Package 'Biocomb'

October 12, 2022

Type Package

Title Feature Selection and Classification with the Embedded Validation Procedures for Biomedical Data Analysis

Version 0.4

Date 2018-05-18

Author Natalia Novoselova,Junxi Wang,Frank Pessler,Frank Klawonn

Maintainer Natalia Novoselova <novos65@mail.ru>

Description Contains functions for the data analysis with the emphasis on biological data, including several algorithms for feature ranking, feature selection, classification algorithms with the embedded validation procedures.
The functions can deal with numerical as well as with nominal features. Includes also the functions for calculation
of feature AUC (Area Under the ROC Curve) and HUM (hypervolume under manifold) values and construction 2D- and 3D- ROC curves.
Provides the calculation of Area Above the RCC (AAC) values and construction of Relative Cost Curves
(RCC) to estimate the classifier performance under unequal misclassification costs problem.
There exists the special function to deal with missing values, including different imputing schemes.

License GPL (>= 3)

Imports rgl, MASS, e1071, randomForest, pROC, ROCR, arules, pamr, class, nnet, rpart, FSelector, RWeka, grDevices, graphics, stats, utils

LinkingTo Rcpp

Depends R (>= 2.13.0),gtools,Rcpp (>= 0.12.1)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-05-18 18:33:55 UTC

R topics documented:

Biocomb-package *Tools for Data Mining*

Description

Functions to make the data analysis with the emphasis on biological data. They can deal with both numerical and nominal features. Biocomb includes functions for several feature ranking, feature selection algorithms. The feature ranking is based on several criteria: information gain, symmetrical uncertainty, chi-squared statistic etc. There are a number of features selection algorithms: Chi2 algorithm, based on chi-squared test, fast correlation-based filter algorithm, feature weighting algorithm (RelieF), sequential forward search algorithm (CorrSF), Correlation-based feature selection algorithm (CFS). Package includes several classification algorithms with embedded feature selection and validation schemes. It includes also the functions for calculation of feature AUC (Area Under the ROC Curve) values with statistical significance analysis, calculation of Area Above the RCC (AAC) values. For two- and multi-class problems it is possible to use functions for HUM (hypervolume under manifold) calculation and construction 2D- and 3D- ROC curves. Relative Cost Curves (RCC) are provided to estimate the classifier performance under unequal misclassification costs.
Biocomb has a special function to deal with missing values, including different imputing schemes.

Details

Package:	Biocomb
Type:	Package
Version:	0.3
Date:	2016-08-14
License:	GPL (>= 3)

Biocomb package presents the functions for two stages of data mining process: feature selection and classification. One of the main functions of Biocomb is the `select.process` function. It presents the infrostructure to perform the feature ranking or feature selection for the data set with two or more class labels. Functions `compute.aucs`, `select.inf.gain`, `select.inf.symm` and `select.inf.chi2` calculate the different criterion measure for each feature in the dataset. Function `select.fast.filter` realizes the fast correlation-based filter method. Function `chi2.algorithm` performes Chi2 discretization algorithms with feature selection. Function `select.forward.Corr` is designed for the sequential forward features search according to the correlation measure. Function `select.forward.wrapper` is the realization of the wrapper feature selection method with sequential forward search strategy. The auxiliary function `ProcessData` performs the discretization of the numerical features and is called from the several functions for feature selection. The second main function of the Biocomb is `classifier.loop` which presents the infrastructure for the classifier construction with the embedded feature selection and using the different schemes for the performance validation. The functions `compute.aucs`, `compute.auc.permutation`, `pauc`, `pauclog`, `compute.auc.random` are the functions for calculation of feature AUC (Area Under the ROC Curve) values with statistical significance analysis. The functions `plotRoc.curves` is assigned for the construction of the ROC curve in 2D-space. The functions `cost.curve` plots the RCC and calculates the corresponding AAC to estimate the classifier performance under unequal misclassification costs problem. The function `input_miss` deals with missing value problem and realizes the two methods of missing value imputing. The function `generate.data.miss` allows to generate the dataset with missing values from the input dataset in order to test the algorithms, which are designed to deal with missing values problem. The functions `CalculateHUM_seq`,

CalculateHUM_ROC, CalculateHUM_Plot are for HUM calculation and construction 2D- and 3D-ROC curves.

Function

select.process	Perform the features ranking or features selection
compute.aucs	Calculate the AUC values
select.inf.gain	Calculate the Information Gain criterion
select.inf.symm	Calculate the Symmetrical uncertainty criterion
select.inf.chi2	Calculate the chi-squared statistic
select.fast.filter	Select the feature subset with fast correlation-based filter method
chi2.algorithm	Select the feature subset with Chi2 discretization algorithm.
select.forward.Corr	Select the feature subset with forward search strategy and correlation measure
select.forward.wrapper	Select the feature subset with a wrapper method
ProcessData	Perform the discretization of the numerical features
classifier.loop	Perform the classification with the embedded feature selection
pauc	Calculate the p-values of the statistical significance of the two-class difference
pauclog	Calculate the logarithm of p-values of the statistical significance
compute.auc.permutation	Compute the p-value of the significance of the AUC using the permutation test
compute.auc.random	Compute the p-value of the significance of the AUC using random sample generation
plotRoc.curves	Plot the ROC curve in 2D-space
CalculateHUM_seq	Calculate a maximal HUM value and the corresponding permutation of class labels
CalculateHUM_Ex	Calculate the HUM values with exaustive serach for specified number of class labels
CalculateHUM_ROC	Function to construct and plot the 2D- or 3d-ROC curve
CalcGene	Compute the HUM value for one feature
CalcROC	Compute the point coordinates to plot the 2D- or 3D-ROC curve
CalculateHUM_Plot	Plot the 2D-ROC curve
Calculate3D	Plot the 3D-ROC curve
cost.curve	Plot the RCC and calculate the AAC for unequal misclassification costs
input_miss	Perform the missing values imputation
generate.data.miss	Generate the dataset with missing values

Dataset

This package comes with two simulated datasets and a real dataset of leukemia patients with 72 cases and 101 features. The last feature is the class (disease labels).

Installing and using

To install this package, make sure you are connected to the internet and issue the following command in the R prompt:

```
install.packages("Biocomb")
```

To load the package in R:

```
library(Biocomb)
```

Author(s)

Natalia Novoselova, Junxi Wang,Frank Pessler,Frank Klawonn

Maintainer: Natalia Novoselova <novos65@mail.ru>

References

H. Liu and L. Yu. "Toward Integrating Feature Selection Algorithms for Classification and Clustering", IEEE Trans. on Knowledge and Data Engineering, pdf, 17(4), 491-502, 2005.
L. Yu and H. Liu. "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution". In Proceedings of The Twentieth International Conference on Machine Leaning (ICML-03), Washington, D.C. pp. 856-863. August 21-24, 2003.
Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification?A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.
Olga Montvida and Frank Klawonn Relative cost curves: An alternative to AUC and an extension to 3-class problems,Kybernetika 50 no. 5, 647-660, 2014

See Also

CRAN packages **arules** or **discretization** for feature discretization. CRAN packages **pROC** for ROC curves. CRAN packages **FSelector** for chi-squared test, forward search strategy. CRAN packages **pamr** for nearest shrunken centroid classifier, CRAN packages **MASS**, **e1071**, **randomForest**,**class**, **nnet**, **rpart** are used in this package.

Examples

```
data(data_test)
# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

# Perform the feature selection using the fast correlation-based filter algorithm
disc="MDL"
threshold=0.2
attrs.nominal=numeric()
out=select.fast.filter(data_test,disc.method=disc,threshold=threshold,
attrs.nominal=attrs.nominal)

# Perform the classification with cross-validation of results
out=classifier.loop(data_test,classifiers=c("svm","lda","rf"),
 feature.selection="auc", flag.feature=FALSE,method.cross="fold-crossval")

# Calculate the coordinates for 2D- or 3D- ROC curve and the optimal threshold point
## Not run: data(data_test)
xllim<--4
xulim<-4
yllim<-30
yulim<-110
```

```
attrs.no=1
pos.Class<-levels(data_test[,ncol(data_test)])[1]
add.legend<-TRUE

aacs<-rep(0,length(attrs.no))
color<-c(1:length(attrs.no))

out <- cost.curve(data_test,attrs.no, pos.Class,col=color[1],add=F,
 xlim=c(xllim,xulim),ylim=c(yllim,yulim))

## End(Not run)
```

CalcGene *Calculate HUM value*

Description

This is the auxiliary function of the Biocomb package. It computes a HUM value for individual feature and returns a "List" object, consisting of HUM value and the best permutation of class labels in "seq" vector. This "seq" vector can be passed to the function CalculateHUM_ROC.

Usage

```
CalcGene(s_data, seqAll, prodValue,thresholds)
```

Arguments

s_data	a list, which contains the vectors of sorted feature values for individual class labels.
seqAll	a numeric matrix of all the permutations of the class labels, where each row corresponds to individual permutation vector.
prodValue	a numeric value, which is the product of the sizes of feature vectors, corresponding to analized class labels.
thresholds	a numeric vector, containing the values of thresholds for calculating ROC curve coordinates.

Details

This function's main job is to compute the maximal HUM value between the all possible permutations of class labels for individual feature, selected for analysis. See the "Value" section to this page for more details.

Value

The data must be provided without missing values in order to process. A returned list consists of the following fields:

HUM	a list of HUM values for the specified number of analyzed features
seq	a list of vectors, each containing the sequence of class labels

Errors

If there exists NA values for features or class labels no HUM value can be calculated and an error is triggered with message "Values are missing".

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.

See Also

CalculateHUM_Ex, CalculateHUM_ROC

Examples

```
data(leukemia72)
# Basic example
# class label must be factor
leukemia72[,ncol(leukemia72)]<-as.factor(leukemia72[,ncol(leukemia72)])

xdata=leukemia72
indexF=3
indexClass=ncol(xdata)
label=levels(xdata[,indexClass])

indexLabel=label[1:2]

indexL=NULL
for(i in 1:length(indexLabel))
{
  indexL=c(indexL,which(label==indexLabel[i]))
}

indexEach=NULL
indexUnion=NULL
for(i in 1:length(label))
{
  vrem=which(xdata[,indexClass]==label[i])
  indexEach=c(indexEach,list(vrem))
  if(length(intersect(label[i],indexLabel))==1)
  indexUnion=union(indexUnion,vrem)
}

s_data=NULL
dataV=xdata[,indexF]
prodValue=1

for (j in 1:length(indexLabel))
{
  vrem=sort(dataV[indexEach[[indexL[j]]]])
  s_data=c(s_data,list(vrem))
```

```
  prodValue = prodValue*length(vrem)
}

len=length(indexLabel)
seq=permutations(len,len,1:len)

#claculate the threshold values
thresholds <- sort(unique(dataV[indexUnion]))
thresholds=(c(-Inf, thresholds) + c(thresholds, +Inf))/2

out=CalcGene(s_data,seq,prodValue,thresholds)
```

`CalcROC` *Calculate ROC points*

Description

This is the auxiliary function of the Biocomb package. It computes a point coordinates for plotting ROC curve and returns a "List" object, consisting of sensitivity and specificity values for 2D-ROC curve and 3D-points for 3D-ROC curve, the optimal threshold values with the corresponding feature values and the accuracy of the classifier (feature).

Usage

```
CalcROC(s_data, seq, thresholds)
```

Arguments

`s_data`	a list, which contains the vectors of sorted feature values for individual class labels.
`seq`	a numeric vector, containing the particular permutation of class labels.
`thresholds`	a numeric vector, containing the values of thresholds for calculating ROC curve coordinates.

Details

This function's main job is to compute the point coordinates to plot the 2D- or 3D-ROC curve, the optimal threshold values and the accuracy of classifier. See the "Value" section to this page for more details. The optimal threshold for two-class problem is the pair of sensitivity and specificity values for the selected feature. The optimal threshold for three-class problem is the 3D-point with the coordinates presenting the fraction of the correctly classified data objects for each class. The calculation of the optimal threshold is described in section "Threshold".

Value

The data must be provided without missing values in order to process. A returned list consists of the following fields:

`Sn`	a specificity values for 2D-ROC construction and the first coordinate for 3D-ROC construction
`Sp`	a sensitivity values for 2D-ROC construction and the second coordinate for 3D-ROC construction
`S3`	the third coordinate for 3D-ROC construction
`optSn`	the optimal specificity value for 2D-ROC construction and the first coordinate of the op-timal threshold for 3D-ROC construction
`optSp`	the optimal sensitivity value for 2D-ROC construction and the second coordinate of the optimal threshold for 3D-ROC construction
`optS3`	the third coordinate of the optimal threshold for 3D-ROC construction
`optThre`	the feature value according to the optimal threshold (optSn,optSp) for two-class problem
`optThre1`	the first feature value according to the optimal threshold (optSn,optSp,optS3) for three-class problem
`optThre2`	the second feature value according to the optimal threshold (optSn,optSp,optS3) for three-class problem
`accuracy`	the accuracy of classifier (feature) for the optimal threshold

Threshold

The optimal threshold value is calculated for two-class problem as the pair "(optSn, optSp)" corresponding to the maximal value of "Sn+Sp". According to "(optSn, optSp)" the corresponding feature threshold value "optThre" is calculated. The optimal threshold value is calculated for three-class problem as the pair "(optSn, optSp,optS3)" corresponding to the maximal value of "Sn+Sp+S3".According to "(optSn, optSp,optS3)" the corresponding feature threshold values "optThre1,optThre2" are calculated. The accuracy of the classifier is the mean value of dQuote(optSn, optSp) for two-class problem and the mean value of "(optSn, optSp,optS3)" for three-class problem.

Errors

If there exists NA values for features or class labels no HUM value can be calculated and an error is triggered with message "Values are missing".

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.

See Also

`CalculateHUM_Ex`, `CalculateHUM_ROC`

Examples

```
data(leukemia72_2)
# Basic example
# class label must be factor
leukemia72_2[,ncol(leukemia72_2)]<-as.factor(leukemia72_2[,ncol(leukemia72_2)])

xdata=leukemia72_2
indexF=1:3
indexClass=ncol(xdata)

label=levels(xdata[,indexClass])
indexLabel=label

out=CalculateHUM_seq(xdata,indexF,indexClass,indexLabel)

HUM<-out$HUM
seq<-out$seq

indexL=NULL
for(i in 1:length(indexLabel))
{
  indexL=c(indexL,which(label==indexLabel[i]))
}

indexEach=NULL
indexUnion=NULL

for(i in 1:length(label))
{
  vrem=which(xdata[,indexClass]==label[i])
  indexEach=c(indexEach,list(vrem))
  if(length(intersect(label[i],indexLabel))==1)
    indexUnion=union(indexUnion,vrem)
}
s_data=NULL
dataV=xdata[,indexF[1]]  #single feature
prodValue=1
for (j in 1:length(indexLabel))
{
  vrem=sort(dataV[indexEach[[indexL[j]]]])

  s_data=c(s_data,list(vrem))
  prodValue = prodValue*length(vrem)
}
#calculate the threshold values for plot of 2D ROC and 3D ROC
thresholds <- sort(unique(dataV[indexUnion]))
thresholds=(c(-Inf, thresholds) + c(thresholds, +Inf))/2

out=CalcROC(s_data,seq[,indexF[1]], thresholds)
```

`Calculate3D` *Plot the 3D-ROC curve*

Description

This function plots the 3D-ROC curve using the point coordinates, computed by the function `CalculateHUM_ROC`. Optionally visualizes the optimal threshold point, which gives the maximal accuracy of the classifier(feature) (see `CalcROC`).

Usage

```
Calculate3D(sel,Sn,Sp,S3,optSn,optSp,optS3,thresholds,HUM,
name,print.optim=TRUE)
```

Arguments

`sel`	a character value, which is the name of the selected feature.
`Sn`	a numeric vector of the x-coordinates of the ROC curve..
`Sp`	a numeric vector of the y-coordinates of the ROC curve.
`S3`	a numeric vector of the z-coordinates of the ROC curve.
`optSn`	the first coordinate of the optimal threshold
`optSp`	the second coordinate of the optimal threshold
`optS3`	the third coordinate of the optimal threshold
`thresholds`	a numeric vector with threshold values to calculate point coordinates.
`HUM`	a numeric vector of HUM values, calculated using function.
`name`	a character vector of class labels.
`print.optim`	a boolean parameter to plot the optimal threshold point on the graph. The default value is TRUE.

Details

This function's main job is to plot the 3D-ROC curve according to the given point coordinates.

Value

The function doesn't return any value.

Errors

If there exists NA values for specificity or sensitivity values, or HUM values the plotting fails and an error is triggered with message "Values are missing"

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.
Natalia Novoselova, Cristina Della Beffa, Junxi Wang, Jialiang Li, Frank Pessler, Frank Klawonn. HUM Calculator and HUM package for R: easy-to-use software tools for multicategory receiver operating characteristic analysis» / Bioinformatics. – 2014. – Vol. 30 (11): 1635-1636 doi:10.1093/bioinformatics/btu086.

See Also

CalculateHUM_seq, CalculateHUM_ROC

Examples

```
data(leukemia72)
# Basic example
# class label must be factor
leukemia72[,ncol(leukemia72)]<-as.factor(leukemia72[,ncol(leukemia72)])

xdata=leukemia72

indexF=names(xdata)[10]

indexClass=ncol(xdata)
label=levels(xdata[,indexClass])
indexLabel=label

out=CalculateHUM_seq(xdata,indexF,indexClass,indexLabel)
HUM<-out$HUM
seq<-out$seq
out=CalculateHUM_ROC(xdata,indexF,indexClass,indexLabel,seq)
Calculate3D(indexF,out$Sn,out$Sp,out$S3,out$optSn,out$optSp,out$optS3,
out$thresholds,HUM,indexLabel[seq])
```

CalculateHUM_Ex *Calculate HUM value*

Description

This function calculates the HUM (hypervolume under manifold) feature values with the exhaustive search, i.e. for all combinations of the defined number of categories from the whole set of available categories. The function is used for ranking the features (in decreasing order of HUM values). HUM values are the extension of the AUC values for more than two classes. It can handle only numerical values. It computes a HUM value and returns a "List" object, consisting of HUM value and the best permutation of class labels in "seq" vector. This "seq" vector can be passed to the function CalculateHUM_ROC for the calculating the coordinates of the 2D or 3D ROC.

Usage

```
CalculateHUM_Ex(data,indexF,indexClass,allLabel,amountL)
```

Arguments

`data`	a dataset, a matrix of feature values for several cases, the additional column with class labels is provided. Class labels could be numerical or character values. The maximal number of classes is ten. The `indexClass` determines the column with class labels.
`indexF`	a numeric or character vector, containing the column numbers or column names of the analyzed features.
`indexClass`	a numeric or character value, containing the column number or column name of the class labels.
`allLabel`	a character vector, containing the column names of the class labels, selected for the analysis.
`amountL`	a number of the class categories to search for all the possible combinations.

Details

This function's main job is to compute the maximal HUM value between the all possible permutations of class labels, selected for analysis. See the "Value" section to this page for more details. Before returning, it will call the `CalcGene` function to calculate the HUM value for each feature (object)..

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the separate column contains class labels. The maximal number of class labels equals 10. The computational efficiency of the function descrease in the case of more than 1000 cases with more than 6 class labels. In order to use all the functions of the package it is necessary to put the class label in the last column of the dataset. The class label features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`. A returned list consists of th the following fields:

`HUM`	a list of HUM values for the specified number of analyzed features
`seq`	a list of vectors, each containing the sequence of class labels

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.
Natalia Novoselova, Cristina Della Beffa, Junxi Wang, Jialiang Li, Frank Pessler, Frank Klawonn. HUM Calculator and HUM package for R: easy-to-use software tools for multicategory receiver operating characteristic analysis» / Bioinformatics. – 2014. – Vol. 30 (11): 1635-1636 doi:10.1093/bioinformatics/btu086.

See Also

CalculateHUM_seq, CalculateHUM_ROC

Examples

```
data(leukemia72)
# Basic example
# class label must be factor
leukemia72[,ncol(leukemia72)]<-as.factor(leukemia72[,ncol(leukemia72)])

xdata=leukemia72
indexF=c(1:2)
indexClass=ncol(xdata)
allLabel=levels(xdata[,indexClass])

amountL=2
out=CalculateHUM_Ex(xdata,indexF,indexClass,allLabel,amountL)
```

CalculateHUM_Plot *Plot 2D-ROC curve*

Description

This function plots the 2D-ROC curve using the point coordinates, computed by the function CalculateHUM_ROC.Optionally visualizes the optimal threshold point, which gives the maximal accuracy of the classifier(feature) (see CalcROC).

Usage

```
CalculateHUM_Plot(sel,Sn,Sp,optSn,optSp,HUM,print.optim=TRUE)
```

Arguments

sel	a character value, which is the name of the selected feature.
Sn	a numeric vector of the x-coordinates of the ROC curve, which is the specificity values of the standard ROC analysis..
Sp	a numeric vector of the y-coordinates of the ROC curve, which is the sensitivity values of the standard ROC analysis..
optSn	the optimal specificity value for 2D-ROC construction
optSp	the optimal sensitivity value for 2D-ROC construction
HUM	a numeric vector of HUM values, calculated using function CalculateHUM_seq or CalculateHUM_Ex.
print.optim	a boolean parameter to plot the optimal threshold point on the graph. The default value is TRUE.

Details

This function's main job is to plot the 2D-ROC curve according to the given point coordinates.

Value

The function doesn't return any value.

Errors

If there exists NA values for specificity or sensitivity values, or HUM values the plotting fails and an error is triggered with message "Values are missing".

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.
Natalia Novoselova, Cristina Della Beffa, Junxi Wang, Jialiang Li, Frank Pessler, Frank Klawonn. HUM Calculator and HUM package for R: easy-to-use software tools for multicategory receiver operating characteristic analysis» / Bioinformatics. – 2014. – Vol. 30 (11): 1635-1636 doi:10.1093/bioinformatics/btu086.

See Also

CalculateHUM_seq, CalculateHUM_ROC

Examples

```
data(leukemia72)
# Basic example
# class label must be factor
leukemia72[,ncol(leukemia72)]<-as.factor(leukemia72[,ncol(leukemia72)])

xdata=leukemia72

indexF=names(xdata)[3]

indexClass=ncol(xdata)
label=levels(xdata[,indexClass])
indexLabel=label[1:2]

out=CalculateHUM_seq(xdata,indexF,indexClass,indexLabel)
HUM<-out$HUM
seq<-out$seq
out=CalculateHUM_ROC(xdata,indexF,indexClass,indexLabel,seq)

CalculateHUM_Plot(indexF,out$Sn,out$Sp,out$optSn,out$optSp,HUM)
```

`CalculateHUM_ROC` *Compute the points for ROC curve*

Description

This is the function for computing the points for ROC curve. It returns a "List" object, consisting of sensitivity and specificity values for 2D-ROC curve and 3D-points for 3D-ROC curve. Also the optimal threshold values are returned. It can handle only numerical values.

Usage

```
CalculateHUM_ROC(data,indexF,indexClass,indexLabel,seq)
```

Arguments

`data`	a dataset, a matrix of feature values for several cases, the additional column with class labels is provided. Class labels could be numerical or character values. The maximal number of classes is ten. The `indexClass` determines the column with class labels.
`indexF`	a numeric or character vector, containing the column numbers or column names of the analyzed features.
`indexClass`	a numeric or character value, containing the column number or column name of the class labels.
`indexLabel`	a character vector, containing the column names of the class labels, selected for the analysis.
`seq`	a numeric matrix, containing the permutation of the class labels for all features.

Details

This function's main job is to compute the point coordinates to plot the 2D- or 3D-ROC curve and the optimal threshold values. See the "Value" section to this page for more details. The function calls the `CalcROC` to calculate the point coordinates, optimal thresholds and accuracy of classifier (feature) in the threshold.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the separate column contains class labels. The maximal number of class labels equals 10. In order to use all the functions of the package it is necessary to put the class label in the last column of the dataset. The class label features must be defined as factors.

Value

The data must be provided without missing values in order to process. A returned list consists of th the following fields:

`Sn`	a specificity values for 2D-ROC construction and the first coordinate for 3D-ROC construction

Sp	a sensitivity values for 2D-ROC construction and the second coordinate for 3D-ROC construction
S3	the third coordinate for 3D-ROC construction
optSn	the optimal specificity value for 2D-ROC construction and the first coordinate of the op-timal threshold for 3D-ROC construction
optSp	the optimal sensitivity value for 2D-ROC construction and the second coordinate of the optimal threshold for 3D-ROC construction
optS3	the third coordinate of the optimal threshold for 3D-ROC construction

Errors

If there exists NA values for features or class labels no HUM value can be calculated and an error is triggered with message "Values are missing".

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.
Natalia Novoselova, Cristina Della Beffa, Junxi Wang, Jialiang Li, Frank Pessler, Frank Klawonn. HUM Calculator and HUM package for R: easy-to-use software tools for multicategory receiver operating characteristic analysis» / Bioinformatics. – 2014. – Vol. 30 (11): 1635-1636 doi:10.1093/bioinformatics/btu086.

See Also

CalculateHUM_Ex, CalculateHUM_seq

Examples

```
data(leukemia72)
# Basic example
# class label must be factor
leukemia72[,ncol(leukemia72)]<-as.factor(leukemia72[,ncol(leukemia72)])

xdata=leukemia72
indexF=1:3
indexClass=ncol(xdata)
label=levels(xdata[,indexClass])
indexLabel=label[1:2]

out=CalculateHUM_seq(xdata,indexF,indexClass,indexLabel)
HUM<-out$HUM
seq<-out$seq
out=CalculateHUM_ROC(xdata,indexF,indexClass,indexLabel,seq)
```

`CalculateHUM_seq` *Calculate HUM value*

Description

This function calculates the features weights using the HUM (hypervolume under manifold) values criterion measure and is used for ranking the features (in decreasing order of HUM values). HUM values are the extension of the AUC values for more than two classes. It can handle only numerical values. It computes a HUM value and returns a "List" object, consisting of HUM value and the best permutation of class labels in "seq" vector. This "seq" vector can be passed to the function `CalculateHUM_ROC` for the calculating the coordinates of the 2D or 3D ROC. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "HUM" for feature selection.

Usage

```
CalculateHUM_seq(data,indexF,indexClass,indexLabel)
```

Arguments

`data`	a dataset, a matrix of feature values for several cases, the additional column with class labels is provided. Class labels could be numerical or character values. The maximal number of classes is ten. The `indexClass` determines the column with class labels.
`indexF`	a numeric or character vector, containing the column numbers or column names of the analyzed features.
`indexClass`	a numeric or character value, containing the column number or column name of the class labels.
`indexLabel`	a character vector, containing the column names of the class labels, selected for the analysis.

Details

This function's main job is to compute the maximal HUM value between the all possible permutations of class labels, selected for analysis. See the "Value" section to this page for more details. Before returning, it will call the `CalcGene` function to calculate the HUM value for each feature (object).

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the separate column contains class labels. The maximal number of class labels equals 10. The computational efficiency of the function descrease in the case of more than 1000 cases with more than 6 class labels. In order to use all the functions of the package it is necessary to put the class label in the last column of the dataset.The class label features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss. A returned list consists of th the following fields:

HUM	a list of HUM values for the specified number of analyzed features
seq	a list of vectors, each containing the sequence of class labels

References

Li, J. and Fine, J. P. (2008): ROC Analysis with Multiple Tests and Multiple Classes: methodology and its application in microarray studies.*Biostatistics*. 9 (3): 566-576.
Natalia Novoselova, Cristina Della Beffa, Junxi Wang, Jialiang Li, Frank Pessler, Frank Klawonn. HUM Calculator and HUM package for R: easy-to-use software tools for multicategory receiver operating characteristic analysis» / Bioinformatics. – 2014. – Vol. 30 (11): 1635-1636 doi:10.1093/bioinformatics/btu086.

See Also

CalculateHUM_Ex, CalculateHUM_ROC

Examples

```
data(leukemia72)
# Basic example
# class label must be factor
leukemia72[,ncol(leukemia72)]<-as.factor(leukemia72[,ncol(leukemia72)])

xdata=leukemia72
indexF=1:2
indexClass=ncol(xdata)
label=levels(xdata[,indexClass])
indexLabel=label[1:2]

out=CalculateHUM_seq(xdata,indexF,indexClass,indexLabel)
```

---

chi2.algorithm *Select the subset of features*

Description

This function selects the subset of features on the basis of the Chi2 discretization algorithm. The algorithm provides the way to select numerical features while discretizing them. It is based on the χ^2 statistic, and consists of two phases of discretization. According to the value of χ^2 statistic for each pair of adjacent intervals the merging of the intervals continues until an inconsistency rate is exceeded. Chi2 algorithms automatically determines a proper χ^2 threshold that keeps the fidelity of the original data. The nominal features must be determined as they didn't take part in

the discretization process but in the process of inconsistency rate calculation. In the process of discretization the irrelevant features are removed. The results is in the form of "list", consisting of two fields: the processed dataset without irrelevant features and the names of the selected features. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "Chi2-algorithm" for feature selection.

Usage

```
chi2.algorithm(matrix,attrs.nominal,threshold)
```

Arguments

`matrix`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
`attrs.nominal`	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.
`threshold`	a numeric threshold value for the inconsistency rate.

Details

This function's main job is to select the subset of informative numerical features using the two phase process of feature values merging according to the χ^2 statistic for the pairs of adjacent intervals. The stopping criterion of merging is the inconsistency rate of the processed dataset. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned data.frame consists of the the following fields:

`data.out`	the processed dataset without irrelevant features (features which have been merged into a single interval)
`subset`	a character vector of the selected feature names

References

H. Liu and L. Yu. "Toward Integrating Feature Selection Algorithms for Classification and Clustering", IEEE Trans. on Knowledge and Data Engineering, pdf, 17(4), 491-502, 2005.

See Also

`input_miss`, `select.process`

Examples

```
# example for dataset without missing values
#p1=Sys.time()
data(data_test)
# not all features to select
xdata=data_test[,c(1:6,ncol(data_test))]
# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])
attrs.nominal=numeric()
threshold=0
out=chi2.algorithm(matrix=xdata,attrs.nominal=attrs.nominal,threshold=threshold)
#Sys.time()-p1
```

`classifier.loop` *Classification and classifier validation*

Description

The main function for the classification and classifier validation. It performs the classification using different classification algorithms (classifiers) with the embedded feature selection and using the different schemes for the performance validation.
It presents the infrostructure to perform classification of the data set with two or more class labels. The function calls several classification methods, including Nearest shrunken centroid ("nsc"), Naive Bayes classifier ("nbc"), Nearest Neighbour classifier ("nn"), Multinomial Logistic Regression ("mlr"), Support Vector Machines ("svm"), Linear Discriminant Analysis ("lda"), Random Forest ("rf"). The function calls the `select.process` in order to perform feature selection for classification, which helps to improve the quality of the classifier. The classifier accuracy is estimated using the embedded validation procedures, including the Repeated random sub-sampling validation ("sub-sampling"), k-fold cross-validation ("fold-crossval") and Leave-one-out cross-validation ("leaveOneOut").

The results is in the form of "list" with the data.frame of classification results for each selected classifier "predictions", matrix with the statistics for the frequency each feature is selected "no.selected", vector or matrix with the number of true classifications for each selected classifier "true.classified".

Usage

```
classifier.loop(dattable,classifiers=c("svm","lda","rf","nsc"),
 feature.selection=c("auc","InformationGain"),
 disc.method="MDL",threshold=0.3, threshold.consis=0,
 attrs.nominal=numeric(), no.feat=20,flag.feature=TRUE,
 method.cross=c("leaveOneOut","sub-sampling","fold-crossval"))
```

Arguments

`dattable`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.

`classifiers` the names of the classifiers.

`feature.selection`
a method of feature ranking or feature selection.

`disc.method` a method used for feature discretization. There are three options "MDL","equal interval width","equal frequency". The discretization options "MDL" assigned to the minimal description length (MDL) discretization algorithm, which is a supervised algorithm. The last two options refer to the unsupervized discretization algorithms.

`threshold` a numeric threshold value for the correlation of feature with class to be included in the final subset. It is used by fast correlation-based filter method (FCBF)

`threshold.consis`
a numeric threshold value for the inconsistency rate. It is used by Chi2 discretization algorithm.

`attrs.nominal` a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

`no.feat` the maximal number of features to be selected.

`flag.feature` logical value; if TRUE the process of classifier construction and validation will be repeated for each subset of features, starting with one feature and upwards.

`method.cross` a character value with the names of the model validation technique for assessing how the classification results will generalize to an independent data set. It includes Repeated random sub-sampling validation, k-fold cross-validation and Leave-one-out cross-validation.

Details

This function's main job is to perform classification with feature selection and the estimation of classification results with the model validation techniques. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned list consists of the the following fields:

`predictions` a data.frame of classification results for each selected classifier

`no.selected` a matrix with the statistics for each feature selection frequency

`true.classified`
a vector or matrix with the number of true classifications for each selected classifier

References

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association, 97(457):77–87, 2002.

See Also

select.process, input_miss

Examples

```
# example for dataset without missing values
data(leukemia72_2)

# class label must be factor
leukemia72_2[,ncol(leukemia72_2)]<-as.factor(leukemia72_2[,ncol(leukemia72_2)])

class.method="svm"
method="InformationGain"
disc<-"MDL"
cross.method<-"fold-crossval"

thr=0.1
thr.cons=0.05
attrs.nominal=numeric()
max.f=10

out=classifier.loop(leukemia72_2,classifiers=class.method,
feature.selection=method,disc.method=disc,
threshold=thr, threshold.consis=thr.cons,attrs.nominal=attrs.nominal,
no.feat=max.f,flag.feature=FALSE,method.cross=cross.method)


# example for dataset with missing values
## Not run:
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
```

```
class.method="svm"
method="InformationGain"
disc<-"MDL"
cross.method<-"fold-crossval"

thr=0.1
thr.cons=0.05
max.f=10

out=classifier.loop(xdata,classifiers=class.method,
feature.selection=method,disc.method=disc,
threshold=thr, threshold.consis=thr.cons,attrs.nominal=attrs.nominal,
no.feat=max.f,flag.feature=FALSE,method.cross=cross.method)

## End(Not run)
```

`compute.auc.permutation` *Calculates the p-values*

Description

This auxiliary function calculates the p-value of the significance of the AUC values using the permutation test (for each input feature). It takes as an input the results of the AUC value calculation using function `compute.aucs`.

The results is in the form of "numeric vector" with p-values for each AUC value.

Usage

```
compute.auc.permutation(aucs,dattable,repetitions=1000)
```

Arguments

`aucs`	a numeric vector of AUC values.
`dattable`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values.
`repetitions`	the number of permutations of feature values.

Details

This auxiliary function's main job is to calculate the p-values of the statistical significance test of the AUC values for each input feature). See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`. A returned data is the following:

`p.values` a numeric vector with the p-values for each feature AUC value

References

David J. Hand and Robert J. Till (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning 45(2), p. 171–186.

See Also

`compute.aucs`, `pauclog`, `pauc`, `compute.auc.random`

Examples

```
# example
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

auc.val=compute.aucs(dattable=data_test)
vauc<-auc.val[,"AUC"]
rep.num<-20

p.values=compute.auc.permutation(aucs=vauc,dattable=data_test,rep.num)
```

`compute.auc.random` *Calculates the p-values*

Description

This auxiliary function calculates the p-value of the significance of the AUC values using the comparison with random sample generation (for each input feature). It takes as an input the results of the AUC value calculation using function `compute.aucs`.

The results is in the form of "numeric vector" with p-values for each AUC value.

Usage

```
compute.auc.random(aucs,dattable,repetitions=10000,correction="none")
```

Arguments

aucs	a numeric vector of AUC values.
dattable	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values.
repetitions	the number of repetitions of random sample' generation.
correction	the method of p-value correction for multiple testing, including Bonferroni-Holm, Bonferroni corrections or without correction.

Details

This auxiliary function's main job is to calculate the p-values of the statistical significance test of the AUC values for each input feature for two-class problem.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels (with two class labels).

The correction methods include the Bonferroni correction ("bonferroni") in which the p-values are multiplied by the number of comparisons and the less conservative corrections by Bonferroni-Holm method ("bonferroniholm"). A pass-through option ("none") is also included.

The correction methods are designed to give strong control of the family-wise error rate. See the "Value" section to this page for more details.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss. A returned data is the following:

pvalues.raw	a numeric vector with the corrected p-values for each feature AUC value

References

Benjamini, Y., and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. Annals of Statistics 29, 1165–1188.

See Also

compute.aucs, pauclog, pauc, compute.auc.permutation

Examples

```
# example
data(datasetF6)

# class label must be factor
datasetF6[,ncol(datasetF6)]<-as.factor(datasetF6[,ncol(datasetF6)])

auc.val=compute.aucs(dattable=datasetF6)
vauc<-auc.val[,"AUC"]
```

```
cors<-"none"
rep.num<-100

pvalues.raw<-compute.auc.random(aucs=vauc,dattable=datasetF6,
 repetitions=rep.num,correction=cors)
```

`compute.aucs` *Ranks the features*

Description

This function calculates the features weights using the AUC (Area Under the ROC Curve) values. It can handle only numerical values. This function performs two-class or multiclass AUC. A multiclass AUC is a mean of AUCs for all combinations of the two class labels. This function measures the worth of a feature by computing the AUC values with respect to the class.The results is in the form of "data.frame". In the case of two-class problem it consists of the three fields: features (Biomarker) names, AUC values and level of the positive class. In the case of more than two classes it consists of two fields: features (Biomarker) names, AUC values. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "auc" for feature selection.

Usage

```
compute.aucs(dattable)
```

Arguments

`dattable`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.

Details

This function's main job is to calculate the weights of the features according to AUC values. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`. A returned data.frame consists of th the following fields:

`Biomarker`	a character vector of feature names

AUC	a numeric vector of AUC values for the features according to class
Positive class	a numeric vector of positive class levels for two-class problem

References

David J. Hand and Robert J. Till (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning 45(2), p. 171–186.

See Also

input_miss, select.process

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

out=compute.aucs(dattable=data_test)

# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# the nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
xdata=xdata[,-attrs.nominal]
# the nominal features are not processed
out=compute.aucs(dattable=xdata)
```

---

cost.curve *Plots the RCC curve for two-class problem*

Description

This function plots the Relative Cost Curves (RCC) and calculates the corresponding Area Above the RCC (AAC) value to estimate the classifier performance under unequal misclassification costs. It is intended for the two-class problem, but the extension to more than two classes will be produced later. RCC is a graphical technique for visualising the performance of binary classifiers over the full range of possible relative misclassification costs. This curve provides helpful information to choose the best set of classifiers or to estimate misclassification costs if those are not known precisely. Area Above the RCC (AAC) is a scalar measure of classifier performance under unequal misclassification costs problem. It can be reasonably used only for two-class problem.

Usage

```
cost.curve(data, attrs.no, pos.Class, AAC=TRUE, n=101, add=FALSE,
xlab="log2(c)",ylab="relative costs", main="RCC",lwd=2,col="black",
xlim=c(-4,4), ylim=(c(20,120)))
```

Arguments

data	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The function is provided for two classes.
attrs.no	a numerical value, containing the column number of the features to construct the RCC.
pos.Class	a level of the class factor to be selected as the positive class for the construction of the RCC cost curve.
AAC	logical value; if TRUE the AAC value will be calculated.
n	the number of points for the construction of RCC curve (it corresponds to the number of cost values).
add	logical value; if TRUE the RCC curve can be added to the existent RCC plot.
xlab	name of the X axis.
ylab	name of the Y axis.
main	name of the RCC plot.
lwd	a positive number for line width.
col	the color value for the RCC plot.
xlim	the vector with two numeric values for the X axis limits, it defines the values for log2(cost).
ylim	the vector with two numeric values for the Y axis limits, it defines the values for the relative cost value.

Details

This function's main job is to plt the RCC curve and calculates the corresponding AAC value to estimate the classifier performance under unequal misclassification costs.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels with two class labels.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss.

References

Olga Montvida and Frank Klawonn Relative cost curves: An alternative to AUC and an extension to 3-class problems,Kybernetika 50 no. 5, 647-660, 2014

See Also

compute.aucs, compute.auc.random, compute.auc.permutation, plotRoc.curves, input_miss

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

xllim<--4
xulim<-4
yllim<-30
yulim<-110

attrs.no=c(1,9)
pos.Class<-levels(data_test[,ncol(data_test)])[1]
add.legend<-TRUE

aacs<-rep(0,length(attrs.no))
color<-c(1:length(attrs.no))

aacs[1] <- cost.curve(data_test, attrs.no[1], pos.Class,col=color[1],add=FALSE,
 xlim=c(xllim,xulim),ylim=c(yllim,yulim))

if(length(attrs.no)>1){
for(i in 2:length(attrs.no)){
      aacs[i]<- cost.curve(data_test, attrs.no[i], pos.Class,
      col=color[i],add=TRUE,xlim=c(xllim,xulim))
    }
    }

if(add.legend){
       legt <- colnames(data_test)[attrs.no]
   for(i in 1:length(attrs.no)){
   legt[i] <- paste(legt[i],", AAC=",round(1000*aacs[i])/1000,sep="")
}
legend("bottomright",legend=legt,col=color,lwd=2)
}
```

datasetF6 *simulated data*

Description

This data file consists of six simulated predictors or variables with three class categories. For each class category the values are independently generated from the normal distribution with the mean μ1, μ2 and μ3 and the variances held at unity. The means are varied such that the problems range from near-separable problems, to near-random.

Usage

```
datasetF6
```

Format

A data.frame containing 300 observations of six variables.

Source

Landgrebe T, Duin R (2006) A simplified extension of the Area under the ROC to the multiclass domain. In: Proceedings 17th Annual Symposium of the Pattern Recognition Association of South Africa. PRASA, pp. 241–245.

See Also

data_test, leukemia72, leukemia72_2, leukemia_miss

Examples

```
# load the dataset
data(datasetF6)
```

data_test *simulated data*

Description

This data file consists of 300 objects with 10 features. The features x1-x5 are informative and define the cluster structure of the dataset. The clusters are generated in the two-dimensional space x1-x2. The values of the features x3-x5 are identically generated as for x2. Features values x1-x5 are normally distributed values with the same standard deviation and different mean values. Features x6-x10 are random variables uniformly distributed in the interval [0, 1] and present the uninformative features.

Usage

```
data_test
```

Format

A data.frame containing 300 observations of 11 variables and class with three labels.

Source

Landgrebe T, Duin R (2006) A simplified extension of the Area under the ROC to the multiclass domain. In: Proceedings 17th Annual Symposium of the Pattern Recognition Association of South Africa. PRASA, pp. 241–245.

See Also

leukemia72, datasetF6, leukemia72_2, leukemia_miss

Examples

```
# load the dataset
data(data_test)
```

---

generate.data.miss *Generate the dataset with missing values*

Description

The function for the generation the dataset with missing values from the input dataset with all the values. It is mainly intended for the testing purposes. The results is in the form of "data.frame" which corresponds to the input data.frame or matrix, where missing values are inserted. The percent of missing values is supplied as the input parameters. The processed dataset can be used in the algorithms for missing value imputation "input_miss" or for any other purposes.

Usage

```
generate.data.miss(data,percent=5,filename=NULL)
```

Arguments

data	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. This data set has not missing values
percent	a numerical value for the percent of the missing values to be inserted into the dataset.
filename	a character name of the output file to save the dataset with missing values.

Details

This function's main job is to generate the dataset with missing values from the input dataset with all the values. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

A returned data.frame corresponds to the input dataset with inserted missing values.

References

McShane LM, Radmacher MD, Freidlin B, Yu R, Li MC, Simon R. Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data. Bioinformatics. 2002 Nov;18(11):1462-9.

See Also

input_miss, select.process, classifier.loop

Examples

```
# example

data(leukemia72_2)

percent =5
f.name=NULL #file name to include
out=generate.data.miss(data=leukemia72_2,percent=percent,filename=f.name)
```

---

input_miss *Process the dataset with missing values*

Description

The main function for handling with missing values. It performs the missing values imputation using two different approachs: imputation with mean values and using the nearest neighbour algorithm. It can handle both numerical and nominal values. The function also delete the features with the number of missing values more then specified threshold. The results is in the form of "list" with the processed dataset and the logical value, which indicates the success or failure of processing. The processed dataset can be used in the algorithms for feature selection "select.process" and classification "classifier.loop".

Usage

```
input_miss(matrix,method.subst="near.value",
attrs.nominal=numeric(),delThre=0.2)
```

Arguments

matrix	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
method.subst	a method of missing value processing. There are two realized methods: substitution with mean value ('mean.value') and nearest neighbour algorithm ('near.value').
attrs.nominal	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.
delThre	the minimal threshold for the deletion of features with missing values. It is in the interval [0,1], where for delThre=0 all features having at least one missing value will be deleted.

Details

This function's main job is to handle the missing values in the dataset. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data are provided with reasonable number of missing values that is preprocessed with one of the imputing methods.

A returned list consists of the the following fields:

data	a processed dataset
flag.miss	logical value; if TRUE the processing is successful, if FALSE the input dataset is returned without processing.

References

McShane LM, Radmacher MD, Freidlin B, Yu R, Li MC, Simon R. Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data. Bioinformatics. 2002 Nov;18(11):1462-9.

See Also

select.process, classifier.loop

Examples

```
# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
```

---

leukemia72 *desease data*

Description

Leukemia dataset includes the bone marrow samples obtained from acute leukemia patients at the time of diagnosis: 25 acute myeloid leukemia (AML) samples; 9 T-lineage acute lymphoblastic leukemia (ALL) samples; and 38 B-lineage ALL samples. After preprocessing, the 100 genes with the largest variation across samples are selected.

Usage

```
leukemia72
```

Format

A data.frame containing 72 observations of 101 variables: 100 features and class with three diagnosis: 38 B-lineage ALL, 9 T-lineage ALL and 25 AML.

Source

Handl J, Knowles J, Kell DB, Computational cluster validation in post-genomic data analysis, Bioinformatics 21:3201-3212, 2005.

See Also

data_test, datasetF6, leukemia72_2, leukemia_miss

Examples

```
# load the dataset
data(leukemia72)
# X95735_at
with(leukemia72, by(X95735_at,Class,mean))

# M27891_at
with(leukemia72,tapply(M27891_at, Class, FUN = mean))
with(leukemia72, table(M27891_at=ifelse(M27891_at<=mean(M27891_at), "1", "2"), Class))
```

---

leukemia72_2 *desease data*

Description

Leukemia dataset includes the bone marrow samples obtained from acute leukemia patients at the time of diagnosis: 25 acute myeloid leukemia (AML) samples; 9 T-lineage acute lymphoblastic leukemia (ALL) samples; and 38 B-lineage ALL samples. After preprocessing, the 100 genes with the largest variation across samples are selected.

Usage

```
leukemia72_2
```

Format

A data.frame containing 72 observations of 101 variables: 100 features and class with two diagnosis: 47 ALL and 25 AML).

Source

Handl J, Knowles J, Kell DB, Computational cluster validation in post-genomic data analysis, Bioinformatics 21:3201-3212, 2005.

See Also

data_test, datasetF6, leukemia72, leukemia_miss

Examples

```
# load the dataset
data(leukemia72_2)
# X95735_at
with(leukemia72_2, by(X95735_at,Class,mean))

# M27891_at
with(leukemia72_2,tapply(M27891_at, Class, FUN = mean))
with(leukemia72_2, table(M27891_at=ifelse(M27891_at<=mean(M27891_at), "1", "2"), Class))
```

leukemia_miss *desease data*

Description

Leukemia dataset includes the bone marrow samples obtained from acute leukemia patients at the time of diagnosis: 25 acute myeloid leukemia (AML) samples; 9 T-lineage acute lymphoblastic leukemia (ALL) samples; and 38 B-lineage ALL samples. After preprocessing, the 100 genes with the largest variation across samples are selected. This dataset is the same as leukemia72 with the 5 percent of missing values.

Usage

```
leukemia_miss
```

Format

A data.frame containing 72 observations of 101 variables: 100 features and class with three diagnosis: 38 B-lineage ALL, 9 T-lineage ALL and 25 AML. It has 5 percent missing values.

Source

Handl J, Knowles J, Kell DB, Computational cluster validation in post-genomic data analysis, Bioinformatics 21:3201-3212, 2005.

See Also

data_test, datasetF6, leukemia72, leukemia72_2

Examples

```
# load the dataset
data(leukemia_miss)
# X95735_at
with(leukemia_miss, by(X95735_at,Class,mean,na.rm=TRUE))

# M27891_at
with(leukemia_miss,tapply(M27891_at, Class, FUN = mean,na.rm=TRUE))
```

pauc *Calculates the p-values*

Description

This auxiliary function calculates the p-value of the statistical significance test of the difference of samples from two classes using AUC values (for each input feature). It takes as an input the results of the AUC value calculation using function `compute.aucs`. It can be reasonably used only for two-class problem. The results is in the form of "numeric vector" with p-values for each features.

Usage

```
pauc(auc,n=100,n.plus=0.5,labels=numeric(),pos=numeric())
```

Arguments

`auc`	a numeric vector of AUC values.
`n`	the whole number of observations for the test.
`n.plus`	the number of cases in the sample with the positive class.
`labels`	the factor with the class labels.
`pos`	the numeric vector with the level of the positive class.

Details

This auxiliary function's main job is to calculate the p-values of the statistical significance test of two samples, defined by negative and positive class labels, i.e. two-class problem. See the "Value" section to this page for more details.

Value

A returned data consists is the following:

`pauc`	a numeric vector with the p-values for each feature

References

David J. Hand and Robert J. Till (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning 45(2), p. 171–186.

See Also

`compute.aucs`, `pauclog`

Examples

```
# example
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

auc.val=compute.aucs(dattable=data_test)
vauc<-auc.val[,"AUC"]
val=levels(data_test[,ncol(data_test)])

if(length(val)==2)
{
 pos=auc.val[,"Positive class"]
 paucv<-pauc(auc=vauc,labels=data_test[,ncol(data_test)],pos=pos)
}else{
 num.size=100
 num.prop=0.5
 paucv<-pauc(auc=vauc,n=num.size,n.plus=num.prop)
}
```

---

pauclog *Calculates the p-values*

Description

This auxiliary function calculates the logarithm of p-values of the statistical significance test of the difference of samples from two classes using AUC values (for each input feature). It takes as an input the results of the AUC value calculation by the function `compute.aucs`. It can be reasonably used only for two-class problem. The results is in the form of "numeric vector" with the logarithms of the p-values for each features.

Usage

```
pauclog(auc,n=100,n.plus=0.5,labels=numeric(),pos=numeric())
```

Arguments

`auc`	a numeric vector of AUC values.
`n`	the whole number of observations for the test.
`n.plus`	the number of cases in the sample with the positive class.
`labels`	the factor with the class labels.
`pos`	the numeric vector with the level of the positive class.

Details

This auxiliary function's main job is to calculate the logarithm of p-values of the statistical significance test of two samples, defined by negative and positive class labels, i.e. two-class problem. See the "Value" section to this page for more details.

Value

A returned data consists is the following:

pauclog	a numeric vector with the logarithm of p-value for each feature

References

David J. Hand and Robert J. Till (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning 45(2), p. 171–186.

See Also

compute.aucs, pauc

Examples

```
# example
data(datasetF6)

# class label must be factor
datasetF6[,ncol(datasetF6)]<-as.factor(datasetF6[,ncol(datasetF6)])

auc.val=compute.aucs(dattable=datasetF6)
vauc<-auc.val[,"AUC"]
val=levels(datasetF6[,ncol(datasetF6)])

if(length(val)==2)
{
 pos=auc.val[,"Positive class"]
 paucv<-pauclog(auc=vauc,labels=datasetF6[,ncol(datasetF6)],pos=pos)
}else{
 num.size=100
 num.prop=0.5
 paucv<-pauclog(auc=vauc,n=num.size,n.plus=num.prop)
}
```

---

plotClass.result *Plots the results of classifier validation schemes*

Description

This function plots the barplots and boxplots, which help in estimation of the results of classifiers' validation, performed by different validation models. It must be called after the performing the classification validation with function classifier.loop.

Usage

```
plotClass.result(true.classified, cross.method, class.method,
flag.feature, feat.num)
```

Arguments

true.classified	a vector or matrix of classification results for one or several classifiers and one or several feature sets. The matrix is the output value of the function classifier.loop.
cross.method	a character value with the names of the model validation technique for assessing how the classification results will generalize to an independent data set. It includes Repeated random sub-sampling validation, k-fold cross-validation and Leave-one-out cross-validation.
class.method	the names of the classifiers.
flag.feature	logical value; if TRUE the process of classifier construction and validation will be repeated for each subset of features, starting with one feature and upwards.
feat.num	the maximal number of features to be selected.

Details

This function's main job is to plot the barplots and boxplots to visually estimate the results of classifiers' validation.

Value

The results is visualization of the plot .

References

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. Journal of the American Statistical Association, 97(457):77–87, 2002.

See Also

select.process, classifier.loop

Examples

```
# example for dataset without missing values

data(leukemia72_2)

# class label must be factor
leukemia72_2[,ncol(leukemia72_2)]<-as.factor(leukemia72_2[,ncol(leukemia72_2)])

class.method=c("svm","nn")
method="InformationGain"
disc<-"MDL"
cross.method<-"fold-crossval"

flag.feature=TRUE
thr=0.1
```

```
thr.cons=0.05
attrs.nominal=numeric()
max.f=10

out=classifier.loop(leukemia72_2,classifiers=class.method,
 feature.selection=method,disc.method=disc,
 threshold=thr, threshold.consis=thr.cons,attrs.nominal=attrs.nominal,
  no.feat=max.f,flag.feature=flag.feature,method.cross=cross.method)

plotClass.result(out$true.classified, cross.method, class.method, flag.feature, max.f)

# example for dataset with missing values
## Not run:
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}

class.method=c("svm","nn")
method="InformationGain"
disc<-"MDL"
cross.method<-"fold-crossval"

flag.feature=TRUE
thr=0.1
thr.cons=0.05
max.f=10

out=classifier.loop(xdata,classifiers=class.method,
 feature.selection=method,disc.method=disc,
 threshold=thr, threshold.consis=thr.cons,attrs.nominal=attrs.nominal,
 no.feat=max.f,flag.feature=flag.feature,method.cross=cross.method)

plotClass.result(out$true.classified, cross.method, class.method, flag.feature, max.f)

## End(Not run)
```

`plotRoc.curves` *Plots the ROC curve for two-class problem*

Description

This function plots the ROC curve for the two-class problem.

Usage

```
plotRoc.curves(dattable,file.name=NULL,colours=NULL,ltys=NULL,
 add.legend=F,curve.names=NULL,include.auc=F,xaxis="",yaxis="",
 line.width=2,headline="",ispercent=F)
```

Arguments

dattable	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values.
file.name	the file name to save the plot.
colours	the color values for each plot if more than one feature or one color value in the case of one feature.
ltys	the line type values for each plot if more than one feature or one line type in the case of one feature.
add.legend	logical value; if TRUE the legend will be plotted at the bottom righ.
curve.names	a character value or vector in the case of more than one feature with curve names to be used in the legend.
include.auc	logical value; if TRUE the AUC value will be included in the legend .
xaxis	character value with the name of X axis.
yaxis	character value with the name of Y axis.
line.width	a positive number for line width.
headline	the character value with the name of the plot.
ispercent	logical value; if TRUE the true positive and false positive values of the plot are in the percents.

Details

This function's main job is to plot the ROC curve for one or more features with the possibility to include the AUC values in the legend.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels with two class labels.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss.

References

David J. Hand and Robert J. Till (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. Machine Learning 45(2), p. 171–186.

See Also

compute.aucs, pauclog, pauc, compute.auc.permutation, input_miss

Examples

```
# example for dataset without missing values
data(leukemia72_2)

# class label must be factor
leukemia72_2[,ncol(leukemia72_2)]<-as.factor(leukemia72_2[,ncol(leukemia72_2)])

add.legend<-TRUE
include.auc<-TRUE

attrs.no=c(1,2)
xdata=leukemia72_2[,c(attrs.no,ncol(leukemia72_2))]
plotRoc.curves(dattable=xdata,add.legend=add.legend,include.auc=include.auc)
```

ProcessData *Select the subset of features*

Description

The auxiliary function performs the discretization of the numerical features and is called from the several functions for feature selection. The discretization options include minimal description length (MDL), equal frequency and equal interval width methods. The results is in the form of "list", consisting of two fields: the processed dataset and the column numbers of the features. When the value of the input parameter "flag"=TRUE the second field will include the column numbers of the features, which have more than single interval after discretization.

Usage

```
ProcessData(matrix,disc.method,attrs.nominal,flag=FALSE)
```

Arguments

matrix	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
disc.method	a method used for feature discretization.The discretization options include minimal description length (MDL), equal frequency and equal interval width methods.
attrs.nominal	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.
flag	a binary logical value. If equals TRUE the output list will contain the processed dataset with the features, having more than one interval after discretization together with their names. In the case of FALSE value the processed dataset with all the features will be returned.

Details

This auxiliary function's main job is to descritize the numerical features using the one of the discretization methods. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned list consists of the the following fields:

`m3`	a processed dataset
`sel.feature`	a numeric vector with the column numbers of the features, having more than one interval value (when "flag"=TRUE). If "flag"=FALSE it return all the column numbers of the dataset.

References

H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," Data Mining and Knowledge Discovery, Vol. 6, No. 4, 2002, pp. 393-423.

See Also

`select.inf.gain`, `select.inf.symm`, `select.inf.chi2`, `select.fast.filter`, `select.process`

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

disc<-"MDL"
attrs.nominal=numeric()
flag=FALSE
out=ProcessData(matrix=data_test,disc.method=disc,
attrs.nominal=attrs.nominal,flag=flag)
```

`select.cfs` *Select the subset of features*

Description

This function selects the subset of features using the best first search strategy on the basis of correlation measure (CFS). CFS evaluates a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them. It can handle both numerical and nominal values. The results is in the form of "data.frame", consisting of the following fields: features (Biomarker) names and the positions of the features in the dataset. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "CFS" for feature selection. The variable "Index" of the data.frame is passed to the classification function.

Usage

```
select.cfs(matrix)
```

Arguments

`matrix`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.

Details

This function's main job is to select the subset of informative features according to best first search strategy using the correlation measure (informative theoretic measure). The measure consideres the individual predictive ability of each feature along with the degree of redundancy between them. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`. A returned list consists of the the following fields:

`Biomarker`	a character vector of feature names
`Index`	a numerical vector of the positions of the features in the dataset

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

input_miss, select.process

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])
out=select.cfs(matrix=data_test)
```

---

select.fast.filter *Select the subset of features*

Description

This function selects the subset of features on the basis of the fast correlation-based filter method (FCBF). It can handle both numerical and nominal values. At first it performs the discretization of the numerical features values, according to several optional discretization methods using the function ProcessData. A fast filter can identify relevant features as well as redundancy among relevant features without pairwise correlation analysis. The overall complexity of FCBF is O(MN logN), where M - number of samples, N - number of features.The results is in the form of "data.frame", consisting of the features (Biomarker) names, values of the information gain and the positions of the features in the dataset. The information gain value is the correlation between the features and the class. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "FastFilter" for feature selection. The variable "NumberFeature" of the data.frame is passed to the classification function.

Usage

```
select.fast.filter(matrix,disc.method,threshold,attrs.nominal)
```

Arguments

matrix	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
disc.method	a method used for feature discretization.The discretization options include minimal description length (MDL), equal frequency and equal interval width methods.

`threshold` a numeric threshold value for the correlation of feature with class to be included in the final subset.

`attrs.nominal` a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

Details

This function's main job is to select the subset of informative features according to correlation between features and class, and between features themselves. See the "Value" section to this page for more details. Before starting it calls the `ProcessData` function to make the discretization of numerical features.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned data.frame consists of the the following fields:

`Biomarker` a character vector of feature names

`Information.Gain` a numeric vector of information gain values for the features according to class

`NumberFeature` a numerical vector of the positions of the features in the dataset

References

L. Yu and H. Liu. "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution". In Proceedings of The Twentieth International Conference on Machine Leaning (ICML-03), Washington, D.C. pp. 856-863. August 21-24, 2003.

See Also

`ProcessData`, `input_miss`, `select.process`

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])
disc<-"MDL"
threshold=0.2
attrs.nominal=numeric()
out=select.fast.filter(data_test, disc.method=disc, threshold=threshold,
attrs.nominal=attrs.nominal)
```

```
# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
disc<-"MDL"
threshold=0.2
out=select.fast.filter(xdata, disc.method=disc, threshold=threshold,
attrs.nominal=attrs.nominal)
```

`select.forward.Corr` *Select the subset of features*

Description

This function selects the subset of features using the forward search strategy on the basis of correlation measure (CFS algorithm with forward search). CFS evaluates a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them. It can handle both numerical and nominal values. At the beginning the discretization of the numerical features values is performed using the function `ProcessData`. At the first step of the method the one-feature subset is selected according to its informative score, which takes into account the average feature to class correlation and the average feature to feature correlation. In the following steps the subset is incrementally extended according to the forward search strategy until the stopping criterion is met. The result is in the form of character vector with the names of the selected features. This function is used internally to perform the classification with feature selection using the function "classifier.loop".

Usage

```
select.forward.Corr(matrix,disc.method,attrs.nominal)
```

Arguments

`matrix`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.

`disc.method`	a method used for feature discretization.The discretization options include minimal description length (MDL), equal frequency and equal interval width methods.
`attrs.nominal`	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

Details

This function's main job is to select the subset of informative features according to forward selection strategy using the correlation measure (informative theoretic measure). The measure consideres the individual predictive ability of each feature along with the degree of redundancy between them. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned value is

`subset`	a character vector of the names of selected features

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

`input_miss`, `select.process`

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])
disc<-"MDL"
attrs.nominal=numeric()
out=select.forward.Corr(matrix=data_test,disc.method=disc,
attrs.nominal=attrs.nominal)
```

`select.forward.wrapper` *Select the subset of features*

Description

This function selects the subset of features using the wrapper method with decision tree algorithm and forward search strategy. It can handle both numerical and nominal values. The wrapper method makes use of the classification algorithm in order to estimate the quality measure of the feature subset. The method uses the built-in cross-validation procedure to estimate the accuracy of classification for the feature subset. At the first step of the method the one-feature subset is selected according to the quality measure. In the following steps the subset is incrementally extended according to the forward search strategy until the stopping criterion is met. The result is in the form of character vector with the names of the selected features. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "CorrSF" for feature selection.

Usage

```
select.forward.wrapper(dattable)
```

Arguments

`dattable`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.

Details

This function's main job is to select the subset of informative features according to forward selection strategy using the wrapper method. The decision tree is used as the classifier to estimate the quality of the feature subset. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned value is

`subset`	a character vector of the names of selected features

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

input_miss, select.process

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

out=select.forward.wrapper(dattable=data_test)
```

---

select.inf.chi2 *Ranks the features*

Description

This function calculates the features weights using the chi-squared (χ^2) statistic and performs the ranking of the features. It can handle both numerical and nominal values. At first it performs the discretization of the numerical features values, according to several optional discretization methods using the function ProcessData. This function measures the worth of a feature by computing the value of the χ^2 statistic with respect to the class.The results is in the form of "data.frame", consisting of the following fields: features (Biomarker) names, values of the chi-squared statistic and the positions of the features in the dataset. The features in the data.frame are sorted according to the chi-squared statistic values. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "Chi-square" for feature selection. The variable "NumberFeature" of the data.frame is passed to the classification function.

Usage

```
select.inf.chi2(matrix,disc.method,attrs.nominal)
```

Arguments

matrix	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
disc.method	a method used for feature discretization.The discretization options include minimal description length (MDL), equal frequency and equal interval width methods.
attrs.nominal	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

Details

This function's main job is to rank the features according to chi-squared statistic. See the "Value" section to this page for more details. Before starting it calls the ProcessData function to make the discretization of numerical features.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss. A returned data.frame consists of the the following fields:

Biomarker	a character vector of feature names
ChiSquare	a numeric vector of chi-squared values for the features according to class
NumberFeature	a numerical vector of the positions of the features in the dataset

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

ProcessData, input_miss, select.process

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])
disc<-"equal interval width"
attrs.nominal=numeric()
out=select.inf.chi2(data_test,disc.method=disc,attrs.nominal=attrs.nominal)

# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])
```

```
delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
disc<-"equal interval width"
out=select.inf.chi2(xdata,disc.method=disc,attrs.nominal=attrs.nominal)
```

`select.inf.gain` *Ranks the features*

Description

This function calculates the features weights using the Information Gain criterion measure and performs the ranking of the features (in decreasing order of Information Gain criteria). It can handle both numerical and nominal values. At first it performs the discretization of the numerical features values, according to several optional discretization methods using the function `ProcessData`. This function measures the worth of a feature by computing the Information Gain criterion measure with respect to the class.The results is in the form of "data.frame", consisting of the following fields: features (Biomarker) names, values of the Information Gain criterion measure and the positions of the features in the dataset. The features in the data.frame are sorted according to the Information Gain uncertainty criterion values. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "InformationGain" for feature selection. The variable "NumberFeature" of the data.frame is passed to the classification function.

Usage

```
select.inf.gain(matrix,disc.method,attrs.nominal)
```

Arguments

`matrix`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
`disc.method`	a method used for feature discretization.The discretization options include minimal description length (MDL), equal frequency and equal interval width methods.
`attrs.nominal`	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

Details

This function's main job is to rank the features according to Information Gain criterion. See the "Value" section to this page for more details. Before starting it calls the `ProcessData` function to make the discretization of numerical features.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`. A returned list consists of the the following fields:

`Biomarker`	a character vector of feature names
`Information.Gain`	a numeric vector of Information gain values for the features
`NumberFeature`	a numerical vector of the positions of the features in the dataset

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

`ProcessData`, `input_miss`, `select.process`

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])
disc<-"equal interval width"
attrs.nominal=numeric()
out=select.inf.gain(data_test,disc.method=disc,attrs.nominal=attrs.nominal)

# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
```

```
{
 xdata=out$data
}
disc<-"equal interval width"
out=select.inf.gain(xdata,disc.method=disc,attrs.nominal=attrs.nominal)
```

`select.inf.symm` *Ranks the features*

Description

This function calculates the features weights using the Symmetrical uncertainty criterion measure and performs the ranking of the features (in decreasing order of Symmetrical uncertainty criteria). It can handle both numerical and nominal values. At first it performs the discretization of the numerical features values, according to several optional discretization methods using the function `ProcessData`. This function measures the worth of a feature by computing the Symmetrical uncertainty criterion measure with respect to the class.The results is in the form of "data.frame", consisting of the following fields: features (Biomarker) names, values of the Symmetrical uncertainty criterion measure and the positions of the features in the dataset. The features in the data.frame are sorted according to the Symmetrical uncertainty criterion values. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "symmetrical.uncertainty" for feature selection. The variable "NumberFeature" of the data.frame is passed to the classification function.

Usage

```
select.inf.symm(matrix,disc.method,attrs.nominal)
```

Arguments

`matrix`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
`disc.method`	a method used for feature discretization.The discretization options include minimal description length (MDL), equal frequency and equal interval width methods.
`attrs.nominal`	a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

Details

This function's main job is to rank the features according to Symmetrical uncertainty criterion. See the "Value" section to this page for more details. Before starting it calls the `ProcessData` function to make the discretization of numerical features.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss. A returned data.frame consists of the the following fields:

Biomarker	a character vector of feature names
SymmetricalUncertainty	a numeric vector of Symmetrical uncertainty criterion values for the features according to class
NumberFeature	a numerical vector of the positions of the features in the dataset

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

ProcessData, input_miss, select.process

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])
disc<-"equal interval width"
attrs.nominal=numeric()
out=select.inf.symm(data_test,disc.method=disc,attrs.nominal=attrs.nominal)

# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
disc<-"equal interval width"
out=select.inf.symm(xdata,disc.method=disc,attrs.nominal=attrs.nominal)
```

`select.process` *Feature ranking and feature selection*

Description

The main function for the feature ranking or feature subset selection. It can handle both numerical and nominal values. It presents the infrastructure to perform the feature ranking or feature selection for the data set with two or more class labels. The function calls several feature ranking methods with different quality measures, including AUC values (functions `compute.aucs`), information gain (function `select.inf.gain`), symmetrical uncertainty (function `select.inf.symm`), chi-squared (χ^2) statistic (function `select.inf.chi2`). It also calls the number of feature selection methods, including fast correlation-based filter method (FCBF) (function `select.fast.filter`), Chi2 discretization algorithm (function `chi2.algorithm`), CFS algorithm with forward search (function `select.forward.Corr`), wrapper method with decision tree algorithm and forward search strategy (function `select.forward.wrapper`). The results is in the form of "numeric vector" with the column numbers of the selected features for features selection algorithms and ordered features' column numbers according to the criteria for feature ranking. The number of features can be limited to the "max.no.features" , which is the function input parameter. The output of the function is used in function "classifier.loop" in the process of classification.

Usage

```
select.process(dattable,method="InformationGain",disc.method="MDL",
threshold=0.2,threshold.consis=0.05,attrs.nominal=numeric(),
max.no.features=10)
```

Arguments

`dattable`	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.
`method`	a method of feature ranking or feature selection. There are 6 methods for feature ranking ("auc", "HUM", "Chi-square", "InformationGain", "symmetrical.uncertainty", "Relief") and 4 methods for feature selection ("FastFilter", "CFS", "CorrSF", "Chi2-algorithm")
`disc.method`	a method used for feature discretization. There are three options "MDL","equal interval width","equal frequency". The discretization options "MDL" assigned to the minimal description length (MDL) discretization algorithm, which is a supervised algorithm. The last two options refer to the unsupervized discretization algorithms.
`threshold`	a numeric threshold value for the correlation of feature with class to be included in the final subset. It is used by fast correlation-based filter method (FCBF)
`threshold.consis`	a numeric threshold value for the inconsistency rate. It is used by Chi2 discretization algorithm.

`attrs.nominal` a numerical vector, containing the column numbers of the nominal features, selected for the analysis.

`max.no.features` the maximal number of features to be selected or ranked.

Details

This function's main job is to present the infrostructure to perform the feature ranking or feature selection for the data set with two or more class labels. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function `input_miss`.

A returned value is

`sel.feat` a vector of column numbers of the selected features for features selection algorithms and ordered features' column numbers according to the criteria for feature ranking

References

H. Liu and L. Yu. "Toward Integrating Feature Selection Algorithms for Classification and Clustering", IEEE Trans. on Knowledge and Data Engineering, pdf, 17(4), 491-502, 2005.
L. Yu and H. Liu. "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution". In Proceedings of The Twentieth International Conference on Machine Leaning (ICML-03), Washington, D.C. pp. 856-863. August 21-24, 2003.

See Also

`select.inf.gain`, `select.inf.symm`, `select.inf.chi2`, `select.fast.filter`, `chi2.algorithm`, `select.forward.Corr`, `select.forward.wrapper`, `input_miss`

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

method="InformationGain"
disc<-"MDL"
thr=0.1
```

```
thr.cons=0.05
attrs.nominal=numeric()
max.f=15

out=select.process(data_test,method=method,disc.method=disc,
threshold=thr, threshold.consis=thr.cons,attrs.nominal=attrs.nominal,
max.no.features=max.f)


# example for dataset with missing values
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}

method="InformationGain"
disc<-"MDL"
thr=0.1
thr.cons=0.05
max.f=15

out=select.process(xdata,method=method,disc.method=disc,
threshold=thr, threshold.consis=thr.cons,attrs.nominal=attrs.nominal,
max.no.features=max.f)
```

`select.relief` *Ranks the features*

Description

This function calculates the features weights basing on a distance between instances. It can handle only numeric. The results is in the form of "data.frame", consisting of the following fields: features (Biomarker) names, weights and the positions of the features in the dataset. The features in the data.frame are sorted according to the weight values. This function is used internally to perform the classification with feature selection using the function "classifier.loop" with argument "Chi-square" for feature selection. The variable "NumberFeature" of the data.frame is passed to the classification function.

Usage

```
select.relief(matrix)
```

Arguments

matrix	a dataset, a matrix of feature values for several cases, the last column is for the class labels. Class labels could be numerical or character values. The maximal number of classes is ten.

Details

This function's main job is to rank the features according to weights. See the "Value" section to this page for more details.

Data can be provided in matrix form, where the rows correspond to cases with feature values and class label. The columns contain the values of individual features and the last column must contain class labels. The maximal number of class labels equals 10. The class label features and all the nominal features must be defined as factors.

Value

The data can be provided with reasonable number of missing values that must be at first preprocessed with one of the imputing methods in the function input_miss. A returned data.frame consists of the the following fields:

Biomarker	a character vector of feature names
Weights	a numeric vector of weight values for the features
NumberFeature	a numerical vector of the positions of the features in the dataset

References

Y. Wang, I.V. Tetko, M.A. Hall, E. Frank, A. Facius, K.F.X. Mayer, and H.W. Mewes, "Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach," Computational Biology and Chemistry, vol. 29, no. 1, pp. 37-46, 2005.

See Also

input_miss, select.process

Examples

```
# example for dataset without missing values
data(data_test)

# class label must be factor
data_test[,ncol(data_test)]<-as.factor(data_test[,ncol(data_test)])

out=select.relief(data_test)

# example for dataset with missing values
```

```
## Not run:
data(leukemia_miss)
xdata=leukemia_miss

# class label must be factor
xdata[,ncol(xdata)]<-as.factor(xdata[,ncol(xdata)])

# nominal features must be factors
attrs.nominal=101
xdata[,attrs.nominal]<-as.factor(xdata[,attrs.nominal])

delThre=0.2
out=input_miss(xdata,"mean.value",attrs.nominal,delThre)
if(out$flag.miss)
{
 xdata=out$data
}
out=select.relief(xdata)

## End(Not run)
```

Index