Package 'MLSeq'

October 16, 2018

Type Package

Title Machine Learning Interface for RNA-Seq Data

Version 1.20.3

Date 2018-06-07

Author Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Izzet Parug Duru, Ahmet Ozturk, Ahmet Ergun Karaagaoglu

Maintainer Gokmen Zararsiz <gokmenzararsiz@hotmail.com>

Depends caret, ggplot2

VignetteBuilder knitr

Suggests knitr, testthat, BiocStyle, VennDiagram, pamr

Imports methods, DESeq2, edgeR, limma, Biobase, SummarizedExperiment, plyr, foreach, utils, sSeq, xtable

biocViews Sequencing, RNASeq, Classification, Clustering

Description This package applies several machine learning methods, including SVM, bagSVM, Random Forest and CART to RNA-Seq data.

License GPL(>=2)

NeedsCompilation no

RoxygenNote 6.0.1

Collate 'all_classes.R' 'all_generics.R' 'voomFunctions.R' 'classify.R' 'helper_functions.R' 'predict.R' 'methods.R' 'onAttach.R' 'package_and_suppl.R' 'plda_nblda_functions.R'

git_url https://git.bioconductor.org/packages/MLSeq

git_branch RELEASE_3_7

git_last_commit e32e56d

git_last_commit_date 2018-06-07

Date/Publication 2018-10-15

R topics documented:

`MLSeq-package` *Machine learning interface for RNA-Seq data*

Description

This package applies machine learning methods, such as Support Vector Machines (SVM), Random Forest (RF), Classification and Regression Trees (CART), Linear Discriminant Analysis (LDA) and more to RNA-Seq data. MLSeq combines well-known differential expression algorithms from bioconductor packages with functions from a famous package `caret`, which has comprehensive machine learning algorithms for classification and regression tasks. Although `caret` has 200+ classification/regression algorithm built-in, approximately 85 classification algorithms are used in `MLSeq` for classifying gene-expression data. See `availableMethods()` for further information.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus, Ahmet Ozturk and Ahmet Ergun Karaagaoglu

---

Maintainers:

Gokmen Zararsiz, <gokmenzararsiz@erciyes.edu.tr>

Dincer Goksuluk <dincer.goksuluk@hacettepe.edu.tr>

Selcuk Korkmaz <selcukorkmaz@hotmail.com>

See Also

availableMethods, getModelInfo

Package:	MLSeq
Type:	Package
License:	GPL (>= 2)

`Available-classifiers` *Available classification/regression methods in* `MLSeq`

Description

This function returns a character vector of available classification/regression methods in `MLSeq`. These methods are imported from `caret` package. See details below.

Usage

```
availableMethods(model = NULL, regex = TRUE, ...)

printAvailableMethods()
```

Arguments

`model`	a character string indicating the name of classification model. If NULL, all the available methods from `MLSeq` is returned. Otherwise, the methods which are complete or partial matches to requested string is returned. See `regex` for details.
`regex`	a logical: should a regular expressions be used? If FALSE, a simple match is conducted against the whole name of the model.
`...`	options to pass to `grepl`.

Details

There are 200+ methods available in `caret`. We import approximately 85 methods which are available for "classification" task. Some of these methods are available for both classification and regression tasks. `availableMethods()` returns a character vector of available methods in `MLSeq`. These names are directly used in `classify` function with arguement `method`. See http://topepo.github.io/caret/available-models.html for a complete list of available methods in `caret`. Run `printAvailableMethods()` to print detailed information about classification methods (prints to R Console).

Value

a requested or complete character vector of available methods.

Note

Available methods in `MLSeq` will be regularly updated. Some of the methods might be removed as well as some others took its place in `MLSeq`. Please check the available methods before fitting the model. This function is inspired from the function `getModelInfo()` in `caret` and some of the code chunks and help texts are used here.

See Also

classify, getModelInfo, train

---

cervical *Cervical cancer data*

Description

Cervical cancer data measures the expressions of 714 miRNAs of human samples. There are 29 tumor and 29 non-tumor cervical samples and these two groups are treated as two separete classes.

Format

A data frame with 58 observations on the following 715 variables.

Source

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2880020/#supplementary-material-sec

References

Witten, D., et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Examples

```
## Not run:
data(cervical)

## End(Not run)
```

---

classify *Fitting classification models to sequencing data*

Description

This function fits classification algorithms to sequencing data and measures model performances using various statistics.

Usage

```
classify(data, method = "rpart", B = 25, ref = NULL,
  class.labels = NULL, preProcessing = c("deseq-vst", "deseq-rlog",
  "deseq-logcpm", "tmm-logcpm", "logcpm"), normalize = c("deseq", "TMM",
  "none"), control = NULL, ...)
```

Arguments

data	a DESeqDataSet object, see the constructor functions DESeqDataSet, DESeqDataSetFromMatrix, DESeqDataSetFromHTSeqCount in DESeq2 package.
method	a character string indicating the name of classification method. Methods are implemented from the caret package. Run availableMethods() for a list of available methods.
B	an integer. It is the number of bootstrap samples for bagging classifiers, for example "bagFDA" and "treebag". Default is 25.
ref	a character string indicating the user defined reference class. Default is NULL. If NULL is selected, first category of class labels is used as reference.
class.labels	a character string indicating the column name of colData(...). Should be given as "character". The column from colData() which matches with given column name is used as class labels of samples. If NULL, first column is used as class labels. Default is NULL.
preProcessing	a character string indicating the name of the preprocessing method. This option consists both the normalization and transformation of the raw sequencing data. Available options are: • deseq-vst: Normalization is applied with deseq median ratio method. Variance stabiling transformation is applied to the normalized data. • deseq-rlog: Normalization is applied with deseq median ratio method. Regularized logarithmic transformation is applied to the normalized data. • deseq-logcpm: Normalization is applied with deseq median ratio method. Log of counts-per-million transformation is applied to the normalized data. • tmm-logcpm: Normalization is applied with trimmed mean of M values (TMM) method. Log of counts-per-million transformation is applied to the normalized data. • logcpm: Normalization is not applied. Log of counts-per-million transformation is used for the raw counts. **IMPORTANT**: See Details for further information.
normalize	a character string indicating the type of normalization. Should be one of 'deseq', 'tmm' and 'none'. Default is 'deseq'. This option should be used with discrete and voom-based classifiers since no transformation is applied on raw counts. For caret-based classifiers, the argument 'preProcessing' should be used.
control	a list including all the control parameters passed to model training process. This arguement should be defined using wrapper functions trainControl for caret-based classifiers, discreteControl for discrete classifiers (PLDA, PLDA2 and NBLDA) and voomControl for voom-based classifiers (voomDLDA, voomDQDA and voomNSC). See related functions for further details.
...	optional arguments passed to selected classifiers.

Details

MLSeq consists both microarray-based and discrete-based classifiers along with the preprocessing approaches. These approaches include both normalization techniques, i.e. deseq median ratio (Anders et al., 2010) and trimmed mean of M values (Robinson et al., 2010) normalization methods, and the transformation techniques, i.e. variance- stabilizing transformation (vst)(Anders and Huber, 2010), regularized logarithmic transformation (rlog)(Love et al., 2014), logarithm of counts per million reads (log-cpm)(Robinson et al., 2010) and variance modeling at observational level (voom)(Law et al., 2014). Users can directly upload their raw RNA-Seq count data, preprocess

their data, build one of the numerous classification models, optimize the model parameters and evaluate the model performances.

MLSeq package consists of a variety of classification algorithms for the classification of RNA-Seq data. These classifiers are categorized into two class: i) microarray-based classifiers after proper transformation, ii) discrete-based classifiers. First option is to transform the RNA-Seq data to bring it hierarchically closer to microarrays and apply microarray-based algorithms. These methods are implemented from the caret package. Run availableMethods() for a list of available methods. Note that voom transformation both exports transformed gene-expression matrix as well as the precision weight matrices in same dimension. Hence, the classifier should consider these two matrices. Zararsiz (2015) presented voom-based diagonal discriminant classifiers and the sparse voom-based nearest shrunken centroids classifier. Second option is to build new discrete-based classifiers to classify RNA-Seq data. Two methods are currently available in the literature. Witten (2011) considered modeling these counts with Poisson distribution and proposed sparse Poisson linear discriminant analysis (PLDA) classifier. The authors suggested a power transformation to deal with the overdispersion problem. Dong et al. (2016) extended this approach into a negative binomial linear discriminant analysis (NBLDA) classifier. More detailed information can be found in referenced papers.

Value

an `MLSeq` object for trained model.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus, Ahmet Ozturk and Ahmet Ergun Karaagaoglu

References

Kuhn M. (2008). Building predictive models in R using the caret package. Journal of Statistical Software, (http://www.jstatsoft.org/v28/i05/)

Anders S. Huber W. (2010). Differential expression analysis for sequence count data. Genome Biology, 11:R106

Witten DM. (2011). Classification and clustering of sequencing data using a poisson model. The Annals of Applied Statistics, 5(4), 2493:2518

Law et al. (2014) Voom: precision weights unlock linear model analysis tools for RNA-Seq read counts, Genome Biology, 15:R29, doi:10.1186/gb-2014-15-2-r29

Witten D. et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-Seq data. Genome Biology, 11:R25, doi:10.1186/gb-2010-11-3-r25

M. I. Love, W. Huber, and S. Anders (2014). Moderated estimation of fold change and dispersion for rna-seq data with deseq2. Genome Biol, 15(12):550,. doi: 10.1186/s13059-014-0550-8.

Dong et al. (2016). NBLDA: negative binomial linear discriminant analysis for rna-seq data. BMC Bioinformatics, 17(1):369, Sep 2016. doi: 10.1186/s12859-016-1208-1.

Zararsiz G (2015). Development and Application of Novel Machine Learning Approaches for RNA-Seq Data Classification. PhD thesis, Hacettepe University, Institute of Health Sciences, June 2015.

See Also

predictClassify, train, trainControl, voomControl, discreteControl

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies
## 1. caret-based classifiers:
# Random Forest (RF) Classification
 rf <- classify(data = data.trainS4, method = "rf",
         preProcessing = "deseq-vst", ref = "T",
         control = trainControl(method = "repeatedcv", number = 5,
                                repeats = 2, classProbs = TRUE))
rf

# 2. Discrete classifiers:
# Poisson Linear Discriminant Analysis
pmodel <- classify(data = data.trainS4, method = "PLDA", ref = "T",
                  class.labels = "condition",normalize = "deseq",
                  control = discreteControl(number = 5, repeats = 2,
                                            tuneLength = 10, parallel = TRUE))
pmodel

# 3. voom-based classifiers:
# voom-based Nearest Shrunken Centroids
vmodel <- classify(data = data.trainS4, normalize = "deseq", method = "voomNSC",
                  class.labels = "condition", ref = "T",
                  control = voomControl(number = 5, repeats = 2, tuneLength = 10))
vmodel

## End(Not run)
```

confusionMat *Accessors for the 'confusionMat' slot.*

Description

This slot stores the confusion matrix for the trained model using `classify` function.

Usage

```
confusionMat(object)

## S4 method for signature 'MLSeq'
confusionMat(object)

## S4 method for signature 'MLSeqModelInfo'
confusionMat(object)
```

Arguments

`object`	an `MLSeq` or `MLSeqModelInfo` object.

Details

`confusionMat` slot includes information about cross-tabulation of observed and predicted classes and corresponding statistics such as accuracy rate, sensitivity, specifity, etc. The returned object is in `confusionMatrix` class of caret package. See `confusionMatrix` for details.

See Also

`confusionMatrix`

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
```

```
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

confusionMat(cart)

## End(Not run)
```

`control` *Accessors for the 'control' slot.*

Description

This slot stores the information about control parameters of selected classification model.

Usage

```
control(object)

control(object) <- value

## S4 method for signature 'MLSeq'
control(object)

## S4 replacement method for signature 'MLSeq,list'
control(object) <- value
```

Arguments

`object`	an `MLSeq` or `MLSeqModelInfo` object.
`value`	a character string. Select reference category for class labels.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]
```

```
# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

control(cart)

## End(Not run)
```

`discrete.train-class` `discrete.train` *object*

Description

This object is the subclass for the `MLSeq.train` class. It contains trained model information for discrete classifiers such as Poisson Linear Discriminant Analysis (PLDA) and Negative Binomial Linear Discriminant Analysis (NBLDA).

Slots

`inputs`: a list with elements used as input for classification task.

`control`: a list with control parameters for discrete classifiers, e.g. PLDA, PLDA2 and NBLDA.

`crossValidatedModel`: a list. It stores the results for cross validation.

`finalModel`: a list. This is the trained model with optimum parameters.

`tuningResults`: a list. It stores the results for tuning parameter if selected classifier has one or more parameters to be optimized.

`callInfo`: a list. call info for selected method.

`discreteControl` *Define controlling parameters for discrete classifiers (NBLDA and PLDA)*

Description

This function sets the control parameters for discrete classifiers (PLDA and NBLDA) while training the model.

Usage

```
discreteControl(method = "repeatedcv", number = 5, repeats = 10,
  rho = NULL, rhos = NULL, beta = 1, prior = NULL, alpha = NULL,
  truephi = NULL, foldIdx = NULL, tuneLength = 30, parallel = FALSE,
  ...)
```

Arguments

`method`	validation method. Support repeated cross validation only ("repeatedcv").
`number`	a positive integer. Number of folds.
`repeats`	a positive integer. Number of repeats.
`rho`	a single numeric value. This parameter is used as tuning parameter in PLDA classifier. It does not effect NBLDA classifier.
`rhos`	a numeric vector. If optimum parameter is searched among given values, this option shpould be used.
`beta`	parameter of Gamma distribution. See PLDA for details.
`prior`	prior probabilities of each class
`alpha`	a numeric value in the interval 0 and 1. It is used to apply power transformation through PLDA method.
`truephi`	a numeric value. If true value of genewise dispersion is known and constant for all genes, this parameter should be used.
`foldIdx`	a list including the fold indexes. Each element of this list is the vector indices of samples which are used as test set in this fold.
`tuneLength`	a positive integer. If there is a tuning parameter in the classifier, this value is used to define total number of tuning parameter to be searched.
`parallel`	if TRUE, parallel computing is performed.
`...`	further arguments. Deprecated.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus, Ahmet Ozturk and Ahmet Ergun Karaagaoglu

See Also

`classify`, `trainControl`, `discreteControl`

Examples

```
1L
```

---

input *Accessors for the 'inputObject' slot of an* MLSeq *object*

Description

MLSeq package benefits from DESeqDataSet structure from bioconductor package DESeq2 for storing gene expression data in a comprehensive structure. This object is used as an input for classification task through classify. The input is stored in inputObject slot of MLSeq object.

Usage

```
input(object)

## S4 method for signature 'MLSeq'
input(object)
```

Arguments

object	an MLSeq object.

See Also

classify, DESeqDataSet

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
```

```
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

input(cart)

## End(Not run)
```

`isUpdated` *Checks if MLSeq object is updated/modified or not.*

Description

These functions are used to check whether the `MLSeq` object is modified and/or updated. It is possible to update classification parameters of `MLSeq` object which is returned by `classify()` function.

Usage

```
isUpdated(object)

isUpdated(object) <- value

isModified(object)

isModified(object) <- value

## S4 method for signature 'MLSeq'
isUpdated(object)

## S4 replacement method for signature 'MLSeq,logical'
isUpdated(object) <- value

## S4 method for signature 'MLSeq'
isModified(object)

## S4 replacement method for signature 'MLSeq,logical'
isModified(object) <- value
```

Arguments

`object`	an `MLSeq` object.
`value`	a logical. Change the state of update info.

Value

a logical.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

isUpdated(cart)
isModified(cart)

## End(Not run)
```

---

metaData *Accessors for the 'metaData' slot of an* MLSeq *object*

Description

This slot stores metada information of MLSeq object.

Usage

```
metaData(object)

## S4 method for signature 'MLSeq'
metaData(object)
```

Arguments

object	an MLSeq object.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                   colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

metaData(cart)

## End(Not run)
```

---

method *Accessors for the 'method'.*

Description

This slot stores the name of selected model which is used in `classify` function. The trained model is stored in slot `trainedModel`. See `trained` for details.

Usage

```
method(object)

method(object) <- value

## S4 method for signature 'MLSeq'
method(object)

## S4 method for signature 'MLSeqModelInfo'
method(object)

## S4 replacement method for signature 'MLSeq,character'
method(object) <- value
```

Arguments

object	an MLSeq object.
value	a character string. One of the available classification methods to replace with current method stored in MLSeq object.

Details

method slot stores the name of the classification method such as "svmRadial" for Radial-based Support Vector Machines, "rf" for Random Forests, "voomNSC" for voom-based Nearest Shrunken Centroids, etc. For the complete list of available methods, see printAvailableMethods and availableMethods.

See Also

trained

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])
```

```
# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

method(cart)

## End(Not run)
```

MLSeq-class MLSeq *object*

Description

For classification, this is the main class for the MLSeq package. It contains all the information including trained model, selected genes, cross-validation results, etc.

Details

Objects can be created by calls of the form new("MLSeq", ...). This type of objects is created as a result of classify function of MLSeq package. It is then used in predict or predictClassify function for predicting the class labels of new samples.

Slots

inputObject: stores the data in DESeqDataSet object.

modelInfo: stores all the information about classification model. The object is from subclass MLSeqModelInfo. See MLSeqModelInfo-class for details.

metaData: metadata for MLSeq object. The object is from subclass MLSeqMetaData. See MLSeqMetaData-class for details.

Note

An MLSeq class stores the results of classify function and offers further slots that are populated during the analysis. The slot inputObject stores the raw and transformed data throughout the classification. The slot modelInfo stores all the information about classification model. These results may contain the classification table and performance measures such as accuracy rate, sensitivity, specifity, positive and negative predictive values, etc. It also contains information on classification method, normalization and transformation used in the classification model. Lastly, the slot metaData stores the information about modified or updated slots in MLSeq object.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus, Ahmet Ozturk and Ahmet Ergun Karaagaoglu

See Also

MLSeqModelInfo-class, MLSeqMetaData-class

MLSeqMetaData-class MLSeqMetaData *object*

Description

This object is a subclass for the MLSeq class. It contains metadata information, i.e. information on modified and/or updated elements, raw data etc..

Details

Objects can be created by calls of the form new("MLSeqMetaData", ...). This type of objects is created as a result of classify function of MLSeq package. It is then used in update function for updating the object in given object.

Slots

updated, modified: a logical. See notes for details.

modified.elements: a list containing the modified elements in MLSeq obejct.

rawData.DESeqDataSet: raw data which is used for classification.

classLabel: a character string indicating the name of class variable.

Note

The function update is used to re-run classification task with modified elements in MLSeq object. This function is useful when one wish to perform classification task with modified options without running classify function from the beginning. MLSeqMetaData object is used to store information on updated and/or modified elements in MLSeq object.

If an MLSeq object is modified, i.e. one or more elements in MLSeq object is replaced using related setter functions such as method, ref etc., the slot modified becomes TRUE. Similarly, the slot updated stores the information that the MLSeq object is updated (or classification task is re-runned) or not. If updated slot is FALSE and modified slot is TRUE, one should run update to obtain the classification results by considering the modified elements.

See Also

update, isUpdated, isModified

MLSeqModelInfo-class MLSeqModelInfo *object*

Description

For classification, this is the subclass for the MLSeq class. This object contains all the information about classification model.

Details

Objects can be created by calls of the form MLSeqModelInfo(...). This type of objects is created as a result of classify function of MLSeq package. It is then used in predictClassify function for predicting the class labels of new samples.

Slots

method, transformation, normalization: these slots store the classification method, transformation technique and normalization method respectively. See notes for details.

preProcessing: See classify for details.

ref: a character string indicating the reference category for cases (diseased subject, tumor sample, etc.)

control: a list with controlling parameters for classification task.

confusionMat: confusion table and accuracy measures for the predictions.

trainedModel: an object of MLSeq.train class. It contains the trained model. See notes for details.

trainParameters: a list with training parameters from final model. These parameters are used for test set before predicting class labels.

call: a call object for classification task.

Note

method, transformation, normalization slots give the information on classifier, transformation and normalization techniques. Since all possible pairs of transformation and normalization are not available in practice, we specify appropriate transformations and normalization techniques with preProcessing argument in classify function. Finally, the information on normalization and transformation is extracted from preProcessing argument.

MLSeq.train is a union class of train from caret package, voom.train and discrete.train from MLSeq package. See related class manuals for details.

See Also

train, voom.train-class, discrete.train-class

modelInfo *Accessors for the 'modelInfo' slot of an* MLSeq *object*

Description

This slot stores all the information about classification model.

Usage

```
modelInfo(object)

## S4 method for signature 'MLSeq'
modelInfo(object)
```

Arguments

object	an MLSeq object.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

modelInfo(cart)
```

```
## End(Not run)
```

`normalization` *Accessors for the 'normalization' slot.*

Description

This slot stores the name of normalization method which is used while normalizing the count data such as "deseq", "tmm" or "none"

Usage

```
normalization(object)

normalization(object) <- value

## S4 method for signature 'MLSeq'
normalization(object)

## S4 method for signature 'MLSeqModelInfo'
normalization(object)

## S4 replacement method for signature 'MLSeq,character'
normalization(object) <- value
```

Arguments

`object`	an `MLSeq` or `MLSeqModelInfo` object.
`value`	a character string. One of the available normalization methods for voom-based classifiers.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
```

```
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

normalization(cart)

## End(Not run)
```

`plot` *Plot accuracy results from 'MLSeq' object*

Description

This generic function is used to plot accuracy results from 'MLSeq' object returned by `classify` function.

Usage

```
## S3 method for class 'MLSeq'
plot(x, y, ...)

## S4 method for signature 'MLSeq,ANY'
plot(x, y, ...)
```

Arguments

`x`	an `MLSeq` object returned from `classify` function.
`y`	this parameter is not used. Deprecated.
`...`	further arguements. Deprecated.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus and Ahmet Ozturk

predict *Extract predictions from* classify() *object*

Description

This function predicts the class labels of test data for a given model.

predictClassify and predict functions return the predicted class information along with trained model. Predicted values are given either as class labels or estimated probabilities of each class for each sample. If type = "raw", as can be seen in the example below, the predictions are extracted as raw class labels.In order to extract estimated class probabilities, one should follow the steps below:

- set classProbs = TRUE within control arguement in classify
- set type = "prob" within predictClassify

Usage

```
## S3 method for class 'MLSeq'
predict(object, test.data, ...)

predictClassify(object, test.data, ...)

## S4 method for signature 'MLSeq'
predict(object, test.data, ...)
```

Arguments

object	a model of MLSeq class returned by classify
test.data	a DESeqDataSet instance of new observations.
...	further arguments to be passed to or from methods. These arguements are used in predict.train from caret package.

Value

MLSeqObject an MLSeq object returned from classify. See details.

Predictions a data frame or vector including either the predicted class probabilities or class labels of given test data.

Note

predictClassify(...) function was used in MLSeq up to package version 1.14.x. This function is alliased with generic function predict. In the upcoming versions of MLSeq package, predictClassify function will be ommitted. Default function for predicting new observations will be predict from version 1.16.x and later.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus, Ahmet Ozturk and Ahmet Ergun Karaagaoglu

See Also

classify, train, trainControl

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

# test set
data.test <- data[ ,ind]
data.test <- as.matrix(data.test + 1)
classts <- data.frame(condition=class[ind, ])

data.testS4 <- DESeqDataSetFromMatrix(countData = data.test,
                                      colData = classts, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))
cart

# predicted classes of test samples for CART method (class probabilities)
pred.cart = predictClassify(cart, data.testS4, type = "prob")
pred.cart

# predicted classes of test samples for RF method (class labels)
pred.cart = predictClassify(cart, data.testS4, type = "raw")
pred.cart

## End(Not run)
```

preProcessing *Accessors for the 'preProcessing' slot of an* MLSeq *object*

Description

MLSeq package benefits from DESeqDataSet structure from bioconductor package DESeq2 for storing gene expression data in a comprehensive structure. This object is used as an input for classification task through classify. The input is stored in inputObject slot of MLSeq object.

Usage

```
preProcessing(object)

preProcessing(object) <- value

## S4 method for signature 'MLSeq'
preProcessing(object)

## S4 replacement method for signature 'MLSeq,character'
preProcessing(object) <- value
```

Arguments

object	an MLSeq object.
value	a character string. Which preProcessing should be replaced with current one?

See Also

classify, DESeqDataSet

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
```

```
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

preProcessing(cart)

## End(Not run)
```

`print.confMat` *Print method for confusion matrix*

Description

This function prints the confusion matrix of the model.

Usage

```
## S3 method for class 'confMat'
print(x, ..., mode = x$mode, digits = max(3,
  getOption("digits") - 3))

## S4 method for signature 'confMat'
print(x, ..., mode = x$mode, digits = max(3,
  getOption("digits") - 3))
```

Arguments

`x`	an object of class `confMat`
`...`	further arguments to be passed to `print.table`
`mode`	see `print.confusionMatrix`
`digits`	see `print.confusionMatrix`

ref *Accessors for the 'ref' slot.*

Description

This slot stores the information about reference category. Confusion matrix and related statistics are calculated using the user-defined reference category.

Usage

```
ref(object)

ref(object) <- value

## S4 method for signature 'MLSeq'
ref(object)

## S4 method for signature 'MLSeqModelInfo'
ref(object)

## S4 replacement method for signature 'MLSeq,character'
ref(object) <- value
```

Arguments

object	an MLSeq or MLSeqModelInfo object.
value	a character string. Select reference category for class labels.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
```

```
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                   colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

ref(cart)

## End(Not run)
```

`selectedGenes` *Accessors for the 'selectedGenes'.*

Description

This slot stores the name of selected genes which are used in the classifier. The trained model is stored in slot `trainedModel`. See `trained` for details.

Usage

```
selectedGenes(object)

## S4 method for signature 'MLSeq'
selectedGenes(object)
```

Arguments

`object`	an `MLSeq` object.

See Also

`trained`

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
```

```
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

selectedGenes(cart)

## End(Not run)
```

---

show *Show method for MLSeq objects*

Description

Prints out the information from the trained model using `classify` function.

Usage

```
show.MLSeq(object)

## S4 method for signature 'MLSeq'
show(object)

## S4 method for signature 'MLSeqModelInfo'
show(object)

## S4 method for signature 'MLSeqMetaData'
show(object)

## S4 method for signature 'voom.train'
show(object)

## S4 method for signature 'discrete.train'
show(object)
```

Arguments

`object`	an `MLSeq` object returned from `classify` function.

See Also

classify

---

trained *Accessors for the 'trainedModel' slot.*

Description

This slot stores the trained model. This object is returned from train function in caret package. Any further request using caret functions is available for trainedModel since this object is in the same class as the returned object from train. See train for details.

Usage

```
trained(object)

## S4 method for signature 'MLSeq'
trained(object)

## S4 method for signature 'MLSeqModelInfo'
trained(object)
```

Arguments

object	an MLSeq or MLSeqModelInfo object.

See Also

train.default, voom.train-class, discrete.train-class

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])
```

```
# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

trained(cart)

## End(Not run)
```

`trainParameters` *Accessors for the 'trainParameters' slot.*

Description

This slot stores the transformation and normalization parameters from train set. These parameters are used to normalize and transform test set using train set parameters.

Usage

```
trainParameters(object)

## S4 method for signature 'MLSeq'
trainParameters(object)

## S4 method for signature 'MLSeqModelInfo'
trainParameters(object)
```

Arguments

`object`	an `MLSeq` or `MLSeqModelInfo` object.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features
```

```
# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

trainParameters(cart)

## End(Not run)
```

---

transformation *Accessors for the 'transformation' slot.*

Description

This slot stores the name of transformation method which is used while transforming the count data (e.g "vst", "rlog", etc.)

Usage

```
transformation(object)

## S4 method for signature 'MLSeq'
transformation(object)

## S4 method for signature 'MLSeqModelInfo'
transformation(object)
```

Arguments

object	an MLSeq or MLSeqModelInfo object.

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
```

```
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))

transformation(cart)

## End(Not run)
```

---

update *Update* MLSeq *objects returnd from* classify()

Description

This function updates the MLSeq object. If one of the options is changed inside MLSeq object, it should be updated to pass its effecs into classification results.

Usage

```
## S3 method for class 'MLSeq'
update(object, ..., env = .GlobalEnv)

## S4 method for signature 'MLSeq'
update(object, ..., env = .GlobalEnv)
```

Arguments

object	a model of MLSeq class returned by classify
...	optional arguements passed to classify function.
env	an environment. Define the environment where the trained model is stored.

Value

same object as an MLSeq object returned from `classify`.

Note

When an `MLSeq` object is updated, new results are updated on the given object. The results before update process are lost when update is done. To keep the results before update, one should copy the MLSeq object to a new object in global environment.

See Also

`classify`

Examples

```
## Not run:
library(DESeq2)
data(cervical)

# a subset of cervical data with first 150 features.
data <- cervical[c(1:150), ]

# defining sample classes.
class <- data.frame(condition = factor(rep(c("N","T"), c(29, 29))))

n <- ncol(data)  # number of samples
p <- nrow(data)  # number of features

# number of samples for test set (30% test, 70% train).
nTest <- ceiling(n*0.3)
ind <- sample(n, nTest, FALSE)

# train set
data.train <- data[ ,-ind]
data.train <- as.matrix(data.train + 1)
classtr <- data.frame(condition = class[-ind, ])

# train set in S4 class
data.trainS4 <- DESeqDataSetFromMatrix(countData = data.train,
                  colData = classtr, formula(~ 1))

# test set
data.test <- data[ ,ind]
data.test <- as.matrix(data.test + 1)
classts <- data.frame(condition=class[ind, ])

data.testS4 <- DESeqDataSetFromMatrix(countData = data.test,
                                      colData = classts, formula(~ 1))

## Number of repeats (repeats) might change model accuracies ##
# Classification and Regression Tree (CART) Classification
cart <- classify(data = data.trainS4, method = "rpart",
          ref = "T", preProcessing = "deseq-vst",
          control = trainControl(method = "repeatedcv", number = 5,
                                 repeats = 3, classProbs = TRUE))
cart
```

```
# Change classification model into "Random Forests" (rf)
method(cart) <- "rf"
rf <- update(cart)

rf

## End(Not run)
```

`voom.train-class` `voom.train` *object*

Description

This object is the subclass for the `MLSeq.train` class. It contains trained model information for voom based classifiers, i.e. "voomDLDA", "voomDQDA" and "voomNSC".

Slots

`weigtedStats`**:** a list with elements of weighted statistics which are used for training the model. Weights are calculated from voom transformation.

`foldInfo`**:** a list containing information on cross-validated folds.

`control`**:** a list with control parameters for voom based classifiers.

`tuningResults`**:** a list. It stores the cross-validation results for tuning parameter(s).

`finalModel`**:** a list. It stores results for trained model with optimum parameters.

`callInfo`**:** a list. call info for related function.

`voomControl` *Define controlling parameters for voom-based classifiers*

Description

This function sets the control parameters for voom based classifiers while training the model.

Usage

```
voomControl(method = "repeatedcv", number = 5, repeats = 10,
  tuneLength = 10)
```

Arguments

`method`	validation method. Support repeated cross validation only ("repeatedcv").
`number`	a positive integer. Number of folds.
`repeats`	a positive integer. Number of repeats.
`tuneLength`	a positive integer. If there is a tuning parameter in the classifier, this value is used to define total number of tuning parameter to be searched.

Author(s)

Gokmen Zararsiz, Dincer Goksuluk, Selcuk Korkmaz, Vahap Eldem, Bernd Klaus, Ahmet Ozturk and Ahmet Ergun Karaagaoglu

See Also

`classify`, `trainControl`, `discreteControl`

Examples

```
1L
```

Index