OFS Adaptive Intelligence Foundation for Anti Money Laundering (AIF4AML)

User Guide

Release 8.0.8.0.0

Aug 2019

ORACLE
FINANCIAL SERVICES

ORACLE

OFS AIF4AMLUser Guide 8.0.8.0.0

Document Control

Version Number	Revision Date	Change Log
1.0	Aug 2019	Created the document for 8.0.8.0.0 release.
1.1	Oct 2019	• Added section Benchmarking and Evaluating OSOT Performance Matrix (Doc 30416823). • Added section Create Stage 1 Data and minor edits (Doc 30416804).

Table of Contents

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 on 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 8.0.8.0.0
  - OFS AIF4AML Administration and Configuration Guide 8.0.8.0.0

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

Acronym	Description
API	Application Programming Interface
DIM	Dimension
EDA	Exploratory Data Analysis
FCC	Financial Crime and Compliance
GLM	Generalized Linear Model
IV	Information Value
NB	Non-behavioral
NB	Naive Bayes
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
WOE	Weight of Evidence
XGBoost	Extreme Gradient Boost

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. Creating Stage 1 Data
6. Loading Datasets from AIF4AML Library Model Groups
7. Loading Behavioral and Non-behavioral Dataframes from Model Group Datasets
8. Applying Transformation on Datasets
9. Selecting NB Variables to Build Models
10. Generating Stage 2 Dataset
11. Generating Data for Exploratory Data Analysis (EDA)
12. Using Feature Selection Techniques
13. Building Models using OREXV Package
14. Using Model Explanation
15. Scoring Customer List
16. Understanding Individual Customer Score for Local Explanations
17. Using Model Evaluation
18. Benchmarking and Evaluating OSOT Performance Matrix
19. Deploying Models
20. Viewing List of Applied Transformations
21. Updating the Transformations' List
22. 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:

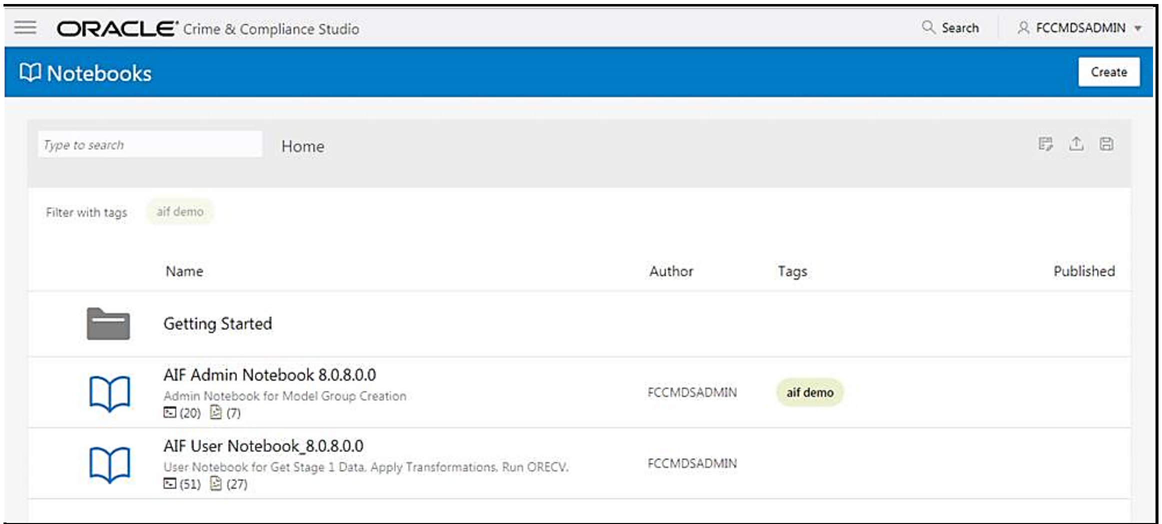
1. Enter the URL for OFS FCC Studio in a web browser to display the Login window.

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.

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.

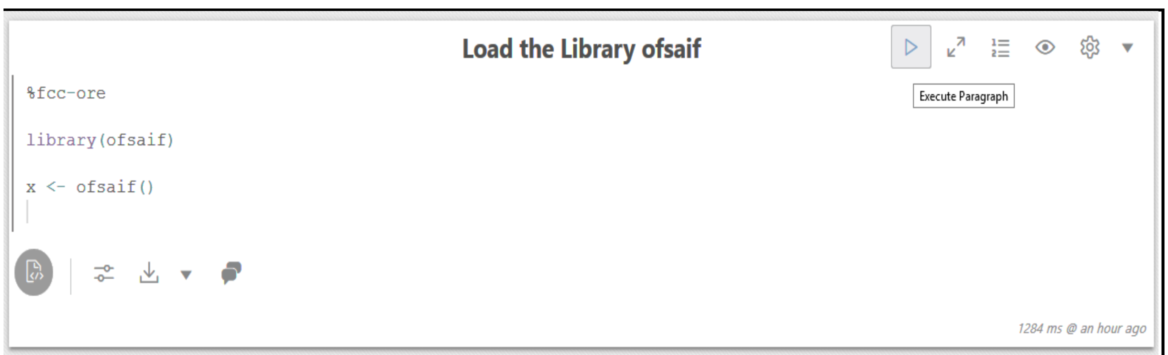


OFS FCC Studio Home Window

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.

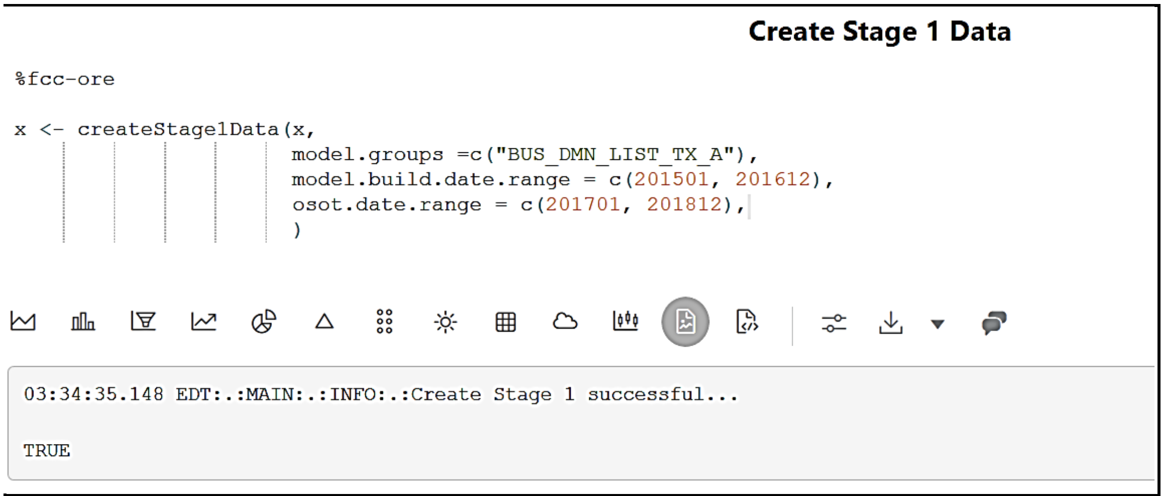
Execute the paragraph instructions to load AIF4AML Library as shown in the following illustration:



Loading AIF4AML Library

2.5 Creating Stage 1 Data

After loading the library, create Stage 1 Data. Execute the paragraph instructions as shown in the following exampe and create the data:



Creating Stage 1 Data

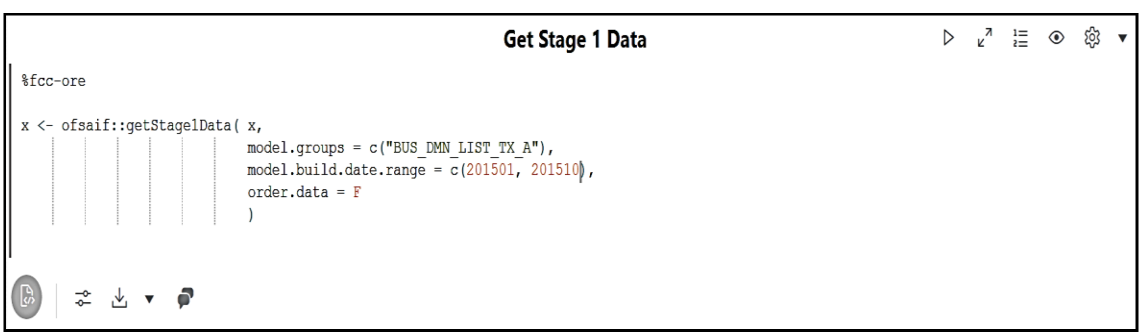
2.6 Loading Datasets from AIF4AML Library Model Groups

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, and the **getStage1** function loads the data and provides handles for the dataset.

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 Model, and the other date range for OSOT dataset; which is used to test the Model.
3. Both OSIT and OSOT - Pass two date ranges, the Model will be built on both OSOT and OSIT methods.

In the paragraph, specify the list of Model Groups for which you want to create the model and execute the paragraph as shown in the following illustrations:



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

2.7 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.

2.7.1 Viewing Dimension of Behavioral and Non-behavioral Dataframes

After loading behavioral and non-behavioral dataframes from model group datasets, you can view the 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 of Model Build, OSOT Behavioral and Non-Behavioral dataframe

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

2.8 Applying Transformation on Datasets

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

2.8.1 Applying Time-series Clustering

Stage-1 data is deep on time-series and you have to collapse to create a single observation for each group-by levels (Customer and model-group). Use time-series clustering to achieve this. The time-series function returns 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 illustration shows an example for transformation applied to time-series clustering model build 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

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  CUTAMLBM-070 BUS_DMN_LIST_TX_A                12                 10
2  CUTAMLBM-071 BUS_DMN_LIST_TX_A                12                 10
3   CUTAMLBM679 BUS_DMN_LIST_TX_A                13                 17
4   CUTAMLBM680 BUS_DMN_LIST_TX_A                 9                  3
5   CUTAMLBM681 BUS_DMN_LIST_TX_A                13                 18
6   CUTAMLBM684 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  CUTAMLBM-070 BUS_DMN_LIST_TX_A                 3                  5
2  CUTAMLBM-071 BUS_DMN_LIST_TX_A                 8                  2
3   CUTAMLBM679 BUS_DMN_LIST_TX_A                 4                  4
4   CUTAMLBM680 BUS_DMN_LIST_TX_A                 3                  5
5   CUTAMLBM681 BUS_DMN_LIST_TX_A                 4                  4
6   CUTAMLBM682 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 the 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.

2.8.2 Applying Transformation - Jump Bitmap

Stage 1 data is deep on time-series and has to be collapsed to create a single observation per Customer for each group by levels. This is achieved using jump bitmap computation. The function returns bitmap created by the following jump calculations:

1. Current month jump over most recent month (is there a recent unexpected jump?).
2. Current month jump over same month last year (is the jump because of seasonality?).
3. Current month jump over previous 12 months' historical mean (is the behavior abnormal?).

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

```
%fcc-ore

bmobj <- ofsaif::jumpBitmap(x = x,data=B, include = c("TOT_DEPST_AM", "TOT_WDRWL_AM"), threshold = c( 50, 100 ) )
```

Transformation - Jump Bitmap Model Build Data

Where input parameters are:

- x: Object of class ofsaif
- B: Behavioral Object
- include: List of features to be included for computing Jump Bitmap
- exclude: List of features to be excluded for computing Jump Bitmap
- threshold: Threshold percentage which should be considered for Jump

The output contains ORE frame with the transformed Jump Bitmap variable. The following illustration is an example:

```
%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                  ___
  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                   ___           7214.56                  ___
  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                   ___          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
```

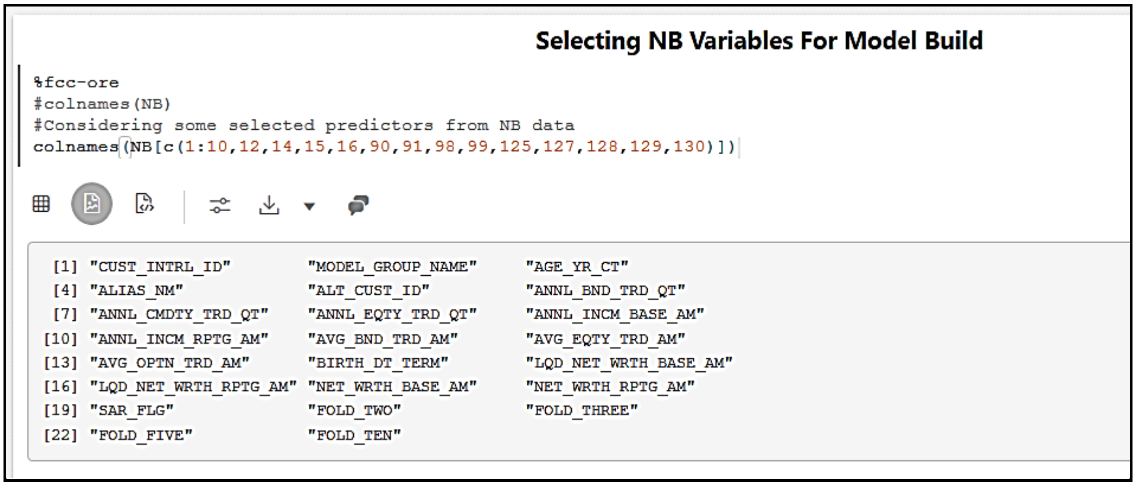
Output of Transformation - Jump Bitmap

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

2.9 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

2.10 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. Create Stage 2 dataset with the time-series transformed data, the jump bitmap 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
```

2. Additionally, you can also define transformations with parameters that are defined by you. This is part of the User Defined Transformation Function (UDTF) and you can update the Stage 2 Data with the UDTF output The following illustration set is an example:

```
%fcc-ore

#Function Skeleton
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
```

Update Stage 2 Data with UDTF output

```
%fcc-ore
B_NB <- getStage2Data( x, f4obj )

print(head( B_NB ) )
```

> **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.

2.10.1 Creating Stage 2 Dataset for OSOT Dataframe

Use this function to convert any new OSOT dataset to Stage 2. Before you call this function, prepare Stage 2 data for model build dataset for the conversion to work.

Create Stage 2 dataset for OSOT dataframe as shown in the following illustration:

```
%fcc-ore
B_NB_OSOT <- ofsaif::CreateStage2ForNewData( x, data = B_OSOT, data.nb = NB_OSOT )
```

OSOT Stage 2 Data Conversion Function Call

Where input parameters are,

- x: Object of class ofsaif
- data: Behavioural data for any new dataset (in this example, OSOT Behavioural Data)
- data.nb: Non Behavioural data for any new dataset (in this example, OSOT Non Behavioural Data)

The output returns Stage 2 converted ORE frame. The following illustration is an example of the output:

```
17:59:57.469 IST:.:MAIN:.:INFO:.: Behavioural data dimension : 530 x 209
17:59:57.486 IST:.:MAIN:.:INFO:.: Non Behavioural data dimension : 236 x 128
17:59:57.515 IST:.:MAIN:.:INFO:.: Model group do not have any user defined transformation functions...
17:59:57.516 IST:.:MAIN:.:INFO:.: Started applying transformations...
17:59:57.623 IST:.:MAIN:.:INFO:.: Current Transformation :
timeSeriesClustering.ofsaif(x = x, data = B, include = c("TOT_DEPST_AM",
    "TOT_WDRWL_AM"), bit.map.type = c("clip", "trend"), max.clusters = 20)
17:59:57.624 IST:.:MAIN:.:INFO:.:
18:00:03.240 IST:.:MAIN:.:INFO:.: Transformation Execution Complete...
 18:00:04.300 IST:.:MAIN:.:INFO:.: Transformation complete...
18:00:04.301 IST:.:MAIN:.:INFO:.: Current Transformation :
bitmapJump.ofsaif(x = x, data = B, include = c("TOT_DEPST_AM",
    "TOT_WDRWL_AM"), threshold.percentage = c(50, 80))
18:00:04.302 IST:.:MAIN:.:INFO:.:
18:00:05.965 IST:.:MAIN:.:INFO:.: Transformation Execution Complete...
 18:00:07.150 IST:.:MAIN:.:INFO:.: Transformation complete...
18:00:07.150 IST:.:MAIN:.:INFO:.: Running transformations successfull...
18:00:07.334 IST:.:MAIN:.:INFO:.:Stage 2 data created...
```

OSOT Stage 2 Data Conversion Output

2.11 Generating Data for Exploratory Data Analysis (EDA)

User provided sampling percentage is used to calculate dataset for EDA.

NOTE If you do not enter the sampling percentage, the system will calculate based on the following formula:

- 0.33 % -> If Num of Bads < 100
- 0.5 % -> If Num of Bads < 150
- 0.66 % -> If Num of Bads < 200
- 0.75 % -> If Num of Bads < (250 + 50*0.2)
- 0.8 -> Otherwise

Provide percentage values and create sampling data as shown in the following to perform EDA on the Stage 2 dataset:

```
%fcc-ore

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

dim(sampleDS)
```

```
[1] 396  66
```

Generate Data for EDA

Where input parameters are,

- x: Object of class ofsaif
- data: Transformed Object from Stage 2
- sample.percentage: Stratified Sampling on the sampling percentage passed

2.12 Using Feature Selection Techniques

Use feature selection techniques to perform various data classification operations such as splitting or ranking of variables to help create a dataset that is relevant, but much smaller in size, and build models for analysis.

The following techniques are supported in this release:

1. Feature Clustering (VARCLUS) - (requires third party package **GPArotation**)
2. Entropy (Mutual Information)
3. BKW Decouple
4. IV (Information Value)

2.12.1 Classifying Using Feature Clustering

Feature Clustering divides a set of variables into disjoint clusters. The process starts with all variables (standardized) grouped in a single cluster and then split into different clusters. The splitting process stops when the number of clusters is equal to the value specified for parameter **no.of.clusters** or the default value specified.

The default value calculation is shown in the following:

When each cluster has a total variation of at least 75% explained by its cluster component, and within each cluster of variables, one variable is chosen based on the Sqload.ratio (highest), which is computed as (1-Sqload.own_cluster)/(1-Sqload.nearest_cluster).

Select variables using Feature Clustering as shown in the following illustration:

```
%fcc-ore
vc <- ofsaif::varSelectVARCLUS( x, data = B_NB,include= NULL, exclude=NULL,label = "SAR_FLG")
print(vc)
```

Feature Clustering Function Call

Where input parameters are,

- x: Object of class ofsaif
- data: Transformed Object from Stage 2 or Non-Behavioral Data
- include: Variables to be included for Feature Selection
- exclude: Variables to be excluded for Feature Selection
- no.of.clusters: Number of Clusters to be formed
- proportion: Minimum variation explained by each of Cluster component in the Cluster
- label: Response Variable

The output consists of the following elements:

- Cluster Summary
- Cluster Members
- Cluster Loadings
- Selected Features

The following illustration is an example of the output:

```
$cluster_summary
  Total_number_of_features Total_variation Total_number_of_clusters
1                       22              22                       15
  Total_variation_explained Total_proportion_explained
1                  20.09873                  0.9135785

$cluster_members
     Cluster Members Cluster_variation Variation_explained
1   Cluster1       5                 5            4.075494
2   Cluster2       2                 2            1.514462
3   Cluster3       1                 1            1.000000
4   Cluster4       1                 1            1.000000
5   Cluster5       2                 2            1.508772
6   Cluster6       1                 1            1.000000
7   Cluster7       1                 1            1.000000
8   Cluster8       1                 1            1.000000
9   Cluster9       1                 1            1.000000
10 Cluster10       1                 1            1.000000
11 Cluster11       1                 1            1.000000
12 Cluster12       1                 1            1.000000
13 Cluster13       2                 2            2.000000
14 Cluster14       1                 1            1.000000
15 Cluster15       1                 1            1.000000
   Proportion_explained
1             0.8150988
2             0.7572308
3             1.0000000
4             1.0000000
5             0.7543861
6             1.0000000
7             1.0000000
8             1.0000000
9             1.0000000
10            1.0000000
11            1.0000000
12            1.0000000
13            1.0000000
14            1.0000000
15            1.0000000

$cluster_loadings
               Variable Cluster_id Sqload.own Sqload.near Sqload.ratio
19 TOT_WDRWL_AM_JMP_P50          1 0.99462252 0.052911992    0.0056779
15 TOT_DEPST_AM_JMP_P80          1 0.99431127 0.029982489    0.0058646
14 TOT_DEPST_AM_JMP_P50          1 0.99433317 0.037560045    0.0058880
20 TOT_WDRWL_AM_JMP_P80          1 0.99457287 0.087514389    0.0059476
16    TOT_DEPST_AM_MEAN          1 0.09765393 0.063529760    0.9635609
9             FOLD_FIVE          2 0.75723082 0.034545249    0.2514558
10             FOLD_TEN          2 0.75723082 0.054380365    0.2567303
6        AVG_BND_TRD_AM          3 1.00000000 0.020433294    0.0000000
8       AVG_OPTN_TRD_AM          4 1.00000000 0.035274128    0.0000000
21    TOT_WDRWL_AM_MEAN          5 0.75225518 0.024431217    0.2539491
18    TOT_WDRWL_AM_CLIP          5 0.75651709 0.449323464    0.4421523
7       AVG_EQTY_TRD_AM          6 1.00000000 0.035692811    0.0000000
3      ANNL_EQTY_TRD_QT          7 1.00000000 0.005409608    0.0000000
4     ANNL_INCM_BASE_AM          8 1.00000000 0.013344986    0.0000000
5     ANNL_INCM_RPTG_AM          9 1.00000000 0.030279180    0.0000000
1             AGE_YR_CT         10 1.00000000 0.035274128    0.0000000
12             FOLD_TWO         11 1.00000000 0.041933422    0.0000000
2       ANNL_BND_TRD_QT         12 1.00000000 0.041933422    0.0000000
17   TOT_DEPST_AM_TREND         13 1.00000000 0.593762286    0.0000000
22   TOT_WDRWL_AM_TREND         13 1.00000000 0.430662612    0.0000000
11           FOLD_THREE         14 1.00000000 0.014800200    0.0000000
13    TOT_DEPST_AM_CLIP         15 1.00000000 0.962436120    0.0000000

$selected.features
 [1] "TOT_WDRWL_AM_JMP_P50" "FOLD_FIVE"            "AVG_BND_TRD_AM"
 [4] "AVG_OPTN_TRD_AM"      "TOT_WDRWL_AM_MEAN"    "AVG_EQTY_TRD_AM"
 [7] "ANNL_EQTY_TRD_QT"     "ANNL_INCM_BASE_AM"    "ANNL_INCM_RPTG_AM"
[10] "AGE_YR_CT"            "FOLD_TWO"             "ANNL_BND_TRD_QT"
[13] "TOT_DEPST_AM_TREND"   "FOLD_THREE"           "TOT_DEPST_AM_CLIP"
```

Feature Clustering Output

2.12.2 Classifying Using Mutual Information (Entropy)

Mutual Information (Entropy) variable selection is a measure of the amount of information that a variable has about another variable. In this technique, the variables give mutual information that each feature has about the response variable (label).

NOTE	
	• Binning in case of numerical variables is done using woe from ofswoelr. • Higher the value of Mutual Information, greater is the information contained by that variable about the response variable.

Select variables using Mutual Information as shown in the following illustration:

```
%fcc-ore

en <- ofsaif::varSelectEntropy( x, data = B_NB,include= NULL, exclude=NULL,label = "SAR_FLG")
print(en)
```

Mutual Information (Entropy) Function Call

Where input parameters are,

- x: Object of class ofsaif
- data: Transformed Object from Stage 2 or Non-Behavioral Data
- include: Variables to be included for Feature Selection
- exclude: Variables to be excluded for Feature Selection
- label: Response Variable

The output consists of mutual information for each Predictor as shown in the following example:

	Predictor	Mutual_Information
1	TOT_DEPST_AM_CLIP	0.502483
2	TOT_WDRWL_AM_CLIP	0.456717
3	TOT_DEPST_AM_TREND	0.449706
4	TOT_WDRWL_AM_TREND	0.407128
5	TOT_DEPST_AM_MEAN	0.153832
6	TOT_WDRWL_AM_MEAN	0.124734
7	AVG_EQTY_TRD_AM	0.097393
8	ANNL_INCM_BASE_AM	0.095246
9	ANNL_INCM_RPTG_AM	0.083734
10	ANNL_BND_TRD_QT	0.067003
11	ANNL_EQTY_TRD_QT	0.064087
12	AVG_BND_TRD_AM	0.063619
13	AVG_OPTN_TRD_AM	0.046780
14	FOLD_TEN	0.015955
15	TOT_WDRWL_AM_JMP_P80	0.014953
16	AGE_YR_CT	0.010523
17	TOT_WDRWL_AM_JMP_P50	0.007038
18	FOLD_FIVE	0.006442
19	TOT_DEPST_AM_JMP_P80	0.002003
20	FOLD_THREE	0.001519
21	TOT_DEPST_AM_JMP_P50	0.000751
22	FOLD_TWO	0.000000

Mutual Information (Entropy) Output

2.12.3 Classifying Using BKW Decouple

BKW Decouple variable selection uses iterative calculation of Generalized Variation Inflation Factor (GVIF) for the input features until none of the features exceeds the threshold provided.

Select variables using BKW Decouple as shown in the following illustration:

```
%fcc-ore

bkw <- ofsaif::varSelectBKW(x = x, data= ore.pull(B_NB[, c(1:23)]),include= NULL, exclude=NULL,vif.thresh=2.5,label = "SAR_FLG", debug.aif = T)
print(bkw)
```

BKW Decouple Function Call

Where input parameters are,

- x: Object of class ofsaif
- data: Transformed Object from Stage 2 or Non-Behavioral Data
- include: Variables to be included for Feature Selection
- exclude: Variables to be excluded for Feature Selection
- vif.thresh: Threshold value to be considered for GVIF (2.5 by default)
- label: Response Variable

The output consists of GVIF for each of the Predictor as shown in the following example:

	Variable	GVIF
1	TOT_DEPST_AM_MEAN	1.814178
2	TOT_WDRWL_AM_MEAN	1.391659
3	AVG_EQTY_TRD_AM	1.285693
4	ANNL_EQTY_TRD_QT	1.236755
5	ANNL_BND_TRD_QT	1.232449
6	AVG_BND_TRD_AM	1.221112
7	AVG_OPTN_TRD_AM	1.174283
8	TOT_DEPST_AM_JMP_P50	1.158265
9	TOT_WDRWL_AM_TREND	1.083110

BKW Decouple Output

2.12.4 Classifying Using Information Value (IV)

Information Value (IV) variable selection computes the IV for each variable and ranks variables on the basis of their importance.

Select variables using IV as shown in the following illustration:

```
%fcc-ore

iv <- ofsaif::varSelectIV( x, data = B_NB,include= NULL, exclude=NULL,iv.thresh=0.1,label = "SAR_FLG")
print(iv)
```

IV Function Call

Where input parameters are,

- x: Object of class ofsaif
- data: Transformed Object from Stage 2 or Non-Behavioral Data
- include: Variables to be included for Feature Selection
- exclude: Variables to be excluded for Feature Selection
- iv.thresh: Threshold value to be considered for IV (0.1 by default)
- label: Response Variable

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

The output consists of IV for each of the Predictor as shown in the following example:

```
                        Variable Total_Information_Value
1       MAX_ATM_TRXN_OUT_AM_MEAN.woe          0.36107263
2                AVG_EQTY_TRD_AM.woe          0.33779940
3                 AVG_BND_TRD_AM.woe          0.31650385
4                ANNL_BND_TRD_QT.woe          0.31020003
5       MAX_CASH_TRXN_IN_AM_MEAN.woe          0.30852764
6           ATM_TRXN_OUT_AM_MEAN.woe          0.29139715
7              ANNL_INCM_BASE_AM.woe          0.29039130
8        MAX_TOT_WDRWL_AM_MEAN.woe            0.28103371
9              ANNL_INCM_RPTG_AM.woe          0.25463284
10         CASH_TRXN_IN_AM_MEAN.woe           0.24754352
11              ANNL_EQTY_TRD_QT.woe          0.21956408
12            TOT_WDRWL_AM_MEAN.woe           0.21849547
13      MAX_ATM_TRXN_IN_AM_MEAN.woe           0.20432563
14       ATM_TRXN_IN_AM_JMP_P50.woe           0.15253415
15   MAX_ATM_TRXN_IN_AM_JMP_P50.woe           0.15253415
16    MAX_TOT_WDRWL_AM_JMP_P100.woe           0.14571390
17           ATM_TRXN_IN_AM_MEAN.woe          0.13444270
18        TOT_WDRWL_AM_JMP_P100.woe           0.11665341
19      CASH_TRXN_IN_AM_JMP_P50.woe           0.11040397
20  MAX_CASH_TRXN_IN_AM_JMP_P50.woe           0.11040397
21     CASH_TRXN_IN_AM_JMP_P100.woe           0.07358539
22 MAX_CASH_TRXN_IN_AM_JMP_P100.woe           0.07118435
23      ATM_TRXN_IN_AM_JMP_P100.woe           0.06602053
24         TOT_WDRWL_AM_JMP_P50.woe           0.06521120
25      ATM_TRXN_OUT_AM_JMP_P50.woe           0.06474072
26  MAX_ATM_TRXN_OUT_AM_JMP_P50.woe           0.06474072
27  MAX_ATM_TRXN_IN_AM_JMP_P100.woe           0.06368625
28      MAX_TOT_WDRWL_AM_JMP_P50.woe          0.06307546
29     ATM_TRXN_OUT_AM_JMP_P100.woe           0.06284014
30 MAX_ATM_TRXN_OUT_AM_JMP_P100.woe           0.06284014
31                     AGE_YR_CT.woe          0.02932485
```

IV Output

2.13 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. **auto.feature.selection -** Enable or disable Auto-selection of variables. If set to T (TRUE) OREXV automatically decides the features that are good to be considered, which is dependent on the feature selection technique. Default is F (FALSE).
  h. **min.validation.data.pct** - Minimum independent hold-out validation data percent (should be between 0.1 and 0.5). Default is 0.2.
  i. **max.cv.runs.per.model** - Maximum of Cross Validation runs per Model. Default is 10.
  j. **max.cv.folds.per.repeat** - Maximum of Cross Validation per repeat. Default is 10.
  k. **data.randomsorted -** Random shuffling of data. Default is T.
  l. **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 =T)
owoelr <- ORECVwoelr()
oxgb <- ORECVxgb()

cls <- OREclassifiers(list(owoelr,onb,oglm,oxgb))
```

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    = 2, #Oversampling
                                             auto.feature.selection  = T, #Auto feature selection
                                             data.randomsorted       = T,
                                             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 The following options are available for validation type cvrun():

1. OSIT
2. OSOT
3. BOTH (includes OSIT and OSOT)

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

```
#Run OREXV
library(orexv)
library(oreclassifiers)

orecv_run_status <- ofsaif::runOnServer(x,
cvr = cvr,
data = B_NB,
```

```
osot.data= B_NB_OSOT,
                 feature.select.method = "iv",feature.select.params =
        list(bin.method="auto",bin.brks=20,iv.thresh=0.5),
        #        feature.select.method = "BKW",feature.select.params =
        list(vif.thresh=2.5),
        #        feature.select.method = "VARCLUS",feature.select.params
        = list(no.of.clusters=NULL,proportion=0.75),
        #        feature.select.params =
        list(bin.method="auto",bin.brks=20,iv.thresh=0.5),
label="SAR_FLG",
id.variable = "CUST_INTRL_ID",
include =NULL ,
exclude = NULL )
```

NOTE For details about use of *runOnServer* and to view outputs, refer to the R Man pages for runonserver using the command **?runonserver**.

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

2.14 Using Model Explanation

Model explanations add additional criteria for model selection, which are important. When you compare models and interpret it, working with model-agnostic explanations is easy because the same method can be used for any type of model.

The following techniques are used in Model Explanation:

- GLM Explanation
- XGBoost Model Explanation
- WOE Model Explanation
- NB Model Explanation

Where input parameters are,

- model.group.name: Model group name for which the explanation is required
- label: Response Variable
- technique: Model Techniques - ODMglm, ODMnb, xgboost, and OFSwoelr
- featImp.method: Type of method for feature importance - “permimportance”, and “shapimportance”
- plot.type: Plot theme - “ofs”, by default, using standard R plots. Supports ggplot2 as well, if ggplot2 and gridExtra are installed

- num.top.featImp: Number of most important featured to be selected for the following techniques - Feature Contribution, Model Response (ICE), and Sensitivity Analysis (OFAT)
- num.bottom.featImp: Number of least important featured to be selected
- feature.list: List of features for which Feature Contribution, Feature Sensitivity(OFAT), and Model Response is required. By default, it is set to NULL, which means that the feature list will be taken from num.top.featImp and num.bottom.featImp
- on.server: Execution on server side (T/F)
- serialize.plot: Serialize the plot (T/F). Set to TRUE for execution on server side and return the plots from Studio
- explain.customer.score: Optional Boolean flag to explain individual Customer Score (T/F). By default, set to False.
- customer.id: Customer id for which local explanation is required

The output is that plots for each technique is rendered in the application.

2.14.1 Creating GLM Explanation

Enter the function call details for GLM Explanation as shown in the following illustration:

```
%fcc-ore
fn_aif_model_explanation(model.group.name = "BUS_DMN_LIST_TX_A",label = "SAR_FLG",
                         technique = "ODMglm",
                         featImp.method = "permimportance",
                         plot.type = c("ofs"),
                         num.top.featImp = 3,
                         num.bottom.featImp = 1,
                         on.server = F,
                         serialize.plot = F)
```

GLM Explanation Function Call

The output appears as shown in the following example:

```
Starting model explanations (default)
Starting: Importance (pfi). 9 features, 40 obs, 5 permutations
    Average pfi time per feature: 1.05 secs. Estimated completion in +0.14 mins. Completed feature 1 of 9
    Average pfi time per feature: 0.93 secs. Estimated completion in +0.06 mins. Completed feature 5 of 9
PERM imp Completed. Elapsed: 0.15 mins
Starting: Feature contribution (shap_s2). 4 features, 40 obs, 5 permutations
    Average SHAP_S time per feature: 3.57 secs. Estimated completion in +0.18 mins. Completed feature 1 of 4
Feature Contribution Completed. Elapsed: 0.26 mins
Starting: PDP compute. 4 features, 40 pdp obs, 20 ice obs
    Average PDP time per feature: 1.17 secs. Estimated completion in +0.06 mins. Completed feature 1 of 4
PDP Completed. Elapsed: 0.11 mins
Starting: Sensitivity. 4 features, 3 local obs
SENS Completed. Elapsed: 0.12 mins
Starting: Feature contribution (shap_s2). 4 features, 3 obs, 5 permutations
    Average SHAP_S time per feature: 4.60 secs. Estimated completion in +0.23 mins. Completed feature 1 of 4
Feature Contribution Completed. Elapsed: 0.30 mins
03:27:43.954 EDT:.:MAIN:.:INFO:.: Transformation Execution Complete...
```

GLM Explanation Output

2.14.2 Creating XGBoost Model Explanation

Enter the function call details for GLM Explanation as shown in the following illustration:

```
%fcc-ore
library(xgboost)
fn_aif_model_explanation(model.group.name = "BUS_DMN_LIST_TX_A",
                         label = "SAR_FLG",
                         technique = "XGBtree",
                         featImp.method = "permimportance",
                         plot.type = c("ofs"),
                         num.top.featImp = 2,
                         num.bottom.featImp = 2,
                         on.server = F,
                         serialize.plot = F,
                         seed = 123)
```

XGBoost Model Explanation Function Call

The output appears as shown in the following example:

```
Starting model explanations (default)
Starting: Importance (pfi). 23 features, 33 obs, 5 permutations
    Average pfi time per feature: 0.14 secs. Estimated completion in +0.05 mins. Completed feature 1 of 23
    Average pfi time per feature: 0.09 secs. Estimated completion in +0.03 mins. Completed feature 5 of 23
    Average pfi time per feature: 0.08 secs. Estimated completion in +0.02 mins. Completed feature 10 of 23
    Average pfi time per feature: 0.08 secs. Estimated completion in +0.01 mins. Completed feature 15 of 23
    Average pfi time per feature: 0.07 secs. Estimated completion in +0.00 mins. Completed feature 20 of 23
PERM imp Completed. Elapsed: 0.02 mins
Starting: Feature contribution (shap_s2). 4 features, 33 obs, 5 permutations
    Average SHAP_S time per feature: 0.12 secs. Estimated completion in +0.01 mins. Completed feature 1 of 4
Feature Contribution Completed. Elapsed: 0.01 mins
Starting: PDP compute. 4 features, 33 pdp obs, 20 ice obs
    Average PDP time per feature: 0.03 secs. Estimated completion in +0.00 mins. Completed feature 1 of 4
PDP Completed. Elapsed: 0.01 mins
Starting: Sensitivity. 4 features, 3 local obs
SENS Completed. Elapsed: 0.01 mins
Starting: Feature contribution (shap_s2). 4 features, 3 obs, 5 permutations
    Average SHAP_S time per feature: 0.24 secs. Estimated completion in +0.01 mins. Completed feature 1 of 4
Feature Contribution Completed. Elapsed: 0.02 mins
```

XGBoost Model Explanation Output

2.14.3 Creating WOE Model Explanation

Enter the function call details for GLM Explanation as shown in the following illustration:

```
%fcc-ore
fn_aif_model_explanation(model.group.name = "BUS_DMN_LIST_TX_A",
                         label = "SAR_FLG",
                         technique = "OFSwoelr",
                         featImp.method = "permimportance",
                         plot.type = c("ofs"),
                         num.top.featImp = 0,
                         num.bottom.featImp = 2,
                         on.server = F,
                         serialize.plot = F)
```

WOE Model Explanation Function Call

The output appears as shown in the following example:

```
Starting model explanations (default)
Starting: Importance (pfi). 5 features, 33 obs, 5 permutations
    Average pfi time per feature: 0.09 secs. Estimated completion in +0.01 mins. Completed feature 1 of 5
    Average pfi time per feature: 0.08 secs. Estimated completion in +0.00 mins. Completed feature 5 of 5
PERM imp Completed. Elapsed: 0.01 mins
Starting: Feature contribution (shap_s2). 2 features, 33 obs, 5 permutations
    Average SHAP_S time per feature: 0.15 secs. Estimated completion in +0.00 mins. Completed feature 1 of 2
Feature Contribution Completed. Elapsed: 0.01 mins
Starting: PDP compute. 2 features, 33 pdp obs, 20 ice obs
    Average PDP time per feature: 0.32 secs. Estimated completion in +0.01 mins. Completed feature 1 of 2
PDP Completed. Elapsed: 0.01 mins
Starting: Sensitivity. 2 features, 3 local obs
SENS Completed. Elapsed: 0.01 mins
Starting: Feature contribution (shap_s2). 2 features, 3 obs, 5 permutations
    Average SHAP_S time per feature: 0.14 secs. Estimated completion in +0.00 mins. Completed feature 1 of 2
Feature Contribution Completed. Elapsed: 0.01 mins
```

WOE Model Explanation Output

2.14.4 Creating NB Model Explanation

Enter the function call details for GLM Explanation as shown in the following illustration:

```
%fcc-ore
fn_aif_model_explanation(model.group.name = "BUS_DMN_LIST_TX_A",
                         label = "SAR_FLG",
                         technique = "ODMnb",
                         featImp.method = "permimportance",
                         plot.type = c("ofs"),
                         num.top.featImp = 1,
                         num.bottom.featImp = 0,
                         on.server = F,
                         serialize.plot = F)
```

NB Model Explanation Function Call

The output appears as shown in the following example:

```
Starting model explanations (default)
Starting: Importance (pfi). 9 features, 40 obs, 5 permutations
    Average pfi time per feature: 2.45 secs. Estimated completion in +0.33 mins. Completed feature 1 of 9
    Average pfi time per feature: 1.57 secs. Estimated completion in +0.10 mins. Completed feature 5 of 9
PERM imp Completed. Elapsed: 0.22 mins
Starting: Feature contribution (shap_s2). 1 features, 40 obs, 5 permutations
    Average SHAP_S time per feature: 4.60 secs. Estimated completion in +0.00 mins. Completed feature 1 of 1
Feature Contribution Completed. Elapsed: 0.08 mins
Starting: PDP compute. 1 features, 40 pdp obs, 20 ice obs
    Average PDP time per feature: 2.40 secs. Estimated completion in +0.00 mins. Completed feature 1 of 1
PDP Completed. Elapsed: 0.04 mins
Starting: Sensitivity. 1 features, 3 local obs
SENS Completed. Elapsed: 0.04 mins
Starting: Feature contribution (shap_s2). 1 features, 3 obs, 5 permutations
    Average SHAP_S time per feature: 5.89 secs. Estimated completion in +0.00 mins. Completed feature 1 of 1
Feature Contribution Completed. Elapsed: 0.10 mins
05:21:43.151 EDT:.:MAIN:.:INFO:.: Transformation Execution Complete...
```

NB Model Explanation Output

2.15 Scoring Customer List

Get a list of customers that you can use to score. The function call is **ofsaif::getCustomerListForScoring**.

Enter the function call details as shown in the following illustration:

```
%fcc-ore
getCustomerListForScoring(model.group.name = "BUS_DMN_LIST_TX_A")
```

Get Customer List for Scoring Function Call

Where the input parameter is: **model.group.name** (Model Group Name for which the list of Customers is required for Local Explanations).

The output displays a list of customers as shown in the following example:

```
 [1] CUTAMLBM684     CUTAMLBM685     CUTAMLBM689     CUTAMLBM708
 [5] CUTAMLBM711     CUTAMLBM713     CUTAMLBM714     CUTAMLBM717
 [9] CUTAMLBM722     CUTAMLBM724     CUTAMLBM730     CUTAMLBM736
[13] CUTAMLBM756     CUTAMLBM757     CUTAMLBM767     CUTAMLBM771
[17] CUTAMLBM774     CUTAMLBM776     CUTAMLBM779     CUTAMLBM780
[21] CUTAMLBM782     CUTAMLBM786     CUTAMLBM787     CUTAMLBM790
[25] CUTAMLBM822     CUTAMLBM825     CUTAMLBM831     CUTAMLBM838
[29] CUTAMLBM841     CUTAMLBM848     CUTAMLBM855     CUTAMLBM863
[33] CUTAMLBM872     CUTAMLBM874     CUTAMLBM883     CUTAMLBM887
[37] CUTAMLBM894     CUTAMLBM897     CUTAMLBM898     OSOTCUTAMLBM899
40 Levels: CUTAMLBM684 CUTAMLBM685 CUTAMLBM689 CUTAMLBM708 ... OSOTCUTAMLBM899
```

Get Customer List for Scoring Function Output

2.16 Understanding Individual Customer Score for Local Explanations

Local Explanation for a Customer score renders plots for two techniques:

- Prediction Attribution
- Sensitivity Analysis (OFAT)

Select a Customer ID from the list of customers output in Scoring Customer List and enter the function calls as shown in the following illustration:

```
%fcc-ore
#Local Explanation for Individual Customer score will give the plots for two techniques: Prediction Attribution and Sensitivity Analysis (OFAT).
fn_aif_model_explanation(model.group.name = "BUS_DMN_LIST_TX_A",
                         label = "SAR_FLG",
                         technique = "ODMnb",
                         featImp.method = "permimportance",
                         plot.type = c("ofs"),
                         num.top.featImp = 2,
                         num.bottom.featImp = 1,
                         on.server = F,
                         serialize.plot = F,
                         explain.customer.score = T,
                         customer.id = "CUTAMLBM684")
```

Individual Customer Score for Local Explanations Function Call

```
Starting: Feature contribution (shap_s2). 9 features, 1 obs, 5 permutations
    Average SHAP_S time per feature: 5.68 secs. Estimated completion in +0.76 mins. Completed feature 1 of 9
    Average SHAP_S time per feature: 5.69 secs. Estimated completion in +0.38 mins. Completed feature 5 of 9
Feature Contribution Completed. Elapsed: 0.86 mins
Starting: Importance (pfi). 9 features, 40 obs, 5 permutations
    Average pfi time per feature: 1.79 secs. Estimated completion in +0.24 mins. Completed feature 1 of 9
    Average pfi time per feature: 1.79 secs. Estimated completion in +0.12 mins. Completed feature 5 of 9
PERM imp Completed. Elapsed: 0.28 mins
Starting: Sensitivity. 3 features, 1 local obs
SENS Completed. Elapsed: 0.16 mins
            CUST_INTRL_ID PREDICTION  TILE
CUTAMLBM684   CUTAMLBM684          1 High3
05:06:43.504 EDT:.:MAIN:.:INFO:.: Transformation Execution Complete...
```

Individual Customer Score for Local Explanations Output

2.17 Using Model Evaluation

Evaluate Models using plots rendered and compare values (high, medium, and low) for the various techniques (GLM, XGBoost, WOE, and NB). The following plots are available:

- Prediction Decile Plot
- Confusion Matrix Plot at different Cut-offs - F1 Score, Precision-Recall, and Kappa

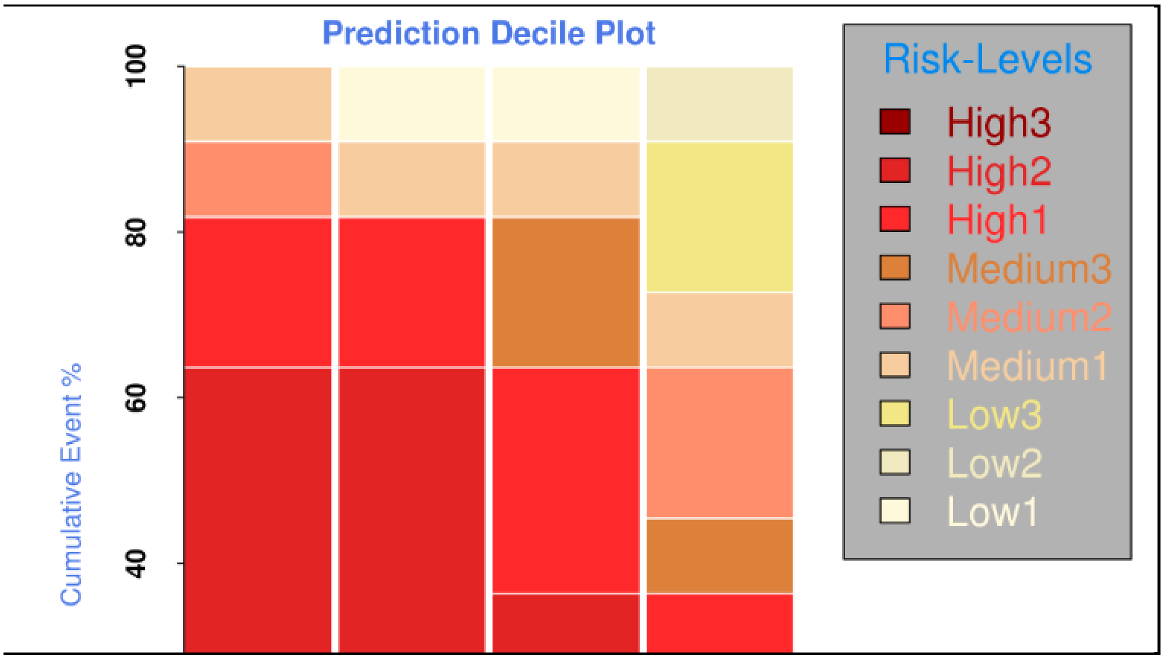
2.17.1 Evaluating Prediction Decile Plot

Enter function calls in the paragraph as shown in the following illustration to render Prediction Decile Plots and evaluate the Models:

```
%fcc-ore

#Decile plot for all the techniques used in the ORECV run
orexv::modelComparePlots(x =ofsaif::getOrecvRunID("BUS_DMN_LIST_TX_A"),plot.type = "decile",as.png.base64=F, theme="ofs", plot_style = "stacked")
orexv::modelComparePlots(x =ofsaif::getOrecvRunID("BUS_DMN_LIST_TX_A"),plot.type = "decile",as.png.base64=F, theme="ofs", plot_style = "grouped")
orexv::modelComparePlots(x =ofsaif::getOrecvRunID("BUS_DMN_LIST_TX_A"),plot.type = "decile",as.png.base64=F, theme="ofs", plot_style = "line")
```

Evaluating Prediction Decile Plot Function Call

Plots are rendered as shown in the following example:



Evaluating Prediction Decile Plot

model	High3	High2	High1	Medium3	Medium2	Medium1	Low3	Low2	Low1
XGBtree	0.9176	0.8287	0.3169	0.2355	0.1855000	0.1529000	0.0976300	0.0568200	0.0529500
OFSwoelr	0.9956	0.9774	0.2262	0.1327	0.0229900	0.0008404	0.0004588	0.0000679	0.0000020
ODMnb	0.9985	0.9985	0.9985	0.2853	0.0002486	0.0002486	0.0002486	0.0002486	0.0002486
ODMglm	0.5081	0.4158	0.3799	0.3420	0.3056000	0.2851000	0.2643000	0.2244000	0.1756000

Evaluating Prediction Decile Plot Model - Techniques Comparison

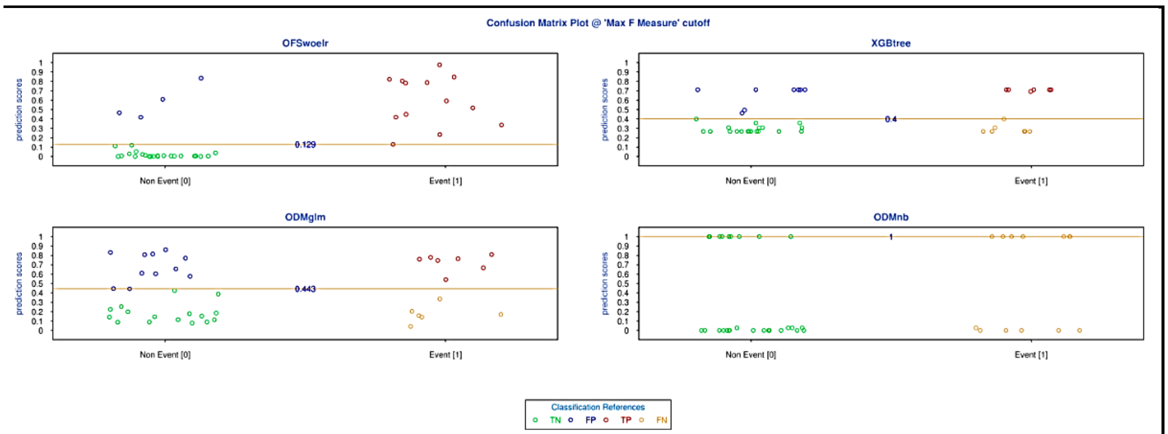
2.17.2 Evaluating Matrix Plot at different Cut-offs - F1 Score, Precision-Recall, and Kappa

Enter function calls in the paragraph as shown in the following illustration to render Matrix Plot at different Cut-offs and evaluate the Models:

```
%fcc-ore

#F1 - Score
orexv::modelComparePlots( ofsaif::getOrecvRunID("BUS_DMN_LIST_TX_A"),"cm",as.png.base64=F, theme="ofs", cutoff_method = "F1")

#Precision-Recall
orexv::modelComparePlots( ofsaif::getOrecvRunID("BUS_DMN_LIST_TX_A"),"cm",as.png.base64=F, theme="ofs", cutoff_method = "PR")

#Kappa
orexv::modelComparePlots( ofsaif::getOrecvRunID("BUS_DMN_LIST_TX_A"),"cm",as.png.base64=F, theme="ofs", cutoff_method = "KA")
```

Evaluating Matrix Plot at different Cut-offs - F1 Score, Precision-Recall, and Kappa Function Call

Plots are rendered as shown in the following example:



Evaluating Matrix Plot at different Cut-offs - F1 Score, Precision-Recall, and Kappa Plot

model	FP	TN	TP	FN	Precison	Recall	F1	Kappa
OFSwoelr	4	23	13	0	0.7647059	1.0000000	0.8666667	0.789
XGBtree	8	19	6	7	0.4285714	0.4615385	0.4444444	0.162
ODMglm	11	16	7	6	0.3888889	0.5384615	0.4516129	0.119
ODMnb	0	27	0	13	NaN	0.0000000	NaN	0.000

Evaluating Matrix Plot at different Cut-offs - F1 Score, Precision-Recall, and Kappa - Techniques Comparison

2.18 Benchmarking and Evaluating OSOT Performance Matrix

Benchmark and evaluate OSOT performance in the Models created by using the details in the following table:

Test	Description	Expected Result
AUC	Area under the ROC Curve - a test for ranking power of a score system. It is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. ROC curve is created by plotting the true positive rate (Sensitivity) against the false positive rate (1-Specificity) at various threshold settings. Sensitivity or True Positive rate measures the proportion of actual positives (in other words, having a condition, such as BAD), which are correctly identified. Specificity measures the proportion of negatives, which are correctly identified (for example, GOOD).	• Random model: **0.5** • Perfect model: **1.0** • Acceptable: **0.7<AUC<0.8** • Excellent: **0.8<AUC<0.9** • Exceptional: **0.9<AUC<1**
KS test	Kolmogorov-Smirnov (KS) Test draws a cumulative BAD distribution curve (BADs in deciles <=n/total BADs) and a cumulative GOOD distribution curve (GOODs in deciles <= n/total GOODs) against the descending ordered scores. The max vertical distance between the two curves is checked. **Observations:** If the score is sorted in descending order (smaller the value on the x-axis, higher the score, as in AIF4AML), then KS figure gives a sensitivity curve and a (1- specificity) curve for each of the sorted scores or ranks. If the score is sorted in ascending order (smaller the value on the x-axis, less is the score), then KS figure gives a specificity curve and a (1- sensitivity) curve for each of the sorted scores or ranks.	Expect most Bad to be concentrated on the first or second Decile.
Rank Ordering	Actual BAD=1 rate in each decile and the average of the risk score in each decile is computed. The average risk score in each decile is in the descending order as the data is already sorted in the descending order. The actual BAD=1 rate in each decile is computed and if found in the descending order the model is said to be rank ordered.	Confirm rank order.
Lorenz Curve	Lorenz Curve is drawn between the cumulative percentage of BAD accounts ((BADs in deciles <=n/total BADs) and the cumulative percentage of ALL accounts.	The curve should reach the ceiling fairly early.

Test	Description	Expected Result
Cumulative Lift Curve	Lift= (Percentage of the BAD=1 in the deciles)/ (Percentage of BAD=1 in the whole population of all the accounts) For a decile, if the lift is more than 1, then the BAD=1 ratio in the deciles is above BAD=1 ratio in the whole population.	Interested in **lift>1** deciles.
Population stability index (PSI)	Compute observed and expected BAD rates for all the deciles, and then calculate PSI as shown in the following equations: PSI: $\sum(Observed - Expected) * \ln(\frac{Observed}{Expected})$	• No significant shift: **<0.1** • Minor Shift: **0.1-0.25** • Significant shift: **>0.25**

2.19 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
```

> **NOTE** For details, prefix the function name with **?** and access the R Man Pages. For example, **?deploy**.

The deployment makes an entry in the table AIF_DEPLOYED_MODEL_GROUPS.

2.20 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" ) )
```

NOTE For details, prefix the function name with **?** and access the R Man Pages. For example, **?deploy**.

The transformations selected would be applied on the prediction dataset.

2.21 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) ) )
```

NOTE For details, prefix the function name with **?** and access the R Man Pages. For example, **?deploy**.

2.22 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)
```

NOTE For details, prefix the function name with **?** and access the R Man Pages. For example, **?deploy**.

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.