# Oracle® Data Mining

User's Guide 12*c* Release 1 (12.1) **E53115-04** 

October 2014



Oracle Data Mining User's Guide, 12c Release 1 (12.1)

E53115-04

Copyright © 2005, 2014, Oracle and/or its affiliates. All rights reserved.

Primary Author: Kathy L. Taylor

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, 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 on 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. 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.

# Contents

⊃re	eface	vi
	Audience	vi
	Related Documentation	vii
	Oracle Data Mining Resources on the Oracle Technology Network	vii
	Application Development and Database Administration Documentation	viii
	Documentation Accessibility	vii
	Conventions	. ix
Ch	nanges in This Release for Oracle Data Mining User's Guide	. x
	Oracle Data Mining User's Guide is New in This Release	. x
	Changes in Oracle Data Mining 12c Release 1 (12.1)	
	New Features	. x
	Desupported Features	xi
	Other Changes	xii
ı	Data Mining With SQL	
	Highlights of the Data Mining API	1-1
	Example: Targeting Likely Candidates for a Sales Promotion	1-2
	Example: Analyzing Preferred Customers	1-3
	Example: Segmenting Customer Data	1-5
2	About the Data Mining API	
	About Mining Models	2-1
	Data Mining Data Dictionary Views	2-2
	USER_MINING_MODELS	2-2
	USER_MINING_MODEL_ATTRIBUTES	2-2
	USER_MINING_MODEL_SETTINGS	2-3
	Data Mining PL/SQL Packages	2-4
	DBMS_DATA_MINING	2-4
	DBMS_DATA_MINING_TRANSFORM	2-5
	DBMS PREDICTIVE ANALYTICS	2-6

Data Mining SQL Scoring Functions	
Preparing the Data	
Data Requirements	
Column Data Types	
Data Sets for Classification and Regression	
Scoring Requirements	
About Attributes	
Data Attributes and Model Attributes	
Target Attribute	
Numericals, Categoricals, and Unstructured Text	
Model Signature	
Scoping of Model Attribute Name	
Model Details	
Using Nested Data	
Nested Object Types	
Example: Transforming Transactional Data for Mining	
Using Market Basket Data	
Example: Creating a Nested Column for Market Basket Analysis	
Handling Missing Values	
Examples: Missing Values or Sparse Data?	
Missing Value Treatment in Oracle Data Mining	
Changing the Missing Value Treatment	
Transforming the Data	
About Transformations	
Preparing the Case Table	
Creating Nested Columns	
Converting Column Data Types	
Text Transformation	
About Business and Domain-Sensitive Transformations	
Understanding Automatic Data Preparation	
Binning	
Normalization	
Outlier Treatment	
How ADP Transforms the Data	
Embedding Transformations in a Model	
Specifying Transformation Instructions for an Attribute	
Building a Transformation List	
Transformation Lists and Automatic Data Preparation	
Oracle Data Mining Transformation Routines	
Understanding Reverse Transformations	

5	Creating a Model	
	Before Creating a Model	. 5-1
	The CREATE_MODEL Procedure	. 5-1
	Choosing the Mining Function	. 5-2
	Choosing the Algorithm	. 5-3
	Supplying Transformations	. 5-4
	Specifying Model Settings	
	Specifying Costs	. 5-6
	Specifying Prior Probabilities	. 5-7
	Specifying Class Weights	
	Model Settings in the Data Dictionary	
	Viewing Model Details	. 5-9
6	Scoring and Deployment	
	About Scoring and Deployment	. 6-1
	Using the Data Mining SQL Functions	. 6-2
	Choosing the Predictors	. 6-2
	Single-Record Scoring	. 6-3
	Prediction Details	. 6-4
	Cluster Details	. 6-4
	Feature Details	. 6-5
	Prediction Details	. 6-5
	Real-Time Scoring	. 6-7
	Dynamic Scoring	. 6-7
	Cost-Sensitive Decision Making	
	DBMS_DATA_MINING.Apply	6-10
7	Mining Unstructured Text	
	About Unstructured Text	. 7-1
	About Text Mining and Oracle Text	. 7-1
	Creating a Model that Includes Text Mining	. 7-2
	Creating a Text Policy	. 7-3
	Configuring a Text Attribute	. 7-4
3	Administrative Tasks for Oracle Data Mining	
	Installing and Configuring a Database for Data Mining	. 8-1
	About Installation	. 8-1
	Enabling or Disabling a Database Option	. 8-2
	Database Tuning Considerations for Data Mining	. 8-2
	Upgrading or Downgrading Oracle Data Mining	. 8-2
	Pre-Upgrade Steps	. 8-3
	Upgrading Oracle Data Mining	. 8-3

	Post Upgrade Steps	8-5
	Downgrading Oracle Data Mining	8-6
	Exporting and Importing Mining Models	8-6
	About Oracle Data Pump	8-6
	Options for Exporting and Importing Mining Models	8-6
	Directory Objects for EXPORT_MODEL and IMPORT_MODEL	8-7
	Using EXPORT_MODEL and IMPORT_MODEL	8-8
	Importing From PMML	8-10
	Controlling Access to Mining Models and Data	8-10
	Creating a Data Mining User	8-10
	System Privileges for Data Mining	8-11
	Object Privileges for Mining Models	8-12
	Auditing and Adding Comments to Mining Models	8-13
	Adding a Comment to a Mining Model	8-13
	Auditing Mining Models	8-14
Α	The Data Mining Sample Programs	
	About the Data Mining Sample Programs	A-1
	Installing the Data Mining Sample Programs	A-2
	The Data Mining Sample Data	A-3

# Index

# **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
- Related Documentation
- Documentation Accessibility
- Conventions

# **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.

# **Related Documentation**

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

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

The following manuals document Oracle Data Mining:

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

### 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=db121&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=db121&id=datminGUI

# **Application Development and Database Administration Documentation**

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

- Oracle Database Concepts
- Oracle Database Administrator's Guide
- Oracle Database Development Guide

# **Documentation Accessibility**

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

### **Access to Oracle Support**

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

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

# **Conventions**

The following text conventions are used in this document:

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

# Changes in This Release for Oracle Data Mining User's Guide

This preface lists changes in this release in 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. 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.

# **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".

Dynamic scoring

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

See "Dynamic Scoring".

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.

- New clustering algorithm: Expectation Maximization
   See the following:
  - Table 4-1
  - Example 1-6
  - Example 1-7
  - Example 1-8
  - Example 6-6
  - "About the Data Mining Sample Programs"
- New feature extraction algorithm: Singular Value Decomposition with Principal Component Analysis

See the following:

- Table 4-1
- Example 6-7
- "About the Data Mining Sample Programs"
- Generalized Linear Models are enhanced to support feature selection and creation.
   See The Data Mining Sample Programs.

# **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.

New support for native double data types, BINARY\_DOUBLE and BINARY\_FLOAT, improves the performance of the SQL scoring functions.

See Preparing the Data.

• Decision Tree algorithm now supports nested data. See Preparing the Data.

# **Data Mining With SQL**

This chapter introduces the Oracle Data Mining application programming interface (API) and shows how it can be used to solve business problems. This chapter contains the following topics:

- Highlights of the Data Mining API
- Example: Targeting Likely Candidates for a Sales Promotion
- **Example: Analyzing Preferred Customers**
- **Example: Segmenting Customer Data**

# Highlights of the Data Mining 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=db121&id=datminGUI

#### Note:

A set of sample data mining programs ship with Oracle Database. The examples in this manual are taken from these samples. See The Data Mining Sample Programs for more information.

### See Also:

Oracle Data Mining Concepts Part I for an introduction to Oracle Data Mining

# **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 would be 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

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

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_V	ALUE	COST		
0	0	0		
0	1	1		
1	0	8		

# **Example: Analyzing Preferred Customers**

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

### 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
__________
           40 4 36
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
              .96
.96
   102365
   102330
              .95
   101733
   101733 .93
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 Will Become a Preferred Customer

This query computes the probability of a first-time customer becoming a preferred customer (an affinity card holder). This query could 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%.

### 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
.209628541 1
.199794636 2
HOUSEHOLD SIZE
CUST_MARITAL_STATUS
                                            .111683067 3
AGE
```

# **Example: Segmenting Customer Data**

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

### **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 3 7 12 17 16	311 294 215 201 123 114 86	
19 15 18	64 56 36	

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

The query in Example 1-6 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 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
```

# **About the Data Mining API**

This chapter provides an overview of the components of the Oracle Data Mining API. This chapter contains the following topics:

- About Mining Models
- Data Mining Data Dictionary Views
- Data Mining PL/SQL Packages
- Data Mining SQL Scoring Functions

# **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 the user. For more information, see:

- "Dynamic Scoring"
- "DBMS PREDICTIVE ANALYTICS"

#### See Also:

- Creating a Model
- Administrative Tasks for Oracle Data Mining

# **Data Mining Data Dictionary Views**

The data dictionary views for Oracle Data Mining are listed in Table 2-1. 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_ATTRI BUTES	Provides information about the attributes of all accessible mining models
ALL_MINING_MODEL_SETTI	Provides information the configuration settings for all accessible mining models

# USER\_MINING\_MODELS

Example 2-1 describes USER\_MINING\_MODELS and shows a sample query.

### Example 2-1 USER\_MINING\_MODELS

describe user_mining_models					
Name	Null?	Туре			
MODEL_NAME	NOT NULL	VARCHAR2(128)			
MINING_FUNCTION		VARCHAR2(30)			
ALGORITHM		VARCHAR2(30)			
CREATION_DATE	NOT NULL	DATE			
BUILD_DURATION		NUMBER			
MODEL_SIZE		NUMBER			
COMMENTS		VARCHAR2(4000)			

The following query returns the models in the user's schema that use the Support Vector Machine algorithm.

```
SELECT mining_function, model_name, algorithm
   FROM user_mining_models
   WHERE algorithm IN 'SUPPORT_VECTOR_MACHINES'
   ORDER BY mining_function;
```

MINING_FUNCTION	MODEL_NAME	ALGORITHM
CLASSIFICATION CLASSIFICATION CLASSIFICATION REGRESSION	T_SVM_CLAS_SAMPLE SVMC_SH_CLAS_SAMPLE	SUPPORT_VECTOR_MACHINES SUPPORT_VECTOR_MACHINES SUPPORT_VECTOR_MACHINES SUPPORT_VECTOR_MACHINES

### See Also:

- · Creating a Model
- ALL\_MINING\_MODELS in Oracle Database Reference

# USER\_MINING\_MODEL\_ATTRIBUTES

Example 2-2 describes USER\_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 USER\_MINING\_MODEL\_ATTRIBUTES

describe user_mining_model_attrib	utes Null?	Type
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 in the user's schema. The model has both categorical and numerical attributes and includes one attribute that is unstructured text.

```
SELECT model_name, attribute_name, attribute_type, target
FROM user_mining_model_attributes
WHERE model_name IN 'T_SVM_CLAS_SAMPLE';
```

MODEL_NAME	ATTRIBUTE_NAME	ATTRIBUTE_TYPE	TAR
T SVM CLAS SAMPLE	PRINTER SUPPLIES	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	CUST INCOME LEVEL	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	AGE	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	HOME_THEATER_PACKAGE	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	CUST_GENDER	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	CUST_MARITAL_STATUS	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	BOOKKEEPING_APPLICATION	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	EDUCATION	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	HOUSEHOLD_SIZE	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	FLAT_PANEL_MONITOR	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	COUNTRY_NAME	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	OS_DOC_SET_KANJI	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	COMMENTS	TEXT	NO
T_SVM_CLAS_SAMPLE	OCCUPATION	CATEGORICAL	NO
T_SVM_CLAS_SAMPLE	Y_BOX_GAMES	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	BULK_PACK_DISKETTES	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	YRS_RESIDENCE	NUMERICAL	NO
T_SVM_CLAS_SAMPLE	AFFINITY_CARD	CATEGORICAL	YES

### See Also:

- "About Attributes"
- ALL\_MINING\_MODEL\_ATTRIBUTES in Oracle Database Reference

# USER\_MINING\_MODEL\_SETTINGS

Example 2-3 describes USER\_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 2-3 USER\_MINING\_MODEL\_SETTINGS

describe user_mining_model_settings			
Name	Null?	Type	
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 in the user's schema. The model uses the Singular Value Decomposition algorithm for feature extraction.

```
SELECT model_name, setting_name, setting_value, setting_type
   FROM user_mining_model_settings
   WHERE model_name IN 'SVD_SH_SAMPLE'
   ORDER BY setting_name;
```

MODEL_NAME	SETTING_NAME	SETTING_VALUE	SETTING
SVD_SH_SAMPLE	ALGO_NAME	ALGO_SINGULAR_VALUE_DECOMP	INPUT
SVD_SH_SAMPLE	FEAT_NUM_FEATURES	10	DEFAULT
SVD_SH_SAMPLE	ODMS_APPROXIMATE_COMPUTATION	ODMS_APPR_COMP_DISABLE	DEFAULT
SVD_SH_SAMPLE	PREP_AUTO	OFF	DEFAULT
SVD_SH_SAMPLE	SVDS_SCORING_MODE	SVDS_SCORING_SVD	DEFAULT
SVD_SH_SAMPLE	SVDS_U_MATRIX_OUTPUT	SVDS_U_MATRIX_ENABLE	INPUT

### See Also:

- "Specifying Model Settings"
- ALL\_MINING\_MODEL\_SETTINGS in Oracle Database Reference

# **Data Mining PL/SQL Packages**

The PL/SQL interface to Oracle Data Mining is implemented in three packages, as shown in Table 2-2.

Table 2-2 Data Mining PL/SQL Packages

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

### DBMS\_DATA\_MINING

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

### DDL in DBMS DATA MINING

Table 2-3 describes the DDL operations for mining models.

Table 2-3 DDL for Mining Models

DDL	DBMS_DATA_MINING	Description
Create model	CREATE_MODEL	Creates a model
Drop model	DROP_MODEL	Drops a model
Rename model	RENAME_MODEL	Renames a model
Add cost matrix	ADD_COST_MATRIX	Adds a cost matrix to a classification model
Remove cost matrix	REMOVE_COST_MATRIX	Removes the cost matrix from a classification model
Alter reverse expression	ALTER_REVERSE_EXPRES SION	Alters the reverse transformation expression associated with a model

The DBMS\_DATA\_MINING package contains a number of functions that return information about mining models. For example, the query in Example 2-4 returns details about feature 1 of the feature extraction model named NMF\_SH\_Sample.

### Example 2-4 Sample Model Details Query

### See Also:

- Creating a Model
- DBMS\_DATA\_MINING in Oracle Database PL/SQL Packages and Types Reference

### DBMS\_DATA\_MINING\_TRANSFORM

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 should be interpreted as unstructured text, or that the column should be excluded from Automatic Data Preparation.

### See Also:

- Transforming the Data
- DBMS\_DATA\_MINING\_TRANSFORM in Oracle Database PL/SQL Packages and Types Reference

### Transformation Methods in DBMS DATA MINING TRANSFORM

Table 2-4 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 Example 2-5 create an 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 2-5 Sample Embedded Transformation

# DBMS\_PREDICTIVE\_ANALYTICS

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 Example 2-6 lists attributes in the view mining\_data\_build\_v in order of their importance in predicting affinity\_card.

### Example 2-6 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;
```

### See Also:

DBMS\_PREDICTIVE\_ANALYTICS in Oracle Database PL/SQL Packages and Types Reference

# **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. Table 2-5 lists the Data Mining SQL functions.

Table 2-5 Data Mining SQL Functions

Function	Description
CLUSTER_ID	Returns the ID of the predicted cluster
CLUSTER_DETAILS	Returns detailed information about the predicted cluster
CLUSTER_DISTANCE	Returns the distance from the centroid of the predicted cluster
CLUSTER_PROBABILIT Y	Returns the probability of a case belonging to a given cluster
CLUSTER_SET	Returns a list of all possible clusters to which a given case belongs along with the associated probability of inclusion
FEATURE_ID	Returns the ID of the feature with the highest coefficient value
FEATURE_DETAILS	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
PREDICTION	Returns the best prediction for the target
PREDICTION_BOUNDS	(GLM only) Returns the upper and lower bounds of the interval wherein the predicted values (linear regression) or probabilities (logistic regression) lie.
PREDICTION_COST	Returns a measure of the cost of incorrect predictions
PREDICTION_DETAILS	Returns detailed information about the prediction
PREDICTION_PROBABI	Returns the probability of the prediction
PREDICTION_SET	Returns the results of a classification model, including the predictions and associated probabilities for each case

Example 2-7 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 2-7 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
    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
        7
               215
       12
               201
       17
                 123
                114
       16
       14
                 86
       19
                 64
       15
                 56
       18
                  36
```

#### See Also:

- Scoring and Deployment
- Oracle Database SQL Language Reference for details about the Data Mining SQL functions

# **Preparing the Data**

This chapter explains how to create a table or view that can be used to build a model. This chapter contains the following topics:

- Data Requirements
- About Attributes
- Using Nested Data
- Using Market Basket Data
- Handling Missing Values

# **Data Requirements**

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 Example 3-1 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. See "Using Market Basket Data".

### Example 3-1 Sample Case Table

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

### **Column Data Types**

The columns of the case table hold the attributes that describe each case. In Example 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
```

#### See Also:

- "Using Nested Data" for information about Oracle Data Mining nested types
- Mining Unstructured Text for information about data types that can be treated as unstructured text data
- Oracle Database SQL Language Reference for information about Oracle data types

# **Data Sets for Classification and Regression**

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.

# **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.

Automatic and embedded data transformations are discussed in Transforming the Data.

### **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. A table of employee data might contain attributes such as job title, date of hire, salary, age, gender, and so on.

### **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, might be an attribute used by an algorithm to build a model. Internally, the model attribute SIZE would 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, would not correspond to a model attribute. The data attribute would be SALES\_PROD, but each product with its corresponding sales figure (each row in the nested column) would be a model attribute.

Transformations also cause a discrepancy between data attributes and model attributes. For example, a transformation could apply a calculation to two data attributes and store the result in a new attribute. The new attribute would be 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.

#### See Also:

- "Using Nested Data"
- Transforming the Data for information about transformations

### **Target Attribute**

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, as shown in Example 2-2 Example 2-2.

### **Numericals, Categoricals, and Unstructured Text**

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. For information about mining unstructured data, see Mining Unstructured Text.

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.

# **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 should 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 may cause the model to ignore certain columns. Other columns may 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.

### **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 Example 3-2.

### 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
```

### **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.

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.

Example 3-3 shows the definition of the GET\_MODEL\_DETAILS function for an Attribute Importance model.

### Example 3-3 Model Details for an Attribute Importance Model

The syntax of the GET\_MODEL\_DETAILS function for Attribute Importance models is shown as follows.

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.

# **Using Nested Data**

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 might 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.

### See Also:

"Example: Creating a Nested Column for Market Basket Analysis".

# **Nested Object 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. You can learn more about collection types in *Oracle Database Object-Relational Developer's Guide*.

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 Example 3-4.

### Example 3-4 Oracle Data Mining Nested Data Types

describe dm_nested_binary_double Name	Null?	Туре
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) BINARY_DOUBLE
describe dm_nested_binary_doubles DM_NESTED_BINARY_DOUBLES TABLE OF SYS.D Name	M_NESTED_B Null?	INARY_DOUBLE Type
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) BINARY_DOUBLE
describe <b>dm_nested_binary_float</b> Name	Null?	Туре
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) BINARY_FLOAT
describe dm_nested_binary_floats DM_NESTED_BINARY_FLOATS TABLE OF SYS.DM		<del>-</del>
Name ATTRIBUTE_NAME VALUE	Null?	Type VARCHAR2(4000) BINARY_FLOAT
describe dm_nested_numerical Name	Null?	Туре
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) NUMBER
describe dm_nested_numericals DM_NESTED_NUMERICALS TABLE OF SYS.DM_NE Name	STED_NUMER	ICAL Type
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) NUMBER
describe <b>dm_nested_categorical</b> Name	Null?	Туре
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) VARCHAR2(4000)
describe dm_nested_categoricals DM_NESTED_CATEGORICALS TABLE OF SYS.DM_ Name	NESTED_CAT	EGORICAL Type
ATTRIBUTE_NAME VALUE		VARCHAR2(4000) VARCHAR2(4000)

# **Example: Transforming Transactional Data for Mining**

Example 3-5 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 3-6 shows how this data could be transformed for mining. The case ID column would be PRODUCT. SALES\_PER\_REGION, a nested column of type DM\_NESTED\_NUMERICALS, would be 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 3-7. (Note that the presentation in this example is conceptual only. The data is not actually pivoted before being processed.)

Example 3-5 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 3-6 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 3-7 Model Attributes Derived From SALES\_PER\_REGION

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

3-8 User's Guide

# **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 might 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 <code>ODMS\_ITEM\_ID\_COLUMN\_NAME</code> and <code>ODMS\_ITEM\_VALUE\_COLUMN\_NAME</code> 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

# **Example: Creating a Nested Column for Market Basket Analysis**

Association models can be built on native transactional data or on nested data. Example 3-8 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 guery returns two rows from the transformed data.

```
(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))
```

"Handling Missing Values".

## Example 3-8 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
```

# **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 may 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.

# **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.

## 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 would 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 would often be represented as nested data for mining.

Since missing values in a nested column always indicate sparsity, you should 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 users' movie ratings of a large movie database, the missing ratings would be unknown (missing at random), but Oracle Data Mining would treat the data as sparse and infer a rating of zero for the missing value.

## 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 may be 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.

## 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 Table 3-2.

#### Note:

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

Table 3-2 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 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	GLM, NMF, k-Means, and SVM replaces missing categorical values with the mode. 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.
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. DT, MDL, and NB and replace sparse	The algorithm handles sparse data.

Missing Data	EM, GLM, NMF, k-Means, SVD, SVM	DT, MDL, NB, OC	Apriori
		numerical data with zeros.	
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.

## **Changing the Missing Value Treatment**

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 would interpret 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. See *Oracle Database SQL Language Reference* for details about the NVL function.

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, you could 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.

# **Transforming the Data**

This chapter explains how to transform the data for building a model or for scoring. This chapter contains the following topics:

- About Transformations
- Preparing the Case Table
- Understanding Automatic Data Preparation
- Embedding Transformations in a Model
- Understanding Reverse Transformations

## **About Transformations**

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. (See "Scoring Requirements".)

# **Preparing the 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.

#### See Also:

Preparing the Data

## **Creating 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" for information about converting transactional data to nested columns

# **Converting Column Data Types**

You must convert the data type of a column if its type will cause 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 will be interpreted as a numerical 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')

## **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.

## See Also:

Oracle Data Mining User's Guide for details

## **About Business and Domain-Sensitive Transformations**

Some transformations are dictated by the definition of the business problem. For example, you might 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 might 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 might be converted to age.

Domain knowledge can be very important in deciding how to prepare the data. For example, some algorithms might 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. (See "Outlier Treatment".)

# **Understanding Automatic Data Preparation**

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.

## See Also:

"Automatic Data Preparation" in *Oracle Database PL/SQL Packages and Types Reference* 

# **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.

## **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).

## **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 may 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 may be perfectly valid. For example, in census data, the earnings for some of the richest individuals may vary significantly from the general population. This information should not be treated as an outlier, since it is an important part of the data. Domain knowledge is usually needed to determine outlier handling.

## **How ADP Transforms the Data**

Table 4-1 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.	
Expectatio n Maximizati on	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.	
SVD	Feature Extraction	Numerical attributes are normalized with outlier-sensitive normalization.	

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

- "Transformations in DBMS\_DATA\_MINING\_TRANSFORM" in *Oracle Database PL/SQL Packages and Types Reference*
- Part III of *Oracle Data Mining Concepts* for more information about algorithm-specific data preparation

# **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
settings_table_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,
settings_schema_name IN VARCHAR2 DEFAULT NULL,
settings_schema_name IN VARCHAR2 DEFAULT NULL,

xform_list IN TRANSFORM_LIST DEFAULT
NULL);
```

# **Specifying Transformation Instructions for an Attribute**

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 Table 4-2.

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

Field	Description
attribute_name and attribute_subnam e	These fields identify the attribute, as described in "Scoping of Model Attribute Name"
expression	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
	CASE WHEN age < 19 THEN 'child' ELSE 'adult'

Field	Description
	Expression and reverse expressions are stored in expression_rec objects. See "Expression Records" for details.
reverse_expressi on	A SQL expression for reversing the transformation. For example, this expression reverses the transformation of the age attribute:
	DECODE(age,'child','(-Inf,19)','[19,Inf)')
attribute_spec	<ul> <li>Specifies special treatment for the attribute. The attribute_spec field can be null or it can have one or more of these values:</li> <li>FORCE_IN — For GLM, forces the inclusion of the attribute in the model build when the ftr_selection_enable setting is enabled. (ftr_selection_enable 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.</li> <li>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.</li> <li>TEXT — Indicates that the attribute contains unstructured text. ADP has no effect on this setting. TEXT may optionally include subsettings POLICY_NAME, TOKEN_TYPE, and MAX_FEATURES.</li> <li>See Example 4-1 and Example 4-2.</li> </ul>

## **Expression Records**

The transformation expressions in a transformation record are expression\_rec objects.

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.

## **Attribute Specifications**

The attribute specification in a transformation record defines characteristics that are specific to this attribute. If not null, the attribute specification can include values <code>FORCE\_IN, NOPREP</code>, or <code>TEXT</code>, as described in Table 4-2.

#### 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 will not be 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)"
```

See "Configuring a Text Attribute" for more examples.

## **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\_TRANFORM 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

## **SET\_TRANSFORM**

The SET\_TRANSFORM procedure adds a single transformation record to a transformation list.

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.

#### 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 should 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.

## GET MODEL TRANSFORMATIONS and GET TRANSFORM 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.

GET\_MODEL\_TRANSFORMATIONS returns a table of dm\_transform objects. Each dm\_transform has these fields

The components of a transformation list are transform\_rec, not dm\_transform. The fields of a transform\_rec are described in Table 4-2. 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 in "Operational Notes" in *Oracle Database PL/SQL Packages and Types Reference*
- DBMS\_DATA\_MINING\_TRANSFORM.SET\_TRANSFORM in Oracle Database PL/SQL Packages and Types Reference
- DBMS\_DATA\_MINING.CREATE\_MODEL in Oracle Database PL/SQL Packages and Types Reference
- DBMS\_DATA\_MINING.GET\_MODEL\_TRANSFORMATIONS in Oracle Database PL/SQL Packages and Types Reference

# **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 will be embedded in the model. You will 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.

## **Oracle Data Mining Transformation Routines**

Oracle Data Mining provides routines that implement various transformation techniques in the DBMS\_DATA\_MINING\_TRANSFORM package. Details about the package are in *Oracle Database PL/SQL Packages and Types Reference*.

## **Binning Routines**

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 might be 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.

Table 4-3 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 should usually be used with outlier treatment. (See "Routines for Outlier Treatment".)
Quantile Binning	Quantile binning is a numerical binning technique. Quantiles are computed using the SQL analytic function NTILE. 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.

#### **Normalization Routines**

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 will be mapped to a tiny range, resulting in a significant loss of information. (See"Routines for Outlier Treatment".)

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, shift = 0, and scale = max{abs(max), abs(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.

#### **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.
Windsorizing	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.

# **Understanding 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 may 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 SVM and 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 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 might not resemble the columns of data used to train the model. However, attributes that undergo embedded transformations, whether initiated by 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.

#### See Also:

GET\_MODEL\_DETAILS, GET\_MODEL\_TRANSFORMATIONS, and ALTER\_REVERSE\_EXPRESSION in *Oracle Database PL/SQL Packages and Types Reference*.

# **Creating a Model**

This chapter explains how to create data mining models and query model details. This chapter contains the following topics:

- Before Creating a Model
- The CREATE\_MODEL Procedure
- Specifying Model Settings
- Viewing Model Details

# **Before Creating a Model**

As described in "About Mining Models", 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). See "DBMS\_DATA\_MINING" for an overview.

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 may specify model settings to influence the behavior of the model behavior. The preparation steps are summarized in Table 5-1.

Table 5-1 Preparation for Creating a Mining Model

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

# 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 may be created with configuration settings and user-specified transformations.

```
PROCEDURE CREATE MODEL(
                      model_name
                                                IN VARCHAR2,
                      mining_function IN VARCHAR2, wining_function IN VARCHAR2,
```

```
data_table_name IN VARCHAR2,
case_id_column_name IN VARCHAR2,
target_column_name IN VARCHAR2
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);
```

## **Choosing the Mining Function**

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 Table 5-2 for the mining\_function parameter to CREATE\_MODEL.

Table 5-2 Mining Model Functions

Mining_Function 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.
ATTRIBUTE_IMPORTANCE	Attribute Importance is a predictive mining function. An attribute importance model identifies the relative importance of attributes in predicting a given outcome.  Attribute Importance models use the Minimum Description Length algorithm.
CLASSIFICATION	Classification is a predictive mining function. A classification model uses historical data to predict a categorical target. Classification models can use Naive Bayes, Decision Tree, Logistic Regression, or Support Vector Machines. The default is Naive Bayes. 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).
CLUSTERING	Clustering is a descriptive mining function. A clustering model identifies natural groupings within a data set.

Mining_Function Value	Description
	Clustering models can use k-Means, O-Cluster, or Expectation Maximization. The default is $k$ -Means.
FEATURE_EXTRACTION	Feature Extraction is a descriptive mining function. A feature extraction model creates a set of optimized attributes. Feature extraction models can use Non-Negative Matrix Factorization or Singular Value Decomposition (which can also be used for Principal Component Analysis). The default is Non-Negative Matrix Factorization.
REGRESSION	Regression is a predictive mining function. A regression model uses historical data to predict a numerical target.  Regression models can use Support Vector Machines or Linear Regression. The default is Support Vector Machine.

Oracle Data Mining Concepts for an introduction to mining functions

# **Choosing the Algorithm**

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
ALGO_APRIORI_ASSOCIATION_ RULES	Apriori	_	association
ALGO_DECISION_TREE	Decision Tree	_	classification
ALGO_EXPECTATION_MAXIMIZA TION	Expectation Maximization		
ALGO_GENERALIZED_LINEAR_M ODEL	Generalized Linear Model	_	classification and regression
ALGO_KMEANS	k-Means	yes	clustering
ALGO_NAIVE_BAYES	Naive Bayes	yes	classification
ALGO_NONNEGATIVE_MATRIX_F ACTOR	Non-Negative Matrix Factorization	yes	feature extraction
ALGO_O_CLUSTER	O-Cluster	_	clustering
ALGO_SINGULAR_VALUE_DECOM P	Singular Value Decomposition (can also be used for Principal Component Analysis)	_	feature extraction

ALGO_NAME Value	Algorithm	Default?	Mining Model Function
ALGO_SUPPORT_VECTOR_MACHI NES	Support Vector Machine	yes	default regression algorithm regression, classification, and anomaly detection (classification with no target)

Oracle Data Mining Concepts for an introduction to the algorithms supported by Oracle Data Mining

## **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.

## **Creating a Transformation List**

There are two 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 could 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 <code>country\_id</code> is the target of a supervised model, the reverse transformation is also applied to the scored target.

#### Transformation List and Automatic Data Preparation

The transformation list argument to CREATE\_MODEL interacts with the PREP\_AUTO setting, which controls Automatic Data Preparation (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.

- "Embedding Transformations in a Model"
- "Operational Notes" for DBMS DATA MINING TRANSFORM in Oracle Database PL/SQL Packages and Types Reference

# **Specifying Model Settings**

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 Table 5-4 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 creates a settings table for an 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 documented in Oracle Database PL/SQL Packages and Types Reference and referenced in Table 5-5 and Table 5-6.

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 Oracle Database PL/SQL Packages and Types Reference
Global model characteristics	See "Global Settings" in Oracle Database PL/SQL Packages and Types Reference
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>

Algorithm	Description
Expectation Maximization	See "Algorithm Settings: Expectation Maximization" 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 $Oracle\ Database\ PL/SQL\ Packages\ and\ Types\ Reference$
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</i> PL/SQL Packages and Types Reference
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</i> PL/SQL Packages and Types Reference
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 symc_sh_sample_settings (
   setting_name VARCHAR2(30),
   setting_value VARCHAR2(4000));

BEGIN
   INSERT INTO symc_sh_sample_settings (setting_name, setting_value) VALUES
     (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);
   INSERT INTO symc_sh_sample_settings (setting_name, setting_value) VALUES
   (dbms_data_mining.syms_kernel_function, dbms_data_mining.syms_linear);
   COMMIT;
END;
//
```

# **Specifying Costs**

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 Table 5-7.

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

Note: Valid target data types are listed in Table 3-1.

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, 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.

#### See Also:

Oracle Data Mining Concepts for information about costs

## **Specifying Prior Probabilities**

The CLAS\_PRIORS\_TABLE\_NAME setting specifies the name of a table of prior probabilities to be used in building a Naive Bayes model. Prior probabilities can be used to offset differences in distribution between the build data and the actual population. The priors table must have the columns shown in Table 5-8.

Table 5-8 Priors Table Required Columns

Column Name	Data Type
target_value	valid target data type
prior_probability	NUMBER

#### See Also:

- Table 3-1
- Oracle Data Mining Concepts for information about priors

# **Specifying Class Weights**

The CLAS\_WEIGHTS\_TABLE\_NAME setting specifies the name of a table of class weights to be used to bias a logistic regression (GLM classification) or SVM classification model to favor higher weighted classes. The weights table must have the columns shown in Table 5-9.

Table 5-9 Class Weights Table Required Columns

Column Name	Data Type
target_value	valid target data type
class_weight	NUMBER

#### See Also:

- Table 3-1
- Oracle Data Mining Concepts for information about class weights

## **Model Settings in the Data Dictionary**

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 Table 5-10.

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_valu e	Value of the setting.
setting_type	INPUT if the value is specified by a user. ${\tt DEFAULT}$ if the value is systemgenerated.

The following query lists the settings for the 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

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

## See Also:

Oracle Database PL/SQL Packages and Types Reference for details about model settings

# **Viewing Model Details**

Model details describe model attributes, rules, statistics, and other information about the model. The DBMS\_DATA\_MINING package supports a separate GET\_MODEL\_DETAILS function for each algorithm. Global details are also available for Generalized Linear Models, Expectation Maximization, Singular Value Decompostion, and Association Rules.

Model details reverse the transformations applied to the attributes, thus enabling the information to be easily understood by a user. You can obtain the transformations embedded in the model by invoking the

DBMS DATA MINING.GET MODEL TRANSFORMATIONS function.

The query in Example 5-3 returns the coefficients for several attribute values in a GLM regression model called GLMR\_SH\_Regr\_sample. Additional details available for this algorithm include: standard error, test statistic, p value, standard coefficient, lower coefficient limit, and upper coefficient limit.

The query in Example 5-4 returns global details for the same model.

## Example 5-3 Model Details for GLM Regression

```
SELECT attribute_name, attribute_value, coefficient
    {\tt FROM\ TABLE(dbms\_data\_mining.get\_model\_details\_glm('GLMR\_SH\_Regr\_sample'))}
    WHERE attribute_name IN
('AFFINITY_CARD','BULK_PACK_DISKETTES','COUNTRY_NAME')
   ORDER BY class, attribute_name, attribute_value;
```

ATTRIBUTE_NAME	ATTRIBUTE_VALUE	COEFFICIENT
AFFINITY_CARD		58234968
BULK_PACK_DISKETTES		99684665
COUNTRY_NAME	Argentina	-1.2032688
COUNTRY_NAME	Australia	.000541598
COUNTRY_NAME	Brazil	5.29534224
COUNTRY_NAME	Canada	4.02414761
COUNTRY_NAME	China	.878394982
COUNTRY_NAME	Denmark	-2.9852215
COUNTRY_NAME	France	-1.0946872
COUNTRY_NAME	Germany	-1.6345684
COUNTRY_NAME	Italy	-1.2749328
COUNTRY_NAME	Japan	-6.259627
COUNTRY_NAME	New Zealand	5.07675762
COUNTRY_NAME	Poland	2.20458524
COUNTRY_NAME	Saudi Arabia	.443146197
COUNTRY_NAME	Singapore	-4.9472244
COUNTRY_NAME	South Africa	.493327068
COUNTRY_NAME	Spain	-3.0895076
COUNTRY_NAME	Turkey	-5.9014625
COUNTRY_NAME	United Kingdom	2.25154714

#### Example 5-4 Global Details for GLM Regression

SELECT \* FROM TABLE(dbms\_data\_mining.get\_model\_details\_global('GLMR\_SH\_Regr\_sample')) ORDER BY global\_detail\_name;

GLOBAL_DETAIL_NAME	GLOBAL_DETAIL_VALUE
ADJUSTED_R_SQUARE	.732
AIC	5943.057
COEFF_VAR	18.165
CORRECTED_TOTAL_DF	1499.000
CORRECTED TOT SS	278740.504

DEPENDENT_MEAN	38.892
ERROR_DF	1420.000
ERROR_MEAN_SQUARE	49.908
ERROR_SUM_SQUARES	70869.218
F_VALUE	52.291
GMSEP	52.722
HOCKING_SP	.035
J_P	52.570
MODEL_CONVERGED	1.000
MODEL_DF	79.000
MODEL_F_P_VALUE	.000
MODEL_MEAN_SQUARE	2609.739
MODEL_SUM_SQUARES	206169.407
NUM_PARAMS	80.000
NUM_ROWS	1500.000
ROOT_MEAN_SQ	7.065
R_SQ	.746
SBIC	6368.114
VALID_COVARIANCE_MATRIX	.000

# **Scoring and Deployment**

This chapter explains the scoring and deployment features of Oracle Data Mining. This chapter contains the following sections:

- About Scoring and Deployment
- Using the Data Mining SQL Functions
- **Prediction Details**
- **Real-Time Scoring**
- Dynamic Scoring
- Cost-Sensitive Decision Making
- DBMS DATA MINING.Apply

# **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 could 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 will be 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.

#### See Also:

- "Using Parallel Execution" in Oracle Database VLDB and Partitioning Guide
- "In-Database Scoring" in Oracle Data Mining Concepts
- "Exporting and Importing Mining Models"

# **Using the Data Mining SQL Functions**

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".

#### See Also:

- "Scoring Requirements"
- Table 2-5 for a list of the data mining functions
- Oracle Database SQL Language Reference for syntax of the data mining SQL functions

# **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 uses all the predictors.

The query in Example 6-2 uses only gender, marital status, occupation, and income as predictors.

The query in Example 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.

```
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,
                  'L: 300,000 and above' AS cust_income_level) = 1
  GROUP BY cust_gender
  ORDER BY cust_gender;
       CNT AVG_AGE
F
        30 38
        186
                   43
```

## 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;
       CNT AVG_AGE
- -----
       25 38
213 43
F
```

## 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;
  CNT AVG_AGE
C
       30 38
186 43
```

Example 6-3 Using Some Predictors and an Expression

# Single-Record Scoring

The data mining functions can produce a score for a single record, as shown in Example 6-4 and Example 6-5.

Example 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 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

## 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)
```

## **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.

## **Cluster Details**

For the most likely cluster assignments of customer 100955 (probability of assignment > 20%), the query in Example 6-6 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"</pre>
                 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>
```

## **Feature Details**

The query in Example 6-7 returns the three attributes that have the greatest impact on the top 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;
PROJ1DET
<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>
```

## **Prediction Details**

The query in Example 6-8 returns the attributes that are most important in predicting the age of customer 100010. The prediction functions apply a GLM 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"/>
              <a href="Attribute name="OS_DOC_SET_KANJI" actualValue="0" weight="0" rank="4"/>
              <Attribute name="BOOKKEEPING_APPLICATION" actualValue="1" weight="-.004"</pre>
rank="5"/>
              </Details>
```

The query in Example 6-9 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 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"</pre>
        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"</pre>
        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"</pre>
        <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"/>
 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 Example 6-10 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"/>
   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>
```

# **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 Example 6-11 uses a Decision Tree model named dt\_sh\_clas\_sample to predict the probability that customer 101488 will use an affinity card. A customer representative could retrieve this information in real time when talking to this customer on the phone. Based on the query result, the representative might offer an extra-value card, since there is a 73% chance that the customer will use a card.

## Example 6-11 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
```

# **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.

Example 6-12 shows a dynamic scoring query. The example identifies the rows in the input data that contain unusual customer age values.

#### Example 6-12 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"</pre>
                                  weight=".059" rank="1"/>
                                  <Attribute name="Y_BOX_GAMES" actualValue="0"</pre>
                                  weight=".059" rank="2"/>
                                  <Attribute name="AFFINITY_CARD" actualValue="0"</pre>
                                  weight=".059" rank="3"/>
                                  <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"</pre>
                                  weight=".059" rank="4"/>
                                  <Attribute name="YRS_RESIDENCE" actualValue="4"</pre>
                                  weight=".059" rank="5"/>
                                  </Details>
 101285 79 42.1753571
                         36.82 <Details algorithm="Support Vector Machines">
                                  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"</pre>
                                  weight=".059" rank="1"/>
                                  <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".059"</pre>
                                  rank="2"/>
                                  <Attribute name="CUST_MARITAL_STATUS" actualValue="Mabsent"</pre>
                                  weight=".059" rank="3"/>
                                  <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"</pre>
                                  rank="4"/>
                                  <Attribute name="OCCUPATION" actualValue="Prof." weight=".059"</pre>
                                  rank="5"/>
                                  </Details>
100694 77 41.0396722
                           35.96 <Details algorithm="Support Vector Machines">
                                  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"</pre>
                                  weight=".059" rank="1"/>
                                  <Attribute name="EDUCATION" actualValue="&lt; Bach."</pre>
                                  weight=".059" rank="2"/>
                                  <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"</pre>
                                  rank="3"/>
                                  <Attribute name="CUST_ID" actualValue="100694" weight=".059"</pre>
                                  rank="4"/>
                                  <Attribute name="COUNTRY_NAME" actualValue="United States of</pre>
                                  America" weight=".059" rank="5"/>
                                  </Details>
 100308 81 45.3252491
                           35.67 <Details algorithm="Support Vector Machines">
                                  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"</pre>
                                  weight=".059" rank="1"/>
                                  <Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"</pre>
                                  rank="2"/>
                                  <Attribute name="HOUSEHOLD_SIZE" actualValue="2" weight=".059"</pre>
                                  rank="3"/>
                                  <Attribute name="FLAT_PANEL_MONITOR" actualValue="1"</pre>
                                  weight=".059" rank="4"/>
                                  <Attribute name="CUST_GENDER" actualValue="F" weight=".059"</pre>
                                  rank="5"/>
                                  </Details>
101256 90 54.3862214
                           35.61 <Details algorithm="Support Vector Machines">
                                  <Attribute name="YRS_RESIDENCE" actualValue="9" weight=".059"</pre>
                                  rank="1"/>
                                  <Attribute name="HOME_THEATER_PACKAGE" actualValue="1"</pre>
                                  weight=".059" rank="2"/>
                                  <Attribute name="EDUCATION" actualValue="&lt; Bach."</pre>
                                  weight=".059" rank="3"/>
                                  <a href="Attribute name="Y_BOX_GAMES" actualValue="0" weight=".059"
                                  rank="4"/>
                                  <Attribute name="COUNTRY_NAME" actualValue="United States of</pre>
                                  America" weight=".059" rank="5"/>
                                  </Details>
```

# **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 will be 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 Table 6-1. The cost matrix specifies costs for a binary target. The matrix indicates that the algorithm should 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

#### See Also:

Example 1-1

#### Example 6-13 Sample Queries With Costs

The table nbmodel\_costs contains the cost matrix described in Table 6-1.

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.

```
dbms_data_mining.add_cost_matrix('nbmodel', 'nbmodel_costs');
```

The following query takes the cost matrix into account when scoring mining\_data\_apply\_v. The output will be restricted to those rows where a prediction of 1 is less costly then a prediction of 0.

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 ), "10" is specified in the inline costs.

The same query based on probability instead of costs is shown below.

# 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 Table 6-2.

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 will be used with APPLY must have a case ID.

A case ID is not required for models that will be 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.

Example 6-14 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).

### See Also:

DBMS\_DATA\_MINING. APPLY in Oracle Database PL/SQL Packages and Types Reference

#### Example 6-14 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
```

# **Mining Unstructured Text**

This chapter explains how to use Oracle Data Mining to mine unstructured text. This chapter includes the following topics:

- About Unstructured Text
- About Text Mining and Oracle Text
- Creating a Model that Includes Text Mining
- Creating a Text Policy
- Configuring a Text Attribute

## **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.

## **About 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 the user to Oracle Text and uses the results in the model creation process.

### See Also:

Oracle Text Application Developer's Guide

## Creating a Model that Includes Text Mining

Oracle Data Mining supports unstructured text within columns of VARCHAR2, CHAR, CLOB, BLOB, and BFILE, as described in Table 7-1.

Table 7-1 Column Data Types That May Contain Unstructured Text

Data Type	Description
BFILE and BLOB	Oracle Data Mining interprets BLOB and BFILE 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 CREATE_MODEL returns an error.
CLOB	Oracle Data Mining interprets CLOB as text.
CHAR	Oracle Data Mining interprets CHAR as categorical by default. You can identify columns of CHAR as text when you create the model.
VARCHAR 2	Oracle Data Mining interprets VARCHAR2 with data length > 4000 as text.  Oracle Data Mining interprets VARCHAR2 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 Table 7-2 control the term extraction process for text attributes in a model. Instructions for specifying model settings are in "Specifying Model Settings".

Table 7-2 Model Settings for Text

Setting Name	Data Type	Setting Value	Description
ODMS_TEXT_POLICY_N AME	VARCHAR2(4	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".
ODMS_TEXT_MAX_FEAT URES	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, as described in "Creating a Text Policy".
- 2. Specify the model configuration settings that are described in Table 7-2.

- Specify which columns should be treated as text and, optionally, provide text transformation instructions for individual attributes. See "Configuring a Text Attribute".
- Pass the model settings and text transformation instructions to DBMS\_DATA\_MINING.CREATE\_MODEL. See "Embedding Transformations in a 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).

## **Creating a Text Policy**

An Oracle Text policy specifies how text content should 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 Table 7-3.

Table 7-3 CTX DDL.CREATE POLICY Procedure Parameters

Parameter Name	Description
policy_name	Name of the new policy object. Oracle Text policies and text indexes share the same namespace.
filter	Specifies how the documents should be converted to plain text for indexing. Examples are: CHARSET_FILTER for character sets and NULL_FILTER for plain text, HTML and XML.  For filter values, see "Filter Types" in <i>Oracle Text Reference</i> .
section_grou p	Identifies sections within the documents. For example, HTML_SECTION_GROUP defines sections in HTML documents. For section_group values, see "Section Group Types" in <i>Oracle Text Reference</i> . Note: You can specify any section group that is supported by CONTEXT indexes.
lexer	Identifies the language that is being indexed. For example, BASIC_LEXER 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 lexer values, see "Lexer Types" in <i>Oracle Text Reference</i> .

Parameter Name	Description
stoplist	Specifies words and themes to exclude from term extraction. For example, the word "the" would typically be 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 should 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> .

### See Also:

CTX\_DDL.CREATE\_POLICY in Oracle Text Reference

## **Configuring a Text Attribute**

As shown in Table 7-1, 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" in Transforming the Data.

You can provide transformation instructions for any text attribute by qualifying the TEXT keyword in the attribute specification with the subsettings described in Table 7-4.

Table 7-4 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)
TOKEN_TYPE	The following values are supported:	(TOKEN_TYPE:THEME)
	NORMAL (the default)	
	STEM	
	THEME	

Subsetting Name	Description	Example
	See "Token Types in an Attribute Specification"	
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;
```

### See Also:

- "Specifying Transformation Instructions for an Attribute"
- DBMS\_DATA\_MINING.SET\_TRANSFORM in Oracle Database PL/SQL Packages and Types Reference

### Example 7-1 A Sample Attribute Specification for Text

This expression specifies that text transformation for the attribute should 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)"
```

# **Administrative Tasks for Oracle Data Mining**

This chapter explains how to perform administrative tasks related to Oracle Data Mining. The chapter contains the following topics:

- Installing and Configuring a Database for Data Mining
- Upgrading or Downgrading Oracle Data Mining
- **Exporting and Importing Mining Models**
- Controlling Access to Mining Models and Data
- Auditing and Adding Comments to Mining Models

## Installing and Configuring a Database for Data Mining

This section contains the following topics:

- About Installation
- Enabling or Disabling a Database Option
- Database Tuning Considerations for Data Mining

### 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:

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

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.

#### See Also:

**Installing and Upgrading** page of the Oracle Database online documentation library for your platform-specific installation instructions: http:// www.oracle.com/pls/db121/homepage

## **Enabling or Disabling a Database Option**

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 or Disabling Database Options" in the installation guide for your platform. For example:

- Oracle Database Installation Guide for Linux
- Oracle Database Installation Guide for Microsoft Windows

## **Database Tuning Considerations for Data Mining**

DBAs managing production databases that support Oracle Data Mining should 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. (Scoring operations are discussed in Scoring and Deployment of this manual.)

Database memory management can have a major impact on data mining. The correct sizing of PGA memory is very important for model building, complex queries, and batch scoring. From a data mining perspective, the SGA is generally less of a concern. However, the SGA should 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.

#### See Also:

- "Database Performance Fundamentals" and "Tuning Database Memory" in Oracle Database Performance Tuning Guide
- "Using Parallel Execution" in Oracle Database VLDB and Partitioning Guide

# **Upgrading or Downgrading Oracle Data Mining**

This section contains the following topics:

- Pre-Upgrade Steps
- Upgrading Oracle Data Mining
- Post Upgrade Steps
- Downgrading Oracle Data Mining

## 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 databases.

### **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.

### **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:

- Use the Data Miner Classic user interface to delete the mining activities.
- 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 should be no tables or views with the prefix DM4J\$ in any schema in the database after you complete these steps.

## **Upgrading 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.

#### See Also:

Oracle Database Upgrade Guide for complete database upgrade instructions

### 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.

### 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 10*g* is upgraded to 12*c*, 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 12*c*, you can no longer switch to the Data Mining Scoring Engine (DMSE). The Scoring Engine does not exist in Oracle Database 11*g* or 12*c*.

### Upgrading from Release 11g

If you upgrade Oracle Database 11*g* to Oracle Database 12*c*, and the database was previously upgraded from Oracle Database 10*g*, then theDMSYS schema may still be present. If the upgrade process detects DMSYS, it displays a warning message and drops DMSYS during the upgrade.

### **Using Export/Import to Upgrade Data Mining Models**

If you wish, 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 12*c*.

#### Caution:

Do not import data mining models that were created in Java. They are not supported in Oracle Database 12c.

### Export/Import Release 10g Data Mining Models

To export models from an instance of Oracle Database 10*g* 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 12*c*.

```
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 12*c* 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.

### 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".

#### 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 12*c* 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;
SOL>EXIT;
```

## Post Upgrade Steps

After upgrading the database, check the DBA\_MINING\_MODELS view in the upgraded database. The newly upgraded mining models should be listed in this view.

After you have verified the upgrade and confirmed that there will be no need to downgrade, you should set the initialization parameter COMPATIBLE to 12.1.

### Note:

The CREATE MINING MODEL privilege must be granted to Data Mining user accounts that will be used to create mining models. Refer to "Creating a Data Mining User" and "Controlling Access to Mining Models and Data" for more information.

## **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 will be aborted.

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;
```

## **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.

This section contains the following topics

- About Oracle Data Pump
- Options for Exporting and Importing Mining Models
- Directory Objects for EXPORT\_MODEL and IMPORT\_MODEL
- Using EXPORT\_MODEL and IMPORT\_MODEL
- Importing From PMML

## **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.

## **Options for Exporting and Importing Mining Models**

Options for exporting and importing mining models are described in Table 8-1.

Table 8-1 Export and Import Options for Oracle Data Mining
--

Task	Description
Export or import a full database	(DBA only) Use expdp to export a full database and impdp to import a full database. All mining models in the database are included.
Export or import a schema	Use expdp to export a schema and impdp to import a schema. All mining models in the schema are included.
Export or import individual models within a database	Use DBMS_DATA_MINING.EXPORT_MODEL to export individual models and DBMS_DATA_MINING.IMPORT_MODEL to import individual models. These procedures can export and import a single mining model, all mining models, or mining models that match specific criteria.  By default, IMPORT_MODEL imports models back into the schema from which they were exported. You can specify the schema_remap parameter to import models into a different schema. You can specify tablespace_remap with schema_remap to import models into a schema that uses a different tablespace.  You may need special privileges in the database to import models into a different schema. These privileges are granted by the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, which are only available to privileged users (such as SYS or a user with the DBA role). You do not need these roles to export or import models within your own schema.  To import models, you must have the same database privileges as the user who created the dump file set. Otherwise, a DBA with full system privileges must import the models.
Export or import individual models to or from a remote database	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 EXPORT_MODEL or IMPORT_MODEL.  To create a private database link, you must have the CREATE DATABASE LINK system privilege. To create a public database link, you must have the CREATE PUBLIC DATABASE LINK system privilege. Also, you must have the CREATE SESSION system privilege on the remote Oracle Database. Oracle Net must be installed on both the local and remote Oracle Databases.

### See Also:

- DBMS\_DATA\_MINING.IMPORT\_MODEL in Oracle Database PL/SQL Packages and Types Reference
- DMBS\_DATA\_MINING.EXPORT\_MODEL in Oracle Database PL/SQL Packages and Types Reference
- CREATE DATABASE LINK in Oracle Database SQL Language Reference

## Directory Objects for EXPORT\_MODEL and IMPORT\_MODEL

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;

#### See Also:

CREATE DIRECTORY in Oracle Database SQL Language Reference

## Using EXPORT\_MODEL and IMPORT\_MODEL

The examples in this section illustrate various export and import scenarios with EXPORT\_MODEL and IMPORT\_MODEL. The examples use the directory object dmdir shown in Example 8-1 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_MAXIMIZADM1 DT_SH_CLAS_SAMPLE CLASSIFICATION DECISION_TREEDM2 SVD_SH_SAMPLE FEATURE_EXTRACTION SINGULAR_VALUE_DECOM	

### See Also:

Oracle Database PL/SQL Packages and Types Reference for more examples

### Example 8-1 Creating the Directory Object

### Example 8-2 Exporting All Models From DM1

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 dml\_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 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 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 dm1_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 dm1_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 dm1_exp_513.log are created in /scratch/dmuser/expimp
```

## **Importing From 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 Predictive Model Markup Language (PMML). The models must be of type RegressionModel, either linear regression or binary logistic regression.

#### See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about PMML import

## **Controlling Access to Mining Models and Data**

This section contains the following topics:

- Creating a Data Mining User
- System Privileges for Data Mining
- Object Privileges for Mining Models

## **Creating a Data Mining User**

A Data Mining user is a database user account that has privileges for performing data mining activities. Example 8-6 shows how to create a database user. Example 8-7 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.

### See Also:

*Oracle Database SQL Language Reference* for the complete syntax of the CREATE USER statement

### 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
       OUOTA 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 would be 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 should be assigned according to standard DBA practices, depending on system load and system resources.

To login as dmuser, type the following.

```
CONNECT dmuser
Enter password: password
```

### **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 Example 8-7 assign a set of basic data mining privileges to the dmuser account. Some of these privileges may not be 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;
```

## **System Privileges for Data Mining**

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 Table 8-2 control operations on mining models.

Table 8-2 System Privileges for Data Mining

System Privilege	Allows you to
CREATE MINING MODEL	Create mining models in your own schema.
CREATE ANY MINING MODEL	Create mining models in any schema.
ALTER ANY MINING MODEL	Change the name or cost matrix of any mining model in any schema.
DROP ANY MINING MODEL	Drop any mining model in any schema.
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. (See "Adding a Comment to a Mining Model".)
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;

## **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 Table 8-3 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.

#### Grant Object Privileges on Mining Models Example 8-9

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;

## Auditing and Adding Comments to Mining Models

Mining model objects support SQL COMMENT and AUDIT statements.

## 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.

See Table 8-2.

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; 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:

Oracle Database SQL Language Reference for details about SQL COMMENT statements

## **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

# The Data Mining Sample Programs

This appendix describes the data mining sample programs that ship with Oracle Database. This appendix contains these topics:

- About the Data Mining Sample Programs
- Installing the Data Mining Sample Programs
- The Data Mining Sample Data

## **About the Data Mining Sample Programs**

MINING\_FUNCTION

You can learn a great deal about the Oracle Data Mining 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. Example A-1 shows this directory listing on a Linux system.

Note that the directory listing in Example A-1 includes one file, dmhpdemo.sql, that is *not* a data mining program.

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 Example A-2.

### 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 dmsvrdem.sql dmsvrdem.sql
\begin{array}{lll} {\tt dmdtdemo.sq1} & {\tt dmnmdemo.sq1} & {\tt dmsvrdem.sq1} \\ {\tt dmdtxvlddemo.sq1} & {\tt dmocdemo.sq1} & {\tt dmtxtnmf.sq1} \end{array}
dmemdemo.sql dmsh.sql
                                            dmtxtsvm.sql
                     dmshgrants.sql
dmglcdem.sql
dmglrdem.sql
                     dmstardemo.sql
dmhpdemo.sql
                       dmsvcdem.sql
```

#### Example A-2 Models Created by the Sample Programs

```
SELECT mining_function, algorithm, model_name FROM user_mining_models
   ORDER BY mining_function;
                              ALGORITHM
```

MODEL NAME

APRIORI\_ASSOCIATION\_RULES AR\_SH\_SAMPLE
GENERALIZED\_LINEAR\_MODEL ASSOCIATION\_RULES CLASSIFICATION GLMC\_SH\_CLAS\_SAMPLE CLASSIFICATION SUPPORT\_VECTOR\_MACHINES T\_SVM\_CLAS\_SAMPLE SUPPORT\_VECTOR\_MACHINES CLASSIFICATION SVMC\_SH\_CLAS\_SAMPLE CLASSIFICATION SUPPORT\_VECTOR\_MACHINES SVMO SH CLAS SAMPLE CLASSIFICATION NAIVE BAYES NB\_SH\_CLAS\_SAMPLE DECISION\_TREE DT\_SH\_CLAS\_SAMPLE
EXPECTATION\_MAXIMIZATION EM\_SH\_CLUS\_SAMPLE
O CLUSTEP CLASSIFICATION DECISION\_TREE CLUSTERING CLUSTERING O CLUSTER OC\_SH\_CLUS\_SAMPLE CLUSTERING KMEANS KM\_SH\_CLUS\_SAMPLE KMEANS DM\_STAR\_CLUSTER CLUSTERING SINGULAR\_VALUE\_DECOMP FEATURE\_EXTRACTION SVD\_SH\_SAMPLE NONNEGATIVE\_MATRIX\_FACTOR NMF\_SH\_SAMPLE
NONNEGATIVE\_MATRIX\_FACTOR T\_NMF\_SAMPLE FEATURE\_EXTRACTION FEATURE\_EXTRACTION REGRESSION SUPPORT\_VECTOR\_MACHINES SVMR\_SH\_REGR\_SAMPLE REGRESSION GENERALIZED\_LINEAR\_MODEL

\_\_\_\_\_

## **Installing the Data Mining Sample Programs**

GLMR\_SH\_REGR\_SAMPLE

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 dmshgrants.sql by a system administrator
- Execution of dmsh.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/db121/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 Table A-1.

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
CREATE SESSION	log in to a database session
CREATE TABLE	create tables, such as the settings tables for CREATE_MODEL
CREATE VIEW	create views, such as the views of tables in the SH schema
CREATE MINING MODEL	create data mining models
EXECUTE ON ctxsys.ctx_ddl	execute procedures in the ctxsys.ctx_ddl 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
```

## 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
```

The dmshgrants script grants SELECT access to the tables in SH. The dmsh.sql script creates views of the SH tables in the schema of the data mining user. The views are described in Table A-2.

Table A-2 The Data Mining Sample Data

View Name	Description
MINING_DATA	Joins and filters data
MINING_DATA_BUILD_V	Data for building models
MINING_DATA_TEST_V	Data for testing models
MINING_DATA_APPLY_V	Data to be scored
MINING_BUILD_TEXT	Data for building models that include text
MINING_TEST_TEXT	Data for testing models that include text

View Name	Description
MINING_APPLY_TEXT	Data, including text columns, to be scored
MINING_DATA_ONE_CLAS S_V	Data for anomaly detection

The association rules program creates its own transactional data.

## Index

#### C **Symbols** {{Emphasis Role='Italic'}}k{{/Emphasis}}-Means, 5-3 case ID, 3-1, 3-2, 3-5, 6-11 case table, 3-1, 4-2 categorical attributes, 7-1 Α chopt utility, 8-2 ADP, 5-4 class weights, 5-7 Advanced Analytics option, 8-1, A-2 classification, 2-1, 3-2, 3-4, 5-2, 5-3 algorithms clipping, 4-4 parallel execution, 8-2 CLUSTER\_DETAILS, 1-5, 2-7 ALL\_MINING\_MODEL\_ATTRIBUTES, 2-2 CLUSTER DISTANCE, 2-7 ALL\_MINING\_MODEL\_SETTINGS, 2-2, 5-8 CLUSTER\_ID, 1-5, 2-7, 2-8 ALL\_MINING\_MODELS, 2-2 CLUSTER\_PROBABILITY, 2-7 anomaly detection, 2-1, 3-2, 5-2, 5-4, 6-11 CLUSTER\_SET, 1-5, 2-7 APPLY, 6-1 clustering, 1-5, 2-1, 3-2, 5-3 Apriori, 3-9, 4-4, 5-3 COMMENT, 8-12 association rules, 5-2, 5-3 cost matrix, 2-5, 5-6, 6-9, 8-12 attribute importance, 2-1, 3-5, 5-2, 5-3 cost-sensitive prediction, 6-9 attribute specification, 4-6, 7-4, 7-5 attributes D categorical, 3-4, 7-1 data attributes, 3-3 data data dictionary, 2-2 categorical, 3-4 model attributes, 3-3, 3-4 dimensioned, 3-7 nested, 3-2 for sample programs, A-3 numerical, 3-4, 7-1 market basket, 3-9 subname, 3-5 missing values, 3-10 target, 3-4 multi-record case, 3-7 text, 3-4 nested, 3-2unstructured text, 7-1 numerical, 3-4 AUDIT, 8-12, 8-14 preparation, 4-1 Automatic Data Preparation, 1-1, 3-3, 4-3 READ access, 8-11 SELECT access, 8-11 single-record case, 3-1 В sparse, 3-10 binning transactional, 3-9 equi-width, 4-9 unstructured text, 3-4 quantile, 4-9 data mining supervised, 4-3, 4-9 applications of, 1-1 top-n frequency, 4-9 database tuning for, 8-2

DDL for, 2-5

privileges for, 8-1, 8-10, A-2

build data, 3-2

data mining (continued)	M
sample programs, A-1	market basket data, 3-9
scoring, 5-2, 6-1	MDL, 4-4
data types	memory, 8-2
nested, 3-6	Minimum Description Length, 4-4, 5-3
Database Upgrade Assistant, 8-4	mining functions
DBMS_DATA_MINING, 2-4, 5-2	supervised, 5-2
DBMS_DATA_MINING_TRANSFORM, 2-4, 2-5 DBMS_DBTDICTIVE_ANALYTICS_1_4_2_4_2_6	unsupervised, 5-2
DBMS_PREDICTIVE_ANALYTICS, 1-4, 2-4, 2-6	mining models
Decision Tree, 4-4, 5-2, 5-3, 6-7	adding a comment, 2-1, 8-13
desupported features Java API, 8-3	applying, 8-12
·	auditing, 2-1, 8-14
directory objects, <i>8</i> -7 DMEIDMSYS, <i>8</i> -4	changing the name, 8-12
	created by sample programs, A-1
downgrading, 8-6	data dictionary, 2-2
	object privileges, 8-12, 8-13
E	privileges for, 2-1
Encoded and Maria and A	SQL DDL, 2-5
Expectation Maximization, 4-4	
EXPLAIN, 2-7	upgrading, <i>8-3</i> viewing model details, <i>8-12</i>
exporting, 8-4, 8-6	missing value treatment, 3-11
	model attributes
F	categorical, 3-4
	derived from nested column, 3-5
feature extraction, 2-1, 3-2, 5-3	numerical, 3-4
FEATURE_DETAILS, 2-7	scoping of name, 3-5
FEATURE_ID, 2-7	text, 3-4
FEATURE_SET, 2-7	
FEATURE_VALUE, 2-7	model details, 2-5, 3-5
	model signature, 3-4
G	models algorithms, 5-3
	created by sample programs, A-1
Generalized Linear Models, 4-4	deploying, 6-1
GLM, 5-3	·
graphical user interface, 1-1	privileges for, 8-11
	settings, 2-2, 5-8
l	testing, 3-2
	training, 3-2
importing, 8-4, 8-6	transparency, 1-1
installation 0.1.4.2	
Oracle Database, 8-1, A-2	N
Oracle Database Examples, A-2	Naine Barrer 4 4 5 2 5 2
sample data mining programs, A-2	Naive Bayes, 4-4, 5-2, 5-3
sample schemas, $A$ -2	nested data, 3-6, 7-2
	Non-Negative Matrix Factorization, 4-4, 5-3
K	normalization
	min-max, 4-10
k-Means, 4-4	scale, 4-10
	z-score, 4-10
L	numerical attributes, 7-1
linear regression, 2-7, 5-3	0
logistic regression, 2-7, 5-2	0.01 + 2.6.4.4.5.2
	O-Cluster, 3-6, 4-4, 5-3
	object privileges, 8-12, 8-13
	One-Class SVM, 5-2

Oracle Data Miner, 1-1, 8-3	settings
Oracle Data Miner Classic, 8-3	data dictionary, 2-2
Oracle Data Pump, 8-6	table for specifying, 5-1
Oracle Text, 7-1	SGA, 8-2
outliers, 4-4, 4-10	Singular Value Decomposition, 4-4
	sparse data, 3-10
P	SQL AUDIT, 2-1, 8-14
	SQL COMMENT, 2-1, 8-13
parallel execution, 6-2, 8-2	SQL data mining functions, 2-7
PGA, 8-2	SQL Developer, 1-1
PL/SQL packages, 2-4	STACK, 2-6, 4-7
PMML, 8-10	Support Vector Machine, 4-5, 5-2, 5-4
PREDICTION, 1-2, 1-3, 2-7, 6-7	system privileges, 8-11, A-2
PREDICTION_BOUNDS, 2-7	, .
PREDICTION_COST, 2-7	Т
PREDICTION_DETAILS, 2-7, 6-7	<u> </u>
PREDICTION_PROBABILITY, 1-3, 2-7, 6-7	target, 3-4, 3-5, 7-2
PREDICTION_SET, 2-7	test data, 3-2, 5-1
predictive analytics, 1-1, 1-4, 2-1	text attributes, 7-2, 7-4
prior probabilities, 5-7	text mining, 2-6, 7-1
priors table, 5-7	text policy, 7-3
privileges	text terms, 7-1
for creating mining models, 8-5	training data, 5-1
for data mining, 8-1, 8-7	transactional data, 3-1, 3-7, 3-9
for data mining sample programs, $A-3$	transformations
for exporting and importing, 8-7	attribute-specific, 2-5, 2-6
required for data mining, 8-11	embedded, 2-5, 2-6, 3-3, 4-1
	reverse, 2-5
R	user-specified, 3-3
	_ transparency, 3-5
regression, 2-1, 3-2, 3-4, 5-3, 5-4	trimming, 4-10
reverse transformations, 2-5, 3-5	
	U
S	
	upgrading
sample programs	exporting and importing, 8-4
configuration scripts, 8-10	from Release 10g, 8-4
data used by, $A-3$	from Release 11g, 8-4
directory listing of, <i>A-1</i>	pre-upgrade steps, 8-3
installing, $A-2$	using Database Upgrade Assistant, 8-4
models created by, <i>A-1</i>	users
Oracle Database Examples, $A-2$	assigning data mining privileges to, 8-11
requirements, $A-2$	creating, 8-10
sample schemas, A-2	privileges for data mining, 8-5, 8-10
scoring	
data, 3-2	W
dynamic, 1-3, 2-1, 6-7	
parallel execution, 6-2	weights, 5-7
privileges for, 8-12	windsorize, 4-10
requirements, 3-2	
SQL functions, 2-7	X
transparency, 1-1	
Scoring Engine, 8-4	XFORM, 2-6