Package ‘nlcv’

June 29, 2018

**Type** Package

**Title** Nested Loop Cross Validation

**Version** 0.3.5

**Date** 2018-06-29

**Author** Willem Talloen, Tobias Verbeke

**Maintainer** Laure Cougnaud <laure.cougnaud@openanalytics.eu>

**Description**

Nested loop cross validation for classification purposes for misclassification error rate estimation. The package supports several methodologies for feature selection: random forest, Student t-test, limma, and provides an interface to the following classification methods in the 'MLInterfaces' package: linear, quadratic discriminant analyses, random forest, bagging, prediction analysis for microarray, generalized linear model, support vector machine (svm and ksvm). Visualizations to assess the quality of the classifier are included: plot of the ranks of the features, scores plot for a specific classification algorithm and number of features, misclassification rate for the different number of features and classification algorithms tested and ROC plot. For further details about the methodology, please check:


**Depends** R (>= 2.10), a4Core, MLInterfaces (>= 1.22.0), xtable

**Imports** limma, MASS, methods, graphics, Biobase, multtest, RColorBrewer, pamr, randomForest, ROCR, ipred, e1071, kernlab

**Suggests** RUnit, ALL

**License** GPL-3

**RoxygenNote** 6.0.1.9000

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-06-29 21:49:59 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>compareOrig</td>
<td>2</td>
</tr>
<tr>
<td>confusionMatrix.nlcv</td>
<td>3</td>
</tr>
<tr>
<td>inTrainingSample</td>
<td>4</td>
</tr>
<tr>
<td>limmaTwoGroups</td>
<td>5</td>
</tr>
<tr>
<td>mcrPlot</td>
<td>5</td>
</tr>
<tr>
<td>nlcv</td>
<td>7</td>
</tr>
<tr>
<td>nlcvRF_R</td>
<td>8</td>
</tr>
<tr>
<td>nlcvRF_SHS</td>
<td>9</td>
</tr>
<tr>
<td>nlcvRF_SS</td>
<td>9</td>
</tr>
<tr>
<td>nlcvRF_WHS</td>
<td>10</td>
</tr>
<tr>
<td>nlcvRF_WS</td>
<td>10</td>
</tr>
<tr>
<td>nlcvTT_R</td>
<td>11</td>
</tr>
<tr>
<td>nlcvTT_SHS</td>
<td>11</td>
</tr>
<tr>
<td>nlcvTT_SS</td>
<td>11</td>
</tr>
<tr>
<td>nlcvTT_WHS</td>
<td>12</td>
</tr>
<tr>
<td>nlcvTT_WS</td>
<td>12</td>
</tr>
<tr>
<td>nldaI</td>
<td>12</td>
</tr>
<tr>
<td>pamrI</td>
<td>13</td>
</tr>
<tr>
<td>pamrConverter</td>
<td>14</td>
</tr>
<tr>
<td>pamrML</td>
<td>14</td>
</tr>
<tr>
<td>pamrTrain</td>
<td>16</td>
</tr>
<tr>
<td>predict.pamrML</td>
<td>17</td>
</tr>
<tr>
<td>print.nlcvConfusionMatrix</td>
<td>17</td>
</tr>
<tr>
<td>print.pamrML</td>
<td>18</td>
</tr>
<tr>
<td>print.summary.mcrPlot</td>
<td>18</td>
</tr>
<tr>
<td>rankDistributionPlot</td>
<td>19</td>
</tr>
<tr>
<td>rocPlot</td>
<td>19</td>
</tr>
<tr>
<td>scoresPlot</td>
<td>20</td>
</tr>
<tr>
<td>summary.mcrPlot</td>
<td>21</td>
</tr>
<tr>
<td>topTable-methods</td>
<td>22</td>
</tr>
<tr>
<td>xtable.confusionMatrix</td>
<td>23</td>
</tr>
<tr>
<td>xtable.summary.mcrPlot</td>
<td>24</td>
</tr>
</tbody>
</table>

Index 25

**compareOrig**  
function to compare the original matrix of correct classes to each component of the output object for a certain classifier

**Description**

function to compare the original matrix of correct classes to each component of the output object for a certain classifier
Usage
   compareOrig(nlcvObj, techn)

Arguments
   nlcvObj      return of the nlcv function
   techn       technique for which the comparison to correct classes should be made

Value
   list with for each number of features selected, a matrix of logical values indicating whether the
   classifier results correspond (TRUE) or not (FALSE) to the original values to be classified

Description
   The observed and predicted classes are cross-tabulated for a given classification technique used in
   the nested loop cross validation. The predicted class that is used to construct the confusion matrix
   is the class that was predicted most of the time (>= 50%) across all runs of the nested loop.

Usage
   ## S3 method for class 'nlcv'
   confusionMatrix(x, tech, proportions = TRUE, ...)

Arguments
   x          object for which a confusionMatrix should be produced, e.g. one produced by
   the nlcv function; for the print method, it is the object to be printed
   tech       string indicating the classification technique for which the confusion matrix
               should be returned
   proportions logical indicating whether the cells of the matrix should contain proportions
               (TRUE) or raw counts (FALSE)
   ...        Dots argument to pass additional parameters to the confusionMatrix or print
               methods

Value
   confusionMatrix produces an object of class confusionMatrix which directly inherits from the
   ftable class (representing the confusion matrix)

Author(s)
   Willem Talloen and Tobias Verbeke
inTrainingSample  

Function to define a learning sample based on balanced sampling

Description

This function takes in a factor with class labels of the total dataset, draws a sample (balanced with respect to the different levels of the factor) and returns a logical vector indicating whether the observation is in the learning sample (TRUE) or not (FALSE).

Usage

inTrainingSample(y, propTraining = 2/3, classdist = c("balanced", "unbalanced"))

Arguments

- y: factor with the class labels for the total data set
- propTraining: proportion of the data that should be in a training set; the default value is 2/3.
- classdist: distribution of classes; allows to indicate whether your distribution 'balanced' or 'unbalanced'. The sampling strategy for each run is adapted accordingly.

Value

logical vector indicating for each observation in y whether the observation is in the learning sample (TRUE) or not (FALSE)

Author(s)

 Willem Talloen and Tobias Verbeke

Examples

### this example demonstrates the logic of sampling in case of unbalanced distribution of classes
y <- factor(c(rep("A", 21), rep("B", 80)))
nlcv:::inTrainingSample(y, 2/3, "unbalanced")

table(y[nlcv:::inTrainingSample(y, 2/3, "unbalanced")]) # should be 14, 14 (for A, B resp.)
table(y[!nlcv:::inTrainingSample(y, 2/3, "unbalanced")]) # should be 7, 66 (for A, B resp.)
limmaTwoGroups

Description

Wrapper around limma for the comparison of two groups

Usage

limmaTwoGroups(object, group)

Arguments

object object of class ExpressionSet

Arguments

group string indicating the variable defining the two groups to be compared

Details

Basically, the wrapper combines the lmFit, eBayes and topTable steps

Value

topTable output for the second (i.e. slope) coefficient of the linear model.

Author(s)

Tobias Verbeke

References


http://www.bepress.com/sagmb/vol3/iss1/art3

mcrPlot

Description

plots for each classification technique and a given number of features used the mean misclassification rate (mcr) and its standard error across all runs of the nested loop cross-validation.

Usage

mcrPlot(nlcvObj, plot = TRUE, optimalDots = TRUE, rescale = FALSE, layout = TRUE, ...)
Arguments

**nlcvObj**  
Object of class `nlcv` as produced by the `nlcv` function

**plot**  
logical. If FALSE, nothing is plotted.

**optimalDots**  
Boolean indicating whether dots should be displayed on a panel below the graph to mark the optimal number of features for a given classification technique

**rescale**  
if TRUE, the upper limit of y-axis is dependent on the data (maximum mcr value); defaults to FALSE which implies limits c(0,1)

**layout**  
boolean indicating whether `mcrPlot` should prespecify a layout for a single plot (default, TRUE) or whether the user takes care of the layout (FALSE)

**...**  
Dots argument to pass additional graphical parameters (such as main) to the `plot` function

Value

An MCR plot is output to the device of choice. The dots represent the mean MCR across runs. The vertical lines below and above the dots represent the standard deviation of the MCR values across runs.

Below the plot coloured solid dots (one for each classification technique) indicate for which number of features a given technique reached its minimum MCR.

The function invisibly returns an object of class `mcrPlot` which is a list with components:

- **meanMcrMatrix** matrix with for each number of features (rows) and classification technique (columns) the mean of the MCR values across all runs of the `nlcv` procedure.

- **sdMcrMatrix** matrix with for each number of features (rows) and classification technique (columns) the sd of the MCR values across all runs of the `nlcv` procedure.

The `summary` method for the `mcrPlot` object returns a matrix with for each classification technique, the optimal number of features as well as the associated mean MCR and standard deviation of the MCR values.

Author(s)

Willem Talloen and Tobias Verbeke

See Also

`nlcv`
Description
This function first proceeds to a feature selection and then applies five different classification algorithms.

Usage
nlcv(eset, classVar = "type", nRuns = 2, propTraining = 2/3, classdist = c("balanced", "unbalanced"), nFeatures = c(2, 3, 5, 7, 10, 15, 20, 25, 30, 35), fsMethod = c("randomForest", "t.test", "limma", "none"), classifMethods = c("dlda", "randomForest", "bagg", "pam", "svm"), fsPar = NULL, initialGenes = seq(length.out = nrow(eset)), geneID = "ID", storeTestScores = FALSE, verbose = FALSE, seed = 123)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eset</td>
<td>ExpressionSet object containing the genes to classify</td>
</tr>
<tr>
<td>classVar</td>
<td>String giving the name of the variable containing the observed class labels, should be contained in the phenoData of eset</td>
</tr>
<tr>
<td>nRuns</td>
<td>Number of runs for the outer loop of the cross-validation</td>
</tr>
<tr>
<td>propTraining</td>
<td>Proportion of the observations to be assigned to the training set. By default propTraining = 2/3.</td>
</tr>
<tr>
<td>classdist</td>
<td>distribution of classes; allows to indicate whether your distribution is 'balanced' or 'unbalanced'. The sampling strategy for each run is adapted accordingly.</td>
</tr>
<tr>
<td>nFeatures</td>
<td>Numeric vector with the number of features to be selected from the features kept by the feature selection method. For each number n specified in this vector the classification algorithms will be run using only the top n features.</td>
</tr>
<tr>
<td>fsMethod</td>
<td>Feature selection method; one of &quot;randomForest&quot; (default), &quot;t.test&quot;, &quot;limma&quot; or &quot;none&quot;.</td>
</tr>
<tr>
<td>classifMethods</td>
<td>character vector with the classification methods to be used in the analysis; elements can be chosen among &quot;dlda&quot;, &quot;randomForest&quot;, &quot;bagg&quot;, &quot;pam&quot;, &quot;svm&quot;, &quot;glm&quot;, &quot;lda&quot;, &quot;nlda&quot;, &quot;dlda&quot;, &quot;ksvm&quot;. The first 5 methods are selected by default</td>
</tr>
<tr>
<td>fsPar</td>
<td>List of further parameters to pass to the feature selection method; currently the default for &quot;randomForest&quot; is an empty list() whereas for &quot;t.test&quot;, one can specify the particular test to be used (the default being list(test = &quot;f&quot;).</td>
</tr>
<tr>
<td>initialGenes</td>
<td>Initial subset of genes in the ExpressionSet on which to apply the nested loop cross validation procedure. By default all genes are selected.</td>
</tr>
<tr>
<td>geneID</td>
<td>string representing the name of the gene ID variable in the fData of the expression set to use; this argument was added for people who use e.g. both Entrez IDs and Ensemble gene IDs</td>
</tr>
</tbody>
</table>
storeTestScores should the test scores be stored in the nlcv object? Defaults to FALSE

verbose Should the output be verbose (TRUE) or not (FALSE).

seed integer with seed, set at the start of the cross-validation.

Value

The result is an object of class 'nlcv'. It is a list with two components, output and features.

Output component is a list of five components, one for each classification algorithm used. Each of these components has as many components as there are elements in the nFeatures vector. These components contain both the error rates for each run (component errorRate) and the predicted labels for each run (character matrix labelsMat).

The features list is a list with as many components as there are runs. For each run, a named vector is given with the variable importance measure for each gene. For t test based feature selection, P-values are used; for random forest based feature selection the variable importance measure is given.

Note

The variable importance measure used is the third column of the output returned by the randomForest function.

Author(s)

Willem Talloen and Tobias Verbeke

nlcvRF_R

Description

This data set contains the nlcv results of selection of features with random forest on a randomly generated dataset.

Usage

nlcvRF_R

Format

nlcv object
Description

This data set contains the nlcv results of selection of features with random forest on a dataset with strong hetero signal.

Usage

nlcvRF_SHS

Format

nlcv object

Description

This data set contains the nlcv results of selection of features with random forest on a dataset with strong signal.

Usage

nlcvRF_SS

Format

nlcv object
nlcvRF_WS  

* nlcv results on weak signal data with random forest feature selection *

---

**Description**

This data set contains the `nlcv` results of selection of features with random forest on a weak signal dataset.

**Usage**

`nlcvRF_WHs`

**Format**

`nlcv` object

---

`nlcvRF_WS`  

* nlcv results on weak hetero signal data with random forest feature selection *

---

**Description**

This data set contains the `nlcv` results of selection of features with random forest on a weak hetero signal dataset.

**Usage**

`nlcvRF_WS`

**Format**

`nlcv` object
\begin{itemize}
\item \texttt{n1cvTT\_R} \textit{n1cv results on random data with t-test feature selection}
\item \texttt{n1cvTT\_SHS} \textit{n1cv results on strong hetero signal data with t-test feature selection}
\item \texttt{n1cvTT\_SS} \textit{n1cv results on strong signal data with t-test feature selection}
\end{itemize}

\textbf{Description}

This data set contains the \texttt{n1cv} results of selection of features with t-test on a randomly generated dataset.

\textbf{Usage}

\texttt{n1cvTT\_R}

\textbf{Format}

\texttt{n1cv object}

\textbf{Description}

This data set contains the \texttt{n1cv} results of selection of features with t-test on a dataset with strong hetero signal.

\textbf{Usage}

\texttt{n1cvTT\_SHS}

\textbf{Format}

\texttt{n1cv object}

\textbf{Description}

This data set contains the \texttt{n1cv} results of selection of features with t-test on a dataset with strong signal.

\textbf{Usage}

\texttt{n1cvTT\_SS}

\textbf{Format}

\texttt{n1cv object}
nldai

nlcvTT_WHS

**Description**

This data set contains the nlcv results of selection of features with t-test on a weak signal dataset.

**Usage**

nlcvTT_WHS

**Format**

nlcv object

nlcvTT_WS

**Description**

This data set contains the nlcv results of selection of features with t-test on a weak hetero signal dataset.

**Usage**

nlcvTT_WS

**Format**

nlcv object

nldaI

**Description**

This interface keeps track of the predictions on the training and test set, contrary to the ldaI interface that is made available in the MLInterfaces package.

**Usage**

nldaI
pamrI

Format

An object of class learnerSchema of length 1.

Details

nldaI is an object of class 'learnerSchema' and can be used as such in calls to MLearn (from MLInterfaces).

See Also

See Also ldaI

pamrI

Instance of a learnerSchema for pamr models

Description

This object is an instance of the learnerSchema object and will be typically used as the method argument of an MLearn call.

Usage

pamrI

Format

An object of class learnerSchema of length 1.

Author(s)

Tobias Verbeke

See Also

MLearn

Examples

```r
set.seed(120)
x <- matrix(rnorm(1000*20), ncol=20)
y <- sample(c(1:4), size=20, replace=TRUE)
alldf <- cbind.data.frame(t(x), y)

# assure it is a factor (otherwise error message)
alldf$y <- factor(alldf$y)
library(MLInterfaces)
(mlobj <- MLearn(y ~ .,
                  data = alldf,
```
pamrML

```
.method = pamrI,
  trainInd = 1:15))
```

pamrIconverter  convert from pamrML to classifierOutput

Description
convert from pamrML to classifierOutput

Usage
pamrIconverter(obj, data, trainInd)

Arguments

- **obj**: object as returned by pamrML i.e. of class pamrML
- **data**: original data used as input for MLearn
- **trainInd**: training indices used as input to MLearn

Value
object of class classifierOutput

pamrML  Wrapper function around the pamr* functions

Description
The pamrML functions are wrappers around pamr.train and pamr.predict that provide a more classical R modelling interface than the original versions.

Usage
pamrML(formula, data, ...)

Arguments

- **formula**: model formula
- **data**: data frame
- **...**: argument for the parmTrain function
Details

The name of the response variable is kept as an attribute in the \texttt{pamrML} object to allow for predict methods that can be easily used for writing converter functions for use in the \texttt{MLInterfaces} framework.

Value

For \texttt{pamrML} an object of class \texttt{pamrML} which adds an attribute to the original object returned by \texttt{pamr.train} (or \texttt{pamrTrain}).

The \texttt{print} method lists the names of the different components of the \texttt{pamrML} object.

The \texttt{predict} method returns a vector of predicted values.

Author(s)

Tobias Verbeke

See Also

\texttt{pamr.train}, \texttt{pamr.predict}

Examples

```r
set.seed(120)
x <- matrix(rnorm(1000*20), ncol=20)
y <- sample(c(1:4), size=20, replace=TRUE)
# for original pam
mydata <- list(x=x, y=y)
mytraindata <- list(x=x[1:15], y=factor(y[1:15]))
mytestdata <- list(x = x[16:20], y = factor(y[16:20]))

# for formula-based methods including \texttt{pamrML}
alldf <- cbind.data.frame(t(mydata$x), y)
traindf <- cbind.data.frame(t(mytraindata$x), y = mytraindata$y)
testdf <- cbind.data.frame(t(mytestdata$x), y = mytestdata$y)

### create \texttt{pamrML} object
pamrMLobj <- pamrML(y ~ ., traindf)
pamrMLobj

### test predict method
predict(object = pamrMLobj, newdata = testdf,
        threshold = 1)  # threshold compulsory
```
Function providing a formula interface to pamr.train

Description
Function that provides a classical R modelling interface, using a formula and data argument

Usage
```
pamrTrain(formula, data, ...)
```

Arguments
- `formula`: formula
- `data`: data frame
- `...`: further arguments to be passed to `pamr.train`

Value
Object that is perfectly identical to the object returned by `pamr.train`

Author(s)
Tobias Verbeke

See Also
- `pamr.train`

Examples
```
set.seed(120)
x <- matrix(rnorm(1000*20), ncol=20)
y <- sample(c(1:4), size=20, replace=TRUE)
alldf <- cbind.data.frame(t(x), y)
pamrTrain(y ~ ., alldf)
```
**predict.pamrML**  

*predict pamrML object*

---

**Description**

predict pamrML object

**Usage**

```r
## S3 method for class 'pamrML'
predict(object, newdata, ...)
```

**Arguments**

- `object`: pamrML object
- `newdata`: new data
- `...`: additional parameters for the `pamr.predict` function

**Value**

output of the `pamr.predict` function

---

**print.nlcvConfusionMatrix**  

*print object nlcvConfusionMatrix*

---

**Description**

print object nlcvConfusionMatrix

**Usage**

```r
## S3 method for class 'nlcvConfusionMatrix'
print(x, ...)
```

**Arguments**

- `x`: object of class `nlcvConfusionMatrix`
- `...`: additional parameters for the `print` function

**Value**

no returned value, the object is printed in the output
print.pamrML  

\textit{print} \texttt{pamrML} \textit{object}

\section*{Description}

\textit{print} \texttt{pamrML} \textit{object}

\section*{Usage}

\begin{verbatim}
## S3 method for class 'pamrML'
print(x, ...)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{1cm} \texttt{object of class pamrML}
  \item \texttt{...} \hspace{1cm} \texttt{additional parameters for the print function}
\end{itemize}

print.summary.mcrPlot  

\textit{print} function for \texttt{summary.mcrPlot} \textit{object}

\section*{Description}

\textit{print} function for \texttt{summary.mcrPlot} \textit{object}

\section*{Usage}

\begin{verbatim}
## S3 method for class 'summary.mcrPlot'
print(x, digits = 2, ...)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{x}  \hspace{1cm} \texttt{Object of class 'summary.mcrPlot' as produced by the function of the same name}
  \item \texttt{digits} \hspace{1cm} \texttt{number of digits to be passed to the default print method}
  \item \texttt{...} \hspace{1cm} \texttt{additional parameters for the print.default function}
\end{itemize}
**rankDistributionPlot**  
*Plot the Distribution of Ranks of Features Across *nlcv* Runs*

**Description**

This plot offers an overview of the distribution of the ranks of the *n* best-ranked features. The order of the features is determined by the median rank of the feature across all *nlcv* runs.

**Usage**

```r
rkDistributionPlot(nlcvObj, n = 5, ...)
```

**Arguments**

- `nlcvObj`  
  object of class *nlcv* as produced by the *nlcv* function
- `n`  
  number of features for which the distribution should be displayed
- `...`  
  additional arguments to the boxplot functions (such as `main`, `sub`, etc.)

**Value**

For each of the *n* features, a boxplot is displayed.

**Author(s)**

Willem Talloen and Tobias Verbeke

**Examples**

```r
{
  data(nlcvRF_SS)
  rankDistributionPlot(nlcvRF_SS, n = 9)
}
```

**rocPlot**  
*Produce a ROC plot for a classification model belonging to a given technique and with a given number of features.*

**Description**

Produce a ROC plot for a classification model belonging to a given technique and with a given number of features.

**Usage**

```r
rocPlot(nlcvObj, tech, nfeat, main = NULL, globalAUCcol = "#FF9900", ...)
```
scoresPlot

Function to Plot a Scores Plot

Description

Function to plot, for a given nested loop cross-validation object, a given classification technique and a given number of features used for the classification, the scores plot. This plot displays the proportion of correctly-classified per sample across all runs of the nested loop cross-validation. The class membership of the samples is displayed using a colored strip (with legend below the plot).

Usage

scoresPlot(nlcvObj, tech, nfeat, plot = TRUE, barPlot = FALSE, layout = TRUE, main = NULL, sub = NULL, ...)

Arguments

nlcvObj Object of class 'nlcv' as produced by the nlcv function
tech string denoting the classification technique used; one of 'dllda', 'bagg', 'pam', 'rf', or 'ksvm'.
nfeat integer giving the number of features; this number should be part of the initial set of number of features that was specified during the nested loop cross-validation (nFeatures argument of the nlcv function)
plot logical. If FALSE, nothing is plotted.
barPlot Should a barplot be drawn (TRUE) or the alternative MCREstimate-type scores plot (the default, FALSE).
### summary.mcrPlot

#### layout
Boolean indicating whether `mcrPlot` should prespecify a layout for a single plot (default, `TRUE`) or whether the user takes care of the layout (`FALSE`).

#### main
Main title for the scores plot; if not supplied, 'Scores Plot' is used as a default.

#### sub
Subtitle for the scores plot; if not supplied, the classification technique and the chosen number of features are displayed.

#### ... Additional graphical parameters to pass to the plot function.

### Value
A scores plot is displayed (for the device specified).

The function invisibly returns a named vector containing (for each sample) the proportion of times the sample was correctly classified (for a given technique and a given number of features used).

### Author(s)
Willem Talloen and Tobias Verbeke

### Description
`summary` function for `mcrPlot` object

### Usage
```r
## S3 method for class 'mcrPlot'
summary(object, ...)  
```

### Arguments
- **object**: Object of class 'mcrPlot' as produced by the function of the same name
- **...**: Additional arguments, not used here
Methods for topTable. topTable extracts the top n most important features for a given classification or regression procedure.

Usage

```r
## S4 method for signature 'nlcv'
topTable(fit, n = 5, method = "percentage")
```

Arguments

- `fit` object resulting from a classification or regression procedure
- `n` number of features that one wants to extract from a table that ranks all features according to their importance in the classification or regression model
- `method` method used to rank the features; one of `percentage` (percentage of runs the feature is selected in the top n), `meanrank` (mean rank of the feature across runs) or `medianrank` (median rank of the feature across runs); `percentage` is the default method

Details

The top n features are extracted across all runs of the nested loop cross-validation. After ranking on their frequency of selection, the top n are retained and returned.

Value

a data frame of one column (`percentage`) with percentages reflecting the frequency of selection of a feature in the top n across all runs; the features are sorted on decreasing frequency.

Methods

- `nlcv`
  nlcv objects are produced by `nlcv`

Author(s)

- `fit = "nlcv"` Willem Talloen and Tobias Verbeke

Examples

```r
data(nlcvRF_SS)
topTable(nlcvRF_SS, n = 7, method = "medianrank")
```
Description

xtable method for confusionMatrix objects

Usage

## S3 method for class 'confusionMatrix'
xtable(x, caption = NULL, label = NULL,
       align = NULL, digits = NULL, display = NULL, ...)

Arguments

- **x**: object of class 'confusionMatrix' as produced by the confusionMatrix
- **caption**: LaTeX caption, see the xtable help page
- **label**: LaTeX label, see the xtable help page
- **align**: alignment specification, see the xtable help page
- **digits**: number of digits to display, see the xtable help page
- **display**: format of the columns, see the xtable help page
- **...**: additional arguments to be passed to xtable

Value

LaTeX table representing the confusion matrix

Author(s)

Willem Talloen and Tobias Verbeke

See Also

confusionMatrix, xtable
Description

xtable method for summary.mcrPlot objects

Usage

```r
## S3 method for class 'summary.mcrPlot'
xtable(x, caption = NULL, label = NULL,
       align = NULL, digits = NULL, display = NULL, ...)
```

Arguments

- `x`: object of class 'summary.mcrPlot' as produced by the summary.mcrPlot
- `caption`: LaTeX caption, see the xtable help page
- `label`: LaTeX label, see the xtable help page
- `align`: alignment specification, see the xtable help page
- `digits`: number of digits to display, see the xtable help page
- `display`: format of the columns, see the xtable help page
- `...`: additional arguments to be passed to xtable

Value

LaTeX table representing the summary of the mcrPlot output, i.e. the optimal number of features, the mean MCR and the standard deviation on the MCR for each of the classification methods used.

Author(s)

Willem Talloen and Tobias Verbeke

See Also

summary.mcrPlot, mcrPlot, xtable

Examples

```r
data(nlcvRF_SS)
mp <- mcrPlot(nlcvRF_SS, plot = FALSE)
smp <- summary(mp)
xtable(smp)
```
Index

*Topic **htest**
  nlcv, 7

*Topic **manip**
  confusionMatrix.nlcv, 3
  inTrainingSample, 4
  mcrPlot, 5
  nldaI, 12
  rankDistributionPlot, 19
  scoresPlot, 20
  topTable-methods, 22
  xtable.confusionMatrix, 23
  xtable.summary.mcrPlot, 24

*Topic **methods**
  topTable-methods, 22

*Topic **models**
  limmaTwoGroups, 5
  pamrI, 13
  pamrML, 14
  pamrTrain, 16

*Topic **regression**
  limmaTwoGroups, 5

  compareOrig, 2
  confusionMatrix, 23
  confusionMatrix.nlcv, 3

  inTrainingSample, 4

  ldaI, 13
  limmaTwoGroups, 5

  mcrPlot, 5, 24
  MLearn, 13

  nlcv, 3, 6, 7, 8–12
  nlcvRF_R, 8
  nlcvRF_SHS, 9
  nlcvRF_SS, 9
  nlcvRF_WHS, 10
  nlcvRF_WS, 10
  nlcvTT_R, 11
  nlcvTT_SHS, 11
  nlcvTT_SS, 11
  nlcvTT_WHS, 12
  nlcvTT_WS, 12
  nldaI, 12
  pamr.predict, 15, 17
  pamr.train, 15, 16
  pamrI, 13
  pamrIConverter, 14
  pamrML, 14
  pamrTrain, 16
  predict.pamrML, 17
  print.nlcvConfusionMatrix, 17
  print.pamrML, 18
  print.summary.mcrPlot, 18

  rankDistributionPlot, 19
  rocPlot, 19

  scoresPlot, 20
  summary.mcrPlot, 21, 24

  topTable(topTable-methods), 22
  topTable, nlcv-method
  (topTable-methods), 22
  topTable-methods, 22

  xtable, 23, 24
  xtable.confusionMatrix, 23
  xtable.summary.mcrPlot, 24

25