers to a data mining function; the term **SQL function** or **SQL Data Mining function** refers to a SQL function for scoring (applying data mining models).

You can specify any of the values in the following table for the *mining_function* parameter to CREATE_MODEL.

Table 5-2 Mining Model Functions

<i>Mining_Function</i> Value	Description
ASSOCIATION	Association is a descriptive mining function. An association model identifies relationships and the probability of their occurrence within a data set. (association rules) Association models use the Apriori algorithm.

Table 5-2 (Cont.) Mining Model Functions

Mining_Function Value	Description
ATTRIBUTE_IMPORTANCE	<p>Attribute Importance is a predictive mining function. An attribute importance model identifies the relative importance of attributes in predicting a given outcome.</p> <p>Attribute Importance models use the Minimum Description Length algorithm.</p>
CLASSIFICATION	<p>Classification is a predictive mining function. A classification model uses historical data to predict a categorical target.</p> <p>Classification models can use Naive Bayes, Decision Tree, Logistic Regression, or Support Vector Machines. The default is Naive Bayes.</p> <p>The classification function can also be used for anomaly detection. In this case, the SVM algorithm with a null target is used (One-Class SVM).</p>
CLUSTERING	<p>Clustering is a descriptive mining function. A clustering model identifies natural groupings within a data set.</p> <p>Clustering models can use k-Means, O-Cluster, or Expectation Maximization. The default is k-Means.</p>
FEATURE_EXTRACTION	<p>Feature Extraction is a descriptive mining function. A feature extraction model creates a set of optimized attributes.</p> <p>Feature extraction models can use Non-Negative Matrix Factorization, Singular Value Decomposition (which can also be used for Principal Component Analysis) or Explicit Semantic Analysis. The default is Non-Negative Matrix Factorization.</p>
REGRESSION	<p>Regression is a predictive mining function. A regression model uses historical data to predict a numerical target.</p> <p>Regression models can use Support Vector Machines or Linear Regression. The default is Support Vector Machine.</p>

Related Topics:

Oracle Data Mining Concepts

5.2.2 Choosing the Algorithm

Learn about providing the algorithm settings for a model.

The ALGO_NAME setting specifies the algorithm for a model. If you use the default algorithm for the mining function, or if there is only one algorithm available for the mining function, you do not need to specify the ALGO_NAME setting. Instructions for specifying model settings are in "Specifying Model Settings".

Table 5-3 Data Mining Algorithms

ALGO_NAME Value	Algorithm	Default?	Mining Model Function
ALGO_AI_MDL	Minimum Description Length	—	attribute importance

Table 5-3 (Cont.) Data Mining Algorithms

ALGO_NAME Value	Algorithm	Default?	Mining Model Function
ALGO_APRIORI_ASSOCIATION_RULES	Apriori	—	association
ALGO_DECISION_TREE	Decision Tree	—	classification
ALGO_EXPECTATION_MAXIMIZATION	Expectation Maximization	—	classification
ALGO_EXPLICIT_SEMANTIC_ANALYSIS	Explicit Semantic Analysis	—	feature extraction
ALGO_EXTENSIBLE_LANG	Language used for extensible algorithm	—	All mining functions are supported
ALGO_GENERALIZED_LINEAR_MODEL	Generalized Linear Model	—	classification and regression
ALGO_KMEANS	<i>k</i> -Means	yes	clustering
ALGO_NAIVE_BAYES	Naive Bayes	yes	classification
ALGO_NONNEGATIVE_MATRIX_FACTOR	Non-Negative Matrix Factorization	yes	feature extraction
ALGO_O_CLUSTER	O-Cluster	—	clustering
ALGO_SINGULAR_VALUE_DECOMPOSITION	Singular Value Decomposition (can also be used for Principal Component Analysis)	—	feature extraction
ALGO_SUPPORT_VECTOR_MACHINES	Support Vector Machine	yes	default regression algorithm regression, classification, and anomaly detection (classification with no target)

Related Topics:

[Specifying Model Settings](#) (page 5-7)

Understand how to configure data mining models at build time.

Oracle Data Mining Concepts

5.2.3 Supplying Transformations

You can optionally specify transformations for the build data in the *xform_list* parameter to CREATE_MODEL. The transformation instructions are embedded in the model and reapplied whenever the model is applied to new data.

5.2.3.1 Creating a Transformation List

The following are the ways to create a transformation list:

- The `STACK` interface in `DBMS_DATA_MINING_TRANSFORM`.

The `STACK` interface offers a set of pre-defined transformations that you can apply to an attribute or to a group of attributes. For example, you can specify supervised binning for all categorical attributes.

- The `SET_TRANSFORM` procedure in `DBMS_DATA_MINING_TRANSFORM`.

The `SET_TRANSFORM` procedure applies a specified SQL expression to a specified attribute. For example, the following statement appends a transformation instruction for `country_id` to a list of transformations called `my_xforms`. The transformation instruction divides `country_id` by 10 before algorithmic processing begins. The reverse transformation multiplies `country_id` by 10.

```
dbms_data_mining_transform.SET_TRANSFORM (my_xforms,
    'country_id', NULL, 'country_id/10', 'country_id*10');
```

The reverse transformation is applied in the model details. If `country_id` is the target of a supervised model, the reverse transformation is also applied to the scored target.

5.2.3.2 Transformation List and Automatic Data Preparation

Understand the interaction between transformation list and Automatic Data Preparation (ADP).

The transformation list argument to `CREATE_MODEL` interacts with the `PREP_AUTO` setting, which controls ADP:

- When ADP is on and you specify a transformation list, your transformations are applied with the automatic transformations and embedded in the model. The transformations that you specify are executed before the automatic transformations.
- When ADP is off and you specify a transformation list, your transformations are applied and embedded in the model, but no system-generated transformations are performed.
- When ADP is on and you do not specify a transformation list, the system-generated transformations are applied and embedded in the model.
- When ADP is off and you do not specify a transformation list, no transformations are embedded in the model; you must separately prepare the data sets you use for building, testing, and scoring the model.

Related Topics:

[Embedding Transformations in a Model](#) (page 4-5)

Oracle Database PL/SQL Packages and Types Reference

5.2.4 About Partitioned Model

Introduces partitioned model to organise and represent multiple models.

Oracle Data Mining supports building of a persistent Oracle Data Mining partitioned model. A partitioned model organizes and represents multiple models as partitions in a single model entity, enabling a user to easily build and manage models tailored to independent slices of data. Persistent means that the partitioned model has an on-disk representation. The product manages the organization of the partitioned model and

simplifies the process of scoring the partitioned model. You must include the partition columns as part of the `USING` clause when scoring.

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

Related Topics:

Oracle Database Reference

Oracle Data Mining User's Guide

5.2.4.1 Partitioned Model Build Process

To build a Partitioned Model, Oracle Data Mining requires a partitioning key. The partition key is set through a build setting in the settings table.

The partitioning key is a comma-separated list of one or more columns (up to 16) from the input data set. The partitioning key horizontally slices the input data based on discrete values of the partitioning key. That is, partitioning is performed as list values as opposed to range partitioning against a continuous value. The partitioning key supports only columns of the data type `NUMBER` and `VARCHAR2`.

During the build process the input data set is partitioned based on the distinct values of the specified key. Each data slice (unique key value) results in its own model partition. This resultant model partition is not separate and is not visible to you as a standalone model. The default value of the maximum number of partitions for partitioned models is 1000 partitions. You can also set a different maximum partitions value. If the number of partitions in the input data set exceed the defined maximum, Oracle Data Mining throws an exception.

The Partitioned Model organizes features common to all partitions and the partition specific features. The common features consist of the following metadata:

- The model name
- The mining function
- The mining algorithm
- A super set of all mining model attributes referenced by all partitions (signature)
- A common set of user-defined column transformations
- Any user-specified or default build settings that are interpreted as global. For example, the Auto Data Preparation (ADP) setting

5.2.4.2 DDL in Partitioned model

Partitioned models are maintained through the following DDL operations:

- [Drop model or drop partition](#) (page 5-6)
- [Add partition](#) (page 5-7)

5.2.4.2.1 Drop Model or Drop Partition

Oracle Data Mining supports dropping a single model partition for a given partition name.

If only a single partition remains, you cannot explicitly drop that partition. Instead, you must either add additional partitions prior to dropping the partition or you may

choose to drop the model itself. When dropping a partitioned model, all partitions are dropped in a single atomic operation. From a performance perspective, Oracle recommends `DROP_PARTITION` followed by an `ADD_PARTITION` instead of leveraging the `REPLACE` option due to the efficient behavior of the `DROP_PARTITION` option.

5.2.4.2.2 Add Partition

Oracle Data Mining supports adding a single partition or multiple partitions to an existing partitioned model.

The addition occurs based on the input data set and the name of the existing partitioned model. The operation takes the input data set and the existing partitioned model as parameters. The partition keys are extracted from the input data set and the model partitions are built against the input data set. These partitions are added to the partitioned model. In the case where partition keys for new partitions conflict with the existing partitions in the model, you can select from the following three approaches to resolve the conflicts:

- `ERROR`: Terminates the `ADD` operation without adding any partitions.
- `REPLACE`: Replaces the existing partition for which the conflicting keys are found.
- `IGNORE`: Eliminates the rows having the conflicting keys.

If the input data set contains multiple keys, then the operation creates multiple partitions. If the total number of partitions in the model increases to more than the user-defined maximum specified when the model was created, then you get an error. The default threshold value for the number of partitions is 1000.

5.2.4.3 Partitioned Model scoring

Learn about scoring of a partitioned model.

The scoring of the partitioned model is the same as that of the non-partitioned model. The syntax of the data mining function remains the same but is extended to provide an optional hint to you. The optional hint can impact the performance of a query which involves scoring a partitioned model.

For scoring a partitioned model, the signature columns used during the build for the partitioning key must be present in the scoring data set. These columns are combined to form a unique partition key. The unique key is then mapped to a specific underlying model partition, and the identified model partition is used to score that row.

The partitioned objects that are necessary for scoring are loaded on demand during the query execution and are aged out depending on the System Global Area (SGA) memory.

Related Topics:

Oracle Database SQL Language Reference

5.3 Specifying Model Settings

Understand how to configure data mining models at build time.

Numerous configuration settings are available for configuring data mining models at build time. To specify settings, create a settings table with the columns shown in the following table and pass the table to `CREATE_MODEL`.

Table 5-4 Settings Table Required Columns

Column Name	Data Type
setting_name	VARCHAR2(30)
setting_value	VARCHAR2(4000)

Example 5-1 (page 5-9) creates a settings table for an Support Vector Machine (SVM) Classification model. Since SVM is not the default classifier, the ALGO_NAME setting is used to specify the algorithm. Setting the SVMS_KERNEL_FUNCTION to SVMS_LINEAR causes the model to be built with a linear kernel. If you do not specify the kernel function, the algorithm chooses the kernel based on the number of attributes in the data.

Some settings apply generally to the model, others are specific to an algorithm. Model settings are referenced in [Table 5-5](#) (page 5-8) and [Table 5-6](#) (page 5-8).

Table 5-5 General Model Settings

Settings	Description
Mining function settings	See "Mining Function Settings" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Algorithm names	See "Algorithm Names" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Global model characteristics	See "Global Settings" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Automatic Data Preparation	See "Automatic Data Preparation" in <i>Oracle Database PL/SQL Packages and Types Reference</i>

Table 5-6 Algorithm-Specific Model Settings

Algorithm	Description
Decision Tree	See "Algorithm Settings: Decision Tree" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Expectation Maximization	See "Algorithm Settings: Expectation Maximization" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Explicit Semantic Analysis	See "Algorithm Settings: Explicit Semantic Analysis" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Generalized Linear Models	See "Algorithm Settings: Generalized Linear Models" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
k-Means	See "Algorithm Settings: k-Means" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Naive Bayes	See "Algorithm Settings: Naive Bayes" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Non-Negative Matrix Factorization	See "Algorithm Settings: Non-Negative Matrix Factorization" in <i>Oracle Database PL/SQL Packages and Types Reference</i>

Table 5-6 (Cont.) Algorithm-Specific Model Settings

Algorithm	Description
O-Cluster	See "Algorithm Settings: O-Cluster" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Singular Value Decomposition	See "Algorithm Settings: Singular Value Decomposition" in <i>Oracle Database PL/SQL Packages and Types Reference</i>
Support Vector Machine	See "Algorithm Settings: Support Vector Machine" in <i>Oracle Database PL/SQL Packages and Types Reference</i>

Example 5-1 Creating a Settings Table for an SVM Classification Model

```

CREATE TABLE svmc_sh_sample_settings (
  setting_name VARCHAR2(30),
  setting_value VARCHAR2(4000));

BEGIN
  INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
    (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);
  INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
    (dbms_data_mining.svms_kernel_function, dbms_data_mining.svms_linear);
  COMMIT;
END;
/

```

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

5.3.1 Specifying Costs

Specify a cost matrix table to build a Decision Tree model.

The CLAS_COST_TABLE_NAME setting specifies the name of a cost matrix table to be used in building a Decision Tree model. A cost matrix biases a classification model to minimize costly misclassifications. The cost matrix table must have the columns shown in the following table:

Table 5-7 Cost Matrix Table Required Columns

Column Name	Data Type
actual_target_value	valid target data type
predicted_target_value	valid target data type
cost	NUMBER

Decision Tree is the only algorithm that supports a cost matrix at build time. However, you can create a cost matrix and associate it with any classification model for scoring.

If you want to use costs for scoring, create a table with the columns shown in [Table 5-7](#) (page 5-9), and use the `DBMS_DATA_MINING.ADD_COST_MATRIX` procedure to add the cost matrix table to the model. You can also specify a cost matrix inline when

invoking a PREDICTION function. [Table 3-1](#) (page 3-4) has details for valid target data types.

Related Topics:

Oracle Data Mining Concepts

5.3.2 Specifying Prior Probabilities

Prior probabilities can be used to offset differences in distribution between the build data and the actual population.

The CLAS_PRIORS_TABLE_NAME setting specifies the name of a table of prior probabilities to be used in building a Naive Bayes model. The priors table must have the columns shown in the following table.

Table 5-8 Priors Table Required Columns

Column Name	Data Type
target_value	valid target data type
prior_probability	NUMBER

Related Topics:

[Target Attribute](#) (page 3-4)

Understand what a **target** means in data mining and understand the different target data types.

Oracle Data Mining Concepts

5.3.3 Specifying Class Weights

Specify class weights table settings in Logistic Regression or Support Vector Machine (SVM) Classification to favour higher weighted classes.

The CLAS_WEIGHTS_TABLE_NAME setting specifies the name of a table of class weights to be used to bias a logistic regression (Generalized Linear Model Classification) or SVM Classification model to favor higher weighted classes. The weights table must have the columns shown in the following table.

Table 5-9 Class Weights Table Required Columns

Column Name	Data Type
target_value	valid target data type
class_weight	NUMBER

Related Topics:

[Target Attribute](#) (page 3-4)

Understand what a **target** means in data mining and understand the different target data types.

Oracle Data Mining Concepts

5.3.4 Model Settings in the Data Dictionary

Explains about ALL/USER/DBA_MINING_MODEL_SETTINGS in data dictionary view.

Information about mining model settings can be obtained from the data dictionary view ALL/USER/DBA_MINING_MODEL_SETTINGS. When used with the ALL prefix, this view returns information about the settings for the models accessible to the current user. When used with the USER prefix, it returns information about the settings for the models in the user's schema. The DBA prefix is only available for DBAs.

The columns of ALL_MINING_MODEL_SETTINGS are described as follows and explained in the following table.

```
SQL> describe all_mining_model_settings
Name                               Null?    Type
-----
OWNER                               NOT NULL VARCHAR2(30)
MODEL_NAME                           NOT NULL VARCHAR2(30)
SETTING_NAME                          NOT NULL VARCHAR2(30)
SETTING_VALUE                          VARCHAR2(4000)
SETTING_TYPE                          VARCHAR2(7)
```

Table 5-10 ALL_MINING_MODEL_SETTINGS

Column	Description
owner	Owner of the mining model.
model_name	Name of the mining model.
setting_name	Name of the setting.
setting_value	Value of the setting.
setting_type	INPUT if the value is specified by a user. DEFAULT if the value is system-generated.

The following query lists the settings for the Support Vector Machine (SVM) Classification model SVMC_SH_CLAS_SAMPLE. The ALGO_NAME, CLAS_WEIGHTS_TABLE_NAME, and SVMS_KERNEL_FUNCTION settings are user-specified. These settings have been specified in a settings table for the model.

Example 5-2 ALL_MINING_MODEL_SETTINGS

```
SQL> COLUMN setting_value FORMAT A35
SQL> SELECT setting_name, setting_value, setting_type
       FROM all_mining_model_settings
       WHERE model_name in 'SVMC_SH_CLAS_SAMPLE';
```

SETTING_NAME	SETTING_VALUE	SETTING
SVMS_ACTIVE_LEARNING	SVMS_AL_ENABLE	DEFAULT
PREP_AUTO	OFF	DEFAULT
SVMS_COMPLEXITY_FACTOR	0.244212	DEFAULT
SVMS_KERNEL_FUNCTION	SVMS_LINEAR	INPUT
CLAS_WEIGHTS_TABLE_NAME	svmc_sh_sample_class_wt	INPUT
SVMS_CONV_TOLERANCE	.001	DEFAULT
ALGO_NAME	ALGO_SUPPORT_VECTOR_MACHINES	INPUT

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

5.3.5 Specifying Mining Model Settings for R Model

The mining model settings for R model determine the characteristics of the model. You can specify the mining model settings in the `mining_model_table`.

You can build R models with the mining model settings by combining together generic settings that do not require an algorithm, such as `ODMS_PARTITION_COLUMNS` and `ODMS_SAMPLING`. The following settings are exclusive to R mining model, and they allow you to specify the R Mining model:

- [ALGO_EXTENSIBLE_LANG](#) (page 5-12)
- [RALG_BUILD_FUNCTION](#) (page 5-13)
- [RALG_BUILD_PARAMETER](#) (page 5-14)
- [RALG_DETAILS_FORMAT](#) (page 5-15)
- [RALG_DETAILS_FUNCTION](#) (page 5-14)
- [RALG_SCORE_FUNCTION](#) (page 5-16)
- [RALG_WEIGHT_FUNCTION](#) (page 5-18)

Related Topics:

[Registered R Scripts](#) (page 5-19)

The `RALG_*_FUNCTION` must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

5.3.5.1 ALGO_EXTENSIBLE_LANG

Use the `ALGO_EXTENSIBLE_LANG` setting to specify the Oracle Data Mining framework with extensible algorithms.

Currently, `R` is the only valid value for `ALGO_EXTENSIBLE_LANG`. When the value for `ALGO_EXTENSIBLE_LANG` is set to `R`, the mining models are built using the R language. You can use the following settings in the `model_setting_table` to specify the build, score, and view of the R model.

- [RALG_BUILD_FUNCTION](#) (page 5-13)
- [RALG_BUILD_PARAMETER](#) (page 5-14)
- [RALG_DETAILS_FUNCTION](#) (page 5-14)
- [RALG_DETAILS_FORMAT](#) (page 5-15)
- [RALG_SCORE_FUNCTION](#) (page 5-16)
- [RALG_WEIGHT_FUNCTION](#) (page 5-18)

Related Topics:

[Registered R Scripts](#) (page 5-19)

The `RALG_*_FUNCTION` must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

5.3.5.2 RALG_BUILD_FUNCTION

Use the `RALG_BUILD_FUNCTION` to specify the name of an existing registered R script for R algorithm mining model build.

You must specify both `RALG_BUILD_FUNCTION` and `ALGO_EXTENSIBLE_LANG` in the `model_setting_table`. The R script defines an R function that has the first input argument of `data.frame` for training data, and it returns an R model object. The first data argument is mandatory. The `RALG_BUILD_FUNCTION` can accept additional model build parameters.

Note: The valid inputs for input parameters are numeric and string scalar data types.

Example 5-3 Example of RALG_BUILD_FUNCTION

This example shows how to specify the name of the R script `MY_LM_BUILD_SCRIPT` that is used to build the model in the `model_setting_table`.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_build_function,'MY_LM_BUILD_SCRIPT');
End;
/
```

The R script `MY_LM_BUILD_SCRIPT` defines an R function that builds the LM model. You must register the script `MY_LM_BUILD_SCRIPT` in the R script repository which uses the existing ORE security restrictions. You can use Oracle R Enterprise API `sys.rqScriptCreate` to register the script. Oracle R Enterprise requires the `RQADMIN` role to register R scripts.

For example:

```
Begin
sys.rqScriptCreate('MY_LM_BUILD_SCRIPT', 'function(data, formula, model.frame)
{lm(formula = formula, data=data, model = as.logical(model.frame))}');
End;
/
```

For Clustering and Feature Extraction mining function model build, the R attributes `dm$nclus` and `dm$nfeat` must be set on the return R model to indicate the number of clusters and features respectively.

The R script `MY_KM_BUILD_SCRIPT` defines an R function that builds the *k*-Means model for Clustering. R attribute `dm$nclus` is set with the number of clusters for the return Clustering model.

```
'function(dat) {dat.scaled <- scale(dat)
  set.seed(6543); mod <- list()
  fit <- kmeans(dat.scaled, centers = 3L)
  mod[[1L]] <- fit
  mod[[2L]] <- attr(dat.scaled, "scaled:center")
  mod[[3L]] <- attr(dat.scaled, "scaled:scale")
```

```
attr(mod, "dm$nclus") <- nrow(fit$centers)
mod}'
```

The R script `MY_PCA_BUILD_SCRIPT` defines an R function that builds the PCA model. R attribute `dm$nfeat` is set with the number of features for the return feature extraction model.

```
'function(dat) {
  mod <- prcomp(dat, retx = FALSE)
  attr(mod, "dm$nfeat") <- ncol(mod$rotation)
mod}'
```

Related Topics:

[RALG_BUILD_PARAMETER](#) (page 5-14)

The `RALG_BUILD_FUNCTION` input parameter specifies a list of numeric and string scalar values in SQL `SELECT` query statement format.

[Registered R Scripts](#) (page 5-19)

The `RALG_*_FUNCTION` must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

5.3.5.2.1 RALG_BUILD_PARAMETER

The `RALG_BUILD_FUNCTION` input parameter specifies a list of numeric and string scalar values in SQL `SELECT` query statement format.

Example 5-4 Example of RALG_BUILD_PARAMETER

The `RALG_BUILD_FUNCTION` input parameters must be a list of numeric and string scalar values. The input parameters are optional.

The syntax of the parameter is:

```
'SELECT value parameter name ...FROM dual'
```

This example shows how to specify a formula for the input argument `'formula'` and a numeric value zero for input argument `'model.frame'` using the `RALG_BUILD_PARAMETER`. These input arguments must match with the function signature of the R script used in `RALG_BUILD_FUNCTION` Parameter.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_build_parameter, 'select ''AGE ~ .'' as "formula", 0 as
"model.frame" from dual');
End;
/
```

Related Topics:

[RALG_BUILD_FUNCTION](#) (page 5-13)

Use the `RALG_BUILD_FUNCTION` to specify the name of an existing registered R script for R algorithm mining model build.

5.3.5.3 RALG_DETAILS_FUNCTION

The `RALG_DETAILS_FUNCTION` specifies the R model metadata that is returned in the `data.frame`.

Use the `RALG_DETAILS_FUNCTION` to specify an existing registered R script that generates model information. The specified R script defines an R function that contains the first input argument for the R model object. The output of the R function

must be a `data.frame`. The columns of the `data.frame` are defined by `RALG_DETAILS_FORMAT`, and can contain only numeric or string scalar types.

Example 5-5 Example of `RALG_DETAILS_FUNCTION`

This example shows how to specify the name of the R script `MY_LM_DETAILS_SCRIPT` in the `model_setting_table`. This script defines the R function that is used to provide the model information.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_details_function, 'MY_LM_DETAILS_SCRIPT');
End;
/
```

In the R script repository, the script `MY_LM_DETAILS_SCRIPT` is registered as:

```
'function(mod) data.frame(name=names(mod$coefficients),
coef=mod$coefficients)'
```

Related Topics:

[Registered R Scripts](#) (page 5-19)

The `RALG_*_FUNCTION` must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

[`RALG_DETAILS_FORMAT`](#) (page 5-15)

Use the `RALG_DETAILS_FORMAT` parameter to specify the names and column types in the model view. It is a string that contains a `SELECT` query to specify a list of numeric and string scalar data types for the name and type of the model view columns.

5.3.5.3.1 `RALG_DETAILS_FORMAT`

Use the `RALG_DETAILS_FORMAT` parameter to specify the names and column types in the model view. It is a string that contains a `SELECT` query to specify a list of numeric and string scalar data types for the name and type of the model view columns.

When `RALG_DETAILS_FORMAT` and `RALG_DETAILS_FUNCTION` are both specified, a model view by the name `DM$VD <model_name>` is created along with an R model in the current schema. The first column of the model view is `PARTITION_NAME`. It has `NULL` value for non-partitioned models. The other columns of the model view are defined by `RALG_DETAILS_FORMAT`.

Example 5-6 Example of `RALG_DETAILS_FORMAT`

This example shows how to specify the name and type of the columns for the generated model view. The model view contains `varchar2` column `attr_name` and number column `coef_value` after the first column `partition_name`.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_details_format, 'select cast(''a'' as varchar2(20)) as
attr_name, 0 as coef_value from dual');
End;
/
```

Related Topics:

[`RALG_DETAILS_FUNCTION`](#) (page 5-14)

The `RALG_DETAILS_FUNCTION` specifies the R model metadata that is returned in the `data.frame`.

5.3.5.4 RALG_SCORE_FUNCTION

Use the `RALG_SCORE_FUNCTION` to specify an existing registered R script for R algorithm mining model score in the `mining_model_table`.

The specified R script defines an R function. The first input argument defines the model object. The second input argument defines the `data.frame` that is used for scoring data.

Example 5-7 Example of RALG_SCORE_FUNCTION

This example shows how the function takes the R model and scores the data in the `data.frame`. The argument object is the R Linear Model. The argument `newdata` contains scoring data in the `data.frame`.

```
function(object, newdata) {res <- predict.lm(object, newdata = newdata, se.fit = TRUE); data.frame(fit=res$fit, se=res$se.fit, df=summary(object)$df[1L])}
```

In this example,

- `object` indicates the LM model
- `newdata` indicates the scoring `data.frame`

The output of the specified R function must be a `data.frame`. Each row represents the prediction for the corresponding scoring data from the input `data.frame`. The columns of the `data.frame` are specific to mining functions, such as:

Regression: A single numeric column for predicted target value, with two optional columns containing standard error of model fit, and the degrees of freedom number. The optional columns are needed for query function `PREDICTION_BOUNDS` to work.

Example 5-8 Example of RALG_SCORE_FUNCTION for Regression

This example shows how to specify the name of the R script `MY_LM_PREDICT_SCRIPT` that is used to score the model in the `model_setting_table`.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_score_function, 'MY_LM_PREDICT_SCRIPT');
End;
/
```

In the R script repository, the script `MY_LM_PREDICT_SCRIPT` is registered as:

```
function(object, newdata) {data.frame(pre = predict(object, newdata = newdata))}
```

Classification: Each column represents the predicted probability of one target class. The column name is the target class name.

Example 5-9 Example of RALG_SCORE_FUNCTION for Classification

This example shows how to specify the name of the R script `MY_LOGITGLM_PREDICT_SCRIPT` that is used to score the logit Classification model in the `model_setting_table`.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_score_function, 'MY_LOGITGLM_PREDICT_SCRIPT');
End;
/
```

In the R script repository, *MY_LOGITGLM_PREDICT_SCRIPT* is registered as follows. It is a logit Classification with two target class "0" and "1".

```
'function(object, newdata) {
  pred <- predict(object, newdata = newdata, type="response");
  res <- data.frame(1-pred, pred);
  names(res) <- c("0", "1");
  res}'
```

Clustering: Each column represents the predicted probability of one cluster. The columns are arranged in order of cluster ID. Each cluster is assigned a cluster ID, and they are consecutive values starting from 1. To support *CLUSTER_DISTANCE* in the R model, the output of R score function returns extra column containing the value of the distance to each cluster in order of cluster ID after the columns for the predicted probability.

Example 5-10 Example of RALG_SCORE_FUNCTION for Clustering

This example shows how to specify the name of the R script *MY_CLUSTER_PREDICT_SCRIPT* that is used to score the model in the *model_setting_table*.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_score_function, 'MY_CLUSTER_PREDICT_SCRIPT');
End;
/
```

In the R script repository, the script *MY_CLUSTER_PREDICT_SCRIPT* is registered as:

```
'function(object, dat){
  mod <- object[[1L]]; ce <- object[[2L]]; sc <- object[[3L]];
  newdata = scale(dat, center = ce, scale = sc);
  centers <- mod$centers;
  ss <- sapply(as.data.frame(t(centers)),
  function(v) rowSums(scale(newdata, center=v, scale=FALSE)^2));
  if (!is.matrix(ss)) ss <- matrix(ss, ncol=length(ss));
  disp <- -1 / (2* mod$tot.withinss/length(mod$cluster));
  distr <- exp(disp*ss);
  prob <- distr / rowSums(distr);
  as.data.frame(cbind(prob, sqrt(ss)))}'
```

This function fetches the centers of the clusters from the R model, and computes the probabilities for belonging to each cluster based on the distance of the score data to the corresponding cluster center.

Feature Extraction: Each column represents the coefficient value of one feature. The columns are arranged in order of feature ID. Each feature is assigned a feature ID, and they are consecutive values starting from 1.

Example 5-11 Example of RALG_SCORE_FUNCTION for Feature Extraction

This example shows how to specify the name of the R script *MY_FEATURE_EXTRACTION_SCRIPT* that is used to score the model in the *model_setting_table*.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_score_function, 'MY_FEATURE_EXTRACTION_SCRIPT');
End;
/
```

In the R script repository, the script *MY_FEATURE_EXTRACTION_SCRIPT* is registered as:

```
'function(object, dat) { as.data.frame(predict(object, dat)) }'
```

The function fetches the centers of the features from the R model, and computes the feature coefficient based on the distance of the score data to the corresponding feature center.

Related Topics:

[Registered R Scripts](#) (page 5-19)

The *RALG_*_FUNCTION* must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

5.3.5.5 RALG_WEIGHT_FUNCTION

Use the *RALG_WEIGHT_FUNCTION* to specify the name of an existing registered R script that computes weight or contribution for each attribute in scoring. The specified R script is used in the query function *PREDICTION_DETAILS* to evaluate attribute contribution.

The specified R script defines an R function containing the first input argument for model object, and the second input argument of *data.frame* for scoring data. When the mining function is Classification, Clustering, or Feature Extraction, the target class name or cluster ID or feature ID is passed by the third input argument to compute the weight for that particular class or cluster or feature. The script returns a *data.frame* containing the contributing weight for each attribute in a row. Each row corresponds to that input scoring *data.frame*.

Example 5-12 Example of RALG_WEIGHT_FUNCTION

This example shows how to specify the name of the R script *MY_PREDICT_WEIGHT_SCRIPT* that computes weight or contribution of R model attributes in the *model_setting_table*.

```
Begin
insert into model_setting_table values
(dbms_data_mining.ralg_weight_function, 'MY_PREDICT_WEIGHT_SCRIPT');
End;
/
```

In the R script repository, the script *MY_PREDICT_WEIGHT_SCRIPT* for Regression is registered as:

```
'function(mod, data) { coef(mod)[-1L]*data }'
```

In the R script repository, the script *MY_PREDICT_WEIGHT_SCRIPT* for logit Classification is registered as:

```
'function(mod, dat, clas) {
  v <- predict(mod, newdata=dat, type = "response");
  v0 <- data.frame(v, 1-v); names(v0) <- c("0", "1");
  res <- data.frame(lapply(seq_along(dat),
    function(x, dat) {
      if(is.numeric(dat[[x]])) dat[,x] <- as.numeric(0)
      else dat[,x] <- as.factor(NA);
      vv <- predict(mod, newdata = dat, type = "response");
      vv = data.frame(vv, 1-vv); names(vv) <- c("0", "1");
      v0[[clas]] / vv[[clas]], dat = dat));
  res
```



```
names(res) <- names(dat);
res}'
```

Related Topics:

[Registered R Scripts](#) (page 5-19)

The RALG_*_FUNCTION must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

5.3.5.6 Registered R Scripts

The RALG_*_FUNCTION must specify R scripts that exist in the R script repository. You can register the R scripts using Oracle R Enterprise.

The RALG_*_FUNCTION includes the following functions:

- RALG_BUILD_FUNCTION
- RALG_DETAILS_FUNCTION
- RALG_SCORE_FUNCTION
- RALG_WEIGHT_FUNCTION

Note: The R scripts must exist in the R script repository for an R model to function.

You can register the R scripts through Oracle Enterprise R (ORE). To register R scripts, you must have the RQADMIN role. After an R model is built, the names of these specified R scripts become model settings. These R scripts must exist in the R script repository for an R model to remain functional.

You can manage the R memory that is used to build, score, and view the R models through Oracle Enterprise R as well.

5.3.5.7 R Model Demonstration Scripts

You can access R model demonstration scripts under `rdbms/demo`

```
dmraidemo.sql  dmrglmdemo.sql  dmrpcademo.sql
dmrardemo.sql  dmrkmdemo.sql  dmrrfdemo.sql
dmrtddemo.sql  dmrrndemo.sql
```

5.4 Model Detail Views

The GET_* interfaces are replaced by model views, and Oracle recommends that users leverage the views instead.

The following are the new model views:

Association:

- [Model Detail Views for Association Rules](#) (page 5-20)
- [Model Detail View for Frequent Itemsets](#) (page 5-26)
- [Model Detail View for Transactional Itemsets](#) (page 5-26)
- [Model Detail View for Transactional Rule](#) (page 5-27)

Classification, Regression, and Anomaly Detection:

- [Model Detail Views for Classification Algorithms](#) (page 5-28)
- [Model Detail Views for Decision Tree](#) (page 5-29)
- [Model Detail Views for Generalized Linear Model](#) (page 5-31)
- [Model Detail Views for Naive Bayes](#) (page 5-37)
- [Model Detail View for Support Vector Machine](#) (page 5-38)

Clustering:

- [Model Detail Views for Clustering Algorithms](#) (page 5-39)
- [Model Detail Views for Expectation Maximization](#) (page 5-42)
- [Model Detail Views for *k*-Means](#) (page 5-45)
- [Model Detail Views for O-Cluster](#) (page 5-46)

Feature Extraction:

- [Model Detail Views for Explicit Semantic Analysis](#) (page 5-47)
- [Model Detail Views for Non-Negative Matrix Factorization](#) (page 5-48)
- [Model Detail Views for Singular Value Decomposition](#) (page 5-50)

Feature Selection:

- [Model Detail View for Minimum Description Length](#) (page 5-52)

Data Preparation and Other:

- [Model Detail View for Binning](#) (page 5-53)
- [Model Detail Views for Global Information](#) (page 5-53)
- [Model Detail View for Normalization and Missing Value Handling](#) (page 5-54)

5.4.1 Model Detail Views for Association Rules

Model detail views for Association Rules describes the rule view for Association Rules. Oracle recommends that users leverage the model details views instead of the `GET_ASSOCIATION_RULES` function.

The rule view `DM$VRmodel_name` describes the generated rules for Association Rules. Depending on the settings of the model, the rule has different set of columns. The following views are displayed when different Global settings are applied without aggregates for transactional and 2-Dimensional inputs.

Transactional Input Without ASSO_AGGREGATES Setting

When `ODMS_ITEM_ID_COLUMN_NAME` is set and `ITEM_VALUE` (`ODMS_ITEM_VALUE_COLUMN_NAME`) is not set, the following is the transactional view:

Name	Type
PARTITION_NAME	VARCHAR2(128)
RULE_ID	NUMBER
RULE_SUPPORT	NUMBER

RULE_CONFIDENCE	NUMBER
RULE_LIFT	NUMBER
RULE_REVCONFIDENCE	NUMBER
ANTECEDENT_SUPPORT	NUMBER
NUMBER_OF_ITEMS	NUMBER
CONSEQUENT_SUPPORT	NUMBER
CONSEQUENT_NAME	VARCHAR2 (4000)
ANTECEDENT	SYS.XMLTYPE

Table 5-11 Rule View Columns for Transactional Inputs

Column Name	Description
PARTITION_NAME	A partition in a partitioned model to retrieve details
RULE_ID	Name or identifier of the target
RULE_SUPPORT	The number of transactions that satisfy the rule.
RULE_CONFIDENCE	The likelihood of a transaction satisfying the rule.
RULE_LIFT	The degree of improvement in the prediction over random chance when the rule is satisfied.
RULE_REVCONFIDENCE	The number of transactions in which the rule occurs divided by the number of transactions in which the consequent occurs.
ANTECEDENT_SUPPORT	The ratio of the number of transactions that satisfy the antecedent to the total number of transactions.
NUMBER_OF_ITEMS	The total number of attributes referenced in the antecedent and consequent of the rule.
CONSEQUENT_SUPPORT	The ratio of the number of transactions that satisfy the consequent to the total number of transactions.
CONSEQUENT_NAME	Name of the consequent

Table 5-11 (Cont.) Rule View Columns for Transactional Inputs

Column Name	Description																														
CONSEQUENT_SUBNAME	<p>For two-dimensional inputs, CONSEQUENT_SUBNAME is used for nested column in the input data table. See, Example: Creating a Nested Column for Market Basket Analysis (page 3-10). In this example, there is a nested column. The CONSEQUENT_SUBNAME is the ATTRIBUTE_NAME part of the nested column. That is, 'O/S Documentation Set - English' and CONSEQUENT_VALUE is the value part of the nested column, which is, 1.</p> <p>For two-dimensional inputs, when ODMS_ITEM_ID_COLUMN_NAME is not set, each item consists of three parts: NAME, SUBNAME and VALUE.</p> <p>The view uses three columns for consequent. The rule view has the following columns:</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr><td>-----</td><td>-----</td></tr> <tr><td>PARTITION_NAME</td><td>VARCHAR2(128)</td></tr> <tr><td>RULE_ID</td><td>NUMBER</td></tr> <tr><td>RULE_SUPPORT</td><td>NUMBER</td></tr> <tr><td>RULE_CONFIDENCE</td><td>NUMBER</td></tr> <tr><td>RULE_LIFT</td><td>NUMBER</td></tr> <tr><td>RULE_REVCONFIDENCE</td><td>NUMBER</td></tr> <tr><td>ANTECEDENT_SUPPORT</td><td>NUMBER</td></tr> <tr><td>NUMBER_OF_ITEMS</td><td>NUMBER</td></tr> <tr><td>CONSEQUENT_SUPPORT</td><td>NUMBER</td></tr> <tr><td>CONSEQUENT_NAME</td><td>VARCHAR2(4000)</td></tr> <tr><td>CONSEQUENT_SUBNAME</td><td>VARCHAR2(4000)</td></tr> <tr><td>CONSEQUENT_VALUE</td><td>VARCHAR2(4000)</td></tr> <tr><td>ANTECEDENT</td><td>SYS.XMLTYPE</td></tr> </tbody> </table>	Name	Type	-----	-----	PARTITION_NAME	VARCHAR2(128)	RULE_ID	NUMBER	RULE_SUPPORT	NUMBER	RULE_CONFIDENCE	NUMBER	RULE_LIFT	NUMBER	RULE_REVCONFIDENCE	NUMBER	ANTECEDENT_SUPPORT	NUMBER	NUMBER_OF_ITEMS	NUMBER	CONSEQUENT_SUPPORT	NUMBER	CONSEQUENT_NAME	VARCHAR2(4000)	CONSEQUENT_SUBNAME	VARCHAR2(4000)	CONSEQUENT_VALUE	VARCHAR2(4000)	ANTECEDENT	SYS.XMLTYPE
Name	Type																														
-----	-----																														
PARTITION_NAME	VARCHAR2(128)																														
RULE_ID	NUMBER																														
RULE_SUPPORT	NUMBER																														
RULE_CONFIDENCE	NUMBER																														
RULE_LIFT	NUMBER																														
RULE_REVCONFIDENCE	NUMBER																														
ANTECEDENT_SUPPORT	NUMBER																														
NUMBER_OF_ITEMS	NUMBER																														
CONSEQUENT_SUPPORT	NUMBER																														
CONSEQUENT_NAME	VARCHAR2(4000)																														
CONSEQUENT_SUBNAME	VARCHAR2(4000)																														
CONSEQUENT_VALUE	VARCHAR2(4000)																														
ANTECEDENT	SYS.XMLTYPE																														
<p>Note: All the types for three parts are VARCHAR2. This column is not applicable when ASSO_AGGREGATES is set.</p>																															

Table 5-11 (Cont.) Rule View Columns for Transactional Inputs

Column Name	Description																												
CONSEQUENT_VALUE	Value of the consequent when ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is set with TYPE as numerical, the view has a CONSEQUENT_VALUE column. In the following view, the TYPE of the CONSEQUENT_VALUE is NUMBER.																												
	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr><td>-----</td><td>-----</td></tr> <tr><td>PARTITION_NAME</td><td>VARCHAR2(128)</td></tr> <tr><td>RULE_ID</td><td>NUMBER</td></tr> <tr><td>RULE_SUPPORT</td><td>NUMBER</td></tr> <tr><td>RULE_CONFIDENCE</td><td>NUMBER</td></tr> <tr><td>RULE_LIFT</td><td>NUMBER</td></tr> <tr><td>RULE_REVCONFIDENCE</td><td>NUMBER</td></tr> <tr><td>ANTECEDENT_SUPPORT</td><td>NUMBER</td></tr> <tr><td>NUMBER_OF_ITEMS</td><td>NUMBER</td></tr> <tr><td>CONSEQUENT_SUPPORT</td><td>NUMBER</td></tr> <tr><td>CONSEQUENT_NAME</td><td>VARCHAR2(4000)</td></tr> <tr><td>CONSEQUENT_VALUE</td><td>NUMBER</td></tr> <tr><td>ANTECEDENT</td><td>SYS.XMLTYPE</td></tr> </tbody> </table>	Name	Type	-----	-----	PARTITION_NAME	VARCHAR2(128)	RULE_ID	NUMBER	RULE_SUPPORT	NUMBER	RULE_CONFIDENCE	NUMBER	RULE_LIFT	NUMBER	RULE_REVCONFIDENCE	NUMBER	ANTECEDENT_SUPPORT	NUMBER	NUMBER_OF_ITEMS	NUMBER	CONSEQUENT_SUPPORT	NUMBER	CONSEQUENT_NAME	VARCHAR2(4000)	CONSEQUENT_VALUE	NUMBER	ANTECEDENT	SYS.XMLTYPE
Name	Type																												
-----	-----																												
PARTITION_NAME	VARCHAR2(128)																												
RULE_ID	NUMBER																												
RULE_SUPPORT	NUMBER																												
RULE_CONFIDENCE	NUMBER																												
RULE_LIFT	NUMBER																												
RULE_REVCONFIDENCE	NUMBER																												
ANTECEDENT_SUPPORT	NUMBER																												
NUMBER_OF_ITEMS	NUMBER																												
CONSEQUENT_SUPPORT	NUMBER																												
CONSEQUENT_NAME	VARCHAR2(4000)																												
CONSEQUENT_VALUE	NUMBER																												
ANTECEDENT	SYS.XMLTYPE																												

When ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is set with TYPE as categorical, the view has a CONSEQUENT_VALUE column.

In the following view, the TYPE of the CONSEQUENT_VALUE is VARCHAR.

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
RULE_ID	NUMBER
RULE_SUPPORT	NUMBER
RULE_CONFIDENCE	NUMBER
RULE_LIFT	NUMBER
RULE_REVCONFIDENCE	NUMBER
ANTECEDENT_SUPPORT	NUMBER
NUMBER_OF_ITEMS	NUMBER
CONSEQUENT_SUPPORT	NUMBER
CONSEQUENT_NAME	VARCHAR2(4000)
CONSEQUENT_VALUE	VARCHAR2(4000)
ANTECEDENT	SYS.XMLTYPE

Table 5-11 (Cont.) Rule View Columns for Transactional Inputs

Column Name	Description
ANTECEDENT	<p>The independent condition in the rule. When this condition exists, the dependent condition in the consequent also exists. The condition is a combination of attribute values called a predicate (DM_PREDICATE). The predicate specifies a condition for each attribute. The condition can specify equality (=), inequality (<>), greater than (>), less than (<), greater than or equal to (>=), or less than or equal to (<=) a given value. Support and confidence for each attribute condition in the antecedent is returned in the predicate. Support is the number of transactions that satisfy the antecedent. Confidence is the likelihood that a transaction satisfies the antecedent.</p> <p>ANTECEDENT also contains the ITEM_VALUE (type number) part for each antecedent item.</p> <hr/> <p>Note: The occurrence of the attribute as a DM_PREDICATE indicates the presence of the item in the transaction. The actual value for attribute_num_value or attribute_str_value is meaningless. For example, the following predicate indicates that 'Mouse Pad' is present in the transaction even though the attribute value is NULL.</p> <pre>DM_PREDICATE('PROD_NAME', 'Mouse Pad', '= ', NULL, NULL, NULL, NULL)</pre>

Transactional Input With ASSO_AGGREGATES Setting

Similar to the view without aggregates setting, there are three transactional cases. The following are the cases:

- Rule view when ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is not set.
- Rule view when ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is set with TYPE as numerical, the view has a CONSEQUENT_VALUE column.
- Rule view when ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is set with TYPE as categorical, the view has a CONSEQUENT_VALUE column.

The view with ASSO_AGGREGATES has columns for the aggregates output (four columns per aggregate). The 2-Dimensional input does not allow aggregates setting.

Example 5-13 Examples

The following example shows profit and sales set to be aggregated:

Name	Type
PARTITION_NAME	VARCHAR2(128)
RULE_ID	NUMBER
RULE_SUPPORT	NUMBER
RULE_CONFIDENCE	NUMBER
RULE_LIFT	NUMBER
RULE_REVCONFIDENCE	NUMBER

ANTECEDENT_SUPPORT	NUMBER
NUMBER_OF_ITEMS	NUMBER
CONSEQUENT_SUPPORT	NUMBER
CONSEQUENT_NAME	VARCHAR2(4000)
ANTECEDENT	SYS.XMLTYPE
ANT_RULE_PROFIT	BINARY_DOUBLE
CON_RULE_PROFIT	BINARY_DOUBLE
ANT_PROFIT	BINARY_DOUBLE
CON_PROFIT	BINARY_DOUBLE
ANT_RULE_SALES	BINARY_DOUBLE
CON_RULE_SALES	BINARY_DOUBLE
ANT_SALES	BINARY_DOUBLE
CON_SALES	BINARY_DOUBLE

Rule view when ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is set with TYPE as numerical, the view has a CONSEQUENT_VALUE column.

Name	Null?	Type

PARTITION_NAME		VARCHAR2(128)
RULE_ID		NUMBER
RULE_SUPPORT		NUMBER
RULE_CONFIDENCE		NUMBER
RULE_LIFT		NUMBER
RULE_REVCONFIDENCE		NUMBER
ANTECEDENT_SUPPORT		NUMBER
NUMBER_OF_ITEMS		NUMBER
CONSEQUENT_SUPPORT		NUMBER
CONSEQUENT_NAME		VARCHAR2(4000)
CONSEQUENT_VALUE		NUMBER
ANTECEDENT		SYS.XMLTYPE
ANT_RULE_PROFIT		BINARY_DOUBLE
CON_RULE_PROFIT		BINARY_DOUBLE
ANT_PROFIT		BINARY_DOUBLE
CON_PROFIT		BINARY_DOUBLE
ANT_RULE_SALES		BINARY_DOUBLE
CON_RULE_SALES		BINARY_DOUBLE
ANT_SALES		BINARY_DOUBLE
CON_SALES		BINARY_DOUBLE

Rule view when ODMS_ITEM_ID_COLUMN_NAME is set and Item_value (ODMS_ITEM_VALUE_COLUMN_NAME) is set with TYPE as categorical, the view has a CONSEQUENT_VALUE column.

Name	Null?	Type

PARTITION_NAME		VARCHAR2(128)
RULE_ID		NUMBER
RULE_SUPPORT		NUMBER
RULE_CONFIDENCE		NUMBER
RULE_LIFT		NUMBER
RULE_REVCONFIDENCE		NUMBER
ANTECEDENT_SUPPORT		NUMBER
NUMBER_OF_ITEMS		NUMBER
CONSEQUENT_SUPPORT		NUMBER
CONSEQUENT_NAME		VARCHAR2(4000)
CONSEQUENT_VALUE		VARCHAR2(4000)
ANTECEDENT		SYS.XMLTYPE
ANT_RULE_PROFIT		BINARY_DOUBLE
CON_RULE_PROFIT		BINARY_DOUBLE

ANT_PROFIT	BINARY_DOUBLE
CON_PROFIT	BINARY_DOUBLE
ANT_RULE_SALES	BINARY_DOUBLE
CON_RULE_SALES	BINARY_DOUBLE
ANT_SALES	BINARY_DOUBLE
CON_SALES	BINARY_DOUBLE

5.4.2 Model Detail View for Frequent Itemsets

Model detail view for Frequent Itemsets describes the frequent itemsets view. Oracle recommends that you leverage model details view instead of the GET_FREQUENT_ITEMSETS function.

The frequent itemsets view DM\$VI`model_name` has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2 (128)
ITEMSET_ID	NUMBER
SUPPORT	NUMBER
NUMBER_OF_ITEMS	NUMBER
ITEMSET	SYS.XMLTYPE

Table 5-12 Frequent Itemsets View

Column Name	Description
PARTITION_NAME	A partition in a partitioned model
ITEMSET_ID	Itemset identifier
SUPPORT	Support of the itemset
NUMBER_OF_ITEMS	Number of items in the itemset
ITEMSET	Frequent itemset The structure of the SYS.XMLTYPE column itemset is the same as the corresponding Antecedent column of the rule view.

5.4.3 Model Detail View for Transactional Itemsets

Model detail view for Transactional Itemsets describes the transactional itemsets view. Oracle recommends that users leverage the model details views.

For the very common case of transactional data without aggregates, DM\$VT`model_name` view provides the itemsets information in transactional format. This view can help improve performance for some queries as compared to the view with the XML column. The transactional itemsets view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
ITEMSET_ID	NUMBER
ITEM_ID	NUMBER
SUPPORT	NUMBER
NUMBER_OF_ITEMS	NUMBER
ITEM_NAME	VARCHAR2(4000)

Table 5-13 Transactional Itemsets View

Column Name	Description
PARTITION_NAME	A partition in a partitioned model
ITEMSET_ID	Itemset identifier
ITEM_ID	Item identifier
SUPPORT	Support of the itemset
NUMBER_OF_ITEMS	Number of items in the itemset
ITEM_NAME	The name of the item

5.4.4 Model Detail View for Transactional Rule

Model detail view for Transactional Rule describes the transactional rule view and transactional itemsets view. Oracle recommends that you leverage model details views.

Transactional data without aggregates also has a transactional rule view DM $\$VAmode_name$. This view can improve performance for some queries as compared to the view with the XML column. The transactional rule view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
RULE_ID	NUMBER
ANTECEDENT_PREDICATE	VARCHAR2(4000)
CONSEQUENT_PREDICATE	VARCHAR2(4000)
RULE_SUPPORT	NUMBER
RULE_CONFIDENCE	NUMBER
RULE_LIFT	NUMBER
RULE_REVCONFIDENCE	NUMBER
RULE_ITEMSET_ID	NUMBER
ANTECEDENT_SUPPORT	NUMBER
CONSEQUENT_SUPPORT	NUMBER
NUMBER_OF_ITEMS	NUMBER

Table 5-14 Transactional Rule View

Column Name	Description
PARTITION_NAME	A partition in a partitioned model
RULE_ID	Rule identifier
ANTECEDENT_PREDICATE	Name of the Antecedent item.
CONSEQUENT_PREDICATE	Name of the Consequent item
RULE_SUPPORT	Support of the rule
RULE_CONFIDENCE	The likelihood a transaction satisfies the rule when it contains the Antecedent.

Table 5-14 (Cont.) Transactional Rule View

Column Name	Description
RULE_LIFT	The degree of improvement in the prediction over random chance when the rule is satisfied
RULE_REVCONFIDENCE	The number of transactions in which the rule occurs divided by the number of transactions in which the consequent occurs
RULE_ITEMSET_ID	Itemset identifier
ANTECEDENT_SUPPORT	The ratio of the number of transactions that satisfy the antecedent to the total number of transactions
CONSEQUENT_SUPPORT	The ratio of the number of transactions that satisfy the consequent to the total number of transactions
NUMBER_OF_ITEMS	Number of items in the rule

5.4.5 Model Detail Views for Classification Algorithms

Model detail view for Classification algorithms describe target map view and scoring cost view which are applicable to all Classification algorithms. Oracle recommends that users leverage the model details views instead of the GET_* function.

The target map view `DM$VTmodel_name` describes the target distribution for Classification models. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
TARGET_VALUE	NUMBER/VARCHAR2
TARGET_COUNT	NUMBER
TARGET_WEIGHT	NUMBER

Table 5-15 Target Map View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
TARGET_VALUE	Target value, numerical or categorical
TARGET_COUNT	Number of rows for a given TARGET_VALUE
TARGET_WEIGHT	Weight for a given TARGET_VALUE

The scoring cost view `DM$VCmodel_name` describes the scoring cost matrix for Classification models. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
ACTUAL_TARGET_VALUE	NUMBER/VARCHAR2
PREDICTED_TARGET_VALUE	NUMBER/VARCHAR2
COST	NUMBER

Table 5-16 Scoring Cost View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
ACTUAL_TARGET_VALUE	A valid target value
PREDICTED_TARGET_VALUE	Predicted target value
COST	Associated cost for the actual and predicted target value pair

5.4.6 Model Detail Views for Decision Tree

Model detail view for Decision Tree describes the split information view, node statistics view, node description view, and the cost matrix view. Oracle recommends that users leverage the model details views instead of GET_MODEL_DETAILS_XML function.

The split information view `DM$VPmodel_name` describes the decision tree hierarchy and the split information for each level in the Decision Tree. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
PARENT	NUMBER
SPLIT_TYPE	VARCHAR2
NODE	NUMBER
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
OPERATOR	VARCHAR2
VALUE	SYS.XMLTYPE

Table 5-17 Split Information View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
PARENT	Node ID of the parent
SPLIT_TYPE	The main or surrogate split
NODE	The node ID
ATTRIBUTE_NAME	The attribute used as the splitting criterion at the parent node to produce this node.
ATTRIBUTE_SUBNAME	Split attribute subname. The value is null for non-nested columns.
OPERATOR	Split operator

Table 5-17 (Cont.) Split Information View

Column Name	Description
VALUE	Value used as the splitting criterion. This is an XML element described using the <Element> tag. For example, <Element>Windy</Element><Element>Hot</Element>.

The node statistics view `DM$VImodel_name` describes the statistics associated with individual tree nodes. The statistics include a target histogram for the data in the node. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
NODE	NUMBER
NODE_SUPPORT	NUMBER
PREDICTED_TARGET_VALUE	NUMBER/VARCHAR2
TARGET_VALUE	NUMBER/VARCHAR2
TARGET_SUPPORT	NUMBER

Table 5-18 Node Statistics View

Parameter	Description
PARTITION_NAME	Partition name in a partitioned model
NODE	The node ID
NODE_SUPPORT	Number of records in the training set that belong to the node
PREDICTED_TARGET_VALUE	Predicted Target value
TARGET_VALUE	A target value seen in the training data
TARGET_SUPPORT	The number of records that belong to the node and have the value specified in the TARGET_VALUE column

Higher level node description can be found in `DM$VOModel_name` view. The `DM$VOModel_name` has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
NODE	NUMBER
NODE_SUPPORT	NUMBER
PREDICTED_TARGET_VALUE	NUMBER/VARCHAR2
PARENT	NUMBER
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
OPERATOR	VARCHAR2
VALUE	SYS.XMLTYPE

Table 5-19 Node Description View

Parameter	Description
PARTITION_NAME	Partition name in a partitioned model
NODE	The node ID
NODE_SUPPORT	Number of records in the training set that belong to the node
PREDICTED_TARGET_VALUE	Predicted Target value
PARENT	The ID of the parent
ATTRIBUTE_NAME	Specifies the attribute name
ATTRIBUTE_SUBNAME	Specifies the attribute subname
OPERATOR	Attribute predicate operator - a conditional operator taking the following values: <i>IN, =, <>, <, >, <=, and >=</i>
VALUE	Value used as the description criterion. This is an XML element described using the <Element> tag. For example, <Element>Windy</Element><Element>Hot</Element>.

The `DM$VModel_name` view describes the cost matrix used by the Decision Tree build. The `DM$VModel_name` view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
ACTUAL_TARGET_VALUE	NUMBER/VARCHAR2
PREDICTED_TARGET_VALUE	NUMBER/VARCHAR2
COST	NUMBER

Table 5-20 Cost Matrix View

Parameter	Description
PARTITION_NAME	Partition name in a partitioned model
ACTUAL_TARGET_VALUE	Valid target value
PREDICTED_TARGET_VALUE	Predicted Target value
COST	Associated cost for the actual and predicted target value pair

5.4.7 Model Detail Views for Generalized Linear Model

Model details views for Generalized Linear Model (GLM) describes the model details view and row diagnostic view for Linear and Logistic Regression. Oracle recommends that users leverage model details views than the `GET_MODEL_DETAILS_GLM` function.

The model details view `DM$VDmodel_name` describes the final model information for both Linear Regression models and Logistic Regression models.

For Linear Regression, the view `DM$VDmodel_name` has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
FEATURE_EXPRESSION	VARCHAR2(4000)
COEFFICIENT	BINARY_DOUBLE
STD_ERROR	BINARY_DOUBLE
TEST_STATISTIC	BINARY_DOUBLE
P_VALUE	BINARY_DOUBLE
VIF	BINARY_DOUBLE
STD_COEFFICIENT	BINARY_DOUBLE
LOWER_COEFF_LIMIT	BINARY_DOUBLE
UPPER_COEFF_LIMIT	BINARY_DOUBLE

For Logistic Regression, the view `DM$VDmodel_name` has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
TARGET_VALUE	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
FEATURE_EXPRESSION	VARCHAR2(4000)
COEFFICIENT	BINARY_DOUBLE
STD_ERROR	BINARY_DOUBLE
TEST_STATISTIC	BINARY_DOUBLE
P_VALUE	BINARY_DOUBLE
STD_COEFFICIENT	BINARY_DOUBLE
LOWER_COEFF_LIMIT	BINARY_DOUBLE
UPPER_COEFF_LIMIT	BINARY_DOUBLE
EXP_COEFFICIENT	BINARY_DOUBLE
EXP_LOWER_COEFF_LIMIT	BINARY_DOUBLE
EXP_UPPER_COEFF_LIMIT	BINARY_DOUBLE

Table 5-21 Model View for Linear and Logistic Regression Models

Column Name	Description
PARTITION_NAME	The name of a feature in the model
TARGET_VALUE	Valid target value
ATTRIBUTE_NAME	The attribute name when there is no subname, or first part of the attribute name when there is a subname. <code>ATTRIBUTE_NAME</code> is the name of a column in the source table or view. If the column is a non-nested, numeric column, then <code>ATTRIBUTE_NAME</code> is the name of the mining attribute. For the intercept, <code>ATTRIBUTE_NAME</code> is null. Intercepts are equivalent to the bias term in SVM models.

Table 5-21 (Cont.) Model View for Linear and Logistic Regression Models

Column Name	Description
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns. When the nested column is numeric, the mining attribute is identified by the combination ATTRIBUTE_NAME - ATTRIBUTE_SUBNAME. If the column is not nested, ATTRIBUTE_SUBNAME is null. If the attribute is an intercept, both the ATTRIBUTE_NAME and the ATTRIBUTE_SUBNAME are null.
ATTRIBUTE_VALUE	A unique value that can be assumed by a categorical column or nested categorical column. For categorical columns, a mining attribute is identified by a unique ATTRIBUTE_NAME . ATTRIBUTE_VALUE pair. For nested categorical columns, a mining attribute is identified by the combination: ATTRIBUTE_NAME . ATTRIBUTE_SUBNAME . ATTRIBUTE_VALUE. For numerical attributes, ATTRIBUTE_VALUE is null.
FEATURE_EXPRESSION	The feature name constructed by the algorithm when feature selection is enabled. If feature selection is not enabled, the feature name is simply the fully-qualified attribute name (<i>attribute_name.attribute_subname</i> if the attribute is in a nested column). For categorical attributes, the algorithm constructs a feature name that has the following form: <i>fully-qualified_attribute_name.attribute_value</i> When feature generation is enabled, a term in the model can be a single mining attribute or the product of up to 3 mining attributes. Component mining attributes can be repeated within a single term. If feature generation is not enabled or, if feature generation is enabled, but no multiple component terms are discovered by the CREATE model process, then FEATURE_EXPRESSION is null.
<hr/> <p>Note: In 12c Release 2, the algorithm does not subtract the mean from numerical components.</p> <hr/>	
COEFFICIENT	The estimated coefficient.
STD_ERROR	Standard error of the coefficient estimate.
TEST_STATISTIC	For Linear Regression, the t-value of the coefficient estimate. For Logistic Regression, the Wald chi-square value of the coefficient estimate.
P_VALUE	Probability of the TEST_STATISTIC under the (NULL) hypothesis that the term in the model is not statistically significant. A low probability indicates that the term is significant, while a high probability indicates that the term can be better discarded. Used to analyze the significance of specific attributes in the model.
VIF	Variance Inflation Factor. The value is zero for the intercept. For Logistic Regression, VIF is null.

Table 5-21 (Cont.) Model View for Linear and Logistic Regression Models

Column Name	Description
STD_COEFFICIENT	Standardized estimate of the coefficient.
LOWER_COEFF_LIMIT	Lower confidence bound of the coefficient.
UPPER_COEFF_LIMIT	Upper confidence bound of the coefficient.
EXP_COEFFICIENT	Exponentiated coefficient for Logistic Regression. For linear regression, EXP_COEFFICIENT is null.
EXP_LOWER_COEFF_LIMIT	Exponentiated coefficient for lower confidence bound of the coefficient for Logistic Regression. For Linear Regression, EXP_LOWER_COEFF_LIMIT is null.
EXP_UPPER_COEFF_LIMIT	Exponentiated coefficient for upper confidence bound of the coefficient for Logistic Regression. For Linear Regression, EXP_UPPER_COEFF_LIMIT is null.

The row diagnostic view `DM$VAmodel_name` describes row level information for both Linear Regression models and Logistic Regression models. For Linear Regression, the view `DM$VAmodel_name` has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CASE_ID	NUMBER/VARHCAR2
TARGET_VALUE	BINARY_DOUBLE
PREDICTED_TARGET_VALUE	BINARY_DOUBLE
Hat	BINARY_DOUBLE
RESIDUAL	BINARY_DOUBLE
STD_ERR_RESIDUAL	BINARY_DOUBLE
STUDENTIZED_RESIDUAL	BINARY_DOUBLE
PRED_RES	BINARY_DOUBLE
COOKS_D	BINARY_DOUBLE

Table 5-22 Row Diagnostic View for Linear Regression

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
CASE_ID	Name of the case identifier
TARGET_VALUE	The actual target value as taken from the input row
PREDICTED_TARGET_VALUE	The model predicted target value for the row
HAT	The diagonal element of the $n \times n$ (n =number of rows) that the Hat matrix identifies with a specific input row. The model predictions for the input data are the product of the Hat matrix and vector of input target values. The diagonal elements (Hat values) represent the influence of the i^{th} row on the i^{th} fitted value. Large Hat values are indicators that the i^{th} row is a point of high leverage, a potential outlier.

Table 5-22 (Cont.) Row Diagnostic View for Linear Regression

Column Name	Description
RESIDUAL	The difference between the predicted and actual target value for a specific input row.
STD_ERR_RESIDUAL	The standard error residual, sometimes called the Studentized residual, re-scales the residual to have constant variance across all input rows in an effort to make the input row residuals comparable. The process multiplies the residual by square root of the row weight divided by the product of the model mean square error and 1 minus the Hat value.
STUDENTIZED_RESIDUAL	Studentized deletion residual adjusts the standard error residual for the influence of the current row.
PRED_RES	The predictive residual is the weighted square of the deletion residuals, computed as the row weight multiplied by the square of the residual divided by 1 minus the Hat value.
COOKS_D	Cook's distance is a measure of the combined impact of the i^{th} case on all of the estimated regression coefficients.

For Logistic Regression, the view `DM$VAmo $_$ name` has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CASE_ID	NUMBER/VARHCAR2
TARGET_VALUE	NUMBER/VARCHAR2
TARGET_VALUE_PROB	BINARY_DOUBLE
Hat	BINARY_DOUBLE
WORKING_RESIDUAL	BINARY_DOUBLE
PEARSON_RESIDUAL	BINARY_DOUBLE
DEVIANCE_RESIDUAL	BINARY_DOUBLE
C	BINARY_DOUBLE
CBAR	BINARY_DOUBLE
DIFDEV	BINARY_DOUBLE
DIFCHISQ	BINARY_DOUBLE

Table 5-23 Row Diagnostic View for Logistic Regression

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
CASE_ID	Name of the case identifier
TARGET_VALUE	The actual target value as taken from the input row
TARGET_VALUE_PROB	Model estimate of the probability of the predicted target value.
Hat	The Hat value concept from Linear Regression is extended to Logistic Regression by multiplying the Linear Regression Hat value by the variance function for Logistic Regression, the predicted probability multiplied by 1 minus the predicted probability.

Table 5-23 (Cont.) Row Diagnostic View for Logistic Regression

Column Name	Description
WORKING_RESIDUAL	<p>The working residual is the residual of the working response. The working response is the response on the linearized scale. For Logistic Regression it has the form: the i^{th} row residual divided by the variance of the i^{th} row prediction. The variance of the prediction is the predicted probability multiplied by 1 minus the predicted probability.</p> <p>WORKING_RESIDUAL is the difference between the working response and the linear predictor at convergence.</p>
PEARSON_RESIDUAL	<p>The Pearson residual is a re-scaled version of the working residual, accounting for the weight. For Logistic Regression, the Pearson residual multiplies the residual by a factor that is computed as square root of the weight divided by the variance of the predicted probability for the i^{th} row.</p> <p>RESIDUAL is 1 minus the predicted probability of the actual target value for the row.</p>
DEVIANCE_RESIDUAL	<p>The DEVIANCE_RESIDUAL is the contribution to the model deviance of the i^{th} observation. For Logistic Regression it has the form the square root of 2 times the $\log(1 + e^{\eta}) - \eta$ for the non-reference class and -square root of 2 time the $\log(1 + e^{\eta})$ for the reference class, where η is the linear prediction (the prediction as if the model were a Linear Regression).</p>
C	<p>Measures the overall change in the fitted logits due to the deletion of the i^{th} observation for all points including the one deleted (the i^{th} point). It is computed as the square of the Pearson residual multiplied by the Hat value divided by the square of 1 minus the Hat value.</p> <p>Confidence interval displacement diagnostics that provides scalar measure of the influence of individual observations.</p>
CBAR	<p>C and CBAR are extensions of Cooks' distance for Logistic Regression. CBAR measures the overall change in the fitted logits due to the deletion of the i^{th} observation for all points excluding the one deleted (the i^{th} point). It is computed as the square of the Pearson residual multiplied by the Hat value divided by (1 minus the Hat value)</p> <p>Confidence interval displacement diagnostic which measures the influence of deleting an individual observation.</p>
DIFDEV	<p>A statistic that measures the change in deviance that occurs when an observation is deleted from the input. It is computed as the square of the deviance residual plus CBAR.</p>
DIFCHISQ	<p>A statistic that measures the change in the Pearson chi-square statistic that occurs when an observation is deleted from the input. It is computed as CBAR divided by the Hat value.</p>

5.4.8 Model Detail Views for Naive Bayes

Model Detail Views for Naive Bayes describes prior view and result view. Oracle recommends that users leverage the model details views instead of the `GET_MODEL_DETAILS_NB` function.

The prior view `DM$VPmodel_name` describes the priors of the targets for Naive Bayes. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
TARGET_NAME	VARCHAR2(128)
TARGET_VALUE	NUMBER/VARCHAR2
PRIOR_PROBABILITY	BINARY_DOUBLE
COUNT	NUMBER

Table 5-24 Prior View for Naive Bayes

Column Name	Description
PARTITION_NAME	The name of a feature in the model
TARGET_NAME	Name of the target column
TARGET_VALUE	Target value, numerical or categorical
PRIOR_PROBABILITY	Prior probability for a given TARGET_VALUE
COUNT	Number of rows for a given TARGET_VALUE

The Naive Bayes result view `DM$VVmodel_view` describes the conditional probabilities of the Naive Bayes model. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
TARGET_NAME	VARCHAR2(128)
TARGET_VALUE	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
CONDITIONAL_PROBABILITY	BINARY_DOUBLE
COUNT	NUMBER

Table 5-25 Result View for Naive Bayes

Column Name	Description
PARTITION_NAME	The name of a feature in the model
TARGET_NAME	Name of the target column
TARGET_VALUE	Target value, numerical or categorical
ATTRIBUTE_NAME	Column name

Table 5-25 (Cont.) Result View for Naive Bayes

Column Name	Description
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Mining attribute value for the column ATTRIBUTE_NAME or the nested column ATTRIBUTE_SUBNAME (if any).
CONDITIONAL_PROBABILITY	Conditional probability of a mining attribute for a given target
COUNT	Number of rows for a given mining attribute and a given target

5.4.9 Model Detail View for Support Vector Machine

Model Detail View for Support Vector Machine describes linear coefficient view. Oracle recommends that users leverage the model details views instead of the `GET_MODEL_DETAILS_SVM` function.

The linear coefficient view `DM$VLMODEL_name` describes the coefficients of a linear SVM algorithm. The `target_value` field in the view is present only for Classification and has the type of the target. Regression models do not have a `target_value` field.

The `reversed_coefficient` field shows the value of the coefficient after reversing the automatic data preparation transformations. If data preparation is disabled, then `coefficient` and `reversed_coefficient` have the same value. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
TARGET_VALUE	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
COEFFICIENT	BINARY_DOUBLE
REVERSED_COEFFICIENT	BINARY_DOUBLE

Table 5-26 Linear Coefficient View for Support Vector Machine

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
TARGET_VALUE	Target value, numerical or categorical
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Value of a categorical attribute

Table 5-26 (Cont.) Linear Coefficient View for Support Vector Machine

Column Name	Description
COEFFICIENT	Projection coefficient value
REVERSED_COEFFICIENT	Coefficient transformed on the original scale

5.4.10 Model Detail Views for Clustering Algorithms

Oracle Data Mining supports these clustering algorithms: Expectation Maximization, *k*-Means, and Orthogonal Partitioning Clustering (O-Cluster).

All clustering algorithms share the following views:

- Cluster description `DM$VDmodel_name`
- Attribute statistics `DM$VAmode_name`
- Histogram statistics `DM$VHmodel_name`
- Rule statistics `DM$VRmodel_name`

The cluster description view `DM$VDmodel_name` describes cluster level information about a clustering model. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CLUSTER_ID	NUMBER
CLUSTER_NAME	NUMBER/VARCHAR2
RECORD_COUNT	NUMBER
PARENT	NUMBER
TREE_LEVEL	NUMBER
LEFT_CHILD_ID	NUMBER
RIGHT_CHILD_ID	NUMBER

Table 5-27 Cluster Description View for Clustering Algorithm

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
CLUSTER_ID	The ID of a cluster in the model
CLUSTER_NAME	Specifies the label of the cluster
RECORD_COUNT	Specifies the number of records
PARENT	The ID of the parent
TREE_LEVEL	Specifies the number of splits from the root
LEFT_CHILD_ID	The ID of the child cluster on the left side of the split
RIGHT_CHILD_ID	The ID of the child cluster on the right side of the split

The attribute view `DM$VAmodeI_name` describes attribute level information about a Clustering model. The values of the mean, variance, and mode for a particular cluster can be obtained from this view. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CLUSTER_ID	NUMBER
CLUSTER_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
MEAN	BINARY_DOUBLE
VARIANCE	BINARY_DOUBLE
MODE_VALUE	VARCHAR2(4000)

Table 5-28 Attribute View for Clustering Algorithm

Column Name	Description
PARTITION_NAME	A partition in a partitioned model
CLUSTER_ID	The ID of a cluster in the model
CLUSTER_NAME	Specifies the label of the cluster
ATTRIBUTE_NAME	Specifies the attribute name
ATTRIBUTE_SUBNAME	Specifies the attribute subname
MEAN	The field returns the average value of a numeric attribute
VARIANCE	The variance of a numeric attribute
MODE_VALUE	The mode is the most frequent value of a categorical attribute

The histogram view `DM$VHmodeI_name` describes histogram level information about a Clustering model. The bin information as well as bin counts can be obtained from this view. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CLUSTER_ID	NUMBER
CLUSTER_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
BIN_ID	NUMBER
LOWER_BIN_BOUNDARY	BINARY_DOUBLE
UPPER_BIN_BOUNDARY	BINARY_DOUBLE
ATTRIBUTE_VALUE	VARCHAR2(4000)
COUNT	NUMBER

Table 5-29 Histogram View for Clustering Algorithm

Column Name	Description
PARTITION_NAME	A partition in a partitioned model

Table 5-29 (Cont.) Histogram View for Clustering Algorithm

Column Name	Description
CLUSTER_ID	The ID of a cluster in the model
CLUSTER_NAME	Specifies the label of the cluster
ATTRIBUTE_NAME	Specifies the attribute name
ATTRIBUTE_SUBNAME	Specifies the attribute subname
BIN_ID	Bin ID
LOWER_BIN_BOUNDARY	Numeric lower bin boundary
UPPER_BIN_BOUNDARY	Numeric upper bin boundary
ATTRIBUTE_VALUE	Categorical attribute value
COUNT	Histogram count

The rule view `DM$VRmodel_name` describes the rule level information about a Clustering model. The information is provided at attribute predicate level. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CLUSTER_ID	NUMBER
CLUSTER_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
OPERATOR	VARCHAR2(2)
NUMERIC_VALUE	NUMBER
ATTRIBUTE_VALUE	VARCHAR2(4000)
SUPPORT	NUMBER
CONFIDENCE	BINARY_DOUBLE
RULE_SUPPORT	NUMBER
RULE_CONFIDENCE	BINARY_DOUBLE

Table 5-30 Rule View for Clustering Algorithm

Column Name	Description
PARTITION_NAME	A partition in a partitioned model
CLUSTER_ID	The ID of a cluster in the model
CLUSTER_NAME	Specifies the label of the cluster
ATTRIBUTE_NAME	Specifies the attribute name
ATTRIBUTE_SUBNAME	Specifies the attribute subname
OPERATOR	Attribute predicate operator - a conditional operator taking the following values: <i>IN</i> , =, <>, <, >, <=, and >=

Table 5-30 (Cont.) Rule View for Clustering Algorithm

Column Name	Description
NUMERIC_VALUE	Numeric lower bin boundary
ATTRIBUTE_VALUE	Categorical attribute value
SUPPORT	Attribute predicate support
CONFIDENCE	Attribute predicate confidence
RULE_SUPPORT	Rule level support
RULE_CONFIDENCE	Rule level confidence

5.4.11 Model Detail Views for Expectation Maximization

Model detail views for Expectation Maximization (EM) describes the differences in the views for EM against those of Clustering views. Oracle recommends that user leverage the model details views instead of the `GET_MODEL_DETAILS_EM` function.

The following views are the differences in the views for Expectation Maximization against Clustering views. For an overview of the different Clustering views, refer to "Model Detail Views for Clustering Algorithms".

The component view `DM$Vmodel_name` describes the EM components. The component view contains information about their prior probabilities and what cluster they map to. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
COMPONENT_ID	NUMBER
CLUSTER_ID	NUMBER
PRIOR_PROBABILITY	BINARY_DOUBLE

Table 5-31 Component View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
COMPONENT_ID	Unique identifier of a component
CLUSTER_ID	The ID of a cluster in the model
PRIOR_PROBABILITY	Component prior probability

The mean and variance component view `DM$VMmodel_name` provides information about the mean and variance parameters for the attributes by Gaussian distribution models. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
COMPONENT_ID	NUMBER
ATTRIBUTE_NAME	VARCHAR2(4000)

MEAN	BINARY_DOUBLE
VARIANCE	BINARY_DOUBLE

The frequency component view `DM$VFmodel_name` provides information about the parameters of the multi-valued Bernoulli distributions used by the EM model. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
COMPONENT_ID	NUMBER
ATTRIBUTE_NAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
FREQUENCY	BINARY_DOUBLE

Table 5-32 Frequency Component View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
COMPONENT_ID	Unique identifier of a component
ATTRIBUTE_NAME	Column name
ATTRIBUTE_VALUE	Categorical attribute value
FREQUENCY	The frequency of the multivalued Bernoulli distribution for the attribute/value combination specified by ATTRIBUTE_NAME and ATTRIBUTE_VALUE.

For 2-Dimensional columns, EM provides an attribute ranking similar to that of Attribute Importance. This ranking is based on a rank-weighted average over Kullback–Leibler divergence computed for pairs of columns. This unsupervised Attribute Importance is shown in the `DM$VImodel_name` view and has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_IMPORTANCE_VALUE	BINARY_DOUBLE
ATTRIBUTE_RANK	NUMBER

Table 5-33 2-Dimensional Attribute Ranking for Expectation Maximization

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
ATTRIBUTE_NAME	Column name
ATTRIBUTE_IMPORTANCE_VALUE	Importance value
ATTRIBUTE_RANK	An attribute rank based on the importance value

The pairwise Kullback–Leibler divergence is reported in the `DM$VBmodel_name` view. This metric evaluates how much the observed joint distribution of two attributes diverges from the expected distribution under the assumption of independence. That is, the higher the value, the more dependent the two attributes are. The dependency value is scaled based on the size of the grid used for each pairwise computation. That ensures that all values fall within the [0; 1] range and are comparable. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
ATTRIBUTE_NAME_1	VARCHAR2(128)
ATTRIBUTE_NAME_2	VARCHAR2(128)
DEPENDENCY	BINARY_DOUBLE

Table 5-34 Kullback-Leibler Divergence for Expectation Maximization

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
ATTRIBUTE_NAME_1	Name of an attribute 1
ATTRIBUTE_NAME_2	Name of an attribute 2
DEPENDENCY	Scaled pairwise Kullback-Leibler divergence

The projection table `DM$VPmodel_name` shows the coefficients used by random projections to map nested columns to a lower dimensional space. The view has rows only when nested or text data is present in the build data. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
FEATURE_NAME	VARCHAR2(4000)
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
COEFFICIENT	NUMBER

Table 5-35 Projection table for Expectation Maximization

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_NAME	Name of feature
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Categorical attribute value
COEFFICIENT	Projection coefficient. The representation is sparse; only the non-zero coefficients are returned.

Related Topics:

[Model Detail Views for Clustering Algorithms](#) (page 5-39)

Oracle Data Mining supports these clustering algorithms: Expectation Maximization, *k*-Means, and Orthogonal Partitioning Clustering (O-Cluster).

5.4.12 Model Detail Views for *k*-Means

Model detail views for *k*-Means (KM) describes cluster description view and scoring view. Oracle recommends that you leverage model details view instead of GET_MODEL_DETAILS_KM function.

This section describes the differences in the views for *k*-Means against the Clustering views. For an overview of the different views, refer to "Model Detail Views for Clustering Algorithms". For *k*-Means, the cluster description view DM\$VD`model_name` has an additional column:

Name	Type
DISPERSION	BINARY_DOUBLE

Table 5-36 Cluster Description for *k*-Means

Column Name	Description
DISPERSION	A measure used to quantify whether a set of observed occurrences are dispersed compared to a standard statistical model.

The scoring view DM\$VC`model_name` describes the centroid of each leaf clusters:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CLUSTER_ID	NUMBER
CLUSTER_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
VALUE	BINARY_DOUBLE

Table 5-37 Scoring View for *k*-Means

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
CLUSTER_ID	The ID of a cluster in the model
CLUSTER_NAME	Specifies the label of the cluster
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Categorical attribute value

Table 5-37 (Cont.) Scoring View for k-Means

Column Name	Description
VALUE	Specifies the centroid value

Related Topics:

[Model Detail Views for Clustering Algorithms](#) (page 5-39)

Oracle Data Mining supports these clustering algorithms: Expectation Maximization, *k*-Means, and Orthogonal Partitioning Clustering (O-Cluster).

5.4.13 Model Detail Views for O-Cluster

Model Detail Views for O-Cluster describes the statistics views. Oracle recommends that user leverage the model details views instead of the `GET_MODEL_DETAILS_OC` function.

The following are the differences in the views for O-Cluster against Clustering views. For an overview of the different clustering views, refer to "Model Detail Views for Clustering Algorithms". The OC algorithm uses the same descriptive statistics views as Expectation Maximization (EM) and *k*-Means (KM). The following are the statistics views:

- Cluster description `DM$VDmodel_name`
- Attribute statistics `DM$VAmode_name`
- Rule statistics `DM$VRmodel_name`
- Histogram statistics `DM$VHmodel_name`

The Cluster description view `DM$VDmodel_name` describes the O-Cluster components. The cluster description view has additional fields that specify the split predicate. The view has the following schema:

Name	Type
-----	-----
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
OPERATOR	VARCHAR2(2)
VALUE	SYS.XMLTYPE

Table 5-38 Description View

Column Name	Description
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
OPERATOR	Split operator
VALUE	List of split values

The structure of the `SYS.XMLTYPE` is as follows:

<Element>splitvall</Element>

The OC algorithm uses a histogram view `DM$VHmodel_name` with a different schema than EM and k-Means (KM). The view has the following schema:

Name	Type
-----	-----
PARTITON_NAME	VARCHAR2(128)
CLUSTER_ID	NUMBER
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
BIN_ID	NUMBER
LABEL	VARCHAR2(4000)
COUNT	NUMBER

Table 5-39 Histogram Component View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
CLUSTER_ID	Unique identifier of a component
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
BIN_ID	Unique identifier
LABEL	Bin label
COUNT	Bin histogram count

Related Topics:

[Model Detail Views for Clustering Algorithms](#) (page 5-39)

Oracle Data Mining supports these clustering algorithms: Expectation Maximization, *k*-Means, and Orthogonal Partitioning Clustering (O-Cluster).

5.4.14 Model Detail Views for Explicit Semantic Analysis

Model Detail Views for Explicit Semantic Analysis (ESA) describes attribute statistics view and feature view. Oracle recommends that users leverage the model details views.

ESA algorithm has the following descriptive statistics views:

- Attribute statistics `DM$VAmode_name`
- Features `DM$VFmodel_name`

The view `DM$VAmode_name` has the following schema:

PARTITION_NAME	VARCHAR2(128)
FEATURE_ID	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)

ATTRIBUTE_VALUE	VARCHAR2(4000)
COEFFICIENT	BINARY_DOUBLE

Table 5-40 Attribute View for Explicit Semantic Analysis

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_ID	Unique identifier of a feature as it appears in the training data
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Categorical attribute value
COEFFICIENT	A measure of the weight of the attribute with respect to the feature

The view `DM$VFmodel_name` has a unique row for every feature in one view. This feature is helpful if the model was pre-built and the source training data are not available. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
FEATURE_ID	NUMBER/VARCHAR2

Table 5-41 Feature View for Explicit Semantic Analysis

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_ID	Unique identifier of a feature as it appears in the training data

5.4.15 Model Detail Views for Non-Negative Matrix Factorization

Model detail views for Non-Negative Matrix Factorization (NMF) describes encoding H matrix view and H inverse matrix view. Oracle recommends that users leverage the model details views instead of the `GET_MODEL_DETAILS_NMF` function.

The NMF algorithm has two matrix content views:

- Encoding (H) matrix `DM$VEmodel_name`
- H inverse matrix `DM$VImodel_name`

The view `DM$VEmodel_name` describes the encoding (H) matrix of an NMF model. The `FEATURE_NAME` column type may be either `NUMBER` or `VARCHAR2`. The view has the following schema definition.

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)

FEATURE_ID	NUMBER
FEATURE_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
COEFFICIENT	BINARY_DOUBLE

Table 5-42 Encoding H Matrix View for Non-Negative Matrix Factorization

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_ID	The ID of a feature in the model
FEATURE_NAME	The name of a feature in the model
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Specifies the value of attribute
COEFFICIENT	The attribute encoding that represents its contribution to the feature

The view `DM$VImodel_view` describes the inverse H matrix of an NMF model. The `FEATURE_NAME` column type may be either `NUMBER` or `VARCHAR2`. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
FEATURE_ID	NUMBER
FEATURE_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
COEFFICIENT	BINARY_DOUBLE

Table 5-43 Inverse H Matrix View for Non-Negative Matrix Factorization

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_ID	The ID of a feature in the model
FEATURE_NAME	The name of a feature in the model
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Specifies the value of attribute

Table 5-43 (Cont.) Inverse H Matrix View for Non-Negative Matrix Factorization

Column Name	Description
COEFFICIENT	The attribute encoding that represents its contribution to the feature

5.4.16 Model Detail Views for Singular Value Decomposition

Model detail views for Singular Value Decomposition (SVD) describes S Matrix view, right-singular vectors view, and left-singular vector view. Oracle recommends that users leverage the model details views instead of the `GET_MODEL_DETAILS_SVD` function.

The `DM$VEModel_name` view leverages the fact that each singular value in the SVD model has a corresponding principal component in the associated Principal Components Analysis (PCA) model to relate a common set of information for both classes of models. For a SVD model, it describes the content of the S matrix. When PCA scoring is selected as a build setting, the variance and percentage cumulative variance for the corresponding principal components are shown as well. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
FEATURE_ID	NUMBER
FEATURE_NAME	NUMBER/VARCHAR2
VALUE	BINARY_DOUBLE
VARIANCE	BINARY_DOUBLE
PCT_CUM_VARIANCE	BINARY_DOUBLE

Table 5-44 S Matrix View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_ID	The ID of a feature in the model
FEATURE_NAME	The name of a feature in the model
VALUE	The matrix entry value
VARIANCE	The variance explained by a component. This column is only present for SVD models with setting <code>dbms_data_mining.svds_scoring_mode</code> set to <code>dbms_data_mining.svds_scoring_pca</code> . This column is non-null only if the build data is centered, either manually or because of the following setting: <code>dbms_data_mining.prep_auto</code> is set to <code>dbms_data_mining.prep_auto_on</code> .

Table 5-44 (Cont.) S Matrix View

Column Name	Description
PCT_CUM_VARIANCE	<p>The percent cumulative variance explained by the components thus far. The components are ranked by the explained variance in descending order.</p> <p>This column is only present for SVD models with setting <code>dbms_data_mining.svds_scoring_mode</code> set to <code>dbms_data_mining.svds_scoring_pca</code></p> <p>This column is non-null only if the build data is centered, either manually or because of the following setting: <code>dbms_data_mining.prep_auto</code> is set to <code>dbms_data_mining.prep_auto_on</code>.</p>

The SVD `DM$VVmodel_view` describes the right-singular vectors of SVD model. For a PCA model it describes the principal components (eigenvectors). The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
FEATURE_ID	NUMBER
FEATURE_NAME	NUMBER/VARCHAR2
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_VALUE	VARCHAR2(4000)
VALUE	BINARY_DOUBLE

Table 5-45 Right-singular Vectors of Singular Value Decomposition

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
FEATURE_ID	The ID of a feature in the model
FEATURE_NAME	The name of a feature in the model
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_VALUE	Categorical attribute value. For numerical attributes, <code>ATTRIBUTE_VALUE</code> is null.
VALUE	The matrix entry value

The view `DM$VUmodel_name` describes the left-singular vectors of a SVD model. For a PCA model, it describes the projection of the data in the principal components. This view does not exist unless the settings `dbms_data_mining.svds_u_matrix_output` is set to `dbms_data_mining.svds_u_matrix_enable`. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
CASE_ID	NUMBER/VARHCAR2
FEATURE_ID	NUMBER
FEATURE_NAME	NUMBER/VARCHAR2
VALUE	BINARY_DOUBLE

Table 5-46 Left-singular Vectors of Singular Value Decomposition or Projection Data in Principal Components

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
CASE_ID	Unique identifier of the row in the build data described by the U matrix projection.
FEATURE_ID	The ID of a feature in the model
FEATURE_NAME	The name of a feature in the model
VALUE	The matrix entry value

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

5.4.17 Model Detail View for Minimum Description Length

Model detail view for Minimum Description Length (for calculating Attribute Importance) describes Attribute Importance view. Oracle recommends that users leverage the model details views instead of the GET_MODEL_DETAILS_AI function.

The Attribute Importance view `DM$VAmodel_name` describes the Attribute Importance as well as the Attribute Importance rank. The view has the following schema:

Name	Type
PARTITION_NAME	VARCHAR2(128)
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
ATTRIBUTE_IMPORTANCE_VALUE	BINARY_DOUBLE
ATTRIBUTE_RANK	NUMBER

Table 5-47 Attribute Importance View for Minimum Description Length

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
ATTRIBUTE_IMPORTANCE_VALUE	Importance value
ATTRIBUTE_RANK	Rank based on importance

5.4.18 Model Detail View for Binning

The binning view DM\$VB describes the bin boundaries used in the automatic data preparation.

The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
BIN_ID	NUMBER
LOWER_BIN_BOUNDARY	BINARY_DOUBLE
UPPER_BIN_BOUNDARY	BINARY_DOUBLE
ATTRIBUTE_VALUE	VARCHAR2(4000)

Table 5-48 Model Details View for Binning

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
ATTRIBUTE_NAME	Specifies the attribute name
ATTRIBUTE_SUBNAME	Specifies the attribute subname
BIN_ID	Bin ID (or bin identifier)
LOWER_BIN_BOUNDARY	Numeric lower bin boundary
UPPER_BIN_BOUNDARY	Numeric upper bin boundary
ATTRIBUTE_VALUE	Categorical value

5.4.19 Model Detail Views for Global Information

Model detail views for Global Information describes global statistics view, alert view, and computed settings view. Oracle recommends that users leverage the model details views instead of GET_MODEL_DETAILS_GLOBAL function.

The global statistics view DM\$VG $model_name$ describes global statistics related to the model build. Examples include the number of rows used in the build, the convergence status, and the model quality metrics. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
NAME	VARCHAR2(30)
NUMERIC_VALUE	NUMBER
STRING_VALUE	VARCHAR2(4000)

Table 5-49 Global Statistics View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model

Table 5-49 (Cont.) Global Statistics View

Column Name	Description
NAME	Name of the statistic
NUMERIC_VALUE	Numeric value of the statistic
STRING_VALUE	Categorical value of the statistic

The alert view `DM$VWmodel_name` lists alerts issued during the model build. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
ERROR_NUMBER	BINARY_DOUBLE
ERROR_TEXT	VARCHAR2(4000)

Table 5-50 Alert View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
ERROR_NUMBER	Error number (valid when event is Error)
ERROR_TEXT	Error message

The computed settings view `DM$VSmodel_name` lists the algorithm computed settings. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
SETTING_NAME	VARCHAR2(30)
SETTING_VALUE	VARCHAR2(4000)

Table 5-51 Computed Settings View

Column Name	Description
PARTITION_NAME	Partition name in a partitioned model
SETTING_NAME	Name of the setting
SETTING_VALUE	Value of the setting

5.4.20 Model Detail View for Normalization and Missing Value Handling

The Normalization and Missing Value Handling View `DM$VN` describes the normalization parameters used in Automatic Data Preparation (ADP) and the missing value replacement when a `NULL` value is encountered. Missing value replacement applies only to the twodimensional columns and does not apply to the nested columns.

The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
ATTRIBUTE_NAME	VARCHAR2(128)
ATTRIBUTE_SUBNAME	VARCHAR2(4000)
NUMERIC_MISSING_VALUE	BINARY_DOUBLE
CATEGORICAL_MISSING_VALUE	VARCHAR2(4000)
NORMALIZATION_SHIFT	BINARY_DOUBLE
NORMALIZATION_SCALE	BINARY_DOUBLE

Table 5-52 Normalization and Missing Value Handling View

Column Name	Description
PARTITION_NAME	A partition in a partitioned model
ATTRIBUTE_NAME	Column name
ATTRIBUTE_SUBNAME	Nested column subname. The value is null for non-nested columns.
NUMERIC_MISSING_VALUE	Numeric missing value replacement
CATEGORICAL_MISSING_VALUE	Categorical missing value replacement
NORMALIZATION_SHIFT	Normalization shift value
NORMALIZATION_SCALE	Normalization scale value

Scoring and Deployment

Explains the scoring and deployment features of Oracle Data Mining.

- [About Scoring and Deployment](#) (page 6-1)
- [Using the Data Mining SQL Functions](#) (page 6-2)
- [Prediction Details](#) (page 6-4)
- [Real-Time Scoring](#) (page 6-7)
- [Dynamic Scoring](#) (page 6-8)
- [Cost-Sensitive Decision Making](#) (page 6-9)
- [DBMS_DATA_MINING.Apply](#) (page 6-11)

6.1 About Scoring and Deployment

Scoring is the application of models to new data. In Oracle Data Mining, scoring is performed by SQL language functions.

Predictive functions perform Classification, Regression, or Anomaly detection. Clustering functions assign rows to clusters. Feature Extraction functions transform the input data to a set of higher order predictors. A scoring procedure is also available in the DBMS_DATA_MINING PL/SQL package.

Deployment refers to the use of models in a target environment. Once the models have been built, the challenges come in deploying them to obtain the best results, and in maintaining them within a production environment. Deployment can be any of the following:

- Scoring data either for batch or real-time results. Scores can include predictions, probabilities, rules, and other statistics.
- Extracting model details to produce reports. For example: clustering rules, decision tree rules, or attribute rankings from an Attribute Importance model.
- Extending the business intelligence infrastructure of a data warehouse by incorporating mining results in applications or operational systems.
- Moving a model from the database where it was built to the database where it used for scoring (export/import)

Oracle Data Mining supports all of these deployment scenarios.

Note:

Oracle Data Mining scoring operations support parallel execution. When parallel execution is enabled, multiple CPU and I/O resources are applied to the execution of a single database operation.

Parallel execution offers significant performance improvements, especially for operations that involve complex queries and large databases typically associated with decision support systems (DSS) and data warehouses.

Related Topics:

Oracle Database VLDB and Partitioning Guide

Oracle Data Mining Concepts

[Exporting and Importing Mining Models](#) (page 8-7)

You can export data mining models to flat files to back up work in progress or to move models to a different instance of Oracle Database Enterprise Edition (such as from a development database to a test database).

6.2 Using the Data Mining SQL Functions

Learn about the benefits of SQL functions in data mining.

The data mining SQL functions provide the following benefits:

- Models can be easily deployed within the context of existing SQL applications.
- Scoring operations take advantage of existing query execution functionality. This provides performance benefits.
- Scoring results are pipelined, enabling the rows to be processed without requiring materialization.

The data mining functions produce a score for each row in the selection. The functions can apply a mining model schema object to compute the score, or they can score dynamically without a pre-defined model, as described in "Dynamic Scoring".

Related Topics:

[Dynamic Scoring](#) (page 6-8)

[Scoring Requirements](#) (page 3-2)

[Table 2-4](#) (page 2-9)

Oracle Database SQL Language Reference

6.2.1 Choosing the Predictors

The data mining functions support a `USING` clause that specifies which attributes to use for scoring. You can specify some or all of the attributes in the selection and you can specify expressions. The following examples all use the `PREDICTION` function to find the customers who are likely to use an affinity card, but each example uses a different set of predictors.

The query in [Example 6-1](#) (page 6-3) uses all the predictors.

The query in [Example 6-2](#) (page 6-3) uses only gender, marital status, occupation, and income as predictors.

The query in [Example 6-3](#) (page 6-3) uses three attributes and an expression as predictors. The prediction is based on gender, marital status, occupation, and the assumption that all customers are in the highest income bracket.

Example 6-1 Using All Predictors

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

C	CNT	AVG_AGE
F	25	38
M	213	43

Example 6-2 Using Some Predictors

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

C	CNT	AVG_AGE
F	30	38
M	186	43

Example 6-3 Using Some Predictors and an Expression

```
SELECT cust_gender, COUNT(*) AS cnt, ROUND(AVG(age)) AS avg_age
   FROM mining_data_apply_v
   WHERE PREDICTION(dt_sh_clas_sample USING
                    cust_gender, cust_marital_status, occupation,
                    'I: 300,000 and above' AS cust_income_level) = 1
   GROUP BY cust_gender
   ORDER BY cust_gender;
```

C	CNT	AVG_AGE
F	30	38
M	186	43

6.2.2 Single-Record Scoring

The data mining functions can produce a score for a single record, as shown in [Example 6-4](#) (page 6-4) and [Example 6-5](#) (page 6-4).

[Example 6-4](#) (page 6-4) returns a prediction for customer 102001 by applying the classification model `NB_SH_Clas_sample`. The resulting score is 0, meaning that this customer is unlikely to use an affinity card.

[Example 6-5](#) (page 6-4) returns a prediction for 'Affinity card is great' as the comments attribute by applying the text mining model `T_SVM_Clas_sample`. The resulting score is 1, meaning that this customer is likely to use an affinity card.

Example 6-4 Scoring a Single Customer or a Single Text Expression

```
SELECT PREDICTION (NB_SH_Clas_Sample USING *)
      FROM sh.customers where cust_id = 102001;

PREDICTION(NB_SH_CLAS_SAMPLEUSING*)
-----
                                0
```

Example 6-5 Scoring a Single Text Expression

```
SELECT
      PREDICTION(T_SVM_Clas_sample USING 'Affinity card is great' AS comments)
      FROM DUAL;

PREDICTION(T_SVM_CLAS_SAMPLEUSING'AFFINITYCARDISGREAT'ASCOMMENTS)
-----
                                                                1
```

6.3 Prediction Details

Prediction details are XML strings that provide information about the score. Details are available for all types of scoring: clustering, feature extraction, classification, regression, and anomaly detection. Details are available whether scoring is dynamic or the result of model apply.

The details functions, `CLUSTER_DETAILS`, `FEATURE_DETAILS`, and `PREDICTION_DETAILS` return the actual value of attributes used for scoring and the relative importance of the attributes in determining the score. By default, the functions return the five most important attributes in descending order of importance.

6.3.1 Cluster Details

For the most likely cluster assignments of customer 100955 (probability of assignment > 20%), the query in the following example produces the five attributes that have the most impact for each of the likely clusters. The clustering functions apply an Expectation Maximization model named `em_sh_clus_sample` to the data selected from `mining_data_apply_v`. The "5" specified in `CLUSTER_DETAILS` is not required, because five attributes are returned by default.

Example 6-6 Cluster Details

```
SELECT S.cluster_id, probability prob,
      CLUSTER_DETAILS(em_sh_clus_sample, S.cluster_id, 5 USING T.*) det
      FROM
      (SELECT v.*, CLUSTER_SET(em_sh_clus_sample, NULL, 0.2 USING *) pset
      FROM mining_data_apply_v v
      WHERE cust_id = 100955) T,
      TABLE(T.pset) S
      ORDER BY 2 DESC;

CLUSTER_ID  PROB  DET
-----
14 .6761 <Details algorithm="Expectation Maximization" cluster="14">
      <Attribute name="AGE" actualValue="51" weight=".676" rank="1"/>
      <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".557" rank="2"/>
      <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".412" rank="3"/>
      <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".171" rank="4"/>
      <Attribute name="BOOKKEEPING_APPLICATION"actualValue="1" weight="-.003"
      rank="5"/>
      </Details>
```

```

3 .3227 <Details algorithm="Expectation Maximization" cluster="3">
  <Attribute name="YRS_RESIDENCE" actualValue="3" weight=".323" rank="1"/>
  <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".265" rank="2"/>
  <Attribute name="EDUCATION" actualValue="HS-grad" weight=".172" rank="3"/>
  <Attribute name="AFFINITY_CARD" actualValue="0" weight=".125" rank="4"/>
  <Attribute name="OCCUPATION" actualValue="Crafts" weight=".055" rank="5"/>
</Details>

```

6.3.2 Feature Details

The query in the following example returns the three attributes that have the greatest impact on the top Principal Components Analysis (PCA) projection for customer 101501. The `FEATURE_DETAILS` function applies a Singular Value Decomposition model named `svd_sh_sample` to the data selected from `svd_sh_sample_build_num`.

Example 6-7 Feature Details

```

SELECT FEATURE_DETAILS(svd_sh_sample, 1, 3 USING *) projldet
  FROM svd_sh_sample_build_num
  WHERE CUST_ID = 101501;

```

```

PROJLDET
-----

```

```

<Details algorithm="Singular Value Decomposition" feature="1">
<Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".352" rank="1"/>
<Attribute name="Y_BOX_GAMES" actualValue="0" weight=".249" rank="2"/>
<Attribute name="AGE" actualValue="41" weight=".063" rank="3"/>
</Details>

```

6.3.3 Prediction Details

The query in the following example returns the attributes that are most important in predicting the age of customer 100010. The prediction functions apply a Generalized Linear Model Regression model named `GLMR_SH_Regr_sample` to the data selected from `mining_data_apply_v`.

Example 6-8 Prediction Details for Regression

```

SELECT cust_id,
  PREDICTION(GLMR_SH_Regr_sample USING *) pr,
  PREDICTION_DETAILS(GLMR_SH_Regr_sample USING *) pd
  FROM mining_data_apply_v
  WHERE CUST_ID = 100010;

```

```

CUST_ID  PR  PD
-----  -  -

```

```

100010 25.45 <Details algorithm="Generalized Linear Model">
  <Attribute name="FLAT_PANEL_MONITOR" actualValue="1" weight=".025" rank="1"/>
  <Attribute name="OCCUPATION" actualValue="Crafts" weight=".019" rank="2"/>
  <Attribute name="AFFINITY_CARD" actualValue="0" weight=".01" rank="3"/>
  <Attribute name="OS_DOC_SET_KANJI" actualValue="0" weight="0" rank="4"/>
  <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight="-.004" rank="5"/>
</Details>

```

The query in the following example returns the customers who work in Tech Support and are likely to use an affinity card (with more than 85% probability). The prediction functions apply an Support Vector Machine (SVM) Classification model named `svmc_sh_clas_sample` to the data selected from `mining_data_apply_v`. The

query includes the prediction details, which show that education is the most important predictor.

Example 6-9 Prediction Details for Classification

```
SELECT cust_id, PREDICTION_DETAILS(svmc_sh_clas_sample, 1 USING *) PD
   FROM mining_data_apply_v
  WHERE PREDICTION_PROBABILITY(svmc_sh_clas_sample, 1 USING *) > 0.85
        AND occupation = 'TechSup'
 ORDER BY cust_id;
```

CUST_ID PD

```
-----
100029 <Details algorithm="Support Vector Machines" class="1">
  <Attribute name="EDUCATION" actualValue="Assoc-A" weight=".199" rank="1"/>
  <Attribute name="CUST_INCOME_LEVEL" actualValue="I: 170\,000 - 189\,999" weight=".044"
    rank="2"/>
  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".028" rank="3"/>
  <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".024" rank="4"/>
  <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight=".022" rank="5"/>
</Details>

100378 <Details algorithm="Support Vector Machines" class="1">
  <Attribute name="EDUCATION" actualValue="Assoc-A" weight=".21" rank="1"/>
  <Attribute name="CUST_INCOME_LEVEL" actualValue="B: 30\,000 - 49\,999" weight=".047"
    rank="2"/>
  <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".043" rank="3"/>
  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".03" rank="4"/>
  <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight=".023" rank="5"/>
</Details>

100508 <Details algorithm="Support Vector Machines" class="1">
  <Attribute name="EDUCATION" actualValue="Bach." weight=".19" rank="1"/>
  <Attribute name="CUST_INCOME_LEVEL" actualValue="L: 300\,000 and above" weight=".046"
    rank="2"/>
  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".031" rank="3"/>
  <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".026" rank="4"/>
  <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight=".024" rank="5"/>
</Details>

100980 <Details algorithm="Support Vector Machines" class="1">
  <Attribute name="EDUCATION" actualValue="Assoc-A" weight=".19" rank="1"/>
  <Attribute name="FLAT_PANEL_MONITOR" actualValue="0" weight=".038" rank="2"/>
  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1" weight=".026" rank="3"/>
  <Attribute name="BULK_PACK_DISKETTES" actualValue="1" weight=".022" rank="4"/>
  <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight=".02" rank="5"/>
</Details>
```

The query in the following example returns the two customers that differ the most from the rest of the customers. The prediction functions apply an anomaly detection model named `SVMO_SH_Clas_sample` to the data selected from `mining_data_apply_v`. Anomaly Detection uses a one-class SVM classifier.

Example 6-10 Prediction Details for Anomaly Detection

```
SELECT cust_id, pd FROM
  (SELECT cust_id,
         PREDICTION_DETAILS(SVMO_SH_Clas_sample, 0 USING *) pd,
         RANK() OVER (ORDER BY prediction_probability(
                     SVMO_SH_Clas_sample, 0 USING *) DESC, cust_id) rnk
   FROM mining_data_one_class_v)
 WHERE rnk <= 2
```

```
ORDER BY rnk;
```

```
CUST_ID PD
```

```
-----
102366 <Details algorithm="Support Vector Machines" class="0">
  <Attribute name="COUNTRY_NAME" actualValue="United Kingdom" weight=".078" rank="1"/>
  <Attribute name="CUST_MARITAL_STATUS" actualValue="Divorc." weight=".027" rank="2"/>
  <Attribute name="CUST_GENDER" actualValue="F" weight=".01" rank="3"/>
  <Attribute name="HOUSEHOLD_SIZE" actualValue="9+" weight=".009" rank="4"/>
  <Attribute name="AGE" actualValue="28" weight=".006" rank="5"/>
</Details>

101790 <Details algorithm="Support Vector Machines" class="0">
  <Attribute name="COUNTRY_NAME" actualValue="Canada" weight=".068" rank="1"/>
  <Attribute name="HOUSEHOLD_SIZE" actualValue="4-5" weight=".018" rank="2"/>
  <Attribute name="EDUCATION" actualValue="7th-8th" weight=".015" rank="3"/>
  <Attribute name="CUST_GENDER" actualValue="F" weight=".013" rank="4"/>
  <Attribute name="AGE" actualValue="38" weight=".001" rank="5"/>
</Details>
```

6.3.4 GROUPING Hint

Data mining functions consist of SQL functions such as PREDICTION*, CLUSTER*, FEATURE*, and ORA_DM_*. The GROUPING hint is an optional hint which applies to data mining scoring functions when scoring partitioned models.

This hint results in partitioning the input data set into distinct data slices so that each partition is scored in its entirety before advancing to the next partition. However, parallelism by partition is still available. Data slices are determined by the partitioning key columns used when the model was built. This method can be used with any data mining function against a partitioned model. The hint may yield a query performance gain when scoring large data that is associated with many partitions but may negatively impact performance when scoring large data with few partitions on large systems. Typically, there is no performance gain if you use the hint for single row queries.

Enhanced PREDICTION Function Command Format

```
<prediction function> ::=
  PREDICTION <left paren> /*+ GROUPING */ <prediction model>
    [ <comma> <class value> [ <comma> <top N> ] ]
  USING <mining attribute list> <right paren>
```

The syntax for only the PREDICTION function is given but it is applicable to any Data mining function where PREDICTION, CLUSTERING, and FEATURE_EXTRACTION scoring functions occur.

Example 6-11 Example

```
SELECT PREDICTION(/*+ GROUPING */my_model USING *) pred FROM <input table>;
```

Related Topics:

Oracle Database SQL Language Reference

6.4 Real-Time Scoring

Oracle Data Mining SQL functions enable prediction, clustering, and feature extraction analysis to be easily integrated into live production and operational systems. Because mining results are returned within SQL queries, mining can occur in real time.

With real-time scoring, point-of-sales database transactions can be mined. Predictions and rule sets can be generated to help front-line workers make better analytical decisions. Real-time scoring enables fraud detection, identification of potential liabilities, and recognition of better marketing and selling opportunities.

The query in the following example uses a Decision Tree model named `dt_sh_clas_sample` to predict the probability that customer 101488 uses an affinity card. A customer representative can retrieve this information in real time when talking to this customer on the phone. Based on the query result, the representative can offer an extra-value card, since there is a 73% chance that the customer uses a card.

Example 6-12 Real-Time Query with Prediction Probability

```
SELECT PREDICTION_PROBABILITY(dt_sh_clas_sample, 1 USING *) cust_card_prob
      FROM mining_data_apply_v
      WHERE cust_id = 101488;

CUST_CARD_PROB
-----
          .72764
```

6.5 Dynamic Scoring

The Data Mining SQL functions operate in two modes: by applying a pre-defined model, or by executing an analytic clause. If you supply an analytic clause instead of a model name, the function builds one or more transient models and uses them to score the data.

The ability to score data dynamically without a pre-defined model extends the application of basic embedded data mining techniques into environments where models are not available. Dynamic scoring, however, has limitations. The transient models created during dynamic scoring are not available for inspection or fine tuning. Applications that require model inspection, the correlation of scoring results with the model, special algorithm settings, or multiple scoring queries that use the same model, require a predefined model.

The following example shows a dynamic scoring query. The example identifies the rows in the input data that contain unusual customer age values.

Example 6-13 Dynamic Prediction

```
SELECT cust_id, age, pred_age, age-pred_age age_diff, pred_det FROM
  (SELECT cust_id, age, pred_age, pred_det,
    RANK() OVER (ORDER BY ABS(age-pred_age) DESC) rnk FROM
    (SELECT cust_id, age,
      PREDICTION(FOR age USING *) OVER () pred_age,
      PREDICTION_DETAILS(FOR age ABS USING *) OVER () pred_det
    FROM mining_data_apply_v))
  WHERE rnk <= 5;

CUST_ID  AGE  PRED_AGE  AGE_DIFF  PRED_DET
-----
100910   80  40.6686505  39.33  <Details algorithm="Support Vector Machines">
      <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
      weight=".059" rank="1"/>
      <Attribute name="Y_BOX_GAMES" actualValue="0"
      weight=".059" rank="2"/>
      <Attribute name="AFFINITY_CARD" actualValue="0"
      weight=".059" rank="3"/>
      <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"
      weight=".059" rank="4"/>
```

```

        <Attribute name="YRS_RESIDENCE" actualValue="4"
        weight=".059" rank="5"/>
    </Details>

101285  79 42.1753571  36.82 <Details algorithm="Support Vector Machines">
    <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
    weight=".059" rank="1"/>
    <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".059"
    rank="2"/>
    <Attribute name="CUST_MARITAL_STATUS" actualValue="Mabsent"
    weight=".059" rank="3"/>
    <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"
    rank="4"/>
    <Attribute name="OCCUPATION" actualValue="Prof." weight=".059"
    rank="5"/>
    </Details>

100694  77 41.0396722  35.96 <Details algorithm="Support Vector Machines">
    <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
    weight=".059" rank="1"/>
    <Attribute name="EDUCATION" actualValue="&lt; Bach."
    weight=".059" rank="2"/>
    <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"
    rank="3"/>
    <Attribute name="CUST_ID" actualValue="100694" weight=".059"
    rank="4"/>
    <Attribute name="COUNTRY_NAME" actualValue="United States of
    America" weight=".059" rank="5"/>
    </Details>

100308  81 45.3252491  35.67 <Details algorithm="Support Vector Machines">
    <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
    weight=".059" rank="1"/>
    <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"
    rank="2"/>
    <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".059"
    rank="3"/>
    <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"
    weight=".059" rank="4"/>
    <Attribute name="CUST_GENDER" actualValue="F" weight=".059"
    rank="5"/>
    </Details>

101256  90 54.3862214  35.61 <Details algorithm="Support Vector Machines">
    <Attribute name="YRS_RESIDENCE" actualValue="9" weight=".059"
    rank="1"/>
    <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"
    weight=".059" rank="2"/>
    <Attribute name="EDUCATION" actualValue="&lt; Bach."
    weight=".059" rank="3"/>
    <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"
    rank="4"/>
    <Attribute name="COUNTRY_NAME" actualValue="United States of
    America" weight=".059" rank="5"/>
    </Details>

```

6.6 Cost-Sensitive Decision Making

Costs are user-specified numbers that bias Classification. The algorithm uses positive numbers to penalize more expensive outcomes over less expensive outcomes. Higher numbers indicate higher costs.

The algorithm uses negative numbers to favor more beneficial outcomes over less beneficial outcomes. Lower negative numbers indicate higher benefits.

All classification algorithms can use costs for scoring. You can specify the costs in a cost matrix table, or you can specify the costs inline when scoring. If you specify costs inline and the model also has an associated cost matrix, only the inline costs are used. The `PREDICTION`, `PREDICTION_SET`, and `PREDICTION_COST` functions support costs.

Only the Decision Tree algorithm can use costs to bias the model build. If you want to create a Decision Tree model with costs, create a cost matrix table and provide its name in the `CLAS_COST_TABLE_NAME` setting for the model. If you specify costs when building the model, the cost matrix used to create the model is used when scoring. If you want to use a different cost matrix table for scoring, first remove the existing cost matrix table then add the new one.

A sample cost matrix table is shown in the following table. The cost matrix specifies costs for a binary target. The matrix indicates that the algorithm must treat a misclassified 0 as twice as costly as a misclassified 1.

Table 6-1 Sample Cost Matrix

ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	COST
0	0	0
0	1	2
1	0	1
1	1	0

Example 6-14 Sample Queries With Costs

The table `nbmodel_costs` contains the cost matrix described in [Table 6-1](#) (page 6-10).

```
SELECT * from nbmodel_costs;
```

ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	COST
0	0	0
0	1	2
1	0	1
1	1	0

The following statement associates the cost matrix with a Naive Bayes model called `nbmodel`.

```
BEGIN
  dbms_data_mining.add_cost_matrix('nbmodel', 'nbmodel_costs');
END;
/
```

The following query takes the cost matrix into account when scoring `mining_data_apply_v`. The output is restricted to those rows where a prediction of 1 is less costly than a prediction of 0.

```
SELECT cust_gender, COUNT(*) AS cnt, ROUND(AVG(age)) AS avg_age
FROM mining_data_apply_v
WHERE PREDICTION (nbmodel COST MODEL
  USING cust_marital_status, education, household_size) = 1
```



```
GROUP BY cust_gender
ORDER BY cust_gender;
```

C	CNT	AVG_AGE
F	25	38
M	208	43

You can specify costs inline when you invoke the scoring function. If you specify costs inline and the model also has an associated cost matrix, only the inline costs are used. The same query is shown below with different costs specified inline. Instead of the "2" shown in the cost matrix table (Table 6-1 (page 6-10)), "10" is specified in the inline costs.

```
SELECT cust_gender, COUNT(*) AS cnt, ROUND(AVG(age)) AS avg_age
FROM mining_data_apply_v
WHERE PREDICTION (nbmodel
                  COST (0,1) values ((0, 10),
                                     (1, 0))
                  USING cust_marital_status, education, household_size) = 1
GROUP BY cust_gender
ORDER BY cust_gender;
```

C	CNT	AVG_AGE
F	74	39
M	581	43

The same query based on probability instead of costs is shown below.

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

C	CNT	AVG_AGE
F	73	39
M	577	44

Related Topics:

[Example 1-1 \(page 1-2\)](#)

6.7 DBMS_DATA_MINING.Apply

The APPLY procedure in DBMS_DATA_MINING is a batch apply operation that writes the results of scoring directly to a table.

The columns in the table are mining function-dependent.

Scoring with APPLY generates the same results as scoring with the SQL scoring functions. Classification produces a prediction and a probability for each case; clustering produces a cluster ID and a probability for each case, and so on. The difference lies in the way that scoring results are captured and the mechanisms that can be used for retrieving them.

APPLY creates an output table with the columns shown in the following table:

Table 6-2 APPLY Output Table

Mining Function	Output Columns
classification	CASE_ID PREDICTION PROBABILITY
regression	CASE_ID PREDICTION
anomaly detection	CASE_ID PREDICTION PROBABILITY
clustering	CASE_ID CLUSTER_ID PROBABILITY
feature extraction	CASE_ID FEATURE_ID MATCH_QUALITY

Since APPLY output is stored separately from the scoring data, it must be joined to the scoring data to support queries that include the scored rows. Thus any model that is used with APPLY must have a case ID.

A case ID is not required for models that is applied with SQL scoring functions. Likewise, storage and joins are not required, since scoring results are generated and consumed in real time within a SQL query.

The following example illustrates Anomaly Detection with APPLY. The query of the APPLY output table returns the ten first customers in the table. Each has a probability for being typical (1) and a probability for being anomalous (0).

Example 6-15 Anomaly Detection with DBMS_DATA_MINING.APPLY

```
EXEC dbms_data_mining.apply
    ('SVMO_SH_Clas_sample', 'svmo_sh_sample_prepared',
    'cust_id', 'one_class_output');
```

```
SELECT * from one_class_output where rownum < 11;
```

```

CUST_ID PREDICTION PROBABILITY
-----
101798      1 .567389309
101798      0 .432610691
102276      1 .564922469
102276      0 .435077531
102404      1 .51213544
102404      0 .48786456
101891      1 .563474346
101891      0 .436525654
102815      0 .500663683
102815      1 .499336317
```

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

Mining Unstructured Text

Explains how to use Oracle Data Mining to mine unstructured text.

- [About Unstructured Text](#) (page 7-1)
- [About Text Mining and Oracle Text](#) (page 7-1)
- [Data Preparation for Text Features](#) (page 7-2)
- [Creating a Model that Includes Text Mining](#) (page 7-2)
- [Creating a Text Policy](#) (page 7-4)
- [Configuring a Text Attribute](#) (page 7-5)

7.1 About Unstructured Text

Data mining algorithms act on data that is numerical or categorical. Numerical data is ordered. It is stored in columns that have a numeric data type, such as NUMBER or FLOAT. Categorical data is identified by category or classification. It is stored in columns that have a character data type, such as VARCHAR2 or CHAR.

Unstructured text data is neither numerical nor categorical. Unstructured text includes items such as web pages, document libraries, Power Point presentations, product specifications, emails, comment fields in reports, and call center notes. It has been said that unstructured text accounts for more than three quarters of all enterprise data. Extracting meaningful information from unstructured text can be critical to the success of a business.

7.2 About Text Mining and Oracle Text

Understand what is text mining and oracle text.

Text mining is the process of applying data mining techniques to text terms, also called text features or tokens. Text terms are words or groups of words that have been extracted from text documents and assigned numeric weights. Text terms are the fundamental unit of text that can be manipulated and analyzed.

Oracle Text is a Database technology that provides term extraction, word and theme searching, and other utilities for querying text. When columns of text are present in the training data, Oracle Data Mining uses Oracle Text utilities and term weighting strategies to transform the text for mining. Oracle Data Mining passes configuration information supplied by you to Oracle Text and uses the results in the model creation process.

Related Topics:

Oracle Text Application Developer's Guide

7.3 Data Preparation for Text Features

The model details view for text features is `DM$VXmodel_name`.

The text feature view `DM$VXmodel_name` describes the extracted text features if there are text attributes present. The view has the following schema:

Name	Type
-----	-----
PARTITION_NAME	VARCHAR2(128)
COLUMN_NAME	VARCHAR2(128)
TOKEN	VARCHAR2(4000)
DOCUMENT_FREQUENCY	NUMBER

Table 7-1 Text Feature View for Extracted Text Features

Column Name	Description
PARTITION_NAME	A partition in a partitioned model to retrieve details
COLUMN_NAME	Name of the identifier column
TOKEN	Text token which is usually a word or stemmed word
DOCUMENT_FREQUENCY	A measure of token frequency in the entire training set

7.4 Creating a Model that Includes Text Mining

Learn how to create a model that includes text mining.

Oracle Data Mining supports unstructured text within columns of `VARCHAR2`, `CHAR`, `CLOB`, `BLOB`, and `BFILE`, as described in the following table:

Table 7-2 Column Data Types That May Contain Unstructured Text

Data Type	Description
BFILE and BLOB	Oracle Data Mining interprets <code>BLOB</code> and <code>BFILE</code> as text <i>only if</i> you identify the columns as text when you create the model. If you do not identify the columns as text, then <code>CREATE_MODEL</code> returns an error.
CLOB	Oracle Data Mining interprets <code>CLOB</code> as text.
CHAR	Oracle Data Mining interprets <code>CHAR</code> as categorical by default. You can identify columns of <code>CHAR</code> as text when you create the model.
VARCHAR 2	Oracle Data Mining interprets <code>VARCHAR2</code> with data length > 4000 as text. Oracle Data Mining interprets <code>VARCHAR2</code> with data length <= 4000 as categorical by default. You can identify these columns as text when you create the model.

Note:

Text is not supported in nested columns or as a target in supervised data mining.

The settings described in the following table control the term extraction process for text attributes in a model. Instructions for specifying model settings are in "Specifying Model Settings".

Table 7-3 Model Settings for Text

Setting Name	Data Type	Setting Value	Description
ODMS_TEXT_POLICY_NAME	VARCHAR2 (4000)	Name of an Oracle Text policy object created with CTX_DDL.CREATE_POLICY	Affects how individual tokens are extracted from unstructured text. See " Creating a Text Policy (page 7-4)".
ODMS_TEXT_MAX_FEATURES	INTEGER	1 <= value <= 100000	Maximum number of features to use from the document set (across all documents of each text column) passed to CREATE_MODEL. Default is 3000.

A model can include one or more text attributes. A model with text attributes can also include categorical and numerical attributes.

To create a model that includes text attributes:

1. Create an Oracle Text policy object..
2. Specify the model configuration settings that are described in "[Table 7-3](#) (page 7-3)".
3. Specify which columns must be treated as text and, optionally, provide text transformation instructions for individual attributes.
4. Pass the model settings and text transformation instructions to DBMS_DATA_MINING.CREATE_MODEL.

Note:

All algorithms except O-Cluster can support columns of unstructured text.

The use of unstructured text is not recommended for association rules (Apriori).

Related Topics:

[Specifying Model Settings](#) (page 5-7)

Understand how to configure data mining models at build time.

[Creating a Text Policy](#) (page 7-4)

An Oracle Text policy specifies how text content must be interpreted. You can provide a text policy to govern a model, an attribute, or both the model and individual attributes.

[Configuring a Text Attribute](#) (page 7-5)

Learn how to identify a column as a text attribute and provide transformation instructions for any text attribute.

[Embedding Transformations in a Model](#) (page 4-5)

7.5 Creating a Text Policy

An Oracle Text policy specifies how text content must be interpreted. You can provide a text policy to govern a model, an attribute, or both the model and individual attributes.

If a model-specific policy is present and one or more attributes have their own policies, Oracle Data Mining uses the attribute policies for the specified attributes and the model-specific policy for the other attributes.

The `CTX_DDL.CREATE_POLICY` procedure creates a text policy.

```
CTX_DDL.CREATE_POLICY(
    policy_name    IN VARCHAR2,
    filter         IN VARCHAR2 DEFAULT NULL,
    section_group IN VARCHAR2 DEFAULT NULL,
    lexer         IN VARCHAR2 DEFAULT NULL,
    stoplist      IN VARCHAR2 DEFAULT NULL,
    wordlist      IN VARCHAR2 DEFAULT NULL);
```

The parameters of `CTX_DDL.CREATE_POLICY` are described in the following table.

Table 7-4 CTX_DDL.CREATE_POLICY Procedure Parameters

Parameter Name	Description
<code>policy_name</code>	Name of the new policy object. Oracle Text policies and text indexes share the same namespace.
<code>filter</code>	Specifies how the documents must be converted to plain text for indexing. Examples are: <code>CHARSET_FILTER</code> for character sets and <code>NULL_FILTER</code> for plain text, HTML and XML. For <code>filter</code> values, see "Filter Types" in <i>Oracle Text Reference</i> .
<code>section_group</code>	Identifies sections within the documents. For example, <code>HTML_SECTION_GROUP</code> defines sections in HTML documents. For <code>section_group</code> values, see "Section Group Types" in <i>Oracle Text Reference</i> . Note: You can specify any section group that is supported by <code>CONTEXT</code> indexes.
<code>lexer</code>	Identifies the language that is being indexed. For example, <code>BASIC_LEXER</code> is the lexer for extracting terms from text in languages that use white space delimited words (such as English and most western European languages). For <code>lexer</code> values, see "Lexer Types" in <i>Oracle Text Reference</i> .

Table 7-4 (Cont.) CTX_DDL.CREATE_POLICY Procedure Parameters

Parameter Name	Description
stoplist	Specifies words and themes to exclude from term extraction. For example, the word "the" is typically in the stoplist for English language documents. The system-supplied stoplist is used by default. See "Stoplists" in <i>Oracle Text Reference</i> .
wordlist	Specifies how stems and fuzzy queries must be expanded. A stem defines a root form of a word so that different grammatical forms have a single representation. A fuzzy query includes common misspellings in the representation of a word. See "BASIC_WORDLIST" in <i>Oracle Text Reference</i> .

Related Topics:

Oracle Text Reference

7.6 Configuring a Text Attribute

Learn how to identify a column as a text attribute and provide transformation instructions for any text attribute.

As shown in [Table 7-2](#) (page 7-2), you can identify columns of CHAR, shorter VARCHAR2 (<=4000), BFILE, and BLOB as text attributes. If CHAR and shorter VARCHAR2 columns are not explicitly identified as unstructured text, then CREATE_MODEL processes them as categorical attributes. If BFILE and BLOB columns are not explicitly identified as unstructured text, then CREATE_MODEL returns an error.

To identify a column as a text attribute, supply the keyword TEXT in an **Attribute specification**. The attribute specification is a field (attribute_spec) in a transformation record (transform_rec). Transformation records are components of transformation lists (xform_list) that can be passed to CREATE_MODEL.

Note:

An attribute specification can also include information that is not related to text. Instructions for constructing an attribute specification are in "Embedding Transformations in a Model".

You can provide transformation instructions for any text attribute by qualifying the TEXT keyword in the attribute specification with the subsettings described in the following table.

Table 7-5 Attribute-Specific Text Transformation Instructions

Subsetting Name	Description	Example
POLICY_NAME	Name of an Oracle Text policy object created with CTX_DDL.CREATE_POLICY	(POLICY_NAME:my_policy)

Table 7-5 (Cont.) Attribute-Specific Text Transformation Instructions

Subsetting Name	Description	Example
TOKEN_TYPE	The following values are supported: NORMAL (the default) STEM THEME See " Token Types in an Attribute Specification (page 7-6)"	(TOKEN_TYPE:THEME)
MAX_FEATURES	Maximum number of features to use from the attribute.	(MAX_FEATURES:3000)

Note:

The TEXT keyword is only required for CLOB and longer VARCHAR2 (>4000) when you specify transformation instructions. The TEXT keyword is *always* required for CHAR, shorter VARCHAR2, BFILE, and BLOB — whether or not you specify transformation instructions.

Tip:

You can view attribute specifications in the data dictionary view ALL_MINING_MODEL_ATTRIBUTES, as shown in *Oracle Database Reference*.

Token Types in an Attribute Specification

When stems or themes are specified as the token type, the lexer preference for the text policy must support these types of tokens.

The following example adds themes and English stems to BASIC_LEXER.

```
BEGIN
  CTX_DDL.CREATE_PREFERENCE('my_lexer', 'BASIC_LEXER');
  CTX_DDL.SET_ATTRIBUTE('my_lexer', 'index_stems', 'ENGLISH');
  CTX_DDL.SET_ATTRIBUTE('my_lexer', 'index_themes', 'YES');
END;
```

Example 7-1 A Sample Attribute Specification for Text

This expression specifies that text transformation for the attribute must use the text policy named my_policy. The token type is THEME, and the maximum number of features is 3000.

```
"TEXT(POLICY_NAME:my_policy)(TOKEN_TYPE:THEME)(MAX_FEATURES:3000)"
```

Related Topics:

[Embedding Transformations in a Model](#) (page 4-5)

[Specifying Transformation Instructions for an Attribute](#) (page 4-5)

Learn what is a transformation instruction for an attribute and learn about the fields in a transformation record.

Oracle Database PL/SQL Packages and Types Reference

ALL_MINING_MODEL_ATTRIBUTES

Administrative Tasks for Oracle Data Mining

Explains how to perform administrative tasks related to Oracle Data Mining.

- [Installing and Configuring a Database for Data Mining](#) (page 8-1)
- [Upgrading or Downgrading Oracle Data Mining](#) (page 8-3)
- [Exporting and Importing Mining Models](#) (page 8-7)
- [Controlling Access to Mining Models and Data](#) (page 8-11)
- [Auditing and Adding Comments to Mining Models](#) (page 8-15)

8.1 Installing and Configuring a Database for Data Mining

Learn how to install and configure a database for Data Mining.

- [About Installation](#) (page 8-1)
- [Enabling or Disabling a Database Option](#) (page 8-2)
- [Database Tuning Considerations for Data Mining](#) (page 8-2)

8.1.1 About Installation

Oracle Data Mining is a component of the Oracle Advanced Analytics option to Oracle Database Enterprise Edition.

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

Oracle Data Miner, the graphical user interface to Oracle 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.

To perform data mining activities, you must be able to log on to the Oracle database, and your user ID must have the database privileges described in [Example 8-7](#) (page 8-13).

Related Topics:

<http://www.oracle.com/pls/topic/lookup?ctx=db122&id=datminGUI>

See Also:

Install and Upgrade page of the Oracle Database online documentation library for your platform-specific installation instructions: <http://docs.oracle.com/en/database/database.html>

8.1.2 Enabling or Disabling a Database Option

Learn how you can enable or disable Oracle Advanced Analytics option after the installation.

The Oracle Advanced Analytics option is enabled by default during installation of Oracle Database Enterprise Edition. After installation, you can use the command-line utility `chopt` to enable or disable a database option. For instructions, see "Enabling and Disabling Database Options After Installation" in the installation guide for your platform.

Related Topics:

Oracle Database Installation Guide for Linux

Oracle Database Installation Guide for Microsoft Windows

8.1.3 Database Tuning Considerations for Data Mining

Understand the Database tuning considerations for Data Mining.

DBAs managing production databases that support Oracle Data Mining must follow standard administrative practices as described in *Oracle Database Administrator's Guide*.

Building data mining models and batch scoring of mining models tend to put a DSS-like workload on the system. Single-row scoring tends to put an OLTP-like workload on the system.

Database memory management can have a major impact on data mining. The correct sizing of Program Global Area (PGA) memory is very important for model building, complex queries, and batch scoring. From a data mining perspective, the System Global Area (SGA) is generally less of a concern. However, the SGA must be sized to accommodate real-time scoring, which loads models into the shared cursor in the SGA. In most cases, you can configure the database to manage memory automatically. To do so, specify the total maximum memory size in the tuning parameter `MEMORY_TARGET`. With automatic memory management, Oracle Database dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands.

Most data mining algorithms can take advantage of parallel execution when it is enabled in the database. Parameters in `INIT.ORA` control the behavior of parallel execution.

Related Topics:

Oracle Database Administrator's Guide

[Scoring and Deployment](#) (page 6-1)

Explains the scoring and deployment features of Oracle Data Mining.

Oracle Database Administrator's Guide

Part I Database Performance Fundamentals

Tuning Database Memory

Oracle Database VLDB and Partitioning Guide

8.2 Upgrading or Downgrading Oracle Data Mining

Understand how to upgrade and downgrade Oracle Data Mining.

- [Pre-Upgrade Steps](#) (page 8-3)
- [Upgrading Oracle Data Mining](#) (page 8-3)
- [Post Upgrade Steps](#) (page 8-6)
- [Downgrading Oracle Data Mining](#) (page 8-7)

8.2.1 Pre-Upgrade Steps

Before upgrading, you must drop any data mining models that were created in Java and any mining activities that were created in Oracle Data Miner Classic (the earlier version of Oracle Data Miner).

Caution:

In Oracle Database 12c, Oracle Data Mining does not support a Java API, and Oracle Data Miner Classic cannot run against Oracle Database 12c .

8.2.1.1 Dropping Models Created in Java

If your 10g or 11g database contains models created in Java, use the `DBMS_DATA_MINING.DROP_MODEL` routine to drop the models before upgrading the database.

8.2.1.2 Dropping Mining Activities Created in Oracle Data Miner Classic

If your database contains mining activities from Oracle Data Miner Classic, delete the mining activities and drop the repository before upgrading the database. Follow these steps:

1. Use the Data Miner Classic user interface to delete the mining activities.
2. In SQL*Plus or SQL Developer, drop these tables:

```
DM4J$ACTIVITIES
DM4J$RESULTS
DM4J$TRANSFORMS
```

and these views:

```
DM4J$MODEL_RESULTS_V
DM4J$RESULTS_STATE_V
```

There must be no tables or views with the prefix `DM4J$` in any schema in the database after you complete these steps.

8.2.2 Upgrading Oracle Data Mining

Learn how to upgrade Oracle Data Mining.

After you complete the "Pre-Upgrade Steps", all models and mining metadata are fully integrated with the Oracle Database upgrade process whether you are upgrading from 11g or from 10g releases.

Upgraded models continue to work as they did in prior releases. Both upgraded models and new models that you create in the upgraded environment can make use of the new mining functionality introduced in the new release.

To upgrade a database, you can use Database Upgrade Assistant (DBUA) or you can perform a manual upgrade using export/import utilities.

Related Topics:

[Pre-Upgrade Steps](#) (page 8-3)

Oracle Database Upgrade Guide

8.2.2.1 Using Database Upgrade Assistant to Upgrade Oracle Data Mining

Oracle Database Upgrade Assistant provides a graphical user interface that guides you interactively through the upgrade process.

On Windows platforms, follow these steps to start the Upgrade Assistant:

1. Go to the Windows **Start** menu and choose the Oracle home directory.
2. Choose the **Configuration and Migration Tools** menu.
3. Launch the **Upgrade Assistant**.

On Linux platforms, run the DBUA utility to upgrade Oracle Database.

8.2.2.1.1 Upgrading from Release 10g

In Oracle Data Mining 10g, data mining metadata and PL/SQL packages are stored in the DMSYS schema. In Oracle Data Mining 11g and 12c, DMSYS no longer exists; data mining metadata objects are stored in SYS.

When Oracle Database 10g is upgraded to 12c, all data mining metadata objects and PL/SQL packages are migrated from DMSYS to SYS. The DMSYS schema and its associated objects are removed after a successful migration. When DMSYS is removed, the SYS.DBA_REGISTRY view no longer lists Oracle Data Mining as a component.

After upgrading to Oracle Database 12c, you can no longer switch to the Data Mining Scoring Engine (DMSE). The Scoring Engine does not exist in Oracle Database 11g or 12c.

8.2.2.1.2 Upgrading from Release 11g

If you upgrade Oracle Database 11g to Oracle Database 12c, and the database was previously upgraded from Oracle Database 10g, then the DMSYS schema may still be present. If the upgrade process detects DMSYS, it displays a warning message and drops DMSYS during the upgrade.

8.2.2.2 Using Export/Import to Upgrade Data Mining Models

If required, you can use a less automated approach to upgrading data mining models. You can export the models created in a previous version of Oracle Database and import them into an instance of Oracle Database 12c.

Caution:

Do not import data mining models that were created in Java. They are not supported in Oracle Database 12c.

8.2.2.2.1 Export/Import Release 10g Data Mining Models

Follow the instructions for exporting and importing Data Mining models.

To export models from an instance of Oracle Database 10g to a dump file, follow the instructions in "Exporting and Importing Mining Models". Before importing the models from the dump file, run the DMEIDMSYS script to create the DMSYS schema in Oracle Database 12c.

```
SQL>CONNECT / as sysdba;
SQL>@ORACLE_HOME\RDBMS\admin\dmeidmsys.sql
SQL>EXIT;
```

Note:

The TEMP tablespace must already exist in the Oracle Database 12g database. The DMEIDMSYS script uses the TEMP and SYSAUX tablespaces to create the DMSYS schema.

To import the dump file into the Oracle Database 12c database:

```
%ORACLE_HOME\bin\impdp system<password>
  dumpfile=<dumpfile_name>
  directory=<directory_name>
  logfile=<logfile_name> .....
SQL>CONNECT / as sysdba;
SQL>EXECUTE dmp_sys.upgrade_models();
SQL>ALTER SYSTEM FLUSH SHARED_POOL;
SQL>ALTER SYSTEM FLUSH BUFFER_CACHE;
SQL>EXIT;
```

The `upgrade_models` script migrates all data mining metadata objects and PL/SQL packages from DMSYS to SYS and then drops DMSYS before upgrading the models.

ALTER SYSTEM Statement

You can flush the Database Smart Flash Cache by issuing an `ALTER SYSTEM FLUSH FLASH_CACHE` statement. Flushing the Database Smart Flash Cache can be useful if you need to measure the performance of rewritten queries or a suite of queries from identical starting points.

Related Topics:

[Exporting and Importing Mining Models](#) (page 8-7)

You can export data mining models to flat files to back up work in progress or to move models to a different instance of Oracle Database Enterprise Edition (such as from a development database to a test database).

8.2.2.2 Export/Import Release 11g Data Mining Models

To export models from an instance of Oracle Database 11g to a dump file, follow the instructions in [Exporting and Importing Mining Models](#) (page 8-7).

Caution:

Do not import data mining models that were created in Java. They are not supported in Oracle Database 12c.

To import the dump file into the Oracle Database 12c database:

```
%ORACLE_HOME\bin\impdp system<password>
  dumpfile=<dumpfile_name>
  directory=<directory_name>
  logfile=<logfile_name> .....
SQL>CONNECT / as sysdba;
SQL>EXECUTE dmp_sys.upgrade_models();
SQL>ALTER SYSTEM flush shared_pool;
SQL>ALTER SYSTEM flush buffer_cache;
SQL>EXIT;
```

ALTER SYSTEM Statement

You can flush the Database Smart Flash Cache by issuing an `ALTER SYSTEM FLUSH FLASH_CACHE` statement. Flushing the Database Smart Flash Cache can be useful if you need to measure the performance of rewritten queries or a suite of queries from identical starting points.

8.2.3 Post Upgrade Steps

Perform steps to view the upgraded database.

After upgrading the database, check the `DBA_MINING_MODELS` view in the upgraded database. The newly upgraded mining models must be listed in this view.

After you have verified the upgrade and confirmed that there is no need to downgrade, you must set the initialization parameter `COMPATIBLE` to 12.1.

Note:

The `CREATE MINING MODEL` privilege must be granted to Data Mining user accounts that are used to create mining models.

Related Topics:

[Creating a Data Mining User](#) (page 8-11)

Explains how to create a Data Mining user.

[Controlling Access to Mining Models and Data](#) (page 8-11)

Understand how to create a Data Mining user and grant necessary privileges.

8.2.4 Downgrading Oracle Data Mining

Before downgrading the Oracle Database 12c database back to the previous version, ensure that no Singular Value Decomposition models or Expectation Maximization models are present. These algorithms are only available in Oracle Database 12c. Use the `DBMS_DATA_MINING.DROP_MODEL` routine to drop these models before downgrading. If you do not do this, the database downgrade process terminates.

Issue the following SQL statement in `SYS` to verify the downgrade:

```
SQL>SELECT o.name FROM sys.model$ m, sys.obj$ o
        WHERE m.obj#=o.obj# AND m.version=2;
```

8.3 Exporting and Importing Mining Models

You can export data mining models to flat files to back up work in progress or to move models to a different instance of Oracle Database Enterprise Edition (such as from a development database to a test database).

All methods for exporting and importing models are based on Oracle Data Pump technology.

The `DBMS_DATA_MINING` package includes the `EXPORT_MODEL` and `IMPORT_MODEL` procedures for exporting and importing individual mining models. `EXPORT_MODEL` and `IMPORT_MODEL` use the export and import facilities of Oracle Data Pump.

- [About Oracle Data Pump](#) (page 8-7)
- [Options for Exporting and Importing Mining Models](#) (page 8-8)
- [Directory Objects for EXPORT_MODEL and IMPORT_MODEL](#) (page 8-9)
- [Using EXPORT_MODEL and IMPORT_MODEL](#) (page 8-9)
- [Importing From PMML](#) (page 8-11)

Related Topics:

`EXPORT_MODEL`

`IMPORT_MODEL`

8.3.1 About Oracle Data Pump

Oracle Data Pump consists of two command-line clients and two PL/SQL packages. The command-line clients, `expdp` and `impdp`, provide an easy-to-use interface to the Data Pump export and import utilities. You can use `expdp` and `impdp` to export and import entire schemas or databases.

The Data Pump export utility writes the schema objects, including the tables and metadata that constitute mining models, to a dump file set. The Data Pump import utility retrieves the schema objects, including the model tables and metadata, from the dump file set and restores them in the target database.

`expdp` and `impdp` cannot be used to export/import individual mining models.

See Also:

Oracle Database Utilities for information about Oracle Data Pump and the `expdp` and `impdp` utilities

8.3.2 Options for Exporting and Importing Mining Models

Lists options for exporting and importing mining models.

Options for exporting and importing mining models are described in the following table.

Table 8-1 *Export and Import Options for Oracle Data Mining*

Task	Description
Export or import a full database	(DBA only) Use <code>expdp</code> to export a full database and <code>impdp</code> to import a full database. All mining models in the database are included.
Export or import a schema	Use <code>expdp</code> to export a schema and <code>impdp</code> to import a schema. All mining models in the schema are included.
Export or import individual models within a database	<p>Use <code>DBMS_DATA_MINING.EXPORT_MODEL</code> to export individual models and <code>DBMS_DATA_MINING.IMPORT_MODEL</code> to import individual models. These procedures can export and import a single mining model, all mining models, or mining models that match specific criteria.</p> <p>By default, <code>IMPORT_MODEL</code> imports models back into the schema from which they were exported. You can specify the <code>schema_remap</code> parameter to import models into a different schema. You can specify <code>tablespace_remap</code> with <code>schema_remap</code> to import models into a schema that uses a different tablespace.</p> <p>You may need special privileges in the database to import models into a different schema. These privileges are granted by the <code>EXP_FULL_DATABASE</code> and <code>IMP_FULL_DATABASE</code> roles, which are only available to privileged users (such as <code>SYS</code> or a user with the <code>DBA</code> role). You do not need these roles to export or import models within your own schema.</p> <p>To import models, you must have the same database privileges as the user who created the dump file set. Otherwise, a <code>DBA</code> with full system privileges must import the models.</p>
Export or import individual models to or from a remote database	<p>Use a database link to export individual models to a remote database or import individual models from a remote database. A database link is a schema object in one database that enables access to objects in a different database. The link must be created before you execute <code>EXPORT_MODEL</code> or <code>IMPORT_MODEL</code>.</p> <p>To create a private database link, you must have the <code>CREATE DATABASE LINK</code> system privilege. To create a public database link, you must have the <code>CREATE PUBLIC DATABASE LINK</code> system privilege. Also, you must have the <code>CREATE SESSION</code> system privilege on the remote Oracle Database. Oracle Net must be installed on both the local and remote Oracle Databases.</p>

Related Topics:

[IMPORT_MODEL Procedure](#)

[EXPORT_MODEL Procedure](#)

[Oracle Database SQL Language Reference](#)

8.3.3 Directory Objects for EXPORT_MODEL and IMPORT_MODEL

Learn how to use directory objects to identify the location of the dump file set.

EXPORT_MODEL and IMPORT_MODEL use a directory object to identify the location of the dump file set. A directory object is a logical name in the database for a physical directory on the host computer.

To export data mining models, you must have write access to the directory object and to the file system directory that it represents. To import data mining models, you must have read access to the directory object and to the file system directory. Also, the database itself must have access to file system directory. You must have the CREATE ANY DIRECTORY privilege to create directory objects.

The following SQL command creates a directory object named dmuser_dir. The file system directory that it represents must already exist and have shared read/write access rights granted by the operating system.

```
CREATE OR REPLACE DIRECTORY dmuser_dir AS '/dm_path/dm_mining';
```

The following SQL command gives user dmuser both read and write access to dmuser_dir.

```
GRANT READ,WRITE ON DIRECTORY dmuser_dir TO dmuser;
```

Related Topics:

Oracle Database SQL Language Reference

8.3.4 Using EXPORT_MODEL and IMPORT_MODEL

The examples illustrate various export and import scenarios with EXPORT_MODEL and IMPORT_MODEL.

The examples use the directory object dmdir shown in [Example 8-1](#) (page 8-9) and two schemas, dm1 and dm2. Both schemas have data mining privileges. dm1 has two models. dm2 has one model.

```
SELECT owner, model_name, mining_function, algorithm FROM all_mining_models;
```

OWNER	MODEL_NAME	MINING_FUNCTION	ALGORITHM
DM1	EM_SH_CLUS_SAMPLE	CLUSTERING	EXPECTATION_MAXIMIZATION
DM1	DT_SH_CLAS_SAMPLE	CLASSIFICATION	DECISION_TREE
DM2	SVD_SH_SAMPLE	FEATURE_EXTRACTION	SINGULAR_VALUE_DECOMP

Example 8-1 Creating the Directory Object

```
-- connect as system user
CREATE OR REPLACE DIRECTORY dmdir AS '/scratch/dmuser/expimp';
GRANT READ,WRITE ON DIRECTORY dmdir TO dm1;
GRANT READ,WRITE ON DIRECTORY dmdir TO dm2;
SELECT * FROM all_directories WHERE directory_name IN 'DMDIR';
```

OWNER	DIRECTORY_NAME	DIRECTORY_PATH
SYS	DMDIR	/scratch/dmuser/expimp

Example 8-2 Exporting All Models From DM1

```
-- connect as dm1
BEGIN
  dbms_data_mining.export_model (
    filename => 'all_dm1',
    directory => 'dmdir');
END;
/
```

A log file and a dump file are created in `/scratch/dmuser/expimp`, the physical directory associated with `dmdir`. The name of the log file is `dm1_exp_11.log`. The name of the dump file is `all_dm101.dmp`.

Example 8-3 Importing the Models Back Into DM1

The models that were exported in [Example 8-2](#) (page 8-10) still exist in `dm1`. Since an import does not overwrite models with the same name, you must drop the models before importing them back into the same schema.

```
BEGIN
  dbms_data_mining.drop_model('EM_SH_CLUS_SAMPLE');
  dbms_data_mining.drop_model('DT_SH_CLAS_SAMPLE');
  dbms_data_mining.import_model(
    filename => 'all_dm101.dmp',
    directory => 'DMDIR');
END;
/
SELECT model_name FROM user_mining_models;
```

```
MODEL_NAME
-----
DT_SH_CLAS_SAMPLE
EM_SH_CLUS_SAMPLE
```

Example 8-4 Importing Models Into a Different Schema

In this example, the models that were exported from `dm1` in [Example 8-2](#) (page 8-10) are imported into `dm2`. The `dm1` schema uses the `example` tablespace; the `dm2` schema uses the `sysaux` tablespace.

```
-- CONNECT as sysdba
BEGIN
  dbms_data_mining.import_model (
    filename => 'all_d101.dmp',
    directory => 'DMDIR',
    schema_remap => 'DM1:DM2',
    tablespace_remap => 'EXAMPLE:SYSAUX');
END;
/
-- CONNECT as dm2
SELECT model_name from user_mining_models;
```

```
MODEL_NAME
-----
SVD_SH_SAMPLE
EM_SH_CLUS_SAMPLE
DT_SH_CLAS_SAMPLE
```

Example 8-5 Exporting Specific Models

You can export a single model, a list of models, or a group of models that share certain characteristics.

```
-- Export the model named dt_sh_clas_sample
EXECUTE dbms_data_mining.export_model (
    filename => 'one_model',
    directory => 'DMDIR',
    model_filter => 'name in ('DT_SH_CLAS_SAMPLE')');
-- one_model01.dmp and dml_exp_37.log are created in /scratch/dmuser/expimp

-- Export Decision Tree models
EXECUTE dbms_data_mining.export_model(
    filename => 'algo_models',
    directory => 'DMDIR',
    model_filter => 'ALGORITHM_NAME IN ('DECISION_TREE')');
-- algo_model01.dmp and dml_exp_410.log are created in /scratch/dmuser/expimp

-- Export clustering models
EXECUTE dbms_data_mining.export_model(
    filename => 'func_models',
    directory => 'DMDIR',
    model_filter => 'FUNCTION_NAME = 'CLUSTERING''');
-- func_model01.dmp and dml_exp_513.log are created in /scratch/dmuser/expimp
```

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

8.3.5 Importing From PMML

You can import Regression models represented in Predictive Model Markup Language (PMML).

PMML is an XML-based standard specified by the Data Mining Group (<http://www.dmg.org>). Applications that are PMML-compliant can deploy PMML-compliant models that were created by any vendor. Oracle Data Mining supports the core features of PMML 3.1 for regression models.

You can import regression models represented in PMML. The models must be of type `RegressionModel`, either linear regression or binary logistic regression.

Related Topics:

Oracle Database PL/SQL Packages and Types Reference

8.4 Controlling Access to Mining Models and Data

Understand how to create a Data Mining user and grant necessary privileges.

- [Creating a Data Mining User](#) (page 8-11)
- [System Privileges for Data Mining](#) (page 8-13)
- [Object Privileges for Mining Models](#) (page 8-14)

8.4.1 Creating a Data Mining User

Explains how to create a Data Mining user.

A Data Mining user is a database user account that has privileges for performing data mining activities. [Example 8-6](#) (page 8-12) shows how to create a database user. [Example 8-7](#) (page 8-13) shows how to assign data mining privileges to the user.

Note:

To create a user for the Data Mining sample programs, you must run two configuration scripts as described in "The Data Mining Sample Programs".

Example 8-6 Creating a Database User in SQL*Plus

1. Log in to SQL*Plus with system privileges.

```
Enter user-name: sys as sysdba
Enter password: password
```

2. To create a user named `dmuser`, type these commands. Specify a password of your choosing.

```
CREATE USER dmuser IDENTIFIED BY password
        DEFAULT TABLESPACE USERS
        TEMPORARY TABLESPACE TEMP
        QUOTA UNLIMITED ON USERS;
Commit;
```

The `USERS` and `TEMP` tablespace are included in the pre-configured database that Oracle ships with the database media. `USERS` is used mostly by demo users; it is appropriate for running the sample programs described in "The Data Mining Sample Programs". `TEMP` is the temporary tablespace that is shared by most database users.

Note:

Tablespaces for Data Mining users must be assigned according to standard DBA practices, depending on system load and system resources.

3. To login as `dmuser`, type the following.

```
CONNECT dmuser
Enter password: password
```

See Also:

Oracle Database SQL Language Reference for the complete syntax of the `CREATE USER` statement

Related Topics:

[The Data Mining Sample Programs](#) (page A-1)

Describes the data mining sample programs that ship with Oracle Database.

8.4.1.1 Granting Privileges for Data Mining

You must have the `CREATE MINING MODEL` privilege to create models in your own schema. You can perform any operation on models that you own. This includes applying the model, adding a cost matrix, renaming the model, and dropping the model.

The `GRANT` statements in the following example assign a set of basic data mining privileges to the `dmuser` account. Some of these privileges are not required for all mining activities, however it is prudent to grant them all as a group.

Additional system and object privileges are required for enabling or restricting specific mining activities.

Example 8-7 Privileges Required for Data Mining

```
GRANT CREATE MINING MODEL TO dmuser;
GRANT CREATE SESSION TO dmuser;
GRANT CREATE TABLE TO dmuser;
GRANT CREATE VIEW TO dmuser;
GRANT EXECUTE ON CTXSYS.CTX_DDL TO dmuser;
```

`READ` or `SELECT` privileges are required for data that is not in your schema. For example, the following statement grants `SELECT` access to the `sh.customers` table.

```
GRANT SELECT ON sh.customers TO dmuser;
```

8.4.2 System Privileges for Data Mining

Learn different privileges to control operations on mining models.

A system privilege confers the right to perform a particular action in the database or to perform an action on a type of schema objects. For example, the privileges to create tablespaces and to delete the rows of any table in a database are system privileges.

You can perform specific operations on mining models in other schemas if you have the appropriate system privileges. For example, `CREATE ANY MINING MODEL` enables you to create models in other schemas. `SELECT ANY MINING MODEL` enables you to apply models that reside in other schemas. You can add comments to models if you have the `COMMENT ANY MINING MODEL` privilege.

To grant a system privilege, you must either have been granted the system privilege with the `ADMIN OPTION` or have been granted the `GRANT ANY PRIVILEGE` system privilege.

The system privileges listed in the following table control operations on mining models.

Table 8-2 System Privileges for Data Mining

System Privilege	Allows you to....
<code>CREATE MINING MODEL</code>	Create mining models in your own schema.
<code>CREATE ANY MINING MODEL</code>	Create mining models in any schema.
<code>ALTER ANY MINING MODEL</code>	Change the name or cost matrix of any mining model in any schema.
<code>DROP ANY MINING MODEL</code>	Drop any mining model in any schema.

Table 8-2 (Cont.) System Privileges for Data Mining

System Privilege	Allows you to....
SELECT ANY MINING MODEL	Apply a mining model in any schema, also view model details in any schema.
COMMENT ANY MINING MODEL	Add a comment to any mining model in any schema.)
AUDIT_ADMIN role	Generate an audit trail for any mining model in any schema. (See <i>Oracle Database Security Guide</i> for details.)

Example 8-8 Grant System Privileges for Data Mining

The following statements allow `dmuser` to score data and view model details in any schema as long as `SELECT` access has been granted to the data. However, `dmuser` can only create models in the `dmuser` schema.

```
GRANT CREATE MINING MODEL TO dmuser;
GRANT SELECT ANY MINING MODEL TO dmuser;
```

The following statement revokes the privilege of scoring or viewing model details in other schemas. When this statement is executed, `dmuser` can only perform data mining activities in the `dmuser` schema.

```
REVOKE SELECT ANY MINING MODEL FROM dmuser;
```

Related Topics:

[Adding a Comment to a Mining Model](#) (page 8-15)

Oracle Database Security Guide

8.4.3 Object Privileges for Mining Models

An object privilege confers the right to perform a particular action on a specific schema object. For example, the privilege to delete rows from the `SH.PRODUCTS` table is an example of an object privilege.

You automatically have all object privileges for schema objects in your own schema. You can grant object privilege on objects in your own schema to other users or roles.

The object privileges listed in the following table control operations on specific mining models.

Table 8-3 Object Privileges for Mining Models

Object Privilege	Allows you to....
ALTER MINING MODEL	Change the name or cost matrix of the specified mining model object.
SELECT MINING MODEL	Apply the specified mining model object and view its model details.

Example 8-9 Grant Object Privileges on Mining Models

The following statements allow `dmuser` to apply the model `testmodel` to the `sales` table, specifying different cost matrixes with each apply. The user `dmuser` can also

rename the model `testmodel`. The `testmodel` model and `sales` table are in the `sh` schema, not in the `dmuser` schema.

```
GRANT SELECT ON MINING MODEL sh.testmodel TO dmuser;
GRANT ALTER ON MINING MODEL sh.testmodel TO dmuser;
GRANT SELECT ON sh.sales TO dmuser;
```

The following statement prevents `dmuser` from renaming or changing the cost matrix of `testmodel`. However, `dmuser` can still apply `testmodel` to the `sales` table.

```
REVOKE ALTER ON MINING MODEL sh.testmodel FROM dmuser;
```

8.5 Auditing and Adding Comments to Mining Models

Mining model objects support SQL `COMMENT` and `AUDIT` statements.

8.5.1 Adding a Comment to a Mining Model

Comments can be used to associate descriptive information with a database object. You can associate a comment with a mining model using a SQL `COMMENT` statement.

```
COMMENT ON MINING MODEL schema_name.model_name IS string;
```

Note:

To add a comment to a model in another schema, you must have the `COMMENT ANY MINING MODEL` system privilege.

To drop a comment, set it to the empty `' '` string.

The following statement adds a comment to the model `DT_SH_CLAS_SAMPLE` in your own schema.

```
COMMENT ON MINING MODEL dt_sh_clas_sample IS
'Decision Tree model predicts promotion response';
```

You can view the comment by querying the catalog view `USER_MINING_MODELS`.

```
SELECT model_name, mining_function, algorithm, comments FROM user_mining_models;
```

MODEL_NAME	MINING_FUNCTION	ALGORITHM	COMMENTS
DT_SH_CLAS_SAMPLE	CLASSIFICATION	DECISION_TREE	Decision Tree model predicts promotion response

To drop this comment from the database, issue the following statement:

```
COMMENT ON MINING MODEL dt_sh_clas_sample '';
```

See Also:

- [Table 8-2](#) (page 8-13)
 - *Oracle Database SQL Language Reference* for details about SQL `COMMENT` statements
-
-

8.5.2 Auditing Mining Models

The Oracle Database auditing system is a powerful, highly configurable tool for tracking operations on schema objects in a production environment. The auditing system can be used to track operations on data mining models.

Note:

To audit mining models, you must have the `AUDIT_ADMIN` role.

Unified auditing is documented in *Oracle Database Security Guide*. However, the full unified auditing system is not enabled by default. Instructions for migrating to unified auditing are provided in *Oracle Database Upgrade Guide*.

See Also:

- "Auditing Oracle Data Mining Events" in *Oracle Database Security Guide* for details about auditing mining models
 - "Monitoring Database Activity with Auditing" in *Oracle Database Security Guide* for a comprehensive discussion of unified auditing in Oracle Database
 - "About the Unified Auditing Migration Process for Oracle Database" in *Oracle Database Upgrade Guide* for information about migrating to unified auditing
 - *Oracle Database Upgrade Guide*
-
-

The Data Mining Sample Programs

Describes the data mining sample programs that ship with Oracle Database.

- [About the Data Mining Sample Programs](#) (page A-1)
- [Installing the Data Mining Sample Programs](#) (page A-2)
- [The Data Mining Sample Data](#) (page A-3)

A.1 About the Data Mining Sample Programs

You can learn a great deal about the Oracle Data Mining application programming interface (API) from the data mining sample programs. The programs illustrate typical approaches to data preparation, algorithm selection, algorithm tuning, testing, and scoring.

The programs are easy to use. They include extensive inline comments to help you understand the code. They delete all temporary objects on exit; you can run the programs repeatedly without setup or cleanup.

The data mining sample programs are installed with Oracle Database Examples in the demo directory under Oracle Home. The demo directory contains sample programs that illustrate many features of Oracle Database. You can locate the data mining files by doing a directory listing of `dm* .sql`. The following example shows this directory listing on a Linux system.

Note that the directory listing in the following example includes one file, `dmhpdemo .sql`, that is *not* a data mining program.

Example A-1 Directory Listing of the Data Mining Sample Programs

```
> cd $ORACLE_HOME/rdbms/demo
> ls dm*.sql
dmaidemo.sql      dmkmdemo.sql      dmsvddemo.sql
dmardemo.sql      dmnbdemo.sql      dmsvodem.sql
dmdtdemo.sql      dmnmddemo.sql     dmsvrdem.sql
dmdtxvlddemo.sql dmocdemo.sql      dmtxtnmf.sql
dmemdemo.sql      dmsh.sql           dmtxtsvm.sql
dmglcdem.sql      dmshgrants.sql
dmglrdem.sql      dmstardemo.sql
dmhpdemo.sql      dmsvcdem.sql
```

The data mining sample programs create a set of mining models in the user's schema. After executing the programs, you can list the models with a query like the one in the following example.

Example A-2 Models Created by the Sample Programs

```
SELECT mining_function, algorithm, model_name FROM user_mining_models
       ORDER BY mining_function;
```

MINING_FUNCTION	ALGORITHM	MODEL_NAME
ASSOCIATION_RULES	APRIORI_ASSOCIATION_RULES	AR_SH_SAMPLE
CLASSIFICATION	GENERALIZED_LINEAR_MODEL	GLMC_SH_CLAS_SAMPLE
CLASSIFICATION	SUPPORT_VECTOR_MACHINES	T_SVM_CLAS_SAMPLE
CLASSIFICATION	SUPPORT_VECTOR_MACHINES	SVMC_SH_CLAS_SAMPLE
CLASSIFICATION	SUPPORT_VECTOR_MACHINES	SVMO_SH_CLAS_SAMPLE
CLASSIFICATION	NAIVE_BAYES	NB_SH_CLAS_SAMPLE
CLASSIFICATION	DECISION_TREE	DT_SH_CLAS_SAMPLE
CLUSTERING	EXPECTATION_MAXIMIZATION	EM_SH_CLUS_SAMPLE
CLUSTERING	O_CLUSTER	OC_SH_CLUS_SAMPLE
CLUSTERING	KMEANS	KM_SH_CLUS_SAMPLE
CLUSTERING	KMEANS	DM_STAR_CLUSTER
FEATURE_EXTRACTION	SINGULAR_VALUE_DECOMP	SVD_SH_SAMPLE
FEATURE_EXTRACTION	NONNEGATIVE_MATRIX_FACTOR	NMF_SH_SAMPLE
FEATURE_EXTRACTION	NONNEGATIVE_MATRIX_FACTOR	T_NMF_SAMPLE
REGRESSION	SUPPORT_VECTOR_MACHINES	SVMR_SH_REGR_SAMPLE
REGRESSION	GENERALIZED_LINEAR_MODEL	GLMR_SH_REGR_SAMPLE

A.2 Installing the Data Mining Sample Programs

Learn how to install Data Mining sample programs.

The data mining sample programs require:

- Oracle Database Enterprise Edition with the Advanced Analytics option
- Oracle Database sample schemas
- Oracle Database Examples
- A data mining user account
- Execution of `dmsgrants.sql` by a system administrator
- Execution of `dms.sh.sql` by the data mining user

Follow these steps to install the data mining sample programs:

1. Install or obtain access to Oracle Database 12c Enterprise Edition with the Advanced Analytics option. To install the Database, see the installation instructions for your platform at <http://www.oracle.com/pls/db122/homepage>.
2. Ensure that the sample schemas are installed in the database. The sample schemas are installed by default with Oracle Database. See *Oracle Database Sample Schemas* for details about the sample schemas.
3. Verify that Oracle Database Examples has been installed with the database, or install it locally. Oracle Database Examples loads the Database sample programs into the `rdbms/demo` directory under Oracle home. See *Oracle Database Examples Installation Guide* for installation instructions.
4. Verify that a data mining user account has been created, or create it yourself if you have administrative privileges. See "Creating a Data Mining User".

5. Ask your system administrator to run `dmshgrants.sql`, or run it yourself if you have administrative privileges. `dmshgrants` grants the privileges that are required for running the sample programs. These include `SELECT` access to tables in the `SH` schema as described in "The Data Mining Sample Data" and the system privileges listed in the following table.

Pass the name of the data mining user to `dmshgrants`.

```
SQL> CONNECT sys / as sysdba
Enter password: sys_password
Connected.
SQL> @ $ORACLE_HOME/rdbms/demo/dmshgrants dmuser
```

Table A-1 System Privileges Granted by `dmshgrants.sql` to the Data Mining User

Privilege	Allows the data mining user to
<code>CREATE SESSION</code>	log in to a database session
<code>CREATE TABLE</code>	create tables, such as the settings tables for <code>CREATE_MODEL</code>
<code>CREATE VIEW</code>	create views, such as the views of tables in the <code>SH</code> schema
<code>CREATE MINING MODEL</code>	create data mining models
<code>EXECUTE ON ctxsys.ctx_ddl</code>	execute procedures in the <code>ctxsys.ctx_ddl</code> PL/SQL package; required for text mining

6. Connect to the database as the data mining user and run `dmsh.sql`. This script creates views of the sample data in the schema of the data mining user.

```
SQL> CONNECT dmuser
Enter password: dmuser_password
Connected.
SQL> @ $ORACLE_HOME/rdbms/demo/dmsh
```

Related Topics:

Oracle Database Sample Schemas

Oracle Database Examples Installation Guide

[Creating a Data Mining User](#) (page 8-11)

Explains how to create a Data Mining user.

A.3 The Data Mining Sample Data

The data used by the sample data mining programs is based on these tables in the `SH` schema:

```
SH.CUSTOMERS
SH.SALES
SH.PRODUCTS
SH.SUPPLEMENTARY_DEMOGRAPHICS
SH.COUNTRIES
```

The `dmsgrants` script grants `SELECT` access to the tables in `SH`. The `dms.sh.sql` script creates views of the `SH` tables in the schema of the data mining user. The views are described in the following table:

Table A-2 The Data Mining Sample Data

View Name	Description
<code>MINING_DATA</code>	Joins and filters data
<code>MINING_DATA_BUILD_V</code>	Data for building models
<code>MINING_DATA_TEST_V</code>	Data for testing models
<code>MINING_DATA_APPLY_V</code>	Data to be scored
<code>MINING_BUILD_TEXT</code>	Data for building models that include text
<code>MINING_TEST_TEXT</code>	Data for testing models that include text
<code>MINING_APPLY_TEXT</code>	Data, including text columns, to be scored
<code>MINING_DATA_ONE_CLAS S_V</code>	Data for anomaly detection

The association rules program creates its own transactional data.

A

ADP, [5-5](#)
Advanced Analytics option, [8-1](#), [A-2](#)
algorithms
 parallel execution, [8-2](#)
ALL_MINING_MODEL_ATTRIBUTES, [2-2](#)
ALL_MINING_MODEL_SETTINGS, [2-2](#), [5-11](#)
ALL_MINING_MODEL_VIEWS, [2-2](#)
ALL_MINING_MODEL_XFORMS, [2-2](#)
ALL_MINING_MODELS, [2-2](#)
anomaly detection, [2-1](#), [3-2](#), [5-3](#), [5-4](#), [6-12](#)
APPLY, [6-1](#)
Apriori
 example: calculating aggregates, [3-11](#)
association rules, [5-2](#), [5-4](#)
Association Rules, [5-20](#)
attribute importance, [2-1](#), [3-6](#), [5-3](#)
attribute specification, [4-6](#), [7-5](#), [7-6](#)
attributes
 categorical, [3-5](#), [7-1](#)
 data attributes, [3-3](#)
 data dictionary, [2-2](#)
 model attributes, [3-3](#), [3-4](#)
 nested, [3-2](#)
 numerical, [3-4](#), [7-1](#)
 subname, [3-5](#)
 target, [3-4](#)
 text, [3-5](#)
 unstructured text, [7-1](#)
AUDIT, [8-14](#), [8-16](#)
Automatic Data Preparation, [1-1](#), [3-3](#), [4-3](#)

B

binning
 equi-width, [4-10](#)
 quantile, [4-10](#)
 supervised, [4-3](#), [4-10](#)
 top-n frequency, [4-10](#)
build data, [3-2](#)

C

case ID, [3-1](#), [3-2](#), [3-5](#), [6-12](#)
case table, [3-1](#), [4-2](#)
categorical attributes, [7-1](#)
chopt utility, [8-2](#)
class weights, [5-10](#)
classification, [2-1](#), [3-2](#), [3-4](#), [5-3](#), [5-4](#)
Classification Algorithm, [5-28](#)
clipping, [4-4](#)
CLUSTER_DETAILS, [1-6](#), [2-9](#)
CLUSTER_DISTANCE, [2-9](#)
CLUSTER_ID, [1-5](#), [2-9](#), [2-10](#)
CLUSTER_PROBABILITY, [2-9](#)
CLUSTER_SET, [1-6](#), [2-10](#)
clustering, [1-5](#), [2-1](#), [3-2](#), [5-4](#)
COMMENT, [8-14](#)
cost matrix, [5-9](#), [6-10](#), [8-14](#)
cost-sensitive prediction, [6-10](#)

D

data
 categorical, [3-4](#)
 dimensioned, [3-8](#)
 for sample programs, [A-3](#)
 market basket, [3-9](#)
 missing values, [3-12](#)
 multi-record case, [3-8](#)
 nested, [3-2](#)
 numerical, [3-4](#)
 preparation, [4-1](#)
 READ access, [8-13](#)
 SELECT access, [8-13](#)
 single-record case, [3-1](#)
 sparse, [3-12](#)
 transactional, [3-9](#)
 unstructured text, [3-4](#)
data mining
 applications of, [1-1](#)
 database tuning for, [8-2](#)
 privileges for, [8-1](#), [8-12](#), [A-3](#)

data mining (*continued*)
 sample programs, [A-1](#)
 scoring, [5-2](#), [6-1](#)
Data Mining with SQL
 FEATURE_COMPARE
 ESA, [1-6](#)
Data preparation
 model view
 text features, [7-2](#)
data types
 nested, [3-6](#)
Database Upgrade Assistant, [8-4](#)
DBMS_DATA_MINING, [2-7](#), [5-2](#)
DBMS_DATA_MINING_TRANSFORM, [2-7](#)
DBMS_PREDICTIVE_ANALYTICS, [1-4](#), [2-7](#), [2-9](#)
Decision Tree, [4-4](#), [5-3](#), [5-4](#), [6-8](#)
desupported features
 Java API, [8-3](#)
directory objects, [8-9](#)
DMEIDMSYS, [8-5](#)
downgrading, [8-7](#)

E

Expectation Maximization, [4-4](#)
EXPLAIN, [2-9](#)
Explicit Semantic Analysis, [5-4](#), [5-47](#)
exporting, [8-4](#), [8-7](#)

F

feature extraction, [2-1](#), [3-2](#), [5-3](#), [5-4](#)
FEATURE_COMPARE, [2-10](#)
FEATURE_DETAILS, [2-10](#)
FEATURE_ID, [2-10](#)
FEATURE_SET, [2-10](#)
FEATURE_VALUE, [2-10](#)
Frequent Itemsets, [5-26](#)

G

Generalized Linear Models, [4-4](#)
GLM, [5-4](#)
graphical user interface, [1-1](#)

I

importing, [8-4](#), [8-7](#)
installation
 Oracle Database, [8-1](#), [A-2](#)
 Oracle Database Examples, [A-2](#)
 sample data mining programs, [A-2](#)
 sample schemas, [A-2](#)

K

k-Means, [4-4](#), [5-3](#)

L

linear regression, [2-10](#), [5-3](#)
logistic regression, [2-10](#), [5-3](#)

M

market basket data, [3-9](#)
MDL, [4-4](#)
memory, [8-2](#)
Minimum Description Length, [4-4](#), [5-3](#), [5-52](#)
mining functions
 supervised, [5-2](#)
 unsupervised, [5-2](#)
mining models
 adding a comment, [2-1](#), [8-15](#)
 applying, [8-14](#)
 auditing, [2-1](#), [8-16](#)
 changing the name, [8-14](#)
 created by sample programs, [A-1](#)
 data dictionary, [2-2](#)
 object privileges, [8-14](#)
 privileges for, [2-1](#)
 upgrading, [8-4](#)
 viewing model details, [8-14](#)
missing value treatment, [3-13](#)
model attributes
 categorical, [3-4](#)
 derived from nested column, [3-5](#)
 numerical, [3-4](#)
 scoping of name, [3-5](#)
 text, [3-4](#)
Model Detail View
 model view
 Clustering algorithm, [5-39](#)
 Decision Tree, [5-29](#)
 global, [5-53](#)
 MDL, [5-28](#), [5-52](#)
 SVM, [5-38](#)
Model Detail Views, [5-19](#)
model details, [3-6](#)
Model details
 binning, [5-53](#)
Model Details View
 model view
 EM, [5-42](#)
 GLM, [5-31](#)
 KM, [5-45](#)
 OC, [5-46](#)
model signature, [3-5](#)
models
 algorithms, [5-3](#)
 created by sample programs, [A-1](#)
 deploying, [6-1](#)
 partitions, [2-2](#)

models (*continued*)
privileges for, 8-13
settings, 2-2, 5-11
testing, 3-2
training, 3-2
transparency, 1-1
XFORMS, 2-2

N

Naive Bayes, 4-4, 5-3, 5-4, 5-37
nested data, 3-6, 7-3
Non-Negative Matrix Factorization, 4-4, 5-3, 5-48
normalization
min-max, 4-10
scale, 4-10
z-score, 4-10
Normalization view
model view
missing value handling, 5-54
numerical attributes, 7-1

O

O-Cluster, 3-7, 4-4, 5-3, 5-4
object privileges, 8-14
One-Class SVM, 5-3
ORA_DM_PARTITION_NAME ORA, 2-10
Oracle Data Miner, 1-1, 8-3
Oracle Data Miner Classic, 8-3
Oracle Data Pump, 8-7
Oracle Text, 7-1
outliers, 4-4, 4-11

P

parallel execution, 6-2, 8-2
Partitioned model
partitioned model scoring, 5-7
Partitioned Model
add partition, 5-7
DDL implementation, 5-6
drop model, 5-6
drop partition, 5-6
Partitioned Model Build, 5-6
partitions
data dictionary, 2-2
PGA, 8-2
PL/SQL packages, 2-7
PMML, 8-11
PREDICTION
GROUPING hint, 6-7
PREDICTION_BOUNDS, 2-10
PREDICTION_COST, 2-10
PREDICTION_DETAILS, 2-10, 6-8
PREDICTION_PROBABILITY, 1-3, 2-10, 6-8

PREDICTION_SET, 2-10
predictive analytics, 1-1, 1-4, 2-1
Preparing the Data
Using Retail Analysis Data
Aggregates, 3-11
prior probabilities, 5-10
priors table, 5-10
privileges
for creating mining models, 8-6
for data mining, 8-1, 8-8
for data mining sample programs, A-3
for exporting and importing, 8-8
required for data mining, 8-13

R

R mining model
settings, 5-12
regression, 2-1, 3-2, 3-4, 5-3, 5-4
reverse transformations, 3-6

S

sample programs
configuration scripts, 8-12
data used by, A-4
directory listing of, A-1
installing, A-2
models created by, A-1
Oracle Database Examples, A-2
requirements, A-2
sample schemas, A-2
scoring
data, 3-2
dynamic, 1-3, 2-1, 6-8
parallel execution, 6-2
privileges for, 8-14
requirements, 3-2
SQL functions, 2-9
transparency, 1-1
Scoring Engine, 8-4
settings
data dictionary, 2-2
table for specifying, 5-1
SGA, 8-2
Singular Value Decomposition, 4-5, 5-50
sparse data, 3-12
SQL AUDIT, 2-1, 8-16
SQL COMMENT, 2-1, 8-15
SQL data mining functions, 2-9
SQL Developer, 1-1
STACK, 2-8, 4-8
Static Dictionary Views
ALL_MINING_MODEL_VIEWS, 2-5
Support Vector Machine, 4-5, 5-3, 5-4
system privileges, 8-13, A-2

T

target, [3-4](#), [3-5](#), [7-3](#)
test data, [3-2](#), [5-1](#)
text attributes, [7-2](#), [7-5](#)
text mining, [2-8](#), [7-1](#)
text policy, [7-4](#)
text terms, [7-1](#)
training data, [5-1](#)
transactional data, [3-1](#), [3-8](#), [3-9](#)
Transactional Itemsets, [5-26](#)
Transactional rule, [5-27](#)
transformations
 attribute-specific, [2-8](#)
 embedded, [2-8](#), [3-3](#), [4-1](#)
 user-specified, [3-3](#)
transparency, [3-6](#)
trimming, [4-11](#)

U

upgrading

upgrading (*continued*)

 exporting and importing, [8-4](#)
 from Release 10g, [8-4](#)
 from Release 11g, [8-4](#)
 pre-upgrade steps, [8-3](#)
 using Database Upgrade Assistant, [8-4](#)

Usage scripts, [5-19](#)

users

 assigning data mining privileges to, [8-13](#)
 creating, [8-12](#)
 privileges for data mining, [8-6](#), [8-12](#)

W

weights, [5-10](#)

windsorize, [4-11](#)

X

XFORM, [2-8](#)

XFORMS

 data dictionary, [2-2](#)