Oracle Data Miner (Extension of SQL Developer 4.1)

Use Repository APIs to Manage and Schedule Workflows to run

ORACLE WHITE PAPER | MARCH 2015

Table of Contents

Introduction

Data analysts use the Data Miner workflows to define, build and test their analytic methodologies. When they are satisfied with the results, they can use the script generation feature to hand off a set of SQL scripts to the application developer. The application developer will be able to take these standard Oracle SQL scripts and integrate them easily into their applications.

Data Miner 4.1 ships with a set of repository PL/SQL APIs that allow applications to manage Data Miner projects and workflows directly. The workflow APIs enable applications to execute workflows immediately or schedule workflows to execute using specific time intervals or using defined schedules. The workflow run APIs internally use Oracle Scheduler for scheduling functionality. Moreover, repository views are provided for applications to query project and workflow information. Applications can also monitor workflow execution status and query generated results using these views.

This white paper will focus on how to use the workflow run API to schedule a workflow to run, and use the repository views to monitor the workflow run status and query the generated results. A complete set of project and workflow APIs are described in the Appendix section for your reference.

Requirement

The Data Miner 4.1 or above repository installation is required. User can install the repository via the Data Miner client or running the server scripts located in the Data Miner installation (please refer to the install_scripts_readme.html in the \<sql developer>\dataminer\scripts directory for repository installation details). The repository views are supported only in database 11.2.0.4 and above. To use the repository APIs and views, a user account needs to be granted specific rights. The Data Miner client automatically grants the required rights to a user account upon opening a connection if it detects the rights are missing. Alternatively, administrator can run the user grant script (please refer to the install_scripts_readme.html for “Granting a User the Rights to Access the Repository”) to manually grant the required rights.

PL/SQL APIs

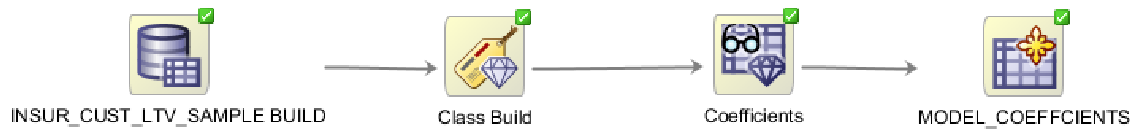
The project PL/SQL APIs can be found in the ODMR_PROJECT package and the workflow PL/SQL APIs can be found in the ODMR_WORKFLOW package; both packages are defined in the ODMRSYS schema (Data Miner repository).

Demo Workflows

This paper will use a build and apply workflows (available in the companion .zip download for this white paper) to demonstrate the use of workflow run API. First, the demo shows how to schedule the build workflow to run on the last day of each month, and then it shows how to schedule the apply workflow to run daily.

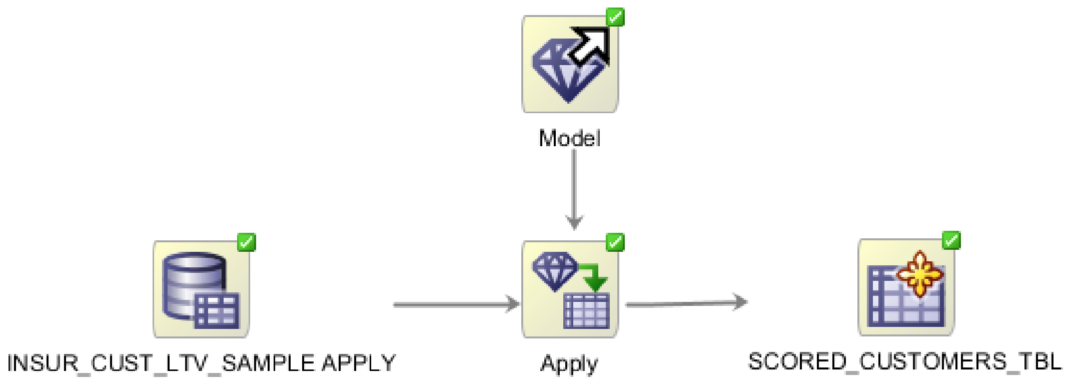
Build Workflow

This workflow builds a SVM classification model, and then stores the model details (coefficients) to a table.



Apply Workflow

This workflow uses the Model node to reference the model built by the above build workflow, and then uses it for scoring.



Workflow Run API

The run APIs have signatures that accept names or ids of a project, a workflow, and specific nodes to run. Moreover, the APIs can accept a start time, an end time, and a run interval or a named schedule, so that you can schedule a workflow to run at specific time and interval or at a defined schedule (schedule can be time based or event based). You can get more details about the run APIs in the appendix.

The demo will use the following version of the WF_RUN API to schedule a workflow to run.

```
FUNCTION WF_RUN(P_PROJECT_NAME          IN VARCHAR2,
                P_WORKFLOW_NAME         IN VARCHAR2,
                P_NODE_NAMES            IN ODMR_OBJECT_NAMES,
                P_RUN_MODE              IN VARCHAR2 DEFAULT 'RUN_NODE_ONLY',
                P_MAX_NUM_THREADS       IN NUMBER DEFAULT NULL,
                P_START_DATE            IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
                P_REPEAT_INTERVAL       IN VARCHAR2 DEFAULT NULL,
                P_END_DATE            IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
                P_JOB_CLASS           IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2;
```

Parameter	Description	
P_PROJECT_NAME	Specify the project that the workflow was created in.	
P_WORKFLOW_NAME	Specify the workflow to run.	
P_NODE_NAMES	Specify the nodes of the workflow to run. The run mode determines how the workflow is run.	
P_RUN_MODE	VALIDATE_ONLY	validate parent nodes of the specified nodes
	RUN_NODE_ONLY	run the specified nodes. If nodes have already been run, they will not be run again (run once). If parent nodes have not been run, they will be run, otherwise they will be ignored
	RERUN_NODE_ONLY	reset status of the specified nodes to READY state, then run these nodes again
	RERUN_NODE_CHILDREN	reset status of the specified nodes and their children nodes to READY state, and then run these nodes again
	RERUN_NODE_PARENTS	reset status of the specified nodes and their parent nodes to READY state, and then run these nodes again
	RERUN_CHILDREN_ONLY	reset status of children nodes of the specified nodes to READY state, and then run these nodes again. Note: the specified nodes will not be run
	RERUN_WORKFLOW	reset status of all nodes to READY state, and then run them all (complete workflow run). Note: p_node_names is ignored
P_MAX_NUM_THREADS	Specify maximum number of parallel model builds across all workflows. Specify NULL for system determined. You may experience with this parameter only if your system has plenty of resources, otherwise set this value to NULL to use the default value.	
P_START_DATE	Specify the first date and time on which this workflow is scheduled to start. If P_START_DATE and P_REPEAT_INTERVAL are left null, then the workflow is scheduled to run	

Parameter	Description
	as soon as possible.
P_REPEAT_INTERVAL	Specify how often the workflow repeats. You can specify the repeat interval by using calendaring or PL/SQL expressions. The expression specified is evaluated to determine the next time the workflow should run. If P_REPEAT_INTERVAL is not specified, the workflow runs only once at the specified start date. See "Calendaring Syntax" for further information.
P_END_DATE	Specify the date and time after which the workflow expires and is no longer run. If no value for P_END_DATE is specified, the job repeats forever.
P_JOB_CLASS	Specify existing job class used to run the workflow. If no value for P_JOB_CLASS is specified, default job class is used.

Schedule Build Workflow to Run

To run the lineage, you can specify all nodes in the lineage and use the RERUN_NODE_ONLY run mode. Or you can specify the MODEL_COEFFCIENTS node and use the RERUN_NODE_PARENTS run mode. Both options accomplish the same result of running all four nodes in the lineage. The demo uses the latter approach in this example. Moreover, it shows how to schedule the workflow to run monthly (MONTHLY) on the last day of the month (BYMONTHDAY=-1) starting at mid night from 12/31/2014 to 12/31/2015 in EST zone. After the WF_RUN call is executed, it polls the workflow running status from the ODMR_USER_PROJECT_WORKFLOW view to determine whether the workflow run completes (i.e. status is neither SCHEDULED nor ACTIVE). At last, it prints out any node failure from the event log along with error messages.

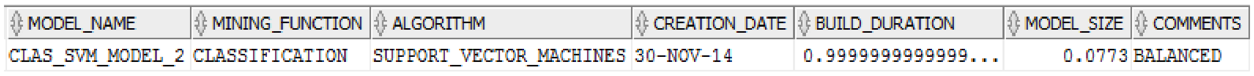
```
CONNECT DMUSER/DMUSER
SET SERVEROUTPUT ON
DECLARE
  v_jobId             VARCHAR2(30) := NULL;
  v_status            VARCHAR2(30) := NULL;
  v_projectName       VARCHAR2(30) := 'Project';
  v_workflow_name     VARCHAR2(30) := 'build_workflow';
  v_node              VARCHAR2(30) := 'MODEL_COEFFCIENTS';
  v_run_mode          VARCHAR2(30) := ODMRSYS.ODMR_WORKFLOW.RERUN_NODE_PARENTS;
  v_failure           NUMBER := 0;
  v_nodes             ODMRSYS.ODMR_OBJECT_NAMES := ODMRSYS.ODMR_OBJECT_NAMES();
BEGIN
  v_nodes.extend();
  v_nodes(v_nodes.count) := v_node;
  v_jobId := ODMRSYS.ODMR_WORKFLOW.WF_RUN(p_project_name => v_projectName,
                              p_workflow_name => v_workflow_name,
                              p_node_names => v_nodes,
                              p_run_mode => v_run_mode,
                              p_start_date => '31-DEC-14 12.00.00 AM AMERICA/NEW_YORK',
                              p_repeat_interval => 'FREQ=MONTHLY;BYMONTHDAY=-1',
                              p_end_date => '31-DEC-15 12.00.00 AM AMERICA/NEW_YORK');
  DBMS_OUTPUT.PUT_LINE('Job: '||v_jobId);
  -- wait for workflow to run to completion
  LOOP
    SELECT STATUS INTO v_status FROM ODMR_USER_PROJECT_WORKFLOW
    WHERE WORKFLOW_NAME = v_workflow_name;
    IF (v_status IN ('SCHEDULED', 'ACTIVE')) THEN
      DBMS_LOCK.SLEEP(10); -- wait for 10 secs
    ELSE
      EXIT; -- workflow run completes
    END IF;
  END LOOP;
  -- print out all failed nodes from the event log
  FOR wf_log IN (
    SELECT node_id, node_name, subnode_id, subnode_name, log_message, log_message_details
    FROM ODMR_USER_WORKFLOW_LOG
    WHERE job_name=v_jobId and log_type='ERR' and log_message IS NOT NULL)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Node Id: '||wf_log.node_id||', '||'Node Name: '||wf_log.node_name);
    IF (wf_log.subnode_id IS NOT NULL) THEN
      DBMS_OUTPUT.PUT_LINE(
                    'Subnode Id: '||wf_log.subnode_id||', '||'Subnode Name: '||wf_log.subnode_name);
    END IF;
    DBMS_OUTPUT.PUT_LINE('Message: '||wf_log.log_message);
    v_failure := v_failure + 1;
  END LOOP;
  IF (v_failure = 0) THEN
    DBMS_OUTPUT.PUT_LINE('Workflow Status: SUCCEEDED');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Workflow Status: FAILURE');
  END IF;
EXCEPTION WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Error: '||SUBSTR(DBMS_UTILITY.FORMAT_ERROR_STACK(), 1, 4000));
END;
```

Query Named Result Objects

After the workflow run completes successfully, you can query all named objects (e.g. table/view in the Create Table node, models in the build nodes, etc) generated by the workflow.

The following query returns the model CLAS_SVM_MODEL_2 information.

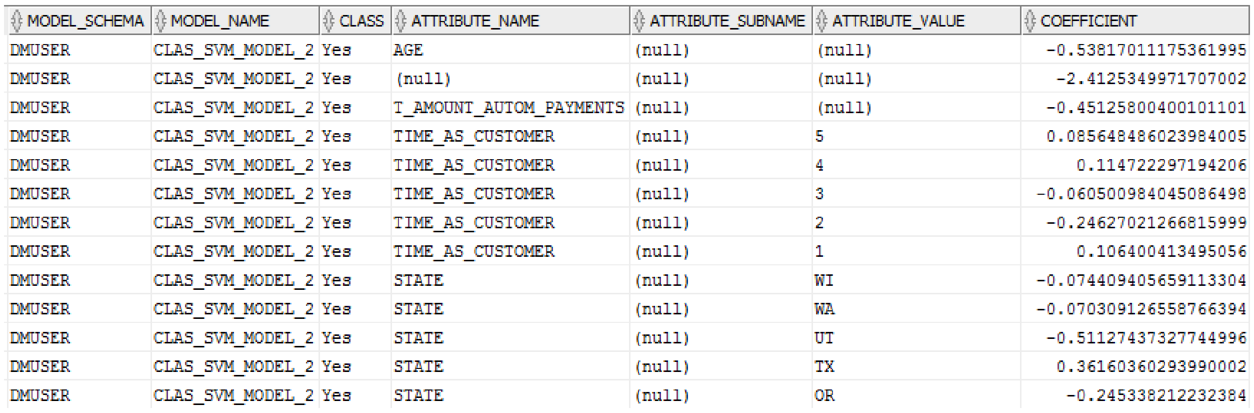
```
SELECT
  *
FROM
  USER_MINING_MODELS
WHERE MODEL_NAME = 'CLAS_SVM_MODEL_2'
```



MODEL_NAME	MINING_FUNCTION	ALGORITHM	CREATION_DATE	BUILD_DURATION	MODEL_SIZE	COMMENTS
CLAS_SVM_MODEL_2	CLASSIFICATION	SUPPORT_VECTOR_MACHINES	30-NOV-14	0.9999999999999...	0.0773	BALANCED

The following query returns the data of the MODEL_COEFFCIENTS table.

```
SELECT
  *
FROM
  MODEL_COEFFCIENTS
```



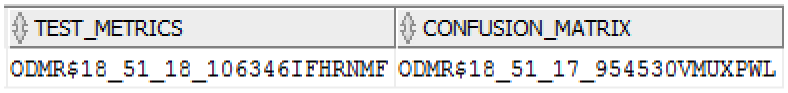
MODEL_SCHEMA	MODEL_NAME	CLASS	ATTRIBUTE_NAME	ATTRIBUTE_SUBNAME	ATTRIBUTE_VALUE	COEFFICIENT
DMUSER	CLAS_SVM_MODEL_2	Yes	AGE	(null)	(null)	-0.53817011175361995
DMUSER	CLAS_SVM_MODEL_2	Yes	(null)	(null)	(null)	-2.4125349971707002
DMUSER	CLAS_SVM_MODEL_2	Yes	T_AMOUNT_AUTOM_PAYMENTS	(null)	(null)	-0.45125800400101101
DMUSER	CLAS_SVM_MODEL_2	Yes	TIME_AS_CUSTOMER	(null)	5	0.085648486023984005
DMUSER	CLAS_SVM_MODEL_2	Yes	TIME_AS_CUSTOMER	(null)	4	0.114722297194206
DMUSER	CLAS_SVM_MODEL_2	Yes	TIME_AS_CUSTOMER	(null)	3	-0.060500984045086498
DMUSER	CLAS_SVM_MODEL_2	Yes	TIME_AS_CUSTOMER	(null)	2	-0.24627021266815999
DMUSER	CLAS_SVM_MODEL_2	Yes	TIME_AS_CUSTOMER	(null)	1	0.106400413495056
DMUSER	CLAS_SVM_MODEL_2	Yes	STATE	(null)	WI	-0.074409405659113304
DMUSER	CLAS_SVM_MODEL_2	Yes	STATE	(null)	WA	-0.070309126558766394
DMUSER	CLAS_SVM_MODEL_2	Yes	STATE	(null)	UT	-0.51127437327744996
DMUSER	CLAS_SVM_MODEL_2	Yes	STATE	(null)	TX	0.36160360293990002
DMUSER	CLAS_SVM_MODEL_2	Yes	STATE	(null)	OR	-0.245338212232384

Query Test Results

You can query the test results from the ODMR_USER_WF_CLAS_TEST_RESULTS (Classification) or ODMR_USER_WF_REGR_TEST_RESULTS (Regression) view.

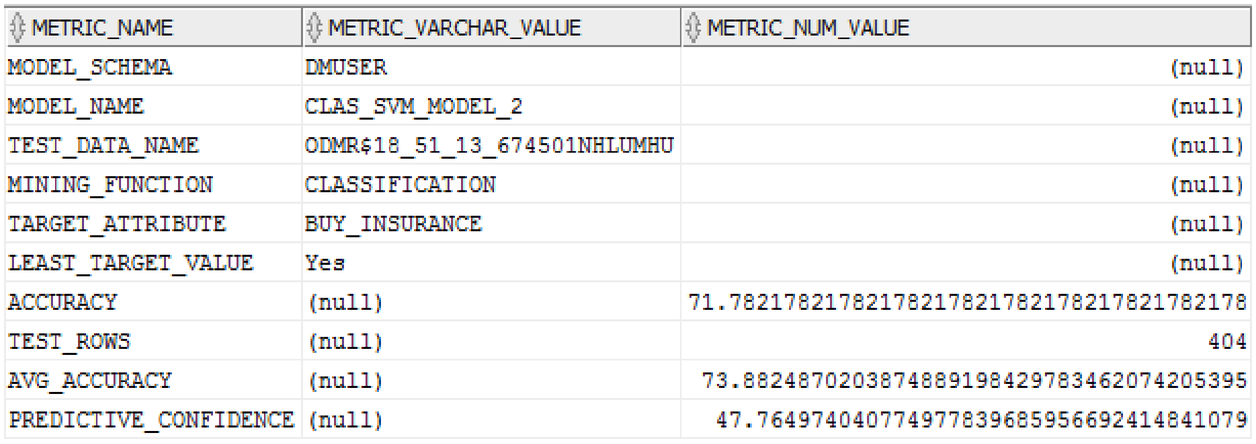
The following query returns the test metrics and confusion matrix results.

```
SELECT
  TEST_METRICS, CONFUSION_MATRIX
FROM
  ODMR_USER_WF_CLAS_TEST_RESULTS
WHERE
  WORKFLOW_NAME = 'build_workflow' AND NODE_NAME = 'Class Build'
```



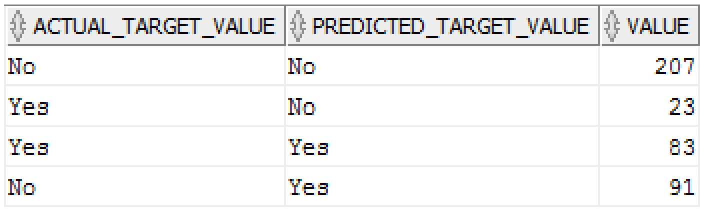
TEST_METRICS	CONFUSION_MATRIX
ODMR$18_51_18_106346IFHRNMF	ODMR$18_51_17_954530VMUXPWL

SELECT * FROM ODMR$18_51_18_106346IFHRNMF



METRIC_NAME	METRIC_VARCHAR_VALUE	METRIC_NUM_VALUE
MODEL_SCHEMA	DMUSER	(null)
MODEL_NAME	CLAS_SVM_MODEL_2	(null)
TEST_DATA_NAME	ODMR$18_51_13_674501NHLUMHU	(null)
MINING_FUNCTION	CLASSIFICATION	(null)
TARGET_ATTRIBUTE	BUY_INSURANCE	(null)
LEAST_TARGET_VALUE	Yes	(null)
ACCURACY	(null)	71.78217821782178217821782178217821782178
TEST_ROWS	(null)	404
AVG_ACCURACY	(null)	73.8824870203874889198429783462074205395
PREDICTIVE_CONFIDENCE	(null)	47.764974040774977839685956692414841079

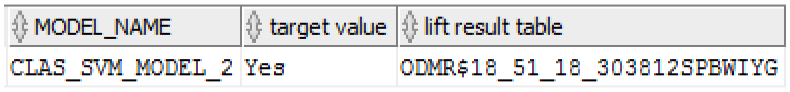
SELECT * FROM ODMR$18_51_17_954530VMUXPWL



ACTUAL_TARGET_VALUE	PREDICTED_TARGET_VALUE	VALUE
No	No	207
Yes	No	23
Yes	Yes	83
No	Yes	91

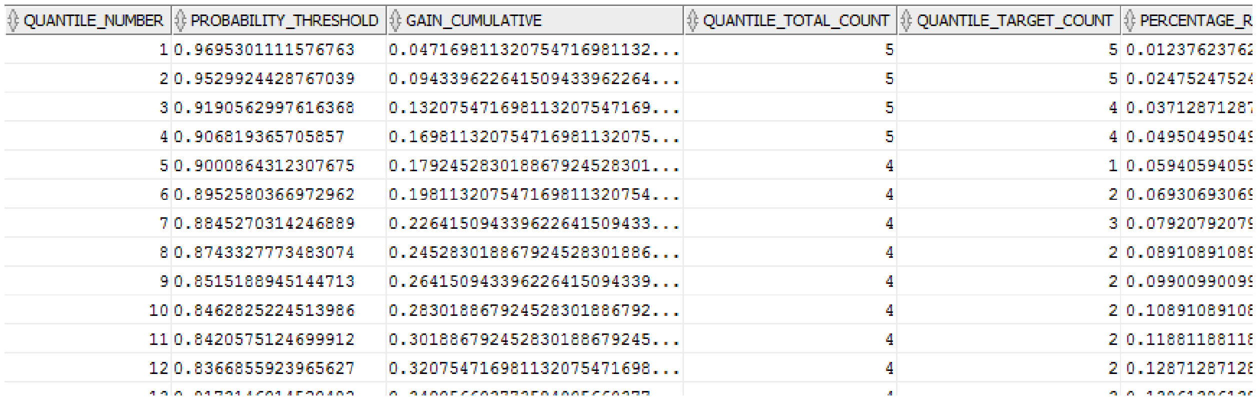
The following query returns the lift result (with target class = 'Yes').

```
SELECT
  MODEL_NAME, a.ATTRIBUTE_NAME "target value", a.VALUE "lift result table"
FROM
  ODMR_USER_WF_CLAS_TEST_RESULTS, TABLE(LIFTS) a
WHERE
  WORKFLOW_NAME = 'build_workflow' AND NODE_NAME = 'Class Build' AND ATTRIBUTE_NAME='Yes'
```



MODEL_NAME	target value	lift result table
CLAS_SVM_MODEL_2	Yes	ODMR$18_51_18_303812SPBWIYG

SELECT * FROM ODMR$18_51_18_303812SPBWIYG



QUANTILE_NUMBER	PROBABILITY_THRESHOLD	GAIN_CUMULATIVE	QUANTILE_TOTAL_COUNT	QUANTILE_TARGET_COUNT	PERCENTAGE_R
1	0.9695301111576763	0.047169811320754716981132...	5	5	0.0123762376
2	0.9529924428767039	0.094339622641509433962264...	5	5	0.0247524752
3	0.9190562997616368	0.132075471698113207547169...	5	4	0.0371287128
4	0.906819365705857	0.169811320754716981132075...	5	4	0.0495049504
5	0.9000864312307675	0.179245283018867924528301...	4	1	0.0594059405
6	0.8952580366972962	0.198113207547169811320754...	4	2	0.0693069306
7	0.8845270314246889	0.226415094339622641509433...	4	3	0.0792079207
8	0.8743327773483074	0.245283018867924528301886...	4	2	0.0891089108
9	0.8515188945144713	0.264150943396226415094339...	4	2	0.0990099009
10	0.8462825224513986	0.283018867924528301886792...	4	2	0.1089108910
11	0.8420575124699912	0.301886792452830188679245...	4	2	0.1188118811
12	0.8366855923965627	0.320754716981132075471698...	4	2	0.1287128712
[illegible]	[illegible]	[illegible]	4	[illegible]	[illegible]

Schedule Apply Workflow to Run

To run the lineage, you can specify the INSUR_CUST_LTV_SAMPLE APPLY node and use the RERUN_NODE_CHILDREN run mode. The demo shows how to schedule the workflow to run daily (DAILY) starting at mid night from 12/31/2014 to 12/31/2015 in EST zone.

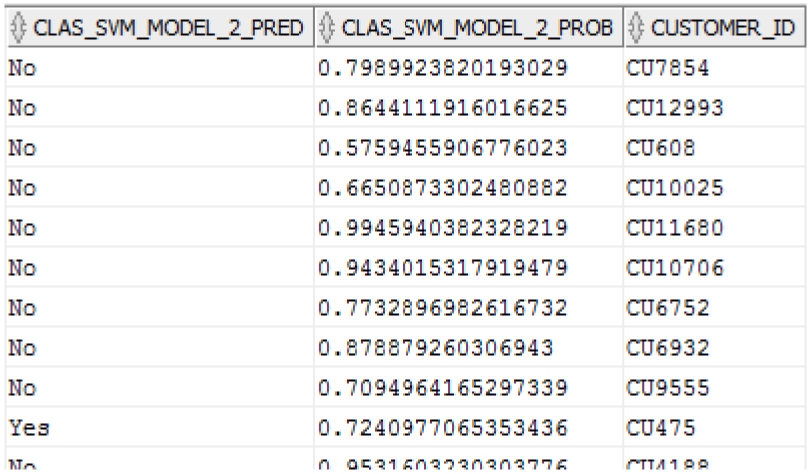
```
CONNECT DMUSER/DMUSER
SET SERVEROUTPUT ON
DECLARE
  v_jobId            VARCHAR2(30) := NULL;
  v_status           VARCHAR2(30) := NULL;
  v_projectName      VARCHAR2(30) := 'Project';
  v_workflow_name    VARCHAR2(30) := 'apply_workflow';
  v_node             VARCHAR2(30) := 'INSUR_CUST_LTV_SAMPLE APPLY';
  v_run_mode         VARCHAR2(30) := ODMRSYS.ODMR_WORKFLOW.RERUN_NODE_CHILDREN;
  v_failure          NUMBER := 0;
  v_nodes            ODMRSYS.ODMR_OBJECT_NAMES := ODMRSYS.ODMR_OBJECT_NAMES();
BEGIN
  v_nodes.extend();
  v_nodes(v_nodes.count) := v_node;
  v_jobId := ODMRSYS.ODMR_WORKFLOW.WF_RUN(p_project_name => v_projectName,
                     p_workflow_name => v_workflow_name,
                     p_node_names => v_nodes,
                     p_run_mode => v_run_mode,
                     p_start_date => '31-DEC-14 12.00.00 AM AMERICA/NEW_YORK',
                     p_repeat_interval => 'FREQ= DAILY',
                     p_end_date => '31-DEC-15 12.00.00 AM AMERICA/NEW_YORK'));
  DBMS_OUTPUT.PUT_LINE('Job: '||v_jobId);
  -- wait for workflow to run to completion
  LOOP
    SELECT STATUS INTO v_status FROM ODMR_USER_PROJECT_WORKFLOW
    WHERE WORKFLOW_NAME = v_workflow_name;
```

```
        IF (v_status IN ('SCHEDULED', 'ACTIVE')) THEN
          DBMS_LOCK.SLEEP(10); -- wait for 10 secs
        ELSE
          EXIT; -- workflow run completes
        END IF;
    END LOOP;
    -- print out all failed nodes (see example above)
EXCEPTION WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: '||SUBSTR(DBMS_UTILITY.FORMAT_ERROR_STACK(), 1, 4000));
END;
```

Query Scoring Result

After the workflow run completes successfully, you can query the scoring result directly.

SELECT * FROM SCORED_CUSTOMERS_TBL



CLAS_SVM_MODEL_2_PRED	CLAS_SVM_MODEL_2_PROB	CUSTOMER_ID
No	0.7989923820193029	CU7854
No	0.864411916016625	CU12993
No	0.5759455906776023	CU608
No	0.6650873302480882	CU10025
No	0.9945940382328219	CU11680
No	0.9434015317919479	CU10706
No	0.7732896982616732	CU6752
No	0.878879260306943	CU6932
No	0.7094964165297339	CU9555
Yes	0.7240977065353436	CU475
No	0.9531603230303776	CU4188

Conclusion

With the workflow APIs, applications can seamlessly integrate the workflow running process. Moreover, the repository views allow applications to monitor the workflow running status and query the generated test results. In addition, applications can use the ODM PL/SQL APIs to query the model results. All the generated results are accessible by the Data Miner, so you can view the results using the Data Miner user interface.

Appendix

PL/SQL APIs

Create a project

This function creates a project using the supplied project name. If the project already exists, it raises an exception. The function returns a project id.

```
FUNCTION PROJECT_CREATE(p_project_name     IN VARCHAR2,
                                          p_comment          IN VARCHAR2 DEFAULT NULL) RETURN NUMBER
```

Parameter	Description
p_project_name	Specify the project to create.
p_comment	Specify the comment to be applied to the project.

Delete a project

This procedure deletes an existing project and contained workflows (all workflow generated objects will be deleted). If any contained workflow is either already running or opened by the Data Miner, it raises an exception.

```
PROCEDURE PROJECT_DELETE(p_project_id IN NUMBER)
```

Parameter	Description
p_project_id	Specify the project to delete.

Delete projects

This procedure deletes multiple projects and their contained workflows (all workflow generated objects will be deleted). If any contained workflow is either already running or opened by the Data Miner, it raises an exception.

```
PROCEDURE PROJECT_DELETE(p_project_ids IN ODMR_OBJECT_IDS)
```

Parameter	Description
p_project_ids	Specify the projects to delete.

Rename a project

This procedure renames an existing project. If a project with the new name already exists, it raises an exception.

```
 PROCEDURE PROJECT_RENAME(p_project_id          IN NUMBER,
                          p_project_name        IN VARCHAR2)
```

Parameter	Description
p_project_id	Specify the project to rename.
p_project_name	Specify the new project name.

Run a workflow

These functions have signatures that accept names or ids of a project, a workflow, and specific nodes to run. The project id, workflow id, and node ids can be queried using the ODMR_USER_WORKFLOW_NODES view (see repository views below). Moreover, the APIs can accept a start time, an end time, and a run interval or a named schedule, so that you can schedule a workflow to run at specific time and interval or at a defined schedule (schedule can be time based or event based). If a workflow has been scheduled to run, it cannot be edited in Data Miner (opened in read only mode) until it is canceled (see Cancel a running workflow API below). If a workflow is either already running or opened by the Data Miner, it cannot be submitted to run (an exception will be raised).

The RERUN_WORKFLOW run mode will run all nodes in a workflow regardless of how these nodes are connected. If a workflow contains two or more separate lineage of nodes, all lineages will be run, but the order of lineage executions is not deterministic.

After the workflow is submitted to run, applications can monitor the workflow running status with the returning job id using the repository views (e.g. ODMR_USER_WORKFLOW_ALL).

```
FUNCTION WF_RUN(P_PROJECT_NAME          IN VARCHAR2,

                P_WORKFLOW_NAME         IN VARCHAR2,

                P_NODE_NAMES            IN ODMR_OBJECT_NAMES,

                P_RUN_MODE              IN VARCHAR2 DEFAULT 'RUN_NODE_ONLY',

                P_MAX_NUM_THREADS       IN NUMBER DEFAULT NULL,

                P_SCHEDULE              IN VARCHAR2 DEFAULT NULL,

                P_JOB_CLASS             IN VARCHAR2 DEFAULT NULL)

 RETURN VARCHAR2


FUNCTION WF_RUN(P_PROJECT_ID            IN NUMBER,

                P_WORKFLOW_ID           IN NUMBER,

                P_NODE_IDS              IN ODMR_OBJECT_IDS,
                           P_RUN_MODE              IN VARCHAR2 DEFAULT 'RUN_NODE_ONLY',
                           P_MAX_NUM_THREADS       IN NUMBER DEFAULT NULL,
                           P_SCHEDULE              IN VARCHAR2 DEFAULT NULL,
                           P_JOB_CLASS             IN VARCHAR2 DEFAULT NULL)
 RETURN VARCHAR2

FUNCTION WF_RUN(P_PROJECT_NAME             IN VARCHAR2,
                           P_WORKFLOW_NAME         IN VARCHAR2,
                           P_NODE_NAMES            IN ODMR_OBJECT_NAMES,
                           P_RUN_MODE              IN VARCHAR2 DEFAULT 'RUN_NODE_ONLY',
                           P_MAX_NUM_THREADS       IN NUMBER DEFAULT NULL,
                           P_START_DATE            IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
                           P_REPEAT_INTERVAL       IN VARCHAR2 DEFAULT NULL,
                           P_END_DATE              IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
                           P_JOB_CLASS             IN VARCHAR2 DEFAULT NULL)
 RETURN VARCHAR2

FUNCTION WF_RUN(P_PROJECT_ID               IN NUMBER,
                           P_WORKFLOW_ID           IN NUMBER,
                           P_NODE_IDS              IN ODMR_OBJECT_IDS,
                           P_RUN_MODE              IN VARCHAR2 DEFAULT 'RUN_NODE_ONLY',
                           P_MAX_NUM_THREADS       IN NUMBER DEFAULT NULL,
                           P_START_DATE            IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
                           P_REPEAT_INTERVAL       IN VARCHAR2 DEFAULT NULL,
                           P_END_DATE              IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
                           P_JOB_CLASS             IN VARCHAR2 DEFAULT NULL)
 RETURN VARCHAR2
```

Parameter	Description
P_PROJECT_NAME	Specify the project that the workflow was created in.
P_PROJECT_ID	Specify the project that the workflow was created in.

Parameter	Description	
P_WORKFLOW_NAME	Specify the workflow to run.	
P_WORKFLOW_ID	Specify the workflow to run.	
P_NODE_NAMES	Specify the nodes of the workflow to run. The run mode determines how the workflow is run.	
P_NODE_IDS	Specify the nodes of the workflow to run. The run mode determines how the workflow is run.	
P_RUN_MODE	VALIDATE_ONLY	validate parent nodes of the specified nodes
	RUN_NODE_ONLY	run the specified nodes. If nodes have already been run, they will not be run again (run once). If parent nodes have not been run, they will be run, otherwise they will be ignored
	RERUN_NODE_ONLY	reset status of the specified nodes to READY state, then run these nodes again
	RERUN_NODE_CHILDREN	reset status of the specified nodes and their children nodes to READY state, and then run these nodes again
	RERUN_NODE_PARENTS	reset status of the specified nodes and their parent nodes to READY state, and then run these nodes again
	RERUN_CHILDREN_ONLY	reset status of children nodes of the specified nodes to READY state, and then run these nodes again. Note: the specified nodes will not be run
	RERUN_WORKFLOW	reset status of all nodes to READY state, and then run them all (complete workflow run). Note: p_node_names is ignored
P_MAX_NUM_THREADS	Specify maximum number of parallel model builds across all workflows. Specify NULL for system determined. You may experience with this parameter only if your system has plenty of resources, otherwise set this value to NULL to use the default value.	
P_SCHEDULE	Specify existing schedule object defined in the Scheduler. If P_SCHEDULE is left	

Parameter	Description
	null, then the workflow is scheduled to run as soon as possible.
P_START_DATE	Specify the first date and time on which this workflow is scheduled to start. If P_START_DATE and P_REPEAT_INTERVAL are left null, then the workflow is scheduled to run as soon as possible.
P_REPEAT_INTERVAL	Specify how often the workflow repeats. You can specify the repeat interval by using calendaring or PL/SQL expressions. The expression specified is evaluated to determine the next time the workflow should run. If P_REPEAT_INTERVAL is not specified, the workflow runs only once at the specified start date. See "Calendaring Syntax" for further information.
P_END_DATE	Specify the date and time after which the workflow expires and is no longer run. If no value for P_END_DATE is specified, the job repeats forever.
P_JOB_CLASS	Specify existing job class used to run the workflow. If no value for P_JOB_CLASS is specified, default job class is used.

Cancel a running workflow

This procedure stops a running workflow or cancels a scheduled workflow to run. If the workflow is not already running or scheduled, it raises an exception.

```
PROCEDURE WF_STOP(p_workflowId IN NUMBER)
```

Parameter	Description
p_workflow_id	Specify the workflow to cancel.

Rename a workflow

This procedure renames an existing workflow. If a workflow with the new name already exists, it raises an exception. If the workflow is either already running or opened by the Data Miner, it raises an exception.

```
PROCEDURE WF_RENAME(p_workflowId        IN NUMBER,
                    p_workflow_name     IN VARCHAR2,
                    p_mode              IN CHAR DEFAULT 'R')
```

Parameter	Description
p_workflow_id	Specify the workflow to rename.

Parameter	Description
p_workflow_name	Specify the new workflow name.
P_mode	Internal use only

Delete a workflow

This procedure deletes a workflow along with all generated objects (e.g. tables, views, models, test results, etc). If the workflow is either already running or opened by the Data Miner, it raises an exception.

```
PROCEDURE WF_DELETE(p_workflowId IN NUMBER)
```

Parameter	Description
p_workflow_id	Specify the workflow to delete.

Import a workflow

This function imports a workflow (exported by the Data Miner) to the specified project. Since workflow is backward compatible, you can import an older version workflow to a newer repository. If the project does not exist, it raises an exception. If the workflow meta data is invalid or incompatible with the current repository (i.e. import a newer workflow to an older repository), it raises an exception. If a workflow with the same name already exists, it raises an exception. During import, it detects if the workflow has object name conflicts with existing workflows in the repository, and the p_force parameter determines whether to abort the import. If p_force is FALSE, it raises an exception with a list of conflicting object names. If p_force is TRUE, it forces the import of the workflow.

```
FUNCTION WF_IMPORT(p_project_id       IN NUMBER,
                   p_workflow_name    IN VARCHAR2,
                   p_workflow_data    IN XMLType,
                   p_comment          IN VARCHAR2,
                   p_force            IN BOOLEAN DEFAULT FALSE) RETURN NUMBER;
```

Parameter	Description
p_project_id	Specify the project to import the workflow.
p_workflow_name	Specify the workflow to import.
p_workflow_data	Specify the workflow meta data. This workflow should be previously exported by the Data Miner and the workflow version should not be newer than what the repository supports.

Parameter	Description
p_comment	Specify the comment to be applied to the workflow.
p_force	Whether to force import if the workflow has object name conflicts with existing workflows in the repository. If p_force = FALSE, raise an exception with a list of conflicting object names. If p_force = TRUE, force the import of the workflow.

Example:

This example creates a new project Project2 and imports the build_workflow.xml to a new workflow workflow2. It assumes a directory object DMUSER_DIR has been created and the build_workflow.xml is placed into the directory points to by the directory object.

```
CONNECT DMUSER/DMUSER
SET SERVEROUTPUT ON
DECLARE
  v_wfId          NUMBER := NULL;
  v_projectId     NUMBER := NULL;
BEGIN
  v_projectId := ODMRSYS.ODMR_PROJECT.PROJECT_CREATE(p_project_name => 'Project2',
                                                      p_comment => 'This is a demo project');
  DBMS_OUTPUT.PUT_LINE('Project: '||v_projectId);

  v_wfId := ODMRSYS.ODMR_WORKFLOW.WF_IMPORT(p_project_id => v_projectId,
                       p_workflow_name => 'workflow2',
                       p_workflow_data => XMLType(bfilename('DMUSER_DIR',
                                      'build_workflow.xml'), nls_charset_id('AL32UTF8')),
                       p_comment => 'This is a build workflow');
  DBMS_OUTPUT.PUT_LINE('Workflow: '||v_wfId);
END;
```

Export a workflow

This function exports a specified workflow. If the workflow is either already running or opened by the Data Miner, it raises an exception. Alternatively, you can query the ODMR_USER_PROJECT_WORKFLOW view (see Repository Views below) for workflows to export.

```
FUNCTION WF_EXPORT(p_workflow_id IN NUMBER) RETURN XMLType;
```

Parameter	Description
p_workflow_id	Specify the workflow to export.

Example 1:

This example uses the WF_EXPORT function to export the specified workflow.

```
CONNECT DMUSER/DMUSER
SET SERVEROUTPUT ON
DECLARE
   v_workflow_id         NUMBER;
   v_workflow_data     XMLType;
BEGIN
   SELECT WORKFLOW_ID INTO v_workflow_id FROM ODMR_USER_PROJECT_WORKFLOW
   WHERE project_name='Project' AND workflow_name='build_workflow';
   v_workflow_data := ODMRSYS.ODMR_WORKFLOW.WF_EXPORT(p_workflow_id => v_workflow_id);
   DBMS_OUTPUT.PUT_LINE('Workflow: '||SUBSTR(v_workflow_data.getClobVal(), 1, 32000));
END;
```

Example 2:

This example shows how you can import a workflow from a remote database via a database link. First, create a database link for the ODMR_USER_PROJECT_WORKFLOW repository view in the remote database, and then query the desired workflow from the view to deploy to the target database using the WF_IMPORT function. Note: this method only works if the source and target user account have the same name.

```
CONNECT DMUSER/DMUSER

-- in the target database, create a database link to the remote database (assume account is DMUSER)
CREATE DATABASE LINK to_remote_user CONNECT TO DMUSER IDENTIFIED BY DMUSER USING
'(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=<host
name>)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=<service name>)))';

SET SERVEROUTPUT ON
DECLARE
   v_wfId                  NUMBER := NULL;
   v_projectId            NUMBER := NULL;
   v_workflowXML     XMLType;
BEGIN
   -- query the desired workflow from the remote database
   SELECT WORKFLOW_DATA INTO v_workflowXML
   FROM ODMR_USER_PROJECT_WORKFLOW@to_remote_user
   WHERE project_name='Project' AND workflow_name='build_workflow';
   -- create a new project to store the workflow in the target database
   v_projectId := ODMRSYS.ODMR_PROJECT.PROJECT_CREATE(p_project_name => 'Deploy Project',
                                                       p_comment => 'Deployment Project');
   DBMS_OUTPUT.PUT_LINE('Project: '||v_projectId);
   -- import the workflow to the target database
   v_wfId := ODMRSYS.ODMR_WORKFLOW.WF_IMPORT(p_project_id => v_projectId,
                                              p_workflow_name => 'build_workflow',
                                              p_workflow_data => v_workflowXML,
                                                  p_comment => 'Deployment Workflow');
  DBMS_OUTPUT.PUT_LINE('Workflow: '||v_wfId);
END;
```

Repository Views

ODMR_USER_PROJECT_WORKFLOW

ODMR_USER_PROJECT_WORKFLOW describes the workflows accessible to the current user.

Column	Datatype	Description	
PROJECT_ID	NUMBER	Project that the workflow was created in	
PROJECT_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in	
PJ_CREATION_TIME	TIMESTAMP(6)	Project creation time stamp	
PJ_LAST_UPDATED_TIME	TIMESTAMP(6)	Project last modified time stamp	
PJ_COMMENTS	VARCHAR2(4000 CHAR)	Project comment	
WORKFLOW_ID	NUMBER	Workflow id	
WORKFLOW_NAME	VARCHAR2(30 CHAR)	Workflow name	
WORKFLOW_DATA	XMLTYPE	Workflow meta data in XML format	
CHAIN_NAME	VARCHAR2(30 CHAR)	Internal use only	
STATUS	VARCHAR2(30 CHAR)	INACTIVE	workflow is idle
		ACTIVE	workflow is running
		QUEUED	workflow is queued to run
		STOPPING	running workflow is being stopped
		STOPPED	running workflow is stopped
		SCHEDULED	workflow is scheduled to run
WF_CREATION_TIME	TIMESTAMP(6)	Workflow creation time stamp	
WF_LAST_UPDATED_TIME	TIMESTAMP(6)	Workflow last modified time stamp	
WF_COMMENTS	VARCHAR2(4000 CHAR)	Workflow comment	

ODMR_USER_WORKFLOW_ALL

ODMR_USER_ WORKFLOW_ALL describes the workflows run status accessible to the current user.

Column	Datatype	Description
WORKFLOW_ID	NUMBER	Workflow
WF_JOB_NAME	VARCHAR2(261)	Scheduler Job used to run the workflow
LOG_DATE	TIMESTAMP(6) WITH TIME ZONE	Log entry time stamp
LOG_ID	NUMBER	Log entry id
NODE_ID	VARCHAR2(261)	Workflow node
SUBNODE_ID	VARCHAR2(261)	Workflow sub node (e.g. model in build node)
NODE_STATUS	VARCHAR2(11)	RUNNING node is running SUCCEEDED node completes successfully FAILED node fails to complete NOT_STARTED node is waiting to run SCHEDULED node is scheduled to run PAUSED node is paused (exception state) STOPPED node run is stopped STALLED node is stalled (exception state)
SUBNODE_STATUS	VARCHAR2(30)	Same as NODE_STATUS above
NODE_START_TIME	TIMESTAMP(6) WITH TIME ZONE	Workflow node start time stamp
NODE_RUN_TIME	INTERVAL DAY(9) TO SECOND(6)	Workflow node run time
ERROR_CODE	NUMBER	Error code returned by the node
LOG_MESSAGE	VARCHAR2(4000 CHAR)	Log message generated by the node

ODMR_USER_WORKFLOW_LOG

ODMR_USER_ WORKFLOW_LOG describes the completed workflow run details accessible to the current user. The Data Miner event log data is populated from this view. In addition, the project and workflow PL/SQL APIs log messages to the repository, so you can query this event log view to trace API calls. In case of error or warning messages, you can find out more details in the LOG_MESSAGE_DETAILS. The number of log entries is automatically trimmed to the max size (100000) specified by the MAX_WORKFLOW_LOG_COUNT parameter in the ODMR$REPOSITORY_PROPERTIES repository table.

Column	Datatype	Description	
LOG_ID	NUMBER	Log entry id	
JOB_NAME	VARCHAR2(30 CHAR)	Scheduler Job used to run the workflow	
PROJ_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in	
PRO_ID	NUMBER	Project that the workflow was created in	
WF_NAME	VARCHAR2(30 CHAR)	Workflow name	
WF_ID	NUMBER	Workflow id	
NODE_NAME	VARCHAR2(30 CHAR)	Workflow node name	
NODE_ID	VARCHAR2(30)	Workflow node id	
SUBNODE_NAME	VARCHAR2(30 CHAR)	Workflow sub node name (e.g. model name in build node)	
SUBNODE_ID	VARCHAR2(30)	Workflow sub node id (e.g. model id in build node)	
LOG_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	Log entry time stamp	
LOG_DURATION	INTERVAL DAY(3) TO SECOND(0)	Log entry duration	
LOG_TYPE	VARCHAR2 (30 CHAR)	WARN	Warning
		ERR	Error
		INFO	Informational
LOG_SUBTYPE	VARCHAR2 (30 CHAR)	START	Signal start of a task
		END	Signal end of a task

Column	Datatype	Description
LOG_TASK	VARCHAR2 (30 CHAR)	When a node is run, it performs one or more following tasks: PROJECT WORKFLOW NODE SUBNODE VALIDATE SAMPLE CACHE STATISTICS FEATURES DATAPREP BUILD TEST APPLY TRANSFORM TEXT BUILDTEXT APPLYTEXT OUTPUT CLEANUP CREATE EXPLORE STATISTICS CREATE HISTOGRAMS CREATE SAMPLE DATA CREATE HISTOGRAM SAMPLE CREATE DATA GUIDE CREATE PROJECT DELETE PROJECT RENAME PROJECT SET COMMENT

Column	Datatype	Description
		CREATE WORKFLOW RUN WORKFLOW RENAME WORKFLOW DELETE WORKFLOW IMPORT WORKFLOW EXPORT WORKFLOW
LOG_MESSAGE	NVARCHAR2(2000)	Log message generated by the node
LOG_MESSAGE_DETAILS	VARCHAR2(4000 CHAR)	Log message details generated by the node

ODMR_USER_WORKFLOW_NODES

ODMR_USER_ WORKFLOW_NODES describes the workflow nodes accessible to the current user.

Column	Datatype	Description	
PROJECT_ID	NUMBER	Project that the workflow was created in	
PROJECT_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in	
WORKFLOW_ID	NUMBER	Workflow id	
WORKFLOW_NAME	VARCHAR2(30 CHAR)	Workflow name	
NODE_TYPE	VARCHAR2(30 CHAR)	Workflow node type	
NODE_NAME	VARCHAR2(30 CHAR)	Workflow node name	
NODE_ID	NUMBER	Workflow node id	
NODE_STATUS	VARCHAR2(10 CHAR)	Invalid	not ready to run
		Warning	run completes with warning
		Ready	ready to run
		Failure	run fails

Column	Datatype	Description
		Complete run succeeds

ODMR_USER_WORKFLOW_MODELS

ODMR_USER_WORKFLOW_MODELS describes the workflow model build nodes accessible to the current user.

Column	Datatype	Description
PROJECT_ID	NUMBER	Project that the workflow was created in
PROJECT_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in
WORKFLOW_ID	NUMBER	Workflow id
WORKFLOW_NAME	VARCHAR2(30 CHAR)	Workflow name
NODE_TYPE	VARCHAR2(30 CHAR)	Workflow node type
NODE_ID	NUMBER	Workflow node id
NODE_NAME	VARCHAR2(30 CHAR)	Workflow node name
NODE_STATUS	VARCHAR2(10 CHAR)	Invalid not ready to run Warning run completes with warning Ready ready to run Failure run fails Complete run succeeds
MODEL_TYPE	VARCHAR2(30 CHAR)	Model types (e.g. NaiveBayesModel)
MODEL_ID	NUMBER	Model id
MODEL_NAME	VARCHAR2(30 CHAR)	Model name
MODEL_STATUS	VARCHAR2(10 CHAR)	See NODE_STATUS above

Column	Datatype	Description
MODEL_CREATIONDATE	VARCHAR2(30 CHAR)	Model creation time

ODMR_USER_WF_CLAS_TEST_RESULTS

ODMR_USER_WF_CLAS_TEST_RESULTS describes the generated classification results accessible to the current user.

Column	Datatype	Description	
PROJECT_ID	NUMBER	Project that the workflow was created in	
PROJECT_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in	
WORKFLOW_ID	NUMBER	Workflow id	
WORKFLOW_NAME	VARCHAR2(30 CHAR)	Workflow name	
NODE_TYPE	VARCHAR2(30 CHAR)	Workflow node type	
NODE_ID	NUMBER	Workflow node id	
NODE_NAME	VARCHAR2(30 CHAR)	Workflow node name	
NODE_STATUS	VARCHAR2(10 CHAR)	Invalid	not ready to run
		Warning	run completes with warning
		Ready	ready to run
		Failure	run fails
		Complete	run succeeds
MODEL_ID	NUMBER	Model id	
MODEL_NAME	VARCHAR2(30 CHAR)	Model name	
MODEL_STATUS	VARCHAR2(10 CHAR)	Warning	run completes with warning
		Ready	ready to run
		Failure	run fails
		Complete	run succeeds
MODEL_CREATIONDATE	VARCHAR2(30 CHAR)	Model creation time	

Column	Datatype	Description
TEST_METRICS	VARCHAR2(128 CHAR)	Test metric result table (contains predictive confidence, accuracy, etc)
CONFUSION_MATRIX	VARCHAR2(128 CHAR)	Test confusion result table
LIFTS	DM_NESTED_CATEGORICALS	Table of DM_NESTED_CATEGORICAL, where ATTRIBUTE_NAME contains target class and VALUE contains lift result table
ROCS	DM_NESTED_CATEGORICALS	Table of DM_NESTED_CATEGORICAL, where ATTRIBUTE_NAME contains target class and VALUE contains ROC result table
ROC_AREA	DM_NESTED_NUMERICALS	Table of DM_NESTED_NUMERICAL, where ATTRIBUTE_NAME contains target class and VALUE contains ROC area under curve value

ODMR_USER_WF_REGR_TEST_RESULTS

ODMR_USER_WF_REGR_TEST_RESULTS describes the generated regression results accessible to the current user.

Column	Datatype	Description	
PROJECT_ID	NUMBER	Project that the workflow was created in	
PROJECT_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in	
WORKFLOW_ID	NUMBER	Workflow id	
WORKFLOW_NAME	VARCHAR2(30 CHAR)	Workflow name	
NODE_TYPE	VARCHAR2(30 CHAR)	Workflow node type	
NODE_ID	NUMBER	Workflow node id	
NODE_NAME	VARCHAR2(30 CHAR)	Workflow node name	
NODE_STATUS	VARCHAR2(10 CHAR)	Invalid	not ready to run
		Warning	run completes with warning
		Ready	ready to run
		Failure	run fails

Column	Datatype	Description
		Complete run succeeds
MODEL_ID	NUMBER	Model id
MODEL_NAME	VARCHAR2(30 CHAR)	Model name
MODEL_STATUS	VARCHAR2(10 CHAR)	Warning run completes with warning
		Ready ready to run
		Failure run fails
		Complete run succeeds
MODEL_CREATIONDATE	VARCHAR2(30 CHAR)	Model creation time
TEST_METRICS	VARCHAR2(128 CHAR)	Test metric result table (contains predictive confidence, root mean square error, etc)
RESIDUAL_PLOT	VARCHAR2(128 CHAR)	Test residual plot table

ODMR_USER_WF_TEST_RESULTS

ODMR_USER_ WF_TEST_RESULTS describes both the generated classification and regression results accessible to the current user.

Column	Datatype	Description
PROJECT_ID	NUMBER	Project that the workflow was created in
PROJECT_NAME	VARCHAR2(30 CHAR)	Project that the workflow was created in
WORKFLOW_ID	NUMBER	Workflow id
WORKFLOW_NAME	VARCHAR2(30 CHAR)	Workflow name
NODE_TYPE	VARCHAR2(30 CHAR)	Workflow node type
NODE_ID	NUMBER	Workflow node id
NODE_NAME	VARCHAR2(30 CHAR)	Workflow node name
NODE_STATUS	VARCHAR2(10 CHAR)	Invalid not ready to run
		Warning run completes with warning
		Ready ready to run

Column	Datatype	Description
		Failure run fails Complete run succeeds
MODEL_ID	NUMBER	Model id
MODEL_NAME	VARCHAR2(30 CHAR)	Model name
MODEL_STATUS	VARCHAR2(10 CHAR)	Warning run completes with warning Ready ready to run Failure run fails Complete run succeeds
MODEL_CREATIONDATE	VARCHAR2(30 CHAR)	Model creation time
TEST_METRICS	VARCHAR2(128 CHAR)	Test metric result table (contains predictive confidence, etc)
CONFUSION_MATRIX	VARCHAR2(128 CHAR)	Test confusion result table
LIFTS	DM_NESTED_CATEGORICALS	Table of DM_NESTED_CATEGORICAL, where ATTRIBUTE_NAME contains target class and VALUE contains lift result table
ROCS	DM_NESTED_CATEGORICALS	Table of DM_NESTED_CATEGORICAL, where ATTRIBUTE_NAME contains target class and VALUE contains ROC result table
ROC_AREA	DM_NESTED_NUMERICALS	Table of DM_NESTED_NUMERICAL, where ATTRIBUTE_NAME contains target class and VALUE contains ROC area under curve value
RESIDUAL_PLOT	VARCHAR2(128 CHAR)	Test residual plot table

Oracle Data Miner (Extension of SQL Developer 4.1)
Use Repository APIs to Manage and Schedule Workflows to run
March 2015
Author: Denny Wong