Package ‘MLAT’

May 30, 2018

Type Package

Title Machine Learning Automated Software

Date 2018-05-29

Version 0.1.0

Author Paulo Cirino Ribeiro Neto

Maintainer Paulo Cirino Ribeiro Neto <paulocirino.neto@gmail.com>

Description The package was created to help R users automate the processes
of benchmarking machine learning methods agains common methods in common datasets.

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 6.0.1

Imports e1071,
class

Suggests ggplot2

R topics documented:

`Accuracy` *Accuracy*

Description

Returns the Accuracy for a classification problem.

Usage

```
Accuracy(Y, Y_hat)
```

Arguments

`Y`	Ground truth numeric vector.
`Y_hat`	Predicted Labels numeric vector.

Value

A numeric value corresponding to the Accuracy of a classification problem

Examples

```
Y = sample(x = c(1,2), size = 10, replace = TRUE)
Y_hat = sample(x = c(1,2), size = 10, replace = TRUE)
Accuracy(Y = Y, Y_hat = Y_hat)
```

`AUC` *AUC*

Description

Returns the Area Under the Curve for a binarry classification problem.

Usage

```
AUC(Y, Y_hat)
```

Arguments

`Y`	Ground truth numeric vector.
`Y_hat`	Predicted Labels numeric vector.

Value

A numeric value corresponding to the AUC of binary classification problem

Examples

```
Y = sample(x = c(1,2), size = 10, replace = TRUE)
Y_hat = sample(x = c(1,2), size = 10, replace = TRUE)
AUC(Y = Y, Y_hat = Y_hat)
```

`BinClassMetrics` *BinClassMetricsNames*

Description

Returns a binaryResultList with binary classificaiton metrics.

Usage

```
BinClassMetrics(Y, Y_hat, MetricsNames = BinClassMetricsNames())
```

Arguments

`Y`	Ground truth numeric vector.
`Y_hat`	Predicted Labels numeric vector.
`MetricsNames`	can be found at BinClassMetricsNames()

Value

A binaryResultList with results

Examples

```
Y = sample(x = c(1,2), size = 10, replace = TRUE)
Y_hat = sample(x = c(1,2), size = 10, replace = TRUE)
BinClassMetrics(Y = Y, Y_hat = Y_hat, MetricsNames = BinClassMetricsNames())
```

`BinClassMetricsNames` *Binary Classification Metrics*

Description

Returns a character string vector containning all binary classification metrics.

Usage

```
BinClassMetricsNames()
```

Value

character string vector with all possible binary classification metrics.

Examples

```
BinClassMetricsNames()
```

CreateAlgo *Create Algorithm*

Description

It is an auxiliary function to help creating new algorithms in the package standarts.

Usage

```
CreateAlgo(algoName, algoFun, task, paramList)
```

Arguments

algoName	A character string that represents the algorithm name.
algoFun	A function class object.
task	A character string vector, cointaning 'MultClass' or/and 'BinClass' and/or 'Regression'.
paramList	A list of all parameters and their values to be tested.

Value

An object of class algorithm.

Examples

```
algoName <- 'myAlgo'
algoFun <- function(X_train, Y_train, X_test, param1){
 set.seed(param1)
 sample(Y_train, size = nrow(X_test), replace = TRUE)}
task <- c('MultClass', 'BinClass')
paramList = list(param1 = 1:3)
CreateAlgo(algoName = algoName, algoFun = algoFun, task = task, paramList = paramList)
```

CreateDataSet *Create DataSet*

Description

Creates a DataSet object type, to be used with the models provided with this package

Usage

```
CreateDataSet(X, Y, Name, type, task)
```

Arguments

X	A Matrix.
Y	A numeric vector of classes or values.
Name	A character string, as dataset name.
type	A character string, the types of values for X (numeric or integer).
task	A character string vector of task to be performed, check GetPossibleTasks().

Value

A DataSet object type.

Examples

```
X <- as.matrix(cbind(runif(n = 100), runif(n = 100)))
Y <- sample(x = c(1, 2), size = 100, replace = TRUE)
Name <- 'randomData'
type <- 'numeric'
task <- 'BinClas'
newData <- CreateDataSet(X = X, Y = Y, Name = Name, type = type, task = task)
```

GetAllMultClassAlgo *Generates all MultClass Classification Alogrithms*

Description

It is an auxiliary that allows to load all of the packages multclass classification alogrithms.

Usage

```
GetAllMultClassAlgo()
```

Value

Returns a list with all MultClass Classification Alogrithms

Examples

```
GetAllMultClassAlgo()
```

GetData *Get DataSet by name*

Description

Get a dataset by name

Usage

```
GetData(datasetName, seed, splitPerc)
```

Arguments

datasetName	A character string, as DataSet name.
seed	For the traint and test split.
splitPerc	Percentage for train of all dataSet, between 0 and 1.

Value

A trainTestDataSet object type.

Examples

```
GetData(datasetName = 'Iris', seed = 123, splitPerc = 0.7)
```

`GetDataSetsNames` *Get Datasets Names*

Description

Get all available datasets names

Usage

```
GetDataSetsNames(task)
```

Arguments

`task`	A character string with desired task, for possible tasks check GetPossibleTasks().

Value

A character string vector with all possible datasets names.

Examples

```
GetDataSetsNames()
```

`GetMetrics` *Get Metrics*

Description

Get all possible task names

Usage

```
GetMetrics(task)
```

Arguments

`A`	valid task, can be ’MultClass’ or ’BinClass’.

Value

All metrics names for that taks

Examples

```
GetPossibleTasks()
```

GetPossibleTasks *Get Possible Tasks*

Description

Get all possible task names

Usage

```
GetPossibleTasks()
```

Value

A character string vector with all possible task names.

Examples

```
GetPossibleTasks()
```

knn *K Nearest Neighbours*

Description

It is the K Nearest Neighbours method

Usage

```
knn(X_train, Y_train, X_test, K)
```

Arguments

X_train	A Matrix of trainning observations.
Y_train	A numeric vector of classes or values of the trainning observations.
X_test	A Matrix of testing observations.
K	An integer as a parameter for the knn method.

Value

predicted labels

Examples

```
X <- as.matrix(cbind(runif(n = 100), runif(n = 100)))
pos <- sample(100, 70)
X_train <- X[pos, ]
X_test <- X[-pos, ]
Y_train <- as.numeric( X_train[, 1] ** 2 - X_train[, 2] > 0)
Y_test <- as.numeric(X_test[, 1] ** 2 - X_test[, 2] > 0)
K <- 5
Y_predicted <- knn(X_train = X_train, Y_train = Y_train, X_test = X_test, K = K)
print(table(Y_test, Y_predicted))
```

LogLoss *LogLoss*

Description

Returns the Logarithmic Loss for classification problem.

Usage

```
LogLoss(Y, Y_hat)
```

Arguments

Y	Ground truth numeric vector.
Y_hat	Predicted Labels numeric vector.

Value

A numeric value corresponding to the LogLoss of binary classification problem

Examples

```
Y = sample(x = c(1,2), size = 10, replace = TRUE)
Y_hat = sample(x = c(1,2), size = 10, replace = TRUE)
LogLoss(Y = Y, Y_hat = Y_hat)
```

MultClassMetrics *MultClassMetrics*

Description

Returns a multiClassResultList with multi class classificaiton metrics.

Usage

```
MultClassMetrics(Y, Y_hat, MetricsNames)
```

Arguments

Y	Ground truth numeric vector.
Y_hat	Predicted Labels numeric vector.
MetricsNames	Metrics names, avilable can be found with MultiClassMetricsNames() .

Value

A multiClassResultList with results

Examples

```
Y = sample(x = c(1,2, 3), size = 20, replace = TRUE)
Y_hat = sample(x = c(1, 2, 3), size = 20, replace = TRUE)
MultClassMetrics(Y = Y, Y_hat = Y_hat, MetricsNames = MultiClassMetricsNames())
```

MultiClassMetricsNames *Multi Class Classification Metrics*

Description

Returns a character string vector containning all multi class classification metrics.

Usage

```
MultiClassMetricsNames()
```

Value

character string vector with all possible multi class classification metrics.

Examples

```
MultiClassMetricsNames()
```

MultiLogLoss *MultiLogLoss*

Description

Returns the Logarithmic Loss for multi class classification problem.

Usage

```
MultiLogLoss(Y, Y_hat)
```

Arguments

Y	Ground truth numeric vector.
Y_hat	Predicted Labels numeric vector.

Value

A numeric value corresponding to the LogLoss of binary classification problem

Examples

```
Y = sample(x = c(1,2), size = 10, replace = TRUE)
Y_hat = sample(x = c(1,2), size = 10, replace = TRUE)
MultiLogLoss(Y = Y, Y_hat = Y_hat)
```

RunTests *Run Tests*

Description

Runs the tests given the methods and datasets

Usage

```
RunTests(cmpTestsFuncsList = cmpTestsFuncsList, task = task,
  dataSetNames = dataSetNames, metrics = NA, nTestsPerParam = 1,
  splitPerc = 0.7, verbose = TRUE)
```

Arguments

cmpTestsFuncsList	is a list of
task	A character string with 'MultClass' or 'BinClas'
dataSetNames	A character string vector with valid dataset names, for options check
metrics	character string vector with all testing metrics or NA for all available metrics
nTestsPerParam	FOOO
splitPerc	TODO
verbose	TRUE/FALSE value for printing partial test

Value

TODO

Examples

```
cmpTestsFuncsList <- GetAllMultClassAlgo()
task <- 'MultClass'
dataSetNames <- c('Iris', 'PimaIndiansDiabetes')
myResult <- RunTests(cmpTestsFuncsList = GetAllMultClassAlgo(),
                     task = 'MultClass',
                     dataSetNames = c('Iris', 'PimaIndiansDiabetes'))
```

squareConfusionTable *Square Confusion Matrix*

Description

Returns a square confusion matrix

Usage

```
squareConfusionTable(Y, Y_hat)
```

Arguments

Y	A numeric vector for the ground truth labels
Y_hat	A numeric vector for the predicted Labels

Value

A confusion table

Examples

```
squareConfusionTable(Y = sample(1:2, size = 10, replace = TRUE), Y_hat = rep(1, 10))
```

svm_linear *Linear Support Vector Machines*

Description

It is the Support Vector Machines without a kernel

Usage

```
svm_linear(X_train, Y_train, X_test, C)
```

Arguments

X_train	A Matrix of trainning observations.
Y_train	A numeric vector of classes or values of the trainning observations.
X_test	A Matrix of testing observations.
C	A numeric value that represents the cost of constraints violation of the regularization term in the Lagrange formulation.

Value

predicted labels

Examples

```
X <- as.matrix(cbind(runif(n = 100), runif(n = 100)))
pos <- sample(100, 70)
X_train <- X[pos, ]
X_test <- X[-pos, ]
Y_train <- as.numeric( X_train[, 1] ** 2 - X_train[, 2] > 0)
Y_test <- as.numeric(X_test[, 1] ** 2 - X_test[, 2] > 0)
C <- 5
Y_predicted <- svm_linear(X_train = X_train, Y_train = Y_train, X_test = X_test, C = C)
print(table(Y_test, Y_predicted))
```

svm_poli *Support Vector Machines with Polinomial Kernel*

Description

It is the Support Vector Machines with a polinomial kernel

Usage

```
svm_poli(X_train, Y_train, X_test, degree, gamma, coef0, C)
```

Arguments

`X_train` A Matrix of trainning observations.

`Y_train` A numeric vector of classes or values of the trainning observations.

`X_test` A Matrix of testing observations.

`degree` A integer that represents the kernel polynomial degree

`gamma` A numeric value as the kernel coefficient.

`coef0` A numeric value for kernel independent term.

`C` A numeric value that represents the cost of constraints violation of the regularization term in the Lagrange formulation.

Value

predicted labels

Examples

```
X <- as.matrix(cbind(runif(n = 100), runif(n = 100)))
pos <- sample(100, 70)
X_train <- X[pos, ]
X_test <- X[-pos, ]
Y_train <- as.numeric( X_train[, 1] ** 2 - X_train[, 2] > 0)
Y_test <- as.numeric(X_test[, 1] ** 2 - X_test[, 2] > 0)
C <- 5
coef0 <- 0
degree <- 5
gamma <- 0.5
Y_predicted <- svm_poli(X_train = X_train, Y_train = Y_train, X_test = X_test, C = C, coef0 = coef0, degree = deg
print(table(Y_test, Y_predicted))
```

svm_radial *Support Vector Machines with Radial Kernel*

Description

It is the Support Vector Machines with a radial kernel

Usage

```
svm_radial(X_train, Y_train, X_test, gamma, C)
```

Arguments

`X_train` A Matrix of trainning observations.

`Y_train` A numeric vector of classes or values of the trainning observations.

`X_test` A Matrix of testing observations.

`gamma` A numeric value as the kernel coefficient.

`C` A numeric value that represents the cost of constraints violation of the regularization term in the Lagrange formulation.

Value

predicted labels

Examples

```
X <- as.matrix(cbind(runif(n = 100), runif(n = 100)))
pos <- sample(100, 70)
X_train <- X[pos, ]
X_test <- X[-pos, ]
Y_train <- as.numeric( X_train[, 1] ** 2 - X_train[, 2] > 0)
Y_test <- as.numeric(X_test[, 1] ** 2 - X_test[, 2] > 0)
C <- 5
gamma <- 0.5
Y_predicted <- svm_radial(X_train = X_train, Y_train = Y_train, X_test = X_test, C = C, gamma = gamma)
print(table(Y_test, Y_predicted))
```

Index