

# **Oracle Financial Services Analytical Applications Infrastructure**

**User Guide**

**Release 8.0.7.0.0**

**May 2019**



## OFS AIF4AML User Guide

Copyright © 2019 Oracle and/or its affiliates. All rights reserved.

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 de-compilation of this software, unless required by law for interoperability, is prohibited.

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

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

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

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

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

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

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

For information on third party licenses, click [here](#).

## Document Control

Version Number	Revision Date	Change Log
1.0	May 2019	Created the document for 8.0.7.0.0 release.

# Table of Contents

<b>1</b>	<b>Preface</b>	<b>5</b>
1.1	Audience	5
1.1.1	<i>Prerequisites for the Audience</i>	5
1.2	Related Documents	5
1.3	Conventions Used in this Guide	5
1.4	Acronyms Used in this Guide	5
<b>2</b>	<b>Using AIF4AML</b>	<b>7</b>
2.1	Knowing the Prerequisites	7
2.2	Logging into OFS FCC Studio	7
2.3	Accessing AIF User Notebooks	8
2.4	Loading the AIF4AML Library	9
2.5	Loading datasets from AIF4AML library model groups	9
2.6	Loading behavioral and non-behavioral dataframes from model group datasets	10
2.6.1	<i>Viewing dimension of behavioral and non-behavioral dataframes</i>	10
2.7	Applying Transformation on data from datasets	11
2.7.1	<i>Applying Time-series Clustering</i>	11
2.7.2	<i>Applying Transformation - Bitmap Jump</i>	13
2.8	Selecting NB Variables to Build Models	15
2.9	Generating Stage 2 Dataset	15
2.10	Generating Uni-Variate Analysis	17
2.11	Building Models using OREXV Package	18
2.12	Deploying Models	21
2.13	Viewing List of Applied Transformations	21
2.14	Updating the Transformations' List	22
2.15	Saving the Run Definition	22

# 1 Preface

This is a user assistance document for users of the Oracle Financial Services (OFS) Adaptive Intelligence Foundation for Anti Money Laundering (AIF4AML) application. The document provides information about using the application through the FCC Studio user-interface.

## 1.1 Audience

This guide is intended for users of AIF4AML who will create models and compare the created systems.

### 1.1.1 Prerequisites for the Audience

This document assumes that you have working knowledge about the following:

- OFSBD pack
- OFSFCC Studio
- Creating Models

## 1.2 Related Documents

This section identifies additional documents related to OFSAIF4AML in the following list:

- OFSBD documents from [OHC](#).
- OFSFCC Studio documents from [OHC](#).
- OFSAIF4AML documents from [OHC](#):
  - OFS AIF4AML Installation Guide 8.0.7.0.0
  - OFS AIF4AML Administration and Configuration Guide 8.0.7.0.0

## 1.3 Conventions Used in this Guide

- Window names are *italicized*.
- Window actions are indicated in **Bold**.

## 1.4 Acronyms Used in this Guide

Acronym	Description
AML	Anti Money Laundering
API	Application Programming Interface
DIM	Dimension

Acronym	Description
EDA	Exploratory Data Analysis
FCC	Financial Crime and Compliance
IV	Information Value
NB	Non-behavioral
OFSAA	Oracle Financial Services Advanced Analytical Applications
OFSAAI	Oracle Financial Services Analytical Applications Infrastructure
OFSBD	Oracle Financial Services Behavior Detection
OHC	Oracle Help Centre
OLAP	Online Analytical Processing
ORE	Oracle R Enterprise
OSIT	Out-of-sample-in-time
OSOT	Out-of-sample-Out-of-time
UI	User Interface
URL	Uniform Resource Locator

## 2 Using AIF4AML

OFS AIF4AML application is a foundation with building-blocks for ML life-cycle, tailored for the AML domain. It uses the familiar notebook environment to rapidly train, test and validate ML models. It has a pre-defined dataset with more than 300 attributes ready for variable analysis. Users can execute models with multiple techniques and compare the results side-by-side.

The application UI for users involves the following topics:

1. [Knowing the Prerequisites](#)
2. [Logging into OFS FCC Studio](#)
3. [Accessing AIF User Notebooks](#)
4. [Loading the AIF4AML Library](#)
5. [Loading datasets from AIF4AML library model groups](#)
6. [Loading behavioral and non-behavioral dataframes from model group datasets](#)
7. [Applying Transformation on the data from the datasets](#)
8. [Applying Transformation - Bitmap Jump](#)
9. [Selecting NB Variables to Build Models](#)
10. [Generating Stage 2 Dataset](#)
11. [Performing OREXV operations on stage 2 dataset](#)
12. [Deploying Models](#)
13. [Viewing List of Applied Transformations](#)
14. [Updating the Transformations' List](#)
15. [Saving the Run Definition](#)

### 2.1 Knowing the Prerequisites

The prerequisites to use AIF4AML is in the following:

1. Users must have the requisite permissions to access OFSFCC Studio.
2. Users must know how to use OFSFCC Studio for features such as Creating Notebooks and Paragraphs.

### 2.2 Logging into OFS FCC Studio

Login to OFS FCC Studio and create Notebooks. The following is the procedure to login to OFS FCC Studio:

1. Enter the URL for OFS FCC Studio in a web browser to display the Login window.



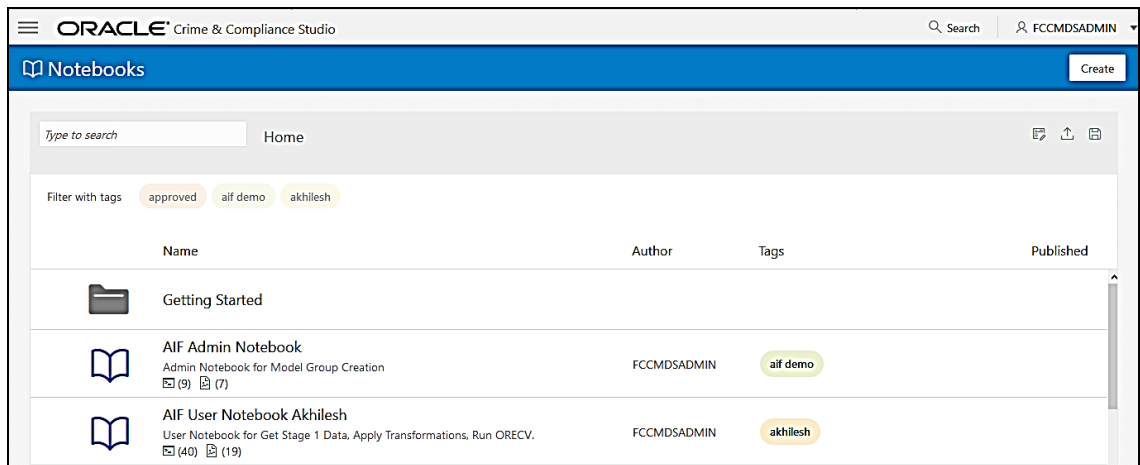
**OFS FCC Studio Login Window**

2. Enter the details in the **Username** and **Password** fields, and click **Login** to display the *OFS FCC Studio Home* window.

**NOTE** The user name and password to login is your OFSAA login ID and password.

## 2.3 Accessing AIF User Notebooks

On the *OFS FCC Studio Home* window, AIF User Notebooks are displayed. These Notebooks are pre-packaged with required APIs that allows you to create Stage 1 data, and build and train models. Click User Notebooks to run the various APIs and functions to create Models. You can also customize the Notebooks to include your transformations, along with the prepackaged ones.



**OFS FCC Studio Home Window**



## 2.4 Loading the AIF4AML Library

After you open an AIF User Notebook, on the Notebook window, the first step is to load libraries from the AIF4AML application.

Execute the paragraph instructions to load AIF4AML Library as shown in the following illustration:

```

Load the Library ofsaif

%fcc-ore
library(ofsaiif)
x <- ofsaif()

```

1284 ms @ an hour ago

Loading AIF4AML Library

## 2.5 Loading datasets from AIF4AML library model groups

After loading the ofsaif library, the next step is to load the Stage 1 data created during CreateStage1 batch (for details on how to run the batch, see the [OFS AIF4AML Administration and Configuration Guide](#)). Stage 1 data output consists of Behavioral and Non-behavioral data.

You can run models in the following methods:

1. OSIT (Out Of Sample In Time) - Pass only one date range, and the test and train sample is considered from the same dataset.
2. OSOT (Out Of Sample Out Of Time) - Pass two date ranges, one for Model Build dataset; which is used to train a dataset, and the other date range for OSOT dataset; which is used to test the Model.
3. Both OSIT and OSOT - Pass a combination of OSIT and OSOT, which requires two date ranges as described for OSOT.

In the paragraph, specify the list of Model Groups that are required for data and execute the paragraph as shown in the following illustrations:

```

Get the Behavioral and Non-Behavioral Object

%fcc-ore
x <- ofsaif::getStage1Data(x, model.groups = c("BUS_DMN_LIST_TX_A") )

```

Loading stage 1 datasets from library without OSOT validations

### NOTE

If you do not specify a Model Group list, the application loads Stage 1 data for all available and active Model Groups.

```

Get the Behavioral and Non-Behavioral Object With OSOT
%fcc-ore
x <- ofsaif::getStage1Data( x,
                           model.groups = c("BUS_DMN_LIST_TX_A"),
                           model.build.date.range = c(201501, 201510),
                           osot.date.range = c(201511, 201601),
                           order.data = F)

```

Loading stage 1 datasets from library with OSOT validations

## 2.6 Loading behavioral and non-behavioral dataframes from model group datasets

After loading the Stage 1 dataset, segregate and create ORE frames for Behavioral and Non Behavioral datasets, which will be used in Model building.

Execute the functions shown in the following illustrations to derive the outputs:

```

%fcc-ore

B <- ofsaif::getBehaviouralFrame(x)
NB <- ofsaif::getNonBehaviouralFrame(x)

```

Loading behavioral and non-behavioral dataframes without OSOT validations

```

%fcc-ore

B_OSOT <- ofsaif::getBehaviouralFrame( x, osot = T )
NB_OSOT <- ofsaif::getNonBehaviouralFrame(x, osot = T )

```

Loading behavioral and non-behavioral dataframes with OSOT validations

### NOTE

In case of OSOT or BOTH, create separate ORE frames on both the Model build dataset and the OSOT dataset as shown in the preceding illustration.

### 2.6.1 Viewing dimension of behavioral and non-behavioral dataframes

After loading behavioral and non-behavioral dataframes from model group datasets, you can view the DIM of dimension of behavioral and non-behavioral dataframes as shown in the following illustration:

```
%fcc-ore

dim(B)
dim(NB)
dim(B_OSOT)
dim(NB_OSOT)
```

### Viewing DIM

You have to now transform the available data. For more details, see the following section.

## 2.7 Applying Transformation on data from datasets

Stage-1 data transformation is achieved using time-series clustering and bit-map jump. The following subsections provide details about how to apply transformation.

### 2.7.1 Applying Time-series Clustering

The time-series function can return the following types of variables to cover both aspects of time-series data:

1. Trend variable to focus more on magnitude (above or below the mean).
2. Direction variable to focus more on direction (increasing or decreasing).

The following illustrations show examples for transformation applied to time-series clustering model build data and time-series clustering OSOT data:

```
%fcc-ore

tsobj <- ofsaif::timeSeriesClustering(x = x,data=B, include = c("TOT_DEPST_AM","TOT_WDRWL_AM"),
                                     bit.map.type = c("clip", "trend"), max.cluster = 20 )
```

#### Transformation - Time Series Clustering Model Build Data

```
%fcc-ore

tsobj_OSOT <- ofsaif::timeSeriesClustering(x = x,data=B_OSOT, include = c("TOT_DEPST_AM","TOT_WDRWL_AM"),
                                           bit.map.type = c("clip", "trend"), max.cluster = 20 )
```

#### Transformation - Time Series Clustering OSOT Data

Where input parameters are,

- x: Object of class ofsaif
- B: Behavioral Object
- include: List of features to be included for time-series clustering
- exclude: List of features to be excluded for time-series clustering
- bit.map.type: Type of Feature Extraction - clip or trend
- max.cluster: Maximum number of clusters to be considered (Default=20)

The output contains ORE frame with the transformed time-series variables. The following illustration is an example:

```

%fcc-ore
print(head(tsobj))
print(head(tsobj_OSOT))
#print(dim(tsobj))
#print(dim(tsobj_OSOT))
    
```

CUST_INTRL_ID	MODEL_GROUP_NAME	TOT_DEPST_AM_CLIP	TOT_DEPST_AM_TREND
1	CUTAMLEB-070 BUS_DMN_LIST_TX_A	12	10
2	CUTAMLEB-071 BUS_DMN_LIST_TX_A	12	10
3	CUTAMLEB679 BUS_DMN_LIST_TX_A	13	17
4	CUTAMLEB680 BUS_DMN_LIST_TX_A	9	3
5	CUTAMLEB681 BUS_DMN_LIST_TX_A	13	18
6	CUTAMLEB684 BUS_DMN_LIST_TX_A	15	1

	TOT_DEPST_AM_MEAN	TOT_WDRWL_AM_CLIP	TOT_WDRWL_AM_TREND	TOT_WDRWL_AM_MEAN
1	4562.20	4	7	19220.60
2	3580254.70	3	18	3546835.80
3	391286.25	11	3	10236529.53
4	26739.00	11	19	156838.60
5	26165.60	1	1	62578.50
6	7214.56	17	20	20780.37

CUST_INTRL_ID	MODEL_GROUP_NAME	TOT_DEPST_AM_CLIP	TOT_DEPST_AM_TREND
1	CUTAMLEB-070 BUS_DMN_LIST_TX_A	3	5
2	CUTAMLEB-071 BUS_DMN_LIST_TX_A	8	2
3	CUTAMLEB679 BUS_DMN_LIST_TX_A	4	4
4	CUTAMLEB680 BUS_DMN_LIST_TX_A	3	5
5	CUTAMLEB681 BUS_DMN_LIST_TX_A	4	4
6	CUTAMLEB682 BUS_DMN_LIST_TX_A	1	1

	TOT_DEPST_AM_MEAN	TOT_WDRWL_AM_CLIP	TOT_WDRWL_AM_TREND	TOT_WDRWL_AM_MEAN
1	461725.667	1	5	567181.333
2	0.000	2	1	0.000
3	28604.190	4	4	2935864.790
4	1910.667	1	5	4290.667
5	3940.500	4	4	5024.000
6	130423.535	6	2	18643.622

**Output of time-series clustering**

**NOTE**

- New Customers (scenario where data is not available for all the months) are assigned a constant Cluster Id: max.cluster + 1.
- Either include or exclude parameter has to be NULL. If both are NULL, all the input attributes are considered for clustering.



```
%fcc-ore
print(head(bmobj))
print(head(bmobj_OSOT))
print(dim(bmobj))
print(dim(bmobj_OSOT))
```



CUST_INTRL_ID	MODEL_GROUP_NAME	TOT_DEPST_AM_JMP_P50
1	CUTAMLBM-070 BUS_DMN_LIST_TX_A	__0
2	CUTAMLBM-071 BUS_DMN_LIST_TX_A	__0
3	CUTAMLBM679 BUS_DMN_LIST_TX_A	__0
4	CUTAMLBM680 BUS_DMN_LIST_TX_A	__0
5	CUTAMLBM681 BUS_DMN_LIST_TX_A	__0
6	CUTAMLBM682 BUS_DMN_LIST_TX_A	__0

TOT_DEPST_AM_JMP_P100	TOT_DEPST_AM_MEAN	TOT_WDRWL_AM_JMP_P50
1	__0 4562.20	__0
2	__0 3580254.70	__1
3	__0 391286.25	__1
4	__0 26739.00	__1
5	__0 26165.60	__1
6	__0 7214.56	__0

TOT_WDRWL_AM_JMP_P100	TOT_WDRWL_AM_MEAN
1	__0 19220.60
2	__1 3546835.80
3	__0 10236529.53
4	__0 156838.60
5	__1 62578.50
6	__0 20780.37

CUST_INTRL_ID	MODEL_GROUP_NAME	TOT_DEPST_AM_JMP_P50
1	CUTAMLBM-070 BUS_DMN_LIST_TX_A	__0
2	CUTAMLBM-071 BUS_DMN_LIST_TX_A	__0
3	CUTAMLBM679 BUS_DMN_LIST_TX_A	__1
4	CUTAMLBM680 BUS_DMN_LIST_TX_A	__0
5	CUTAMLBM681 BUS_DMN_LIST_TX_A	__1
6	CUTAMLBM682 BUS_DMN_LIST_TX_A	__1

TOT_DEPST_AM_JMP_P100	TOT_DEPST_AM_MEAN	TOT_WDRWL_AM_JMP_P50
1	__0 461725.667	__0
2	__0 0.000	__0
3	__0 28604.190	__1
4	__0 1910.667	__0
5	__0 3940.500	__0
6	__0 130423.535	__1

TOT_WDRWL_AM_JMP_P100	TOT_WDRWL_AM_MEAN
1	__0 567181.333
2	__0 0.000
3	__0 2935864.790
4	__0 4290.667
5	__0 5024.000
6	__0 18643.622

[1] 219 8

[1] 219 8

Output of Transformation - Bitmap Jump

**NOTE**

- Either include or exclude parameter has to be NULL. If both are NULL, all the input attributes in the Behavioral frame is considered.
- Possible values of bit:
  - 1 - Jump exceeds the threshold percentage
  - 0 - Jump doesn't exceeds the threshold percentage
  - \_ - Insufficient data to compute jump

## 2.8 Selecting NB Variables to Build Models

As part of the procedure to transform Stage 1 data, select the NB variables required to build models as shown in the following illustration:

**Selecting NB Variables For Model Build**

```

%fcc-ore
#colnames(NB)
#Considering some selected predictors from NB data
colnames(NB[c(1:10,12,14,15,16,90,91,98,99,125,127,128,129,130)])
    
```

📄
🔍
📄
🔄
⬇️
⌵
💬

[1]	"CUST_INTRL_ID"	"MODEL_GROUP_NAME"	"AGE_YR_CT"
[4]	"ALIAS_NM"	"ALT_CUST_ID"	"ANNL_BND_TRD_QT"
[7]	"ANNL_CMDTY_TRD_QT"	"ANNL_EQTY_TRD_QT"	"ANNL_INCM_BASE_AM"
[10]	"ANNL_INCM_RPTG_AM"	"AVG_BND_TRD_AM"	"AVG_EQTY_TRD_AM"
[13]	"AVG_OPTN_TRD_AM"	"BIRTH_DT_TERM"	"LQD_NET_WRTH_BASE_AM"
[16]	"LQD_NET_WRTH_RPTG_AM"	"NET_WRTH_BASE_AM"	"NET_WRTH_RPTG_AM"
[19]	"SAR_FLG"	"FOLD_TWO"	"FOLD_THREE"
[22]	"FOLD_FIVE"	"FOLD_TEN"	

Selecting NB variables to build models

## 2.9 Generating Stage 2 Dataset

Data objects that was transformed previously from the Stage 1 dataset is used to create Stage 2 dataset. The input consists of the object class of OFSAIF and the ORE frames from Stage 1 (comma-separated). The output derived are ORE frames containing Stage 2 dataset.

The following illustrations show an example for how to generate stage 2 dataset:

1. Check the dimension of the transformed datasets as shown in the following:

```
%fcc-ore
dim(tsobj)
dim(bmobj)
dim(nbdata)
```

[1] 219 8

[1] 219 8

[1] 219 23

2. Create Stage 2 dataset with the time-series transformed data, the bitmap-jump transformed data and the non-behavioral data. The following illustration is an example:

```
%fcc-ore
x <- ofsaif::addToStage2(x, tsobj, bmobj, NB[c(1:10, 12, 14, 15, 16, 90, 91, 98, 99, 125, 127, 128, 129, 130)], osot = F)
B_NB <- ofsaif::getStage2Data(x)

dim(B_NB)
```

[1] 219 33

3. Additionally, you can also define transformations with parameters that are defined by you. The following illustration set is an example:

```
%fcc-ore

#Function Skeletion
fn_name <- function( x, B , param1 , param2) {

  #Load the Transformation to be used during Production Scoring
  ofsaif::loadTransformation(x)

  #####
  # FUNCTION BODY
  #####

  #Save the Transformation to be used during Production Scoring
  ofsaif::saveTransformation(x)

  return( ORE_FRAME_OBJECT )

}
```



```

%fcc-ore

fn_user <- function( x , data ) {

  #Load the Transformation to be used during Production Scoring
  ofsaif::loadTransformation(x)

  group.var = c("CUST_INTRL_ID", "MODEL_GROUP_NAME")
  feature.name = "WIRE_TRXN_OUT_CT"

  of <- ore.groupApply(X = data[c("CUST_INTRL_ID", "MODEL_GROUP_NAME", "WIRE_TRXN_OUT_CT")],
  | INDEX = data[c("CUST_INTRL_ID", "MODEL_GROUP_NAME")],
  | FUN = mean,
  | FUN.VALUE = data.frame(CUST_INTRL_ID = "ABC",
  | MODEL_GROUP_NAME = "MG",
  | WIRE_TRXN_OUT_CT = 0))

  #Save the Transformation to be used during Production Scoring
  ofsaif::saveTransformation(x)

  return(of)
}

```

	CUST_INTRL_ID	MODEL_GROUP_NAME	WIRE_TRXN_OUT_CT
1	CUTAMLBM726	MG11	0.2203193
2	CUTAMLBM726	MG11	0.4982200
3	CUTAMLBM726	MG11	0.2055895
4	CUTAMLBM726	MG11	0.2072604
5	CUTAMLBM726	MG11	0.4714470
6	CUTAMLBM726	MG11	0.1847857

**NOTE**

- You can add any user defined transformation apart from Time-series Clustering and Jump.
- You must store the user defined transformation by calling the Function `ofsaif::saveTransformation(x)` for Production Scoring.

## 2.10 Generating Uni-Variate Analysis

Perform Uni-Variate Analysis to compute IV, which shows the amount of information carried by each variable.

1. Create a sample dataset to perform EDA on the Stage 2 dataset as shown in the following (this step is optional):

```
%fcc-ore

sampleDS = ofsaif::edaDataset(x = x, data = tsobj,
                             sample.percentage = 0.2)

dim(sampleDS)
```

[1] 396 66

2. Generate uni-variate analysis as shown in the following example:

```
%fcc-ore

v = ofsaif::edaIV(x = x, data = tsobj)
print(v)
```

	Variable	Total_Information_Value
1	MKTVAL_AM_MEAN.woe	1.014596314262
2	ACCT_ID_CORR_TRADE_CT_MEAN.woe	0.315655916603
3	TOT_EQTY_SELL_CT_MEAN.woe	0.291472690993
4	TOT_EQTY_BUY_CT_MEAN.woe	0.229530209570
5	ONE_MONTH_CMSN_AM_CLIP.woe	0.228186535954
6	TOT_BUY_TRD_CT_MEAN.woe	0.210569399179
7	ACCT_ID_CORR_TRADE_CT_CLIP.woe	0.157115039187
8	TOT_BUY_EQUITY_TRADE_CT_CLIP.woe	0.157115039187
9	MIN_ACCT_TURNOVER_PT_MEAN.woe	0.128373066612
10	BUY_AMT_CLIP.woe	0.059682918268
11	BUY_SUM_CLIP.woe	0.058839595410
12	TOT_BUY_EQUITY_TRADE_CT_MEAN.woe	0.055160200971
13	SELL_AM_CLIP.woe	0.050499123767

**NOTE** Higher the IV of the variable, better is the predicting power of the variable.

## 2.11 Building Models using OREXV Package

Stage 2 dataset prepared previously should be passed to OREXV package to perform model building. The following techniques are supported:

1. ODM NB (Naive Bayes Algorithm)
2. ODM GLM (Generalized Linear Model)
3. WOELR (Weight of Evidence Logistic Regression)
4. XGB (Xtreme Gradient Boosting)

Before you start, you have to be familiar with the characteristics of the following terms that you will use in OREXV:

- OREXV Classifiers
  - Defines the list of classifiers to be used in cross-validation.
  - Takes hyper-parameters for each of the classifiers as input parameters.
  - Along with hyper-parameters, it also takes: include, exclude and mustInclude parameters.
    - include=NULL (default): Column names that you must enter into the model or algorithm.
    - exclude=NULL (default): Column names that you must exclude (either include or exclude should be NULL).
    - mustInclude=NULL (default): Column names that you must use in the model or algorithm.

---

**NOTE** All classifiers for OREXV are part of the *oreclassifiers R* Package.

- OREXV Control Parameters
  - a. **col.na.check** - Check for the columns with NA values (T/F). Default is T.
  - b. **drop.col.na.pct** - Drop columns with percent of NA values specified. Default is 33.
  - c. **col.zero.var.check** - Check for the columns with Zero Variance (T/F). Default is T.
  - d. **find.linear.combos** - Check for the columns with perfect linear combinations (redundant variables) (T/F). Default is T.
  - e. **min.minority.obs.fold** - Minimum number of Minority class to be considered for each fold. Default is 50.
  - f. **auto.data.partition** - Enable auto Data Partition (T/F).
    - If this parameter is set to T, the subsequent four parameters will be used as reference for computing optimal parameters.
    - If this parameter is set to F, the subsequent four parameters will be used as-is.
  - g. **min.validation.data.pct** - Minimum independent hold-out validation data percent (should be between 0.1 to 0.5). Default is 0.2.
  - h. **max.cv.runs.per.model** - Maximum of Cross Validation runs per Model. Default is 10.
  - i. **max.cv.folds.per.repeat** - Maximum of Cross Validation per repeat. Default is 10.
  - j. **max.oversample.ratio** - Maximum Over Sample Ratio. Default is 10.

---

**NOTE** *max.oversample.ratio* adds an equal number of new synthetic minority observation for each existing minority observation.

Perform the following paragraph execution procedures for OREXV operations:

1. Set the classifiers control parameters as shown in the following:

```

#fcc-ore

library(orexv)
library(oreclassifiers)

#Set the Classifier objects
#onb <- ORECVodmNB()
#oglm <- ORECVodmGLM(ridge = F)
#woelr <- ORECVwoelr()
#oxgb <- ORECVxgb()

cls <- OREclassifiers(list(woelr))

```

2. Set the control parameters to run OREXV as shown in the following:

```

orecv_cntrl_param <- OREclassifierTrainCtrl( col.na.check           = T,
drop.col.na.pct         = 0.2,
col.zero.var.check     = T,
find.linear.combos     = F, #Linear combos
min.minority.obs.fold  = 5,
auto.data.partition    = T, #Auto or manual data partition
min.validation.data.pct = 0.2,
max.cv.runs.per.model  = 2,
max.cv.folds.per.repeat = 2,
max.oversample.ratio   = 0, #Oversampling
progress.update.secs   = 33
)

```

3. Create an OREXV trainer object. For example, in the following example, *cvrun* is a trainer object creation function:

```

cvr <- cvrun(models = cls,ctrl = orecv_cntrl_param, validationType="OSIT")

```

**NOTE** OSIT is the only the available options for validation type *cvrun()*..

4. Select data created in [Generating Stage 2 Dataset](#) and run OREXV model training on database server as shown in the following:

```

%fcc-ore

#Run OREXV

orecv_run_status <- ofsaif::runOnServer(x,
                                        cvr,
                                        data = df,
                                        osot.data= df_OSOT,
                                        label="SAR_FLG",
                                        id.variable = "CUST_INTRL_ID",
                                        include =NULL ,
                                        exclude = NULL
                                        mustInclude = NULL)

```

Where the input is,

- *x*: Object of class *ofsaif*.
- *cvr*: *cvr* definition setup with models and run parameters.
- *data*: OSIT Data, a select statement or an *ore.frame* with cross validation data.

- `osot.data`: OSOT Data.
- `label`: character name of the label (target) factor column.
- `id.variable`: id variable name (if any). Default is NULL.
- `include`: Columns from the input ORE frame that must be fed into the model or algorithm.
- `exclude`: Columns from the input ORE frame that must be excluded (either include or exclude should be NULL).
- `mustInclude`: Columns from the input ORE frame that must be used by the model or algorithm

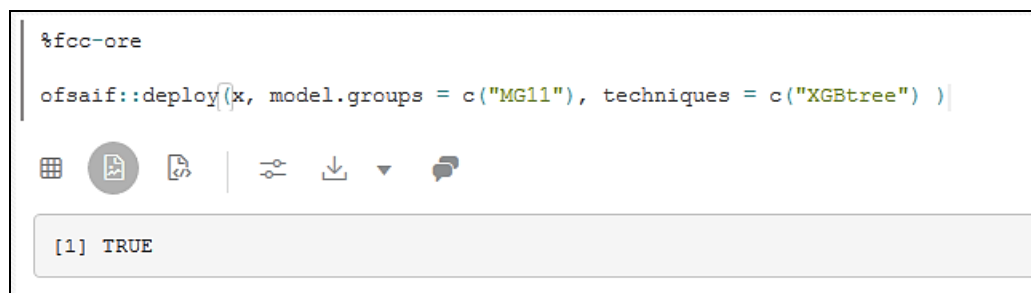
The output is `OREclassifierXtrainer` object containing the cross validation model results.

## 2.12 Deploying Models

After model evaluation, you can deploy models for each of the required model groups using the paragraph in the notebook.

Deploy the model by entering and executing the function as shown in the following:

```
%fcc-ore
ofsaif::deploy(x, model.groups = c("MG11"), techniques = c("XGBtree") )
```



[1] TRUE

The input parameters are:

- `x` - Object of class `ofsaif`
- `model.groups` - List of Model Group Name as R vector.
- `techniques` - List of Techniques as R vector (for example, `c("xgb")`).

## 2.13 Viewing List of Applied Transformations

View the list of applied transformations for the selected model groups by entering and executing the function as shown in the following:

```
%fcc-ore
ofsaif::showAppliedTransformations(x, model.groups = c("MODEL_GROUP_1", "MODEL_GROUP_2") )
```

Where the input is,

- `x` - Object of class `ofsaif`.
- `model.groups` - Names of model groups as R vector. If not passed, by default it shows for all the groups.

The transformations selected would be applied on the prediction dataset.

## 2.14 Updating the Transformations' List

Remove the transformations that are not useful from the object class ofsaif and update the list. Enter and execute in the paragraph the update transformation list function as shown in the following:

```
%fcc-ore
updateTransformationList(x,includeExclude = list( "Model_Group_1" = c(1,2,3,4), "Model_Group_2" = c(-3,-5) ) )
```

Where the input is,

- x - Object of class ofsaif
- includeExclude - List of model groups with a transformation index as shown by showAppliedTransformations() to include or exclude it from the list. Use positive index for include and negative index for exclude.
- (1, 2, 3, 4) is the output result described in section [Viewing List of Applied Transformations](#).

## 2.15 Saving the Run Definition

After deploying the model and updating the transformation list, save the run definition. This function saves the ofsaif object, which has the complete training information and will be used in predictions. The class object is stored in the ORE data store.

Enter and execute the following function in the notebook paragraph to save the run definition:

```
%fcc-ore
ofsaif:saveDefinition(x, cleanup = T)
```

Where the input is,

- x - Object of class ofsaif.
- cleanup - Boolean Flag TRUE/FALSE. If TRUE (recommended), the application cleans or deletes all the temporary tables and objects used while training.

## Send Us Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, indicate the title and part number of the documentation along with the chapter/section/page number (if available) and contact the Oracle Support.

Before sending us your comments, you might like to ensure that you have the latest version of the document wherein any of your concerns have already been addressed. You can access My Oracle Support site which has all the revised/recently released documents.

