Package ‘HPDGClassifier’

March 9, 2017

Type Package

Title Hybrid Probabilistic Decision Graph Classifier

Depends R (>= 3.3.2), polynom, quadprog, codetools, foreach

Imports methods

Suggests infotheo, bnlearn, pROC

Version 0.2

Date 2017-3-9

Author Antonio Fernandez-Alvarez, Jose del Sagrado

Maintainer Jose del Sagrado `<jsagrado@ual.es>`

Description

This package allows the use of probabilistic decision graphs (PDGs) in supervised classification problems. PDGs have been used as classifiers by imposing certain structural restriction, ensuring that all the features are connected to the class. Also the classifiers learnt have the the ability of operating in hybrid domains, where discrete and continuous variables coexist. Hybrid Probabilistic Decision Graphs (HPDG) classifiers defined are based in the use of mixtures of truncated basis functions.

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

R topics documented:

`accuracy` *Gets the proportion of observations correctly classified.*

Description

Gets the proportion of observations correctly classified.

Usage

```
accuracy(classificMatrix)
```

Arguments

`classificMatrix`	a classification matrix.

Value

Returns the proportion of correctly classified observations.

See Also

classificationMatrix, tpr, fpr, precision, fmeasure, specificity

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
tprv <- tpr("a", cm)
fprv <- fpr(cl[1], cm)
prev <- precision("a", cm)
fv <- fmeasure(cl[1], cm)
spev <- specificity (cl[1], cm)
accv <- accuracy(cm)
```

`auc` *Computes the area under the roc curve (AUC) using the trapezoid approximation on a single (TPR, FPR) point.*

Description

Computes the area under the roc curve (AUC) using the trapezoid approximation on a single (TPR, FPR) point.

Usage

```
auc(class_label, classificMatrix)
```

Arguments

`class_label`	the label of the class.
`classificMatrix`	a classification matrix.

Value

Returns the AUC for the given class.

See Also

classificationMatrix, tpr, fpr, precision, fmeasure, specificity, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
aucv <- auc("a", cm)
aucv <- auc(cl[1], cm)
```

`classificationMatrix` *Function for building a classification(i.e. confusion) matrix for comparing the predictions obtained by a classifier on a set of observations with the real class labels associated to each one of them.*

Description

Function for building a classification(i.e. confusion) matrix for comparing the predictions obtained by a classifier on a set of observations with the real class labels associated to each one of them.

Usage

```
classificationMatrix(class_labels, real, predicted)
```

Arguments

`class_labels`	a vector containing all class labels.
`real`	a vector with the real class labels associated to observations.
`predicted`	a vector with the class labels, predicted by a classifier, for observations.

Value

Returns a classification matrix summarizing the classification results obtained for each class.

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
```

`classify` *Function for applying an Hybrid Probabilistic Decision Graph classifier (HPDG) to each observation in a dataset. The HPDG classifier predicts a class for each observation.*

Description

Function for applying an Hybrid Probabilistic Decision Graph classifier (HPDG) to each observation in a dataset. The HPDG classifier predicts a class for each observation.

Usage

```
classify(hpdg_classifier, dataset)
```

Arguments

`hpdg_classifier`	an Hybrid Probabilistic Decision Graph classifier.
`dataset`	a String indicating the data file (the path has to be included also). Data columns are separated by ";" and a header indicates the variables names.

Value

Return a vector of predicted values for the class variable, one for each observation in

See Also

`hpdgclassifier`, score

Examples

```
## Not run:
## Learns a hpdg classifier from a dataset
classifier <- hpdgclassifier("data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)
predicted_values <- classify (classifier, "data/iris.csv")

## End(Not run)
```

`cmplot` *Plots the classes counts in a classification matrix.*

Description

Plots the classes counts in a classification matrix.

Usage

```
cmplot(classificMatrix, short = TRUE)
```

Arguments

`classificMatrix`	a confusion matrix.
`short`	indicates the type of plot. By default a short plot is produced: observations are grouped by class.

Value

Returns a plot of the class counts in a confusion matrix.

See Also

classificationMatrix

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
cmplot(cm)
cmplot(cm, FALSE)
cmplot(cm, short=FALSE)
```

`crossValidation` *Function for performing k-fold cross validation of an Hybrid Probabilistic Decision Graph (HPDG) classifier on a dataset. The HPDG Classifier is learnt on the whole data set. The evaluation results displayed are obtained doing the k-fold cross validation process.*

Description

Function for performing k-fold cross validation of an Hybrid Probabilistic Decision Graph (HPDG) classifier on a dataset. The HPDG Classifier is learnt on the whole data set. The evaluation results displayed are obtained doing the k-fold cross validation process.

Usage

```
crossValidation(folds = NULL, dataset, method = c("ranked", "rankedCR",
  "CR", "NB", "ChowLiu"), densityType = c("MOP", "MTE"), maxIntervals,
  collapseRate, mergeRate, maxTermsDensity)
```

Arguments

`folds`	the number of folds to be used. It has to be greater than 1. By default is set to 5.
`dataset`	a String indicating the data file (the path has to be included also). Data columns are separated by ";" and a header indicates the variables names.
`method`	the method applied to build the classifier: • rankedThe variables are ranked by mutual information and they are connected as: C -> X_1 -> X_2 -> X_3 -> ... • rankedCRThe variables are ranked by mutual information and then each X_i is connected to the leaf that maximizes the CR of the classifier wrt to a test database. • CRIn each step a non included variable Xj is connected to a leaf variable Xi satisfying that <X_i, X_j> = arg max CR(X_i,X_j,pdg,train,test). This is computationally the hardest option. See Nielsen et. al (2009) - Algorithm 2. • NBVariables are arranged following a NB structure. • ChowLiuVariables are arranged in a TAN structure given by the Chow Liu algorithm.
`densityType`	the type of density ("MOP" or "MTE").
`maxIntervals`	maximum number of intervals in which the continuous parents are split.
`collapseRate`	A number k in the formula nCases = k * R(Xi), where R(Xi) is the number of states in the variable Xi and nCases is the minimum number of cases to be allowed to learn the parameters without collapsing nodes. By default this number is set to 0.5.
`mergeRate`	A number between 0 and 1 indicating the percentage of parameters to be checked for merging (non mandatory argument). By default this argument is set to 0.5.
`maxTermsDensity`	the maximum number of terms to be fit in the density.

Value

Return an HPDG classifier learnt using the whole given dataset.

See Also

`hpdgclassifier`, `classificationMatrix`

Examples

```
 ## Not run:
## Learns a hpdg classifier from a dataset and evaluates its performance applying
## k-fold cross validation
classifier <- crossValidation(2, "data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)

## End(Not run)
```

`evaluate` *Function for evaluating an Hybrid Probabilistic Prediction Graph (HPDG) on a dataset. The HPDG classifier was previously builded and is applied to predict a class value for each observation in the given dataset. The evaluation is performed comparing the each predicted class value with the real class value given in the dataset.*

Description

The function displays the evaluation measures computed using these real an predicted class values.

Usage

```
evaluate(hpdg_classifier, dataset)
```

Arguments

`hpdg_classifier`	an Hybrid Probabilistic Decision Graph classifier.
`dataset`	a String indicating the data file (the path has to be included also). Data columns are separated by ";" and a header indicates the variables names.

Value

A classification matrix.

See Also

`hpdgclassifier`, `classificationMatrix`

Examples

```
## Not run:
## Learns a hpdg classifier from a dataset
classifier <- hpdgclassifier("data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)
evaluate (classifier, "data/iris.csv")

## End(Not run)
```

`exhibit` *Function for showing the components of an Hybrid Probabilistic Decision Graph classifier (HPDG): variables, structure and parameters.*

Description

Function for showing the components of an Hybrid Probabilistic Decision Graph classifier (HPDG): variables, structure and parameters.

Usage

```
exhibit(hpdg_classifier)
```

Arguments

`hpdg_classifier`	an Hybrid Probabilistic Decision Graph classifier.

Value

Return in console information about the estructure and parameters of an Hybrid Probabilistic Decision Graph classifier (HPDG)

See Also

`hpdgclassifier`

Examples

```
## Not run:
## Exhibits the components of a hpdg classifier learned from a dataset
classifier <- hpdgclassifier("data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)
exhibit(classifier)

## End(Not run)
```

`fmeasure` *Computes the harmonic mean of sensitivity (i.e. true positive rate) and precision. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).*

Description

Computes the harmonic mean of sensitivity (i.e. true positive rate) and precision. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).

Usage

```
fmeasure(class_label, classificMatrix)
```

Arguments

`class_label`	the label of the class.
`classificMatrix`	a classification matrix.

Value

Returns the f-measure for the given class.

See Also

classificationMatrix,tpr, fpr, precision, specificity, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
fv <- fmeasure("a", cm)
fv <- fmeasure(cl[1], cm)
```

`fpr` *Gets the proportion of false positives (i.e. observations that were incorrectly recognized as belonging to the class). For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).*

Description

Gets the proportion of false positives (i.e. observations that were incorrectly recognized as belonging to the class). For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).

Usage

```
fpr(class_label, classificMatrix)
```

Arguments

`class_label`	the label of the class.
`classificMatrix`	a classification matrix.

Value

Returns the proportion of false positives for the given class.

See Also

classificationMatrix, tpr, precision, fmeasure, specificity, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
fprv <- fpr("a", cm)
fprv <- fpr(cl[1], cm)
```

`hpdgclassifier` *Function for learning Hybrid Probabilistic Decision Graph (HPDG) classifiers from data*

Description

Learns an Hybrid Probabilistic Decision Graph (HPDG) classifier from a given dataset

Usage

```
hpdgclassifier(dataset, method = c("ranked", "rankedCR", "CR", "NB",
  "ChowLiu"), densityType = c("MOP", "MTE"), maxIntervals, collapseRate,
  mergeRate, maxTermsDensity)
```

Arguments

`dataset` a String indicating the data file (the path has to be included also). Data columns are separated by ";" and a header indicates the variables names.

`method` the method applied to build the classifier:

- rankedThe variables are ranked by mutual information and they are connected as: C -> X_1 -> X_2 -> X_3 -> ...
- rankedCRThe variables are ranked by mutual information and then each X_i is connected to the leaf that maximizes the CR of the classifier wrt to a test database.

- CRIn each step a non included variable Xj is connected to a leaf variable Xi satisfying that <X_i, X_j> = arg max CR(X_i,X_j,pdg,train,test). This is computationally the hardest option. See Nielsen et. al (2009) - Algorithm 2.
- NBVariables are arranged following a NB structure.
- ChowLiuVariables are arranged in a TAN structure given by the Chow Liu algorithm.

`densityType`	the type of density ("MOP" or "MTE").
`maxIntervals`	maximum number of intervals in which the continuous parents are split.
`collapseRate`	A number k in the formula nCases = k * R(Xi), where R(Xi) is the number of states in the variable Xi and nCases is the minimum number of cases to be allowed to learn the parameters without collapsing nodes. By default this number is set to 0.5.
`mergeRate`	A number between 0 and 1 indicating the percentage of parameters to be checked for merging (non mandatory argument). By default this argument is set to 0.5.
`maxTermsDensity`	the maximum number of terms to be fit in the density.

Value

The HPDG classifier learnt from data.

See Also

`chow.liu` in package `bnlearn`

Examples

```
## Not run:
## Learns a hpdg classifier from a dataset
classifier <- hpdgclassifier("data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)

## End(Not run)
```

`iris` *Iris Data Set*

Description

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Usage

`iris`

Format

A data frame with 150 rows and 5 variables. The variables are as follows:

Sepal.Length in cm

Sepal.Width in cm

Petal.Length in cm

Petal.Width in cm

Species (the class) is one of Iris Setosa, Iris Versicolour, Iris Virginica.

Source

The dataset was obtained from the *UCI Machine Learning Repository* at `https://archive.ics.uci.edu/ml/datasets/Iris`.

`is.classificationMatrix` *Function to test whether the matrix contains integer values, has the same number of rows and cols and, row and col names coincide*

Description

Function to test whether the matrix contains integer values, has the same number of rows and cols and, row and col names coincide

Usage

```
is.classificationMatrix(m)
```

Arguments

`m`	a matrix

Value

Returns 'TRUE' if matrix contains integer values, has the same number of rows and cols and, row and col names coincide

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
is.classificationMatrix(cm)
```

`precision` *Gets the proportion of true positives from observations classified as belonging to the class. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).*

Description

Gets the proportion of true positives from observations classified as belonging to the class. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).

Usage

```
precision(class_label, classificMatrix)
```

Arguments

`class_label`	the label of the class.
`classificMatrix`	a classification matrix.

Value

Returns the precision for the given class.

See Also

classificationMatrix, tpr, fpr, fmeasure, specificity, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
prev <- precision(cl[1], cm)
prev <- precision("a", cm)
```

`ROCplot` *Plots a ROC curve from a series of labels and scores returned by a classifier. In multiclass cases plots a ROC curve for each class applying a one versus all approach.*

Description

Plots a ROC curve from a series of labels and scores returned by a classifier. In multiclass cases plots a ROC curve for each class applying a one versus all approach.

Usage

```
ROCplot(real, score, single = TRUE, plot_classes = NULL)
```

Arguments

`real`	a vector with the real class labels associated to observations.
`score`	a vector with the probability values assigned to each observation by the classifier. The length of this vector has to coincid with that of the real labels vector.
`single`	an optional boolean. It indicates if a single plot has to be produced or not. By default a single plot is produced.
`plot_classes`	an optional vector with the different class labels whose ROC curves have to be displayed. Only those classes included in the `real` vector are taken into account. By default it is set to NULL.

Value

Returns plots with one ROC curve (binary case) or with a ROC curve per class (multiclass case) using the the class reference formulation (i.e. one versus all) approach. Classes are considered in the order specified by the vector with the `real` class labels.

See Also

classificationMatrix, `pROC`

Examples

```
## Simulates classification results
class_values <- sample(c(TRUE, FALSE), 20, replace = TRUE )
classifier_scores <- runif(20)
ROCplot(class_values, classifier_scores)

## Simulates multiclASS classification results
class_values <- sample( c("a","b","c"), 20, replace = TRUE )
classifier_scores <- runif(20)
# Plot multiclass ROC
# A unique plot
ROCplot(class_values, classifier_scores)
ROCplot(class_values, classifier_scores, plot_classes = c("a", "c"))
# One plot per class
ROCplot(class_values, classifier_scores, FALSE)
ROCplot(class_values, classifier_scores, FALSE, plot_classes = c("a", "c"))
```

`score` *Function for applying an Hybrid Probabilistic Decision Graph classifier (HPDG) to each observation in a dataset. The HPDG classifier predicts a class for each observation and return its probability value.*

Description

Function for applying an Hybrid Probabilistic Decision Graph classifier (HPDG) to each observation in a dataset. The HPDG classifier predicts a class for each observation and return its probability value.

Usage

```
score(hpdg_classifier, dataset)
```

Arguments

`hpdg_classifier`	an Hybrid Probabilistic Decision Graph classifier.
`dataset`	a String indicating the data file (the path has to be included also). Data columns are separated by ";" and a header indicates the variables names.

Value

Return a vector with the probability values associated to the predicted values for the class variable, one for each observation in the given dataset

See Also

`hpdgclassifier`, classify

Examples

```
## Not run:
## Learns a hpdg classifier from a dataset
classifier <- hpdgclassifier("data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)
predicted_values <- classify (classifier, "data/iris.csv")
predicted_scores <- score(classifier, "data/iris.csv")

## End(Not run)
```

`scores` *Function for applying an Hybrid Probabilistic Decision Graph classifier (HPDG) to each observation in a dataset. The HPDG classifier computes the class distribution for each observation and returns it.*

Description

Function for applying an Hybrid Probabilistic Decision Graph classifier (HPDG) to each observation in a dataset. The HPDG classifier computes the class distribution for each observation and returns it.

Usage

```
scores(hpdg_classifier, dataset)
```

Arguments

`hpdg_classifier`	an Hybrid Probabilistic Decision Graph classifier.
`dataset`	a String indicating the data file (the path has to be included also). Data columns are separated by ";" and a header indicates the variables names.

Value

Return a matrix with the class probability distribution associated to each observation in the given dataset

See Also

`hpdgclassifier`, classify, score

Examples

```
## Not run:
## Learns a hpdg classifier from a dataset
classifier <- hpdgclassifier("data/iris.csv", "rankedCR", "MOP", 3, 2, 0.5, 5)
predicted_values <- classify (classifier, "data/iris.csv")
predicted_scores <- score (classifier, "data/iris.csv")
predicted_distributions <- scores(classifier, "data/iris.csv")

## End(Not run)
```

`specificity` *Gets the proportion of negatives (i.e. observations not belonging to the class) that are correctly identified as such. It quantifies the avoiding of false positives. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).*

Description

Gets the proportion of negatives (i.e. observations not belonging to the class) that are correctly identified as such. It quantifies the avoiding of false positives. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).

Usage

```
specificity(class_label, classificMatrix)
```

Arguments

`class_label`	the label of the class.
`classificMatrix`	a classification matrix.

Value

Returns the specificity for the given class.

See Also

classificationMatrix, tpr, fpr, precision, fmeasure, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
spev <- specificity("a", cm)
spev <- specificity (cl[1], cm)
```

`summarize` *Sumarizes the classification results encoded in a classification matrix. Detiled results by class are included. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).*

Description

Sumarizes the classification results encoded in a classification matrix. Detiled results by class are included. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).

Usage

```
summarize(classificMatrix)
```

Arguments

`classificMatrix`	a classification matrix.

Value

Returns a summary of the classifications results, including the confusion matrix itself, detailed results by class and global accuracy.

See Also

classificationMatrix, tpr, fpr, precision, fmeasure, specificity, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
summarize(cm)
```

`tpr` *Gets the proportion of positives (i.e. observations belonging to the class) that are correctly identified as such. It quantifies the avoiding of false negatives. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).*

Description

Gets the proportion of positives (i.e. observations belonging to the class) that are correctly identified as such. It quantifies the avoiding of false negatives. For more than two classes, these results are calculated comparing each class to the remaining classes (i.e. a "one versus all" approach).

Usage

```
tpr(class_label, classificMatrix)
```

Arguments

`class_label`	the label of the class.
`classificMatrix`	a classification matrix.

Value

Returns the proportion of true positives for the given class.

See Also

classificationMatrix, fpr, precision, fmeasure, specificity, accuracy

Examples

```
## Builds a classification matrix
cl <- c("a", "b", "c")
rl <- c("a", "b", "b", "c", "a", "a")
pl <- c("a", "b", "c", "c", "b", "a")
cm <- classificationMatrix(cl, rl, pl)
tprv <- tpr("a", cm)
```

Index