Package 'mfe'

January 31, 2017

Type Package

Title Meta-Feature Extractor

Version 0.1.0

Date 2017-01-27

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, this package allows a comprehensive data characterization, a deep data exploration and a large number of Meta-Learning based data analysis. These concepts are described in the book: Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R. (2009) <doi:10.1007/978-3-540-73263-1>.

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

Depends R (>= 3.3.1),

Imports C50, class, e1071, infotheo, MASS, rpart, stats, utils

Suggests knitr, rmarkdown, testthat

License GPL | file LICENSE

LazyData true

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

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2017-01-31 14:16:53

R topics documented:

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

Description

List the discriminant meta-features

Usage

```
ls.discriminant()
```

Value

A list of discriminant meta-features names

Examples

```
ls.discriminant()
```

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.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.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. To set specific parameters for each group, use the specific characterization method.

Usage

```
metafeatures(...)

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

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

Arguments

`...`	Optional arguments to the summary methods. #' @param formula A formula to define the class column.
`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 or `"all"` to include all them. The details section describes the valid values for this parameter.
`summary`	A list of methods to summarize the results as post-processing functions. 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:

"discriminant" Include all discriminant meta-features. See mf.discriminant for more details.

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

"general" Include all general meta-features. See mf.general for more details.

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

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

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

Value

A numeric vector named by the requested meta-features.

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"))
```

---

mf.discriminant *Discriminant meta-features*

Description

Discriminant meta-features are computed using the discriminant analysis. It is computed using just the numerical attributes.

Usage

```
mf.discriminant(...)

## Default S3 method:
mf.discriminant(x, y, features = "all", ...)

## S3 method for class 'formula'
mf.discriminant(formula, data, features = "all", ...)
```

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.
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" Represents the first canonical discriminant correlations of the numeric attributes in the dataset.

"center.of.gravity" Represents the distance between minority and majority classes in the dataset.

"discfct" Represents the number of discriminant functions normalized by the number of classes"

"cancor.fract" Represents a measure of collinearity of the class means. It is computed using the squares of the canonical correlations instead of the eigenvalues.

"eigen.fract" Represents the relative importance of the largest eigenvalue of the attribute covariance matrix computed from the numeric attributes in the dataset.

"sdratio" Represents the test statistic for homogeneity of covariances. It uses the Box's M test and it is is strictly greater than unity if the covariances differ, and is equal to unity if and only if the M-statistic is zero.

The categorical attributes is replaced by binaries attributes.

Value

A list named by the requested meta-features.

References

Michie, E. D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine Learning , Neural and Statistical Classification. Technometrics, 37(4), 459.

Castiello, C., Castellano, G., & Fanelli, A. M. (2005). Meta-data: Characterization of Input Features for Meta-learning. In Proceedings of the 2nd International Conference on Modeling Decisions for Artificial Intelligence (Vol. 3558, pp. 457-468).

Lindner, G., & Studer, R. (1999). AST: Support for Algorithm Selection with a CBR Approach. Principles of Data Mining and Knowledge Discovery, 1704, 418-423.

Ali, S., & Smith, K. a. (2006). On learning algorithm selection for classification. Applied Soft Computing, 6(2), 119-138.

See Also

Other meta.features: `mf.general`, `mf.infotheo`, `mf.landmarking`, `mf.model.based`, `mf.statistical`

Examples

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

## Extract some metafeatures
mf.discriminant(iris[1:4], iris[5], c("cancor", "cancor.fract"))
```

`mf.general` *General meta-features*

Description

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

Usage

```
mf.general(...)

## Default S3 method:
mf.general(x, y, features = "all", ...)

## S3 method for class 'formula'
mf.general(formula, data, features = "all", ...)
```

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.
`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:

"defective.instances" Represents the proportion of instances with missing values in the dataset.

"dimensionality" Represents the ratio between the number of attributes and the number of instances constituting the dataset.

"majority.class" Represents the proportion of instances that belongs to the majority class. It is also known as default accuracy.

"missing.values" Represents the proportion of missing values in the dataset.

"nattribute" Represents the total number of attributes in the dataset.

"nbinary" Represents the total of binary attributes in the dataset. It includes categorical and numeric values that have just 2 different values.

"nclasse" Represents the total number of output values (classes) in the dataset.

"ninstance" Represents the total number of instances (also named samples or observations) in the dataset.

"nnumeric" Represents the number of numeric attributes in the dataset.

"nsymbolic" Represents the number of categorical attributes in the dataset. This is the same as the number of factor columns.

"pbinary" Represents the proportion of binary attributes in the dataset.

"pnumeric" Represents the proportion of numeric attributes in the dataset.

"psymbolic" Represents the proportion of symbolic attributes in the dataset.

"sdclass" Represents the standard deviation of the class distribution in the dataset.

Value

A list named by the requested meta-features.

References

Michie, E. D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine Learning , Neural and Statistical Classification. Technometrics, 37(4), 459.

Lindner, G., & Studer, R. (1999). AST: Support for Algorithm Selection with a CBR Approach. Principles of Data Mining and Knowledge Discovery, 1704, 418-423.

Castiello, C., Castellano, G., & Fanelli, A. M. (2005). Meta-data: Characterization of Input Features for Meta-learning. In Proceedings of the 2nd International Conference on Modeling Decisions for Artificial Intelligence (Vol. 3558, pp. 457-468).

See Also

Other meta.features: `mf.discriminant`, `mf.infotheo`, `mf.landmarking`, `mf.model.based`, `mf.statistical`

Examples

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

## Extract some metafeatures
small.iris <- iris[1:100, ]
mf.general(small.iris[1:4], small.iris[5], c("nclasse", "dimensionality"))
```

`mf.infotheo` *Information-theoretic meta-features*

Description

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

Usage

```
mf.infotheo(...)

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

## S3 method for class 'formula'
mf.infotheo(formula, data, features = "all",
  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.
features	A list of features names or "all" to include all them.
summary	A list of methods to summarize the results as post-processing functions. 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

TODO describe discretization method The following features are allowed for this method:

"attributes.concentration" Represents the association between the nominal attributes. It is the Goodman and Kruskal's tau measure otherwise known as the concentration coefficient.

"attribute.entropy" Represents the normalized entropy (a measure of randomness) of each attributes in the dataset.

"class.concentration" Represents the association between the nominal attributes and the class. It is the Goodman and Kruskal's tau measure otherwise known as the concentration coefficient.

"class.entropy" Represents the normalized entropy of the class that describes how much information is necessary to specify one class in the dataset.

"equivalent.attributes" Represents the the number of attributes suitable to optimally solve the classification task using the dataset.

"joint.entropy" Represents the total entropy of each attribute and the classe in the dataset.

"mutual.information" Represents the common information shared between each attribute and the class in the dataset.

"noise.signal" Represents the amount of irrelevant information contained in the dataset.

Each one of these meta-features generate multiple values (by attribute and/or class value) and then it is post processed by the summary methods. See the post.processing method for more details about it.

Value

A list named by the requested meta-features.

References

Michie, E. D., Spiegelhalter, D. J., & Taylor, C. C. (1994). Machine Learning , Neural and Statistical Classification. Technometrics, 37(4), 459.

Kalousis, A., & Hilario, M. (2001). MODEL SELECTION VIA META-LEARNING: A COMPARATIVE STUDY. International Journal on Artificial Intelligence Tools, 10(4), 525-554.

Castiello, C., Castellano, G., & Fanelli, A. M. (2005). Meta-data: Characterization of Input Features for Meta-learning. In Proceedings of the 2nd International Conference on Modeling Decisions for Artificial Intelligence (Vol. 3558, pp. 457-468).

See Also

Other meta.features: `mf.discriminant`, `mf.general`, `mf.landmarking`, `mf.model.based`, `mf.statistical`

Examples

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

## Extract some metafeatures
mf.infotheo(iris[1:4], iris[5], c("class.entropy", "joint.entropy"))

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

`mf.landmarking` *Landmarking Meta-features*

Description

Landmarking meta-features include the performance of some simple and efficient learning algorithms.

Usage

```
mf.landmarking(...)

## Default S3 method:
mf.landmarking(x, y, features = "all", summary = c("mean",
  "sd"), map = c("one.vs.all", "one.vs.one"), folds = 10, ...)

## S3 method for class 'formula'
mf.landmarking(formula, data, features = "all",
  summary = c("mean", "sd"), map = c("one.vs.all", "one.vs.one"),
  folds = 10, ...)
```

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.
features	A list of features names or "all" to include all them.
summary	A list of methods to summarize the results as post-processing functions. See post.processing method to more information. (Default: c("mean", "sd"))
map	A list of decomposition strategies for multi-class problems. The options are "one.vs.all" and "one.vs.one" strategy.
folds	The number of k equal size subsamples in k-fold cross-validation.
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:

"decision.stumps" Represents the performance of a single node DT model induced by the most informative attribute.

"elite.nearest.neighbor" Represents the performance of the Elite 1-Nearest Neighbor classifier.

"linear.discriminant" Represents the performance of the Linear Discriminant classifier.

"naive.bayes" Represents the performance of the Naive Bayes classifier.

"nearest.neighbor" Represents the performance of the 1-Nearest Neighbor classifier.

"worst.node" Represents the performance of a single node DT model induced by the less informative attribute.

Value

Each one of these meta-features generate multiple values (by fold and/or attribute) and then it is post processed by the summary methods. See the post.processing method for more details about it.

References

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. G. (2000). Meta-Learning by Landmarking Various Learning Algorithms. In Proceedings of the 17th International Conference on Machine Learning (pp. 743-750)

See Also

Other meta.features: mf.discriminant, mf.general, mf.infotheo, mf.model.based, mf.statistical

Examples

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

## Extract all meta-features using data.frame
mf.landmarking(iris[1:4], iris[5])

## Extract some meta-features
mf.landmarking(Species ~ ., iris, features=c("decision.stumps",
"nearest.neighbor", "linear.discriminant"))

## Extract all meta-features with different summary methods
mf.landmarking(Species ~ ., iris, summary=c("min", "median", "max"))
```

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

Description

Decision Tree (DT) Model Based meta-features are the measures desined to extract characteristics like the depth, the shape and size of a DT model induced from a dataset.

Usage

```
mf.model.based(...)

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

## S3 method for class 'formula'
mf.model.based(formula, data, features = "all",
  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`.
`features`	A list of features names or `"all"` to include all them.
`summary`	A list of methods to summarize the results as post-processing functions. 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:

"average.leaf.corrobation" Represents the.number of examples that belong to each leaf in the DT model divided by the number of examples in the dataset.

"branch.length" Represents the.size of each leaf in the DT model.

"depth" Represents the size of each path in the DT model.

"homogeneity" Represents the number of leaves divided by the strutural shape of the DT model.

"max.depth" Represents the size of the longest path of the DT model.

"nleave" Represents the number of leaves of the DT model.

"nnode" Represents the number of nodes in the DT model.

"nodes.per.attribute" Represents the number of nodes in the DT model divided by the number of predictive attributes.

"nodes.per.instance" Represents the number of leaves in the DT model divided by the number of examples in the dataset.

"nodes.level" Represents the number of nodes per level.

"repeated.nodes" Represents the number of repeated attributes that appear in the DT model.

"shape" Represents the probability of arrive in each leaf given a random walk. We call this as the strutural shape of the DT model.

"variable.importance" Represents the variable importance calculated by Gini index to construct the DT model.

Value

Each one of these meta-features generate multiple values (by leaves and/or nodes) and then it is post processed by the summary methods. See the post.processing method for more details about it.

References

Bensusan, H., Giraud-Carrier, C. G., & Kennedy, C. J. (2000). A Higher-order Approach to Meta-learning. In Proceedings of the 10th International Conference on Inductive Logic Programming (Vol. 35, pp. 1-10).

Peng, Y., Flach, P. A., Soares, C., & Brazdil, P. (2002). Improved Dataset Characterisation for Meta-learning. In Proceedings of the 5th International Conference on Discovery Science (Vol 2534, pp. 141-152)

See Also

Other meta.features: `mf.discriminant`, `mf.general`, `mf.infotheo`, `mf.landmarking`, `mf.statistical`

Examples

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

## Extract all meta-features using data.frame
mf.model.based(iris[1:4], iris[5])

## Extract some meta-features
mf.model.based(Species ~ ., iris, features=c("nnode", "nleave", "depth"))

## Extract all meta-features with different summary methods
mf.model.based(Species ~ ., iris, summary=c("min", "median", "max"))
```

`mf.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

```
mf.statistical(...)

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

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

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`.
`features`	A list of features names or `"all"` to include all them. The details section describes the valid values for this parameter.
`summary`	A list of methods to summarize the results as post-processing functions. 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: TRUE)
`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:

"correlation" Represents the absolute correlation between each pair of the numeric attributes in the dataset.

"covariance" Represents the absolute covariance between each pair of the numeric attributes in the dataset.

"discreteness.degree" Represents the degree of discreetness of each attribute in the dataset. It is measured using a sparsity measure.

"geometric.mean" Represents the geometric mean of the numeric attributes in the dataset.

"harmonic.mean" Represents the harmonic mean of the numeric attributes in the dataset.

"iqr" Represents the interquartile range divided by the standard deviation of the numeric attributes in the dataset.

"kurtosis" Represents the kurtosis of the numeric attributes in the dataset.

"mad" Represents the median absolute deviation of the numeric attributes in the dataset.

"normality" Represents the number of attributes that have a normal distribution of values. For that it is used the Shapiro-Wilk Normality Test with `alpha=0.05`.

"outliers" "Represents the proportion of numeric attributes with outliers."

"skewness" Represents the skewness of the numeric attributes in the dataset.

"standard.deviation" Represents the standard deviation of the numeric attributes in the dataset.

"trim.mean" Represents the trim mean of the numeric attributes in the dataset. It is the aritimetic mean excluding the 20 and hieghest instances.

"variance" Represents the variance (normalization of the standard deviation) of the numeric attributes in the dataset.

Each one of these meta-features generate multiple values (by attribute and/or class value) and then it is post processed by the summary methods. See the post.processing method for more details about it.

The categorical attributes is replaced by binaries attributes.

Value

A list named by the requested meta-features.

References

Castiello, C., Castellano, G., & Fanelli, A. M. (2005). Meta-data: Characterization of Input Features for Meta-learning. In Proceedings of the 2nd International Conference on Modeling Decisions for Artificial Intelligence (Vol. 3558, pp. 457-468).

Ali, S., & Smith, K. a. (2006). On learning algorithm selection for classification. Applied Soft Computing, 6(2), 119-138.

See Also

Other meta.features: `mf.discriminant`, `mf.general`, `mf.infotheo`, `mf.landmarking`, `mf.model.based`

Examples

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

## Extract some meta-features
mf.statistical(iris[1:4], iris[5], c("correlation", "variance"))

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

## Compute the mean for each measure without consider the class information
mf.statistical(Species ~ ., iris, summary="mean", by.class=FALSE)
```

---

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"), ...)
```

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")`
...	Extra values used to the functions.

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"`. Mandatorily, a single value always use the `"non.aggregated"` function.

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

"hist" 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

"mode" Returns the most common value of the distribution. If more than one value are the most common return the first. It does not work as expected for real numbers.

"quantile" 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

Pinto, F., Soares, C., & Mendes-Moreira, J. (2016). Towards Automatic Generation of Metafeatures. In 20th Pacific-Asia Conference (pp. 215-226).

Examples

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

Index