Package 'mldr.resampling'

August 22, 2023

Title Resampling Algorithms for Multi-Label Datasets

Version 0.2.3

Description Collection of the state of the art multi-label resampling algorithms. The objective of these algorithms is to achieve balance in multi-label datasets.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports data.table, e1071, mldr, pbapply, vecsets

Suggests parallel

NeedsCompilation no

Author Miguel Ángel Dávila [cre],
Francisco Charte [aut] (<https://orcid.org/0000-0002-3083-8942>),
María José Del Jesus [aut] (<https://orcid.org/0000-0002-7891-3059>),
Antonio Rivera [aut] (<https://orcid.org/0000-0002-1062-3127>)

Maintainer Miguel Ángel Dávila <madr0008@red.ujaen.es>

Repository CRAN

Date/Publication 2023-08-22 12:20:02 UTC

R topics documented:

`adjustedHammingDist` *Auxiliary function used by MLeNN. Computes the Hamming Distance between two instances*

Description

Auxiliary function used by MLeNN. Computes the Hamming Distance between two instances

Usage

```
adjustedHammingDist(x, y, D)
```

Arguments

`x`	Index of sample 1
`y`	Index of sample 2
`D`	mld `mldr` object in which the instances are located

Value

The Hamming Distance between the instances

`calculateDistances` *Auxiliary function used to calculate the distances between an instance and the ones with a specific active label. Euclidean distance is calculated for numeric attributes, and VDM for non numeric ones.*

Description

Auxiliary function used to calculate the distances between an instance and the ones with a specific active label. Euclidean distance is calculated for numeric attributes, and VDM for non numeric ones.

Usage

```
calculateDistances(sample, rest, label, D, tableVDM = NULL)
```

Arguments

`sample`	Index of the sample whose distances to other samples we want to know
`rest`	Indexes of the samples to which we will calculate the distance
`label`	Label that must be active
`D`	mld `mldr` object with the multilabel dataset to preprocess
`tableVDM`	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A list with the distance to the rest of samples

`calculateTableVDM` *Auxiliary function used to calculate an auxiliary table to make VDM calculation faster*

Description

Auxiliary function used to calculate an auxiliary table to make VDM calculation faster

Usage

```
calculateTableVDM(D)
```

Arguments

`D`	mld `mldr` object with the multilabel dataset to preprocess

Value

A dataframe with tables, useful for VDM calculation

executeAlgorithm *Auxiliary function used by resample. It executes an algorithm, given as a string, and stores the resulting MLD in a arff file*

Description

Auxiliary function used by resample. It executes an algorithm, given as a string, and stores the resulting MLD in a arff file

Usage

```
executeAlgorithm(
  D,
  a,
  P,
  k,
  TH,
  strategy,
  outputDirectory,
  neighbors,
  neighbors2,
  tableVDM
)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
a	String with the name of the algorithm to be applied.
P	Percentage in which the original dataset is increased/decreased (if required by the algorithm)
k	Number of neighbors taken into account for each instance (if required by the algorithm)
TH	Threshold for the Hamming Distance in order to consider an instance different to another one (if required by the algorithm)
strategy	Strategy for choosing the synthetic labels (if required by the algorithm). Possible values: "union", "intersection" and "ranking" (default)
outputDirectory	Route with the directory where the generated ARFF file will be stored
neighbors	Structure with all instances and neighbors in the dataset, useful in MLSOL and MLUL
neighbors2	Structure with some instances and neighbors in the dataset, useful in MLeNN and MLTL
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

Time (in seconds) taken to execute the algorithm (NULL if no algorithm was executed)

`generateInstanceMLSOL` *Auxiliary function used by MLSOL. Creates a synthetic sample based on two other samples, taking into account their types*

Description

Auxiliary function used by MLSOL. Creates a synthetic sample based on two other samples, taking into account their types

Usage

```
generateInstanceMLSOL(seedInstance, refNeigh, t, D)
```

Arguments

`seedInstance`	Index of the sample we are using as "template"
`refNeigh`	Index of the reference neighbor
`t`	types of the instances
`D`	mld `mldr` object with the multilabel dataset to preprocess

Value

A synthetic sample derived from the one passed as a parameter and its neighbors

---

`getAllNeighbors` *Auxiliary function used by MLSOL and MLUL. Computes the kNN of every instance in a dataset*

Description

Auxiliary function used by MLSOL and MLUL. Computes the kNN of every instance in a dataset

Usage

```
getAllNeighbors(D, d, tableVDM = NULL)
```

Arguments

`D`	mld `mldr` object with the multilabel dataset to preprocess
`d`	Vector with the instances of the dataset which have one or more label active (ideally, all of them)
`tableVDM`	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A list of vectors with the indexes of the neighbors for each instance

---

`getAllNeighbors2` *Auxiliary function used by MLeNN and MLTL. Gets the kNN of every instance in a dataset, when compared to some of the rest*

Description

Auxiliary function used by MLeNN and MLTL. Gets the kNN of every instance in a dataset, when compared to some of the rest

Usage

```
getAllNeighbors2(neighbors, d, k)
```

Arguments

`neighbors`	Structure with all the neighbors in the dataset, regardless of which ones to be compared
`d`	Vector with the instances of the dataset which are going to be compared
`k`	Number of neighbors to be retrieved

Value

A list of vectors with the indexes of the neighbors for each instance

---

`getAllReverseNeighbors` *Auxiliary function used by MLUL. For each instance in the dataset, given the neighbors structure, we compute its reverse nearest neighbors*

Description

Auxiliary function used by MLUL. For each instance in the dataset, given the neighbors structure, we compute its reverse nearest neighbors

Usage

```
getAllReverseNeighbors(d, neighbors, k)
```

Arguments

d	Vector with the instances of the dataset which have one or more label active (ideally, all of them)
neighbors	Structure with the neighbors of every instance in the dataset
k	Number of neighbors to be considered

Value

A list of vectors with the indexes of the reverse nearest neighbors of every instance in the dataset

---

getC *Auxiliary function used by MLSOL and MLUL. For each instance in the dataset, we compute, for each label, the proportion of neighbors having an opposite class with respect to the proper instance*

Description

Auxiliary function used by MLSOL and MLUL. For each instance in the dataset, we compute, for each label, the proportion of neighbors having an opposite class with respect to the proper instance

Usage

```
getC(D, d, neighbors, k)
```

Arguments

D	mld `mldr` object with the multilabel dataset to preprocess
d	Vector with the instances of the dataset which have one or more label active (ideally, all of them)
neighbors	Structure with the neighbors of every instance in the dataset
k	Number of neighbors taken into account for each instance

Value

A structure with the proportion of neighbors having an opposite class with respect to an instance and label

getNN *Auxiliary function used to compute the neighbors of an instance*

Description

Auxiliary function used to compute the neighbors of an instance

Usage

```
getNN(sample, rest, label, D, tableVDM = NULL)
```

Arguments

sample	Index of the sample whose neighbors we want to know
rest	Indexes of the samples among which we will search
label	Label that must be active, in order to calculate the distances
D	mld mldr object with the multilabel dataset to preprocess
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A vector with the indexes inside rest of the neighbors

getNumCores *Get the number of cores available for parallel computing*

Description

Get the number of cores available for parallel computing

Usage

```
getNumCores()
```

Value

The number of cores available for parallel computing

Examples

```
getNumCores()
```

getS *Auxiliary function used by MLSOL and MLUL. For non outlier instances, it aggregates the values of C, taking into account the global class imbalance*

Description

Auxiliary function used by MLSOL and MLUL. For non outlier instances, it aggregates the values of C, taking into account the global class imbalance

Usage

```
getS(D, d, C, minoritary)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
d	Vector with the instances of the dataset which have one or more label active (ideally, all of them)
C	Structure with the proportion of neighbors having an opposite class with respect to an instance and label
minoritary	Vector with the minoritary class of each label (normally, 1)

Value

A structure with the proportion of neighbors having an opposite class with respect to an instance and label, normalized by the global class imbalance

getU *Auxiliary function used by MLUL. It computes the influence of each instance with respect to its reverse neighbors*

Description

Auxiliary function used by MLUL. It computes the influence of each instance with respect to its reverse neighbors

Usage

```
getU(D, d, rNeighbors, S)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
d	Vector with the instances of the dataset which have one or more label active (ideally, all of them)
rNeighbors	Structure with the reverse nearest neighbors of each instance of the dataset
S	Structure with the proportion of neighbors having an opposite class with respect to an instance and label, normalized by the global class imbalance

Value

A list of values of influence for each instance with respect to its reverse neighbors

---

getV *Auxiliary function used by MLUL. It calculates, for each instance, how important it is in the dataset*

Description

Auxiliary function used by MLUL. It calculates, for each instance, how important it is in the dataset

Usage

```
getV(w, u)
```

Arguments

w	List of weights for each instance
u	List of influences in reverse neighbors for each instance

Value

A list with the values of importance of each instance in the dataset

getW *Auxiliary function used by MLSOL and MLUL. For non outlier instances, it aggregates the values of S for each label*

Description

Auxiliary function used by MLSOL and MLUL. For non outlier instances, it aggregates the values of S for each label

Usage

```
getW(S)
```

Arguments

S	Structure with the proportion of neighbors having an opposite class with respect to an instance and label, normalized by the global class imbalance

Value

A vector of weights to be considered when oversampling for each instance

initTypes *Auxiliary function used by MLSOL. Categorizes each pair instance-label of the dataset with a type*

Description

Auxiliary function used by MLSOL. Categorizes each pair instance-label of the dataset with a type

Usage

```
initTypes(C, neighbors, k, minoritary, D, d)
```

Arguments

C	List of vectors with one value for each pair instance-label
neighbors	Structure with the k nearest neighbors of each instance of the dataset
k	Number of neighbors to be considered for each instance
minoritary	Vector with the minoritary value of each label (normally, 1)
D	mld `mldr` object with the multilabel dataset to preprocess
d	Vector with the instances of the dataset which have one or more label active (ideally, all of them)

Value

A synthetic sample derived from the one passed as a parameter and its neighbors

LPROS *Randomly clones instances with minoritary labelsets*

Description

This function implements the LP-ROS algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify instances with minoritary labels, and randomly clone them.

Usage

```
LPROS(D, P)
```

Arguments

D	mld `mldr` object with the multilabel dataset to preprocess
P	Percentage in which the original dataset is increased

Value

A mld object containing the preprocessed multilabel dataset

Source

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. Neurocomputing, 163, 3-16.

Examples

```
library(mldr)
LPROS(birds, 25)
```

LPRUS *Randomly deletes instances with majoritary labelsets*

Description

This function implements the LP-RUS algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify instances with majoritary labelsets, and randomly delete them from the original dataset.

Usage

```
LPRUS(D, P)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
P	Percentage in which the original dataset is increased

Value

A mld object containing the preprocessed multilabel dataset

Source

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. Neurocomputing, 163, 3-16.

Examples

```
library(mldr)
LPRUS(birds, 25)
```

MLeNN *Multilabel edited Nearest Neighbor (MLeNN)*

Description

This function implements the MLeNN algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify instances with majoritary labels, and remove its neihgbors which are too different to them, in terms of active labels.

Usage

```
MLeNN(D, TH = 0.5, k = 3, neighbors = NULL, tableVDM = NULL)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
TH	threshold for the Hamming Distance in order to consider an instance different to another one. Defaults to 0.5.
k	number of nearest neighbours to check for each instance. Defaults to 3.
neighbors	Structure with instances and neighbors. If it is empty, it will be calculated by the function
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

An mldr object containing the preprocessed multilabel dataset

Source

Francisco Charte, Antonio J. Rivera, María J. del Jesus, and Francisco Herrera. MLeNN: A First Approach to Heuristic Multilabel Undersampling. Intelligent Data Engineering and Automated Learning – IDEAL 2014. ISBN 978-3-319-10840-7.

`MLRkNNOS` *Reverse-nearest neighborhood based oversampling for imbalanced, multi-label datasets*

Description

This function implements an algorithm that uses the concept of reverse nearest neighbors, in order to create new instances for each label. Then, several radial SVMs, one for each label, are trained in order to predict each label of the synthetic instances.

Usage

```
MLRkNNOS(D, k, tableVDM = NULL)
```

Arguments

`D`	mld `mldr` object with the multilabel dataset to preprocess
`k`	Number of neighbors to be considered when creating a synthetic instance
`tableVDM`	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A mld object containing the preprocessed multilabel dataset

Source

Sadhukhan, P., & Palit, S. (2019). Reverse-nearest neighborhood based oversampling for imbalanced, multi-label datasets. Pattern Recognition Letters, 125, 813-820

MLROS *Randomly clones instances with minoritary labels*

Description

This function implements the ML-ROS algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify instances with minoritary labels, and randomly clone them.

Usage

```
MLROS(D, P)
```

Arguments

D	mld `mldr` object with the multilabel dataset to preprocess
P	Percentage in which the original dataset is increased

Value

A mld object containing the preprocessed multilabel dataset

Source

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. Neurocomputing, 163, 3-16.

Examples

```
library(mldr)
library(mldr.resampling)
MLROS(birds, 25)
```

MLRUS *Randomly deletes instances with majoritary labels*

Description

This function implements the ML-RUS algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify instances with majoritary labels, and randomly delete them from the original dataset.

Usage

```
MLRUS(D, P)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
P	Percentage in which the original dataset is increased

Value

A mld object containing the preprocessed multilabel dataset

Source

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. Neurocomputing, 163, 3-16.

Examples

```
library(mldr)
MLRUS(birds, 25)
```

MLSMOTE *Synthetic oversampling of multilabel instances (MLSMOTE)*

Description

This function implements the MLSMOTE algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify instances with minoritary labels, and generate synthetic instances based on their neighbor instances.

Usage

```
MLSMOTE(D, k, strategy = "ranking", tableVDM = NULL)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
k	Number of neighbors to be considered when creating a synthetic instance
strategy	Strategy for choosing the synthetic labels. Possible values: "union", "intersection" and "ranking" (default)
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A mld object containing the preprocessed multilabel dataset

Source

Charte, F., Rivera, A. J., del Jesus, M. J., & Herrera, F. (2015). MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. Knowledge-Based Systems, 89, 385-397.

`MLSOL` *Multi-label oversampling based on local label imbalance (MLSOL)*

Description

This function implements the MLSOL algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, which applies oversampling on difficult regions of the instance space, in order to help classifiers distinguish labels.

Usage

```
MLSOL(D, P, k, neighbors = NULL, tableVDM = NULL)
```

Arguments

`D`	mld `mldr` object with the multilabel dataset to preprocess
`P`	Percentage in which the original dataset is increased
`k`	Number of neighbors to be considered when computing the neighbors of an instance
`neighbors`	Structure with all instances and neighbors in the dataset. If it is empty, it will be calculated by the function
`tableVDM`	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A mld object containing the preprocessed multilabel dataset

Source

Liu, B., Blekas, K., & Tsoumakas, G. (2022). Multi-label sampling based on local label imbalance. Pattern Recognition, 122, 108294.

MLTL *Multilabel approach for the Tomek Link undersampling algorithm (MLTL)*

Description

This function implements the MLTL algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to identify tomek links (majoritary instances with a very different neighbor), and remove them. It's like MLeNN, with the number of neighbors being 1.

Usage

```
MLTL(D, TH, neighbors = NULL, tableVDM = NULL)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
TH	threshold for the Hamming Distance in order to consider an instance different to another one.
neighbors	Structure with instances and neighbors. If it is empty, it will be calculated by the function
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

An mldr object containing the preprocessed multilabel dataset

Source

Pereira, R. M., Costa, Y. M., & Silla Jr, C. N. (2020). MLTL: A multi-label approach for the Tomek Link undersampling algorithm. Neurocomputing, 383, 95-105.

MLUL *Multi-label undersampling based on local label imbalance (MLUL)*

Description

This function implements the MLUL algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, which applies undersampling, removing difficult instances according to their neighbors.

Usage

```
MLUL(D, P, k, neighbors = NULL, tableVDM = NULL)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
P	Percentage in which the original dataset is decreased
k	Number of neighbors to be considered when computing the neighbors of an instance
neighbors	Structure with all instances and neighbors in the dataset. If it is empty, it will be calculated by the function
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A mld object containing the preprocessed multilabel dataset

Source

Liu, B., Blekas, K., & Tsoumakas, G. (2022). Multi-label sampling based on local label imbalance. Pattern Recognition, 122, 108294.

---

newSample *Auxiliary function used by MLSMOTE. Creates a synthetic sample based on values of attributes and labels of its neighbors*

Description

Auxiliary function used by MLSMOTE. Creates a synthetic sample based on values of attributes and labels of its neighbors

Usage

```
newSample(seedInstance, refNeigh, neighbors, strategy, D)
```

Arguments

seedInstance	Sample we are using as "template"
refNeigh	Reference neighbor
neighbors	Neighbors to take into account
strategy	Strategy for choosing the synthetic labels: union, intersection or ranking
D	mld mldr object with the multilabel dataset to preprocess

Value

A synthetic sample derived from the one passed as a parameter and its neighbors

REMEDIAL *Decouples highly imbalanced labels*

Description

This function implements the REMEDIAL algorithm. It is a preprocessing algorithm for imbalanced multilabel datasets, whose aim is to decouple frequent and rare classes appearing in the same instance. For doing so, it aggregates new instances to the dataset and edit the labels present in them.

Usage

```
REMEDIAL(mld)
```

Arguments

`mld`	`mldr` object with the multilabel dataset to preprocess

Value

An mldr object containing the preprocessed multilabel dataset

Source

F. Charte, A. J. Rivera, M. J. del Jesus, F. Herrera. "Resampling Multilabel Datasets by Decoupling Highly Imbalanced Labels". Proc. 2015 International Conference on Hybrid Artificial Intelligent Systems (HAIS 2015), pp. 489-501, Bilbao, Spain, 2015. Implementation from the original `mldr` package

Examples

```
library(mldr)
REMEDIAL(birds)
```

resample *Interface function of the package. It executes one or several algorithms, given as strings, and stores the resulting MLDs in arff files*

Description

Interface function of the package. It executes one or several algorithms, given as strings, and stores the resulting MLDs in arff files

Usage

```
resample(
  D,
  algorithms,
  P = 25,
  k = 3,
  TH = 0.5,
  strategy = "ranking",
  params,
  outputDirectory = tempdir()
)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
algorithms	String, or string vector, with the name(s) of the algorithm(s) to be applied.
P	Percentage in which the original dataset is increased/decreased, if required by the algorithm(s). Defaults to 25
k	Number of neighbors taken into account for each instance, if required by the algorithm(s). Defaults to 3
TH	Threshold for the Hamming Distance in order to consider an instance different to another one, if required by the algorithm(s). Defaults to 0.5
strategy	Strategy for choosing the synthetic labels, if required by the algorithm. Defaults to ranking
params	Dataframe with 4 columns: name of the algorithm, P, k and TH, in that order, to execute several algorithms with different values for their parameters
outputDirectory	Route with the directory where generated ARFF files will be stored. Defaults to a temporary directory

Value

Dataframe with times (in seconds) taken in to execute each algorithm

Examples

```
library(mldr)
library(mldr.resampling)
resample(birds, "LPROS", P=25)
resample(birds, c("LPROS", "LPRUS"), P=30)
```

setNumCores *Set the number of cores available for parallel computing*

Description

Set the number of cores available for parallel computing

Usage

```
setNumCores(n)
```

Arguments

n	The new value for the number of cores

Value

No return value, called in order to change the number of cores

Examples

```
setNumCores(8)
```

setParallel *Enable/Disable parallel computing*

Description

Enable/Disable parallel computing

Usage

```
setParallel(beParallel)
```

Arguments

beParallel	A boolean indicating if parallel computing is to be enabled (TRUE) or disabled (FALSE)

Value

No return value, called in order to enable parallel computing

Examples

```
setParallel(TRUE)
```

vdm *Auxiliary function used to calculate the Value Difference Metric (VDM) between two instances considering their non numeric attributes*

Description

Auxiliary function used to calculate the Value Difference Metric (VDM) between two instances considering their non numeric attributes

Usage

```
vdm(D, sample, y, label, tableVDM = NULL)
```

Arguments

D	mld mldr object with the multilabel dataset to preprocess
sample	Index of the first sample
y	Index of the second sample
label	Label that will be considered in calculations
tableVDM	Dataframe object containing previous calculations for faster processing. If it is empty, the algorithm will be slower

Value

A value for the distance

Index