Package ‘mfe’

May 5, 2020

Type Package

Title Meta-Feature Extractor

Version 0.1.5

Date 2020-05-04

Description Extracts meta-features from datasets to support the design of recommendation systems based on Meta-Learning. The meta-features, also called characterization measures, are able to characterize the complexity of datasets and to provide estimates of algorithm performance. The package contains not only the standard characterization measures, but also more recent characterization measures. By making available a large set of meta-feature extraction functions, tasks like comprehensive data characterization, deep data exploration and large number of Meta-Learning based data analysis can be performed. These concepts are described in the paper: Rivolli A., Garcia L., Soares c., Vanschoren J. and Carvalho A. (2018) <arXiv:1808.10406>.

URL https://github.com/rivolli/mfe

Depends R (>= 3.3),

Imports cluster, clusterCrit, ECoL(>= 0.3), e1071, infotheo, MASS, rpart, rrcov, stats, utils

Suggests knitr, rmarkdown, testthat

License MIT + file LICENSE

LazyData true

BugReports https://github.com/rivolli/mfe/issues

RoxygenNote 7.1.0

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Adriano Rivolli [aut, cre],
Luis P. F. Garcia [aut],
Andre C. P. L. F. de Carvalho [ths]

Maintainer Adriano Rivolli <rivolli@utfpr.edu.br>

Repository CRAN

Date/Publication 2020-05-05 15:20:02 UTC

R topics documented:

`clustering` *Clustering Meta-features*

Description

Clustering measures extract information about validation index.

Usage

```
clustering(...)

## Default S3 method:
clustering(
  x,
  y,
  features = "all",
  summary = c("mean", "sd"),
  transform = TRUE,
  ...
)

## S3 method for class 'formula'
clustering(
  formula,
  data,
  features = "all",
  summary = c("mean", "sd"),
  transform = TRUE,
  ...
)
```

Arguments

`...`	Further arguments passed to the summarization functions.
`x`	A data.frame contained only the input attributes.
`y`	A factor response vector with one label for each row/component of x.
`features`	A list of features names or `"all"` to include all them.
`summary`	A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)
`transform`	A logical value indicating if the categorical attributes should be transformed. If `FALSE` they will be ignored. (Default: `TRUE`)
`formula`	A formula to define the class column.
`data`	A data.frame dataset contained the input attributes and class. The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"vdu" Calculate the Dunn Index.

"vdb" Calculate the Davies and Bouldin Index.

"int" Calculate the INT index.

"sil" Calculate the mean silhouette value from data.

"pb" Pearson Correlation between class matching and instance distances.

"ch" Calinski and Harabaz index.

"nre" Normalized relative entropy.

"sc" Mean of the number of examples per class.

Value

A list named by the requested meta-features.

References

Bruno A. Pimentel, and Andre C. P. L. F. de Carvalho. A new data characterization for selecting clustering algorithms using meta-learning. Information Sciences, volume 477, pages 203 - 219, 2019.

See Also

Other meta-features: `complexity()`, `concept()`, `general()`, `infotheo()`, `itemset()`, `landmarking()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all meta-features using formula
clustering(Species ~ ., iris)

## Extract some meta-features
clustering(iris[1:4], iris[5], c("vdu", "vdb", "sil"))

## Use another summarization function
clustering(Species ~ ., iris, summary=c("min", "median", "max"))
```

`complexity` *Complexity meta-features*

Description

The complexity group is a set of measures to characterize the complexity of classification problems based on aspects that quantify the linearity of the data, the presence of informative feature, the sparsity and dimensionality of the datasets.

Usage

```
complexity(...)

## Default S3 method:
complexity(x, y, features = "all", summary = c("mean", "sd"), ...)

## S3 method for class 'formula'
complexity(formula, data, features = "all", summary = c("mean", "sd"), ...)
```

Arguments

`...`	Not used.
`x`	A data.frame contained only the input attributes.
`y`	A factor response vector with one label for each row/component of x.
`features`	A list of features names or `"all"` to include all them. The supported values are described in the details section. (Default: `"all"`)

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class

Details

The following features are allowed for classification problems:

"C1" Entropy of class proportions.

"C2" Multi-class imbalance ratio.

"F1" Fisher's discriminant ratio.

"F1v" The directional-vector Fisher's discriminant ratio.

"F2" Overlapping of the per-class bounding boxes.

"F3" Maximum individual feature efficiency.

"F4" Collective feature efficiency.

"L1" Distance of erroneous instances to a linear classifier.

"L2" Training error of a linear classifier.

"L3" Nonlinearity of a linear classifier.

"LSC" Local-Set cardinality average.

"N1" Fraction of points lying on the class boundary.

"N2" Average intra/inter class nearest neighbor distances.

"N3" Leave-one-out error rate of the 1-nearest neighbor algorithm.

"N4" Nonlinearity of the one-nearest neighbor classifier.

"T1" Fraction of maximum covering spheres on data.

"T2" Average number of samples per dimension.

"T3" Average intrinsic dimensionality per number of examples.

"T4" Intrinsic dimensionality proportion.

Also it is possible to ask for a subgroup of features:

"balance" Include the measures C1 and C2.

"dimensionality" Include the measures T2, T3 and T4.

"linearity" Include the measures L1, L2 and L3.

"neighborhood" Include the measures N1, N2, N3, N4, T1 and LSC.

"network" Include the measures Density, ClsCoef and Hubs.

"overlapping" Include the measures F1, F1v, F2, F3 and F4.

Value

A list named by the requested meta-features.

References

Ana C. Lorena, Luis P. F. Garcia, Jens Lehmann, Marcilio C. P. Souto, and Tin Kam Ho. 2019. How Complex Is Your Classification Problem?: A Survey on Measuring Classification Complexity. ACM Comput. Surv. 52, 5.

Lorena, A. C., Maciel, A. I., de Miranda, P. B. C., Costa, I. G., and Prudencio, R. B. C. (2018). Data complexity meta-features for regression problems. Machine Learning, 107(1):209-246.

Ho, T., and Basu, M. (2002). Complexity measures of supervised classification problems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3):289-300.

See Also

Other meta-features: `clustering()`, `concept()`, `general()`, `infotheo()`, `itemset()`, `landmarking()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all metafeatures
complexity(Species ~ ., iris)

## Extract some metafeatures
complexity(iris[30:120, 1:4], iris[30:120, 5], c("F1", "F2", "linearity"))
```

---

concept *Concept Meta-features*

Description

Concept characterization features measure the sparsity of the input space and the irregularity of the input-output distribution measures extract information about validation index.

Usage

```
concept(...)

## Default S3 method:
concept(
  x,
  y,
  features = "all",
  summary = c("mean", "sd"),
  transform = TRUE,
  ...
)

## S3 method for class 'formula'
concept(
  formula,
  data,
  features = "all",
  summary = c("mean", "sd"),
  transform = TRUE,
  ...
)
```

Arguments

`...` Further arguments passed to the summarization functions.

`x` A data.frame contained only the input attributes.

`y` A factor response vector with one label for each row/component of x.

`features` A list of features names or `"all"` to include all them.

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`transform` A logical value indicating if the categorical attributes should be transformed. If `FALSE` they will be ignored. (Default: `TRUE`)

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class. The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"cohesiveness" Example Cohesiveness is a different version of the wgDist measure.

"conceptvar" Concept variation estimates the variability of class labels among examples.

"impconceptvar" Improved concept variation is a different version of the conceptvar measure.

"wgDist" Weighted distance captures how dense or sparse is the example distribution.

Value

A list named by the requested meta-features.

References

Vilalta, R., & Drissi, Y. (2002). A characterization of difficult problems in classification. In M. A. Wani, H. R. Arabnia, K. J. Cios, K. Hafeez, G. Kendall (Eds.), Proceedings ofthe 2002 international conference on machine learning and applications - ICMLA 2002, June 24-27, 2002, Las Vegas, Nevada (pp. 133-138).

Vilalta, R., 1999. Understanding accuracy performance through concept characterization and algorithm analysis. In: ECML Workshop on Recent Advances in Meta-Learning and Future Work. pp. 3-9.

See Also

Other meta-features: `clustering()`, `complexity()`, `general()`, `infotheo()`, `itemset()`, `landmarking()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all meta-features using formula
concept(Species ~ ., iris)

## Extract some meta-features
concept(iris[1:4], iris[5], c("conceptvar"))

## Use another summarization function
concept(Species ~ ., iris, summary=c("min", "median", "max"))
```

`general` *General meta-features*

Description

General meta-features include general information related to the dataset. It is also known as simple measures.

Usage

```
general(...)

## Default S3 method:
general(x, y, features = "all", summary = c("mean", "sd"), ...)

## S3 method for class 'formula'
general(formula, data, features = "all", summary = c("mean", "sd"), ...)
```

Arguments

`...` Not used.

`x` A data.frame contained only the input attributes.

`y` A factor response vector with one label for each row/component of x.

`features` A list of features names or `"all"` to include all them. The supported values are described in the details section. (Default: `"all"`)

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class

Details

The following features are allowed for this method:

"attrToInst" Ratio of the number of attributes per the number of instances, also known as dimensionality.

"catToNum" Ratio of the number of categorical attributes per the number of numeric attributes.

"freqClass" Proportion of the classes values (multi-valued).

"instToAttr" Ratio of the number of instances per the number of attributes.

"nrAttr" Number of attributes.

"nrBin" Number of binary attributes.

"nrCat" Number of categorical attributes.

"nrClass" Number of classes.

"nrInst" Number of instances.

"nrNum" Number of numeric attributes.

"numToCat" Ratio of the number of numeric attributes per the number of categorical attributes.

Value

A list named by the requested meta-features.

References

Donald Michie, David J. Spiegelhalter, Charles C. Taylor, and John Campbell. Machine Learning, Neural and Statistical Classification, volume 37. Ellis Horwood Upper Saddle River, 1994.

Guido Lindner and Rudi Studer. AST: Support for algorithm selection with a CBR approach. In European Conference on Principles of Data Mining and Knowledge Discovery (PKDD), pages 418 - 423, 1999.

Ciro Castiello, Giovanna Castellano, and Anna M. Fanelli. Meta-data: Characterization of input features for meta-learning. In 2nd International Conference on Modeling Decisions for Artificial Intelligence (MDAI), pages 457 - 468, 2005.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `infotheo()`, `itemset()`, `landmarking()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all metafeatures
general(Species ~ ., iris)

## Extract some metafeatures
general(iris[1:100, 1:4], iris[1:100, 5], c("nrAttr", "nrClass"))

## Extract all meta-features without summarize prop.class
general(Species ~ ., iris, summary=c())

## Use another summarization functions
general(Species ~ ., iris, summary=c("sd","min","iqr"))
```

infotheo *Information-theoretic meta-features*

Description

Information-theoretic meta-features are particularly appropriate to describe discrete (categorical) attributes, but they also fit continuous ones so a discretization is required.

Usage

```
infotheo(...)

## Default S3 method:
infotheo(
  x,
  y,
  features = "all",
  summary = c("mean", "sd"),
  transform = TRUE,
  ...
)

## S3 method for class 'formula'
infotheo(
  formula,
  data,
  features = "all",
  summary = c("mean", "sd"),
  transform = TRUE,
  ...
)
```

Arguments

`...`	Further arguments passed to the summarization functions.
`x`	A data.frame contained only the input attributes.
`y`	A factor response vector with one label for each row/component of x.
`features`	A list of features names or `"all"` to include all them. The supported values are described in the details section. (Default: `"all"`)
`summary`	A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)
`transform`	A logical value indicating if the numeric attributes should be transformed. If `FALSE` they will be ignored. (Default: `TRUE`)
`formula`	A formula to define the class column.
`data`	A data.frame dataset contained the input attributes and class The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"attrConc" Attributes concentration. It is the Goodman and Kruskal's tau measure otherwise known as the concentration coefficient computed for each pair of attributes (multi-valued).

"attrEnt" Attributes entropy, a measure of randomness of each attributes in the dataset (multi-valued).

"classConc" Class concentration, similar to "attrConc", however, it is computed for each attribute and the class (multi-valued).

"classEnt" Class entropy, which describes how much information is necessary to specify the class in the dataset.

"eqNumAttr" Equivalent number of attributes, which represents the number of attributes suitable to optimally solve the classification task using the dataset.

"jointEnt" Joint entropy, which represents the total entropy of each attribute and the class (multi-valued).

"mutInf" Mutual information, that is the common information shared between each attribute and the class in the dataset (multi-valued).

"nsRatio" Noise ratio, which describes the amount of irrelevant information contained in the dataset.

This method uses the unsupervised data discretization procedure provided by discretize function, where the default values are used when `transform=TRUE`.

Value

A list named by the requested meta-features.

References

Donald Michie, David J. Spiegelhalter, Charles C. Taylor, and John Campbell. Machine Learning, Neural and Statistical Classification, volume 37. Ellis Horwood Upper Saddle River, 1994.

Alexandros Kalousis and Melanie Hilario. Model selection via meta-learning: a comparative study. International Journal on Artificial Intelligence Tools, volume 10, pages 525 - 554, 2001.

Ciro Castiello, Giovanna Castellano, and Anna Maria Fanelli. Meta-data: Characterization of input features for meta-learning. In 2nd International Conference on Modeling Decisions for Artificial Intelligence (MDAI), pages 457 - 468, 2005.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `general()`, `itemset()`, `landmarking()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all metafeatures
infotheo(Species ~ ., iris)

## Extract some metafeatures
infotheo(iris[1:4], iris[5], c("classEnt", "jointEnt"))

## Extract all meta-features without summarize the results
infotheo(Species ~ ., iris, summary=c())

## Use another summarization functions
infotheo(Species ~ ., iris, summary=c("min", "median", "max"))

## Do not transform the data (using only categorical attributes)
infotheo(Species ~ ., iris, transform=FALSE)
```

itemset *Itemset Meta-features*

Description

Itemset characterization features measure measure the distribution of values of both single attributes and pairs of attributes.

Usage

```
itemset(...)

## Default S3 method:
itemset(x, y, features = "all", summary = c("mean", "sd"), ...)

## S3 method for class 'formula'
itemset(formula, data, features = "all", summary = c("mean", "sd"), ...)
```

Arguments

`...`	Further arguments passed to the summarization functions.
`x`	A data.frame contained only the input attributes.
`y`	A factor response vector with one label for each row/component of x.
`features`	A list of features names or `"all"` to include all them.
`summary`	A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)
`formula`	A formula to define the class column.
`data`	A data.frame dataset contained the input attributes and class. The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"oneitemset" Individual frequency of each attributes' value.

"twoitemset" Correlation information of the two attributes' value pairs.

"clssitemset" It is a two itemset computed using a predictive attribute and the target.

Value

A list named by the requested meta-features.

References

Song, Q., Wang, G., & Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. Pattern Recognition, 45(7), 2672-2689.

Wang, G., Song, Q., & Zhu, X. (2015). An improved data characterization method and its application in classification algorithm recommendation. Applied Intelligence, 43(4), 892-912.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `general()`, `infotheo()`, `landmarking()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all meta-features using formula
itemset(Species ~ ., iris)

## Extract some meta-features
itemset(iris[1:4], iris[5], c("oneitemset"))

## Use another summarization function
itemset(Species ~ ., iris, summary=c("min", "median", "max"))
```

`landmarking` *Landmarking and Subsampling Landmarking Meta-features*

Description

Landmarking measures are simple and fast learners, from which performance can be extracted.

Usage

```
landmarking(...)

## Default S3 method:
landmarking(
  x,
  y,
  features = "all",
  summary = c("mean", "sd"),
  size = 1,
  folds = 10,
  score = "accuracy",
  ...
)

## S3 method for class 'formula'
landmarking(
  formula,
  data,
  features = "all",
  summary = c("mean", "sd"),
  size = 1,
  folds = 10,
  score = "accuracy",
  ...
)
```

Arguments

`...` Further arguments passed to the summarization functions.

`x` A data.frame contained only the input attributes.

`y` A factor response vector with one label for each row/component of x.

`features` A list of features names or `"all"` to include all them.

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`size` The percentage of examples subsampled. Values different from 1 generate the subsampling-based landmarking metafeatures. (Default: 1.0)

`folds` The number of k equal size subsamples in k-fold cross-validation.(Default: 10)

`score` The evaluation measure used to score the classification performance. `c("accuracy","balanced.accura` (Default: `"accuracy"`).

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class. The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"bestNode" Construct a single decision tree node model induced by the most informative attribute to establish the linear separability (multi-valued).

"eliteNN" Elite nearest neighbor uses the most informative attribute in the dataset to induce the 1-nearest neighbor. With the subset of informative attributes is expected that the models should be noise tolerant (multi-valued).

"linearDiscr" Apply the Linear Discriminant classifier to construct a linear split (non parallel axis) in the data to establish the linear separability (multi-valued).

"naiveBayes" Evaluate the performance of the Naive Bayes classifier. It assumes that the attributes are independent and each example belongs to a certain class based on the Bayes probability (multi-valued).

"oneNN" Evaluate the performance of the 1-nearest neighbor classifier. It uses the euclidean distance of the nearest neighbor to determine how noisy is the data (multi-valued).

"randomNode" Construct a single decision tree node model induced by a random attribute. The combination with `"bestNode"` measure can establish the linear separability (multi-valued).

"worstNode" Construct a single decision tree node model induced by the worst informative attribute. The combination with `"bestNode"` measure can establish the linear separability (multi-valued).

Value

A list named by the requested meta-features.

References

Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In 17th International Conference on Machine Learning (ICML), pages 743 - 750, 2000.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `general()`, `infotheo()`, `itemset()`, `model.based()`, `relative()`, `statistical()`

Examples

```
## Extract all meta-features using formula
landmarking(Species ~ ., iris)

## Extract some meta-features
landmarking(iris[1:4], iris[5], c("bestNode", "randomNode", "worstNode"))

## Use another summarization function
landmarking(Species ~ ., iris, summary=c("min", "median", "max"))

## Use 2 folds and balanced accuracy
```

```
landmarking(Species ~ ., iris, folds=2, score="balanced.accuracy")

## Extract the subsapling landmarking
landmarking(Species ~ ., iris, size=0.7)
```

`ls.clustering` *List the best clustering meta-features*

Description

List the best clustering meta-features

Usage

```
ls.clustering()
```

Value

A list of best neighbor meta-features names.

Examples

```
ls.clustering()
```

`ls.complexity` *List the complexity meta-features*

Description

List the complexity meta-features

Usage

```
ls.complexity()
```

Value

A list of complexity meta-features names

Examples

```
ls.complexity()
```

`ls.concept` *List the best concept meta-features*

Description

List the best concept meta-features

Usage

```
ls.concept()
```

Value

A list of concept meta-features names.

Examples

```
ls.concept()
```

`ls.general` *List the general meta-features*

Description

List the general meta-features

Usage

```
ls.general()
```

Value

A list of general meta-features names

Examples

```
ls.general()
```

`ls.infotheo` *List the information theoretical meta-features*

Description

List the information theoretical meta-features

Usage

```
ls.infotheo()
```

Value

A list of information theoretical meta-features names

Examples

```
ls.infotheo()
```

`ls.itemset` *List the itemset meta-features*

Description

List the itemset meta-features

Usage

```
ls.itemset()
```

Value

A list of itemset meta-features names.

Examples

```
ls.itemset()
```

`ls.landmarking` *List the Landmarking meta-features*

Description

List the Landmarking meta-features

Usage

```
ls.landmarking()
```

Value

A list of Landmarking meta-features names.

Examples

```
ls.landmarking()
```

`ls.metafeatures` *List the meta-features groups*

Description

List the meta-features groups

Usage

```
ls.metafeatures()
```

Value

A list of meta-features groups

Examples

```
ls.metafeatures()
```

`ls.model.based` *List the DT model based meta-features*

Description

List the DT model based meta-features

Usage

```
ls.model.based()
```

Value

A list of DT model based meta-features names

Examples

```
ls.model.based()
```

`ls.relative` *List the relative meta-features*

Description

List the relative meta-features

Usage

```
ls.relative()
```

Value

A list of relative meta-features names.

Examples

```
ls.relative()
```

`ls.statistical` *List the statistical meta-features*

Description

List the statistical meta-features

Usage

```
ls.statistical()
```

Value

A list of statistical meta-features names

Examples

```
ls.statistical()
```

`metafeatures` *Extract meta-features from a dataset*

Description

This is a simple way to extract the meta-features from a dataset, where all meta-features from each group is extracted.

Usage

```
metafeatures(...)

## Default S3 method:
metafeatures(x, y, groups = "default", summary = c("mean", "sd"), ...)

## S3 method for class 'formula'
metafeatures(formula, data, groups = "default", summary = c("mean", "sd"), ...)
```

Arguments

`...`	Optional arguments to the summary methods.
`x`	A data.frame contained only the input attributes.
`y`	A factor response vector with one label for each row/component of x.
`groups`	A list of meta-features groups, `"default"` for traditional groups of meta-features or `"all"` to include all them. The details section describes the valid values for this parameter.

`summary`	A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)
`formula`	A formula to define the class column.
`data`	A data.frame dataset contained the input attributes and class The details section describes the valid values for this group.

Details

The following groups are allowed for this method:

"infotheo" Include all information theoretical meta-features. See infotheo for more details.

"general" Include all general (simple) meta-features. See general for more details.

"landmarking" Include all landmarking meta-features. See landmarking for more details.

"model.based" Include all model based meta-features. See model.based for more details.

"statistical" Include all statistical meta-features. See statistical for more details.

"clustering" Include all clustering meta-features. See clustering for more details.

"complexity" Include all complexity meta-features. See complexity for more details.

"concept" Include all concept variation meta-features. See concept for more details.

"itemset" Include all itemset meta-features. See itemset for more details.

Value

A numeric vector named by the meta-features from the specified groups.

Examples

```
## Extract all meta-features
metafeatures(Species ~ ., iris)

## Extract some groups of meta-features
metafeatures(iris[1:4], iris[5], c("general", "statistical", "infotheo"))

## Use another summary methods
metafeatures(Species ~ ., iris, summary=c("min", "median", "max"))
```

`model.based` *Decision Tree Model Based Meta-features*

Description

Decision Tree (DT) Model Based meta-features are the measures designed to extract characteristics of a DT model induced from a dataset.

Usage

```
model.based(...)

## Default S3 method:
model.based(x, y, features = "all", summary = c("mean", "sd"), ...)

## S3 method for class 'formula'
model.based(formula, data, features = "all", summary = c("mean", "sd"), ...)
```

Arguments

`...` Further arguments passed to the summarization functions.

`x` A data.frame contained only the input attributes.

`y` A factor response vector with one label for each row/component of x.

`features` A list of features names or `"all"` to include all them.

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class. The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"leaves" Number of leaves of the DT model.

"leavesBranch" Size of branches, which consists in the level of all leaves of the DT model (multi-valued).

"leavesCorrob" Leaves corroboration, which is the proportion of examples that belong to each leaf of the DT model (multi-valued).

"leavesHomo" Homogeneity, which is the number of leaves divided by the structural shape of the DT model (multi-valued).

"leavesPerClass" Leaves per class, which is the proportion of leaves of the DT model associated with each class (multi-valued).

"nodes" Number of nodes of the DT model.

"nodesPerAttr" Ratio of the number of nodes of the DT model per the number of attributes.

"nodesPerInst" Ratio of the number of nodes of the DT model per the number of instances.

"nodesPerLevel" Number of nodes of the DT model per level (multi-valued).

"nodesRepeated" Repeated nodes, which is the number of repeated attributes that appear in the DT model (multi-valued).

"treeDepth" Tree depth, which is the level of all tree nodes and leaves of the DT model (multi-valued).

"treeImbalance" Tree imbalance (multi-valued).

- **"treeShape"** Tree shape, which is the probability of arrive in each leaf given a random walk. We call this as the structural shape of the DT model (multi-valued).
- **"varImportance"** Variable importance. It is calculated using the Gini index to estimate the amount of information used in the DT model (multi-valued).

Value

A list named by the requested meta-features.

References

Hilan Bensusan, Christophe Giraud-Carrier, and Claire Kennedy. A higher-order approach to meta-learning. In 10th International Conference Inductive Logic Programming (ILP), pages 33 - 42, 2000.

Yonghong Peng, Peter A. Flach, Carlos Soares, and Pavel Brazdil. Improved dataset characterization for meta-learning. In 5th International Conference on Discovery Science (DS), pages 141 - 152, 2002.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `general()`, `infotheo()`, `itemset()`, `landmarking()`, `relative()`, `statistical()`

Examples

```
## Extract all meta-features using formula
model.based(Species ~ ., iris)

## Extract some meta-features
model.based(iris[1:4], iris[5], c("nodes", "leaves", "treeShape"))

## Use another summarization function
model.based(Species ~ ., iris, summary=c("min", "median", "max"))
```

`post.processing` *Post processing meta-features values*

Description

Post-processing alternatives to deal with multiples meta-features values. This method is used by the meta-features characterization methods to summarize the obtained values.

Usage

```
post.processing(measure, summary = c("mean", "sd"), multiple = TRUE, ...)
```

Arguments

`measure`	A list with the meta-features values.
`summary`	The functions to post processing the data. See the details to more information. Default: `c("mean","sd")`
`multiple`	A logical value defining if the measure should return multiple values. (Default: `TRUE`)
`...`	Extra values used to the functions of summarization.

Details

The post processing functions are used to summarize the meta-features. They are organized into three groups: non-aggregated, descriptive statistic and distribution. Currently, the hypothesis testing post processing are not supported.

In practice, there are no difference among the types, so that more than one type and functions can be combined. Usually, these function are used to summarize a set of values for each meta-features. For instance, a measure computed for each attribute can be summarized using the `"mean"` and/or `"sd"`. Necessarily, a single value always use the `"non.aggregated"` function.

In addition to the native functions available in R, the following functions can be used:

"histogram" Computes a histogram of the given data value. The extra parameters '`bins`' can be used to define the number of values to be returned. The parameters '`max`' and '`min`' are used to define the range of the data. The default value for these parameters are respectively `10,min(x)` and `max(x)`.

"kurtosis" See `kurtosis`

"max" See `max`

"mean" See `mean`

"median" See `median`

"min" See `min`

"quantiles" See `quantile`

"sd" See `sd`

"skewness" See `skewness`

"var" See `var`

"non.aggregated" Returns the original value(s) of the meta-feature.

These functions are not restrictive, thus another functions can be applied as post-processing summarization function.

Value

A list with the post-processed meta-features

References

Fabio Pinto, Carlos Soares, and Joao Mendes-Moreira. Towards Automatic Generation of Metafeatures. In 20th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pages 215 - 226, 2016.

Examples

```
post.processing(runif(15))
post.processing(runif(15), c("min", "max"))
post.processing(runif(15), c("quantiles", "skewness"))
post.processing(runif(15), "histogram", bins=5, min=0, max=1)
```

relative *Relative Landmarking Meta-features*

Description

Relative Landmarking measures are landmarking measures using ranking strategy.

Usage

```
relative(...)

## Default S3 method:
relative(
  x,
  y,
  features = "all",
  summary = c("mean", "sd"),
  size = 1,
  folds = 10,
  score = "accuracy",
  ...
)

## S3 method for class 'formula'
relative(
  formula,
  data,
  features = "all",
  summary = c("mean", "sd"),
  size = 1,
  folds = 10,
  score = "accuracy",
  ...
)
```

Arguments

...	Further arguments passed to the summarization functions.
x	A data.frame contained only the input attributes.
y	A factor response vector with one label for each row/component of x.

`features` A list of features names or `"all"` to include all them.

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`size` The percentage of examples subsampled. Values different from 1 generate the subsampling-based relative landmarking metafeatures. (Default: 1.0)

`folds` The number of k equal size subsamples in k-fold cross-validation.(Default: 10)

`score` The evaluation measure used to score the classification performance. `c("accuracy","balanced.accura`
(Default: `"accuracy"`).

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class. The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"bestNode" Construct a single decision tree node model induced by the most informative attribute to establish the linear separability (multi-valued).

"eliteNN" Elite nearest neighbor uses the most informative attribute in the dataset to induce the 1-nearest neighbor. With the subset of informative attributes is expected that the models should be noise tolerant (multi-valued).

"linearDiscr" Apply the Linear Discriminant classifier to construct a linear split (non parallel axis) in the data to establish the linear separability (multi-valued).

"naiveBayes" Evaluate the performance of the Naive Bayes classifier. It assumes that the attributes are independent and each example belongs to a certain class based on the Bayes probability (multi-valued).

"oneNN" Evaluate the performance of the 1-nearest neighbor classifier. It uses the euclidean distance of the nearest neighbor to determine how noisy is the data (multi-valued).

"randomNode" Construct a single decision tree node model induced by a random attribute. The combination with `"bestNode"` measure can establish the linear separability (multi-valued).

"worstNode" Construct a single decision tree node model induced by the worst informative attribute. The combination with `"bestNode"` measure can establish the linear separability (multi-valued).

Value

A list named by the requested meta-features.

References

Johannes Furnkranz, Johann Petrak, Pavel Brazdil, and Carlos Soares. On the use of Fast Subsampling Estimates for Algorithm Recommendation. Technical Report, pages 1-9, 2002.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `general()`, `infotheo()`, `itemset()`, `landmarking()`, `model.based()`, `statistical()`

Examples

```
## Extract all meta-features using formula
relative(Species ~ ., iris)

## Extract some meta-features
relative(iris[1:4], iris[5], c("bestNode", "randomNode", "worstNode"))

## Use another summarization function
relative(Species ~ ., iris, summary=c("min", "median", "max"))

## Use 2 folds and balanced accuracy
relative(Species ~ ., iris, folds=2, score="balanced.accuracy")

## Extract the subsapling relative landmarking
relative(Species ~ ., iris, size=0.7)
```

statistical *Statistical meta-features*

Description

Statistical meta-features are the standard statistical measures to describe the numerical properties of a distribution of data. As it requires only numerical attributes, the categorical data are transformed to numerical.

Usage

```
statistical(...)

## Default S3 method:
statistical(
  x,
  y,
  features = "all",
  summary = c("mean", "sd"),
  by.class = FALSE,
  transform = TRUE,
  ...
)

## S3 method for class 'formula'
statistical(
  formula,
  data,
  features = "all",
  summary = c("mean", "sd"),
  by.class = FALSE,
  transform = TRUE,
  ...
)
```

Arguments

`...` Further arguments passed to the summarization functions.

`x` A data.frame contained only the input attributes.

`y` A factor response vector with one label for each row/component of x.

`features` A list of features names or `"all"` to include all them. The details section describes the valid values for this group.

`summary` A list of summarization functions or empty for all values. See post.processing method to more information. (Default: `c("mean","sd")`)

`by.class` A logical value indicating if the meta-features must be computed for each group of samples belonging to different output classes. (Default: FALSE)

`transform` A logical value indicating if the categorical attributes should be transformed. If `FALSE` they will be ignored. (Default: `TRUE`)

`formula` A formula to define the class column.

`data` A data.frame dataset contained the input attributes and class The details section describes the valid values for this group.

Details

The following features are allowed for this method:

"canCor" Canonical correlations between the predictive attributes and the class (multi-valued).

"gravity" Center of gravity, which is the distance between the instance in the center of the majority class and the instance-center of the minority class.

"cor" Absolute attributes correlation, which measure the correlation between each pair of the numeric attributes in the dataset (multi-valued). This measure accepts an extra argument called `method = c("pearson","kendall","spearman")`. See `cor` for more details.

"cov" Absolute attributes covariance, which measure the covariance between each pair of the numeric attributes in the dataset (multi-valued).

"nrDisc" Number of the discriminant functions.

"eigenvalues" Eigenvalues of the covariance matrix (multi-valued).

"gMean" Geometric mean of attributes (multi-valued).

"hMean" Harmonic mean of attributes (multi-valued).

"iqRange" Interquartile range of attributes (multi-valued).

"kurtosis" Kurtosis of attributes (multi-valued).

"mad" Median absolute deviation of attributes (multi-valued).

"max" Maximum value of attributes (multi-valued).

"mean" Mean value of attributes (multi-valued).

"median" Median value of attributes (multi-valued).

"min" Minimum value of attributes (multi-valued).

"nrCorAttr" Number of attributes pairs with high correlation (multi-valued when `by.class=TRUE`).

"nrNorm" Number of attributes with normal distribution. The Shapiro-Wilk Normality Test is used to assess if an attribute is or not is normally distributed (multi-valued only when `by.class=TRUE`).

"nrOutliers" Number of attributes with outliers values. The Turkey's boxplot algorithm is used to compute if an attributes has or does not have outliers (multi-valued only when `by.class=TRUE`).

"range" Range of Attributes (multi-valued).

"sd" Standard deviation of the attributes (multi-valued).

"sdRatio" Statistic test for homogeneity of covariances.

"skewness" Skewness of attributes (multi-valued).

"sparsity" Attributes sparsity, which represents the degree of discreetness of each attribute in the dataset (multi-valued).

"tMean" Trimmed mean of attributes (multi-valued). It is the arithmetic mean excluding the 20% of the lowest and highest instances.

"var" Attributes variance (multi-valued).

"wLambda" Wilks Lambda.

This method uses simple binarization to transform the categorical attributes when `transform=TRUE`.

Value

A list named by the requested meta-features.

References

Ciro Castiello, Giovanna Castellano, and Anna M. Fanelli. Meta-data: Characterization of input features for meta-learning. In 2nd International Conference on Modeling Decisions for Artificial Intelligence (MDAI), pages 457 - 468, 2005.

Shawkat Ali, and Kate A. Smith. On learning algorithm selection for classification. Applied Soft Computing, volume 6, pages 119 - 138, 2006.

See Also

Other meta-features: `clustering()`, `complexity()`, `concept()`, `general()`, `infotheo()`, `itemset()`, `landmarking()`, `model.based()`, `relative()`

Examples

```
## Extract all meta-features
statistical(Species ~ ., iris)

## Extract some meta-features
statistical(iris[1:4], iris[5], c("cor", "nrNorm"))

## Extract all meta-features without summarize the results
statistical(Species ~ ., iris, summary=c())
```

```
## Use another summarization function
statistical(Species ~ ., iris, summary=c("min", "median", "max"))

## Extract statistical measures using by.class approach
statistical(Species ~ ., iris, by.class=TRUE)

## Do not transform the data (using only categorical attributes)
statistical(Species ~ ., iris, transform=FALSE)
```

Index