

# Package ‘IMP’

July 21, 2025

**Type** Package

**Title** Interactive Model Performance Evaluation

**Version** 1.1

**Date** 2016-1-29

**Description** Contains functions for evaluating & comparing the performance of Binary classification models. Functions can be called either statically or interactively (as Shiny Apps).

**License** GPL

**LazyData** TRUE

**Imports** dplyr, ggplot2, stats, tidyr, shiny

**RoxygenNote** 5.0.1

**URL** <https://github.com/anup50695/IMPPackage>

**NeedsCompilation** no

**Author** Anup Nair [aut, cre]

**Maintainer** Anup Nair <nairanup50695@gmail.com>

**Repository** CRAN

**Date/Publication** 2016-01-29 22:46:46

## Contents

interConfMatrix . . . . .	2
interPerfMeasures . . . . .	3
staticConfMatrix . . . . .	5
staticPerfMeasures . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

interConfMatrix	<i>Interactive confusion matrix</i>
-----------------	-------------------------------------

---

### Description

Interactive version of the staticConfMat function

### Usage

```
interConfMatrix(list_models, model_function = NULL, data = NULL, y = NULL)
```

### Arguments

<code>list_models</code>	A list of one (or more) dataframes for each model whose performance is to be evaluated. Each dataframe should comprise of 2 columns with the first column indicating the class labels (0 or 1) and the second column providing the raw predicted probabilities
<code>model_function</code>	Models can be created interactively, if required. For this option to work, a model function should be passed as an argument. The model function should take a formula as an argument, and return a dataframe as output (dataframe should comprise of 2 columns with the first column indicating the class labels (0 or 1) and the second column providing the raw predicted probabilities) Please refer to the example section for more details
<code>data</code>	The name of the data-set. The Independent Variable (IV) names, for interactive model building, is picked up from this data set
<code>y</code>	The column name of the Dependent Variable (DV), for interactive model building

### Value

This function will launch a ShinyApp. Input parameters (such as the probability threshold, the "t" argument in the static version of this function) can be adjusted through app widgets. The 'Run-Analysis' button in the app, will generate model performance output based on selected input parameters

For interactive Model building, a model function, data set & the dependent variable name should be passed as arguments. Interactive model building option creates additional input widgets in the app. This includes -

A drop down to select independent variables (the names of the variables will be picked up from the data argument)

An input slider to include additional models (upto 4 additional models can be created). Each additional model updates the original model created. For e.g. consider the dataset has 10 IVs: x1-x10. Original model was created by selecting x1-x4 from the drop down list. If we need to create a second model, by including x5 and excluding x3 simply type, "+ x5 - x3" in the input text box

## Examples

```
# Without interactive model development
model_1 <- glm(Species ~ Sepal.Length,data=iris,family=binomial)
model_2 <- glm(Species ~ Sepal.Width, data=iris, family = binomial)
df1 <- data.frame(model_1$y,fitted(model_1))
df2 <- data.frame(model_2$y,fitted(model_2))
## Not run:
#This will launch a Shiny App
interConfMatrix(list_models = list(df1,df2))
## End(Not run)

# With interactive model development
glm_model <- function(formula) {
  glm_model <- glm(formula, data = iris, family = "binomial")
  out <- data.frame(glm_model$y, fitted(glm_model))
  out }
## Not run:
#This will launch a Shiny App
interConfMatrix(model_function=glm_model,data=iris,y="Species")
## End(Not run)
```

---

interPerfMeasures

*Interactive Model Performance Evaluation & Comparison*


---

## Description

Interactive version of the staticPerfMeasures function

## Usage

```
interPerfMeasures(list_models, sample_size_concord = 5000,
  model_function = NULL, data = NULL, y = NULL)
```

## Arguments

- |                                  |   |
|----------------------------------|---|
| <code>list_models</code>         | A list of one (or more) dataframes for each model whose performance is to be evaluated. Each dataframe should comprise of 2 columns with the first column indicating the class labels (0 or 1) and the second column providing the raw predicted probabilities  |
| <code>sample_size_concord</code> | For computing concordance-discordance measures (and c-statistic) a random sample is drawn from each dataset (if <code>nrow(dataset) &gt; 5000</code> ). Default sample size of 5000 can be adjusted by changing the value of this argument  |
| <code>model_function</code>      | Models can be created interactively, if required. For this option to work, a model function should be passed as an argument. The model function should take a formula as an argument, and return a a dataframe as output (dataframe should comprise of 2 columns with the first column indicating the class labels (0 or 1) |

	and the second column providing the raw predicted probabilities) Refer to the example section for more details
data	The name of the data-set. The Independent Variable (IV) names, for interactive model building, is picked up from this data set
y	The column name of the Dependent Variable (DV), for interactive model building

### Value

This function will launch a ShinyApp. Input parameters (such as the number of bins, the "g" argument in the static version of this function) can be adjusted through app widgets. The 'Run-Analysis' button in the app, will generate model performance output basis selected input parameters

For interactive Model building, a model function, data set & the dependent variable name should be passed as arguments. Interactive model building option creates additional input widgets in the app. This includes -

A drop down to select independent variables (the names of the variables will be picked up from the data argument)

An input slider to include additional models (upto 4 additional models can be created). Each additional model updates the original model created. For e.g. consider the dataset has 10 IVs: x1-x10. Original model was created by selecting x1-x4 from the drop down list. If we need to create a second model, by including x5 and excluding x3 simply type, "+ x5 - x3" in the input text box

### Examples

```
# Without interactive model development
model_1 <- glm(Species ~ Sepal.Length,data=iris,family=binomial)
model_2 <- glm(Species ~ Sepal.Width, data=iris, family = binomial)
df1 <- data.frame(model_1$y,fitted(model_1))
df2 <- data.frame(model_2$y,fitted(model_2))

## Not run:
#This will launch a Shiny App
interPerfMeasures(list_models = list(df1,df2))
## End(Not run)

# With interactive model development
glm_model <- function(formula) {
  glm_model <- glm(formula, data = iris, family = "binomial")
  out <- data.frame(glm_model$y, fitted(glm_model))
  out }
## Not run:
#This will launch a Shiny App
interPerfMeasures (model_function = glm_model,data=iris,y="Species")
## End(Not run)
```

---

staticConfMatrix      *Confusion Matrix for Binary Classification Models*

---

## Description

Generates confusion matrix for a specified probability threshold. Also computes the following metrics - Accuracy, True Positive Rate, False Positive Rate & Precision. Multiple models can be passed as arguments to this function

## Usage

```
staticConfMatrix(list_models, t, reps = NULL, reps.all.unique = F)
```

## Arguments

list_models	A list of one (or more) dataframes for each model whose performance is to be evaluated. Each dataframe should comprise of 2 columns with the first column indicating the class labels (0 or 1) and the second column providing the raw predicted probabilities
t	Probability threshold value
reps	Performance measures derived from the confusion matrix (Accuracy, TPR, FPR & Precision) are computed for a range of different probability thresholds. The "reps" argument controls the number of different probability thresholds considered (threshold range given by the sequence - seq(0,1,1/reps))
reps.all.unique	Logical; If set to True, Performance measures are computed for each unique Probability value

## Value

If reps = NULL, the output will be a list with 2 components - a confusion matrix dataframe and a dataframe with the values of the computed metrics (Accuracy,TPR,FPR,Precision). If reps argument is supplied, an additional dataframe containing the metrics values for different probability thresholds is included in the output

## Examples

```
model_1 <- glm(Species ~ Sepal.Length,data=iris,family=binomial)
model_2 <- glm(Species ~ Sepal.Width, data=iris, family = binomial)
df1 <- data.frame(model_1$y,fitted(model_1))
df2 <- data.frame(model_2$y,fitted(model_2))
staticConfMatrix(list(df1,df2),t=0.2)
```

---

staticPerfMeasures      *Model evaluation measures for Binary classification models*

---

### Description

Generates & plots the following performance evaluation & validation measures for Binary Classification Models - Hosmer Lemeshow goodness of fit tests, Calibration plots, Lift index & gain charts & concordance-discordance measures

### Usage

```
staticPerfMeasures(list_models, g, perf_measures = c("hosmer", "calibration",
  "lift", "concord"), sample_size_concord = 5000)
```

### Arguments

<code>list_models</code>	A list of one (or more) dataframes for each model whose performance is to be evaluated. Each dataframe should comprise of 2 columns with the first column indicating the class labels (0 or 1) and the second column providing the raw predicted probabilities
<code>g</code>	The number of groups for binning. The predicted probabilities are binned as follows For Hosmer-Lemshow (HL) test: Predicted probabilities binned as per $g$ unique quantiles i.e. <code>cut_points = unique(quantile(predicted_prob,seq(0,1,1/g)))</code> For Lift-Index & Gain charts: Same as HL test, however if $g > \text{unique}(\text{predicted\_probability})$ , the predicted probabilities are used as such without binning For calibration plots, $g$ equal sized intervals are created (of width $1/g$ each)
<code>perf_measures</code>	Select the required performance evaluation and validation measure/s, from the following options - <code>c('hosmer','calibration','lift','concord')</code> . Default option is All
<code>sample_size_concord</code>	For computing concordance-discordance measures (and c-statistic) a random sample is drawn from each dataset (if <code>nrow(dataset) &gt; 5000</code> ). Default sample size of 5000 can be adjusted by changing the value of this argument

### Value

A nested list with 2 components - a list of dataframes and a list of plots - containing the outcomes of the different performance evaluations carried out.

### Examples

```
model_1 <- glm(Species ~ Sepal.Length, data=iris, family=binomial)
model_2 <- glm(Species ~ Sepal.Width, data=iris, family = binomial)
df1 <- data.frame(model_1$y, fitted(model_1))
df2 <- data.frame(model_2$y, fitted(model_2))
staticPerfMeasures(list(df1,df2),g=10, perf_measures = c("hosmer","lift"))
```

# Index

interConfMatrix, [2](#)  
interPerfMeasures, [3](#)

staticConfMatrix, [5](#)  
staticPerfMeasures, [6](#)