Package 'mldr'

December 19, 2019

Title Exploratory Data Analysis and Manipulation of Multi-Label Data Sets

Version 0.4.3

Date 2019-12-19

Description Exploratory data analysis and manipulation functions for multi-label data sets along with an interactive Shiny application to ease their use.

URL https://github.com/fcharte/mldr

Depends R (>= 3.0.0)

Imports shiny (>= 0.11), XML, circlize, graphics, grDevices, stats, methods

Suggests pROC, knitr, mldr.datasets, testthat

License LGPL (>= 3) | file LICENSE

LazyData true

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Author David Charte [cre] (<https://orcid.org/0000-0002-4830-9512>), Francisco Charte [aut] (<https://orcid.org/0000-0002-3083-8942>), Antonio J. Rivera [aut]

Maintainer David Charte <fdavidcl@ugr.es>

Repository CRAN

Date/Publication 2019-12-19 17:50:10 UTC

R topics documented:

`+.mldr` *Generates a new mldr object joining the rows in the two mldrs given as input*

Description

Generates a new mldr object joining the rows in the two mldrs given as input

Usage

```
## S3 method for class 'mldr'
mldr1 + mldr2
```

Arguments

`mldr1`	First `mldr` object to join
`mldr2`	Second `mldr` object to join

Value

a new `mldr` object with all rows in the two parameters

==.mldr *Checks if two mldr objects have the same structure*

Description

Checks if two mldr objects have the same structure

Usage

```
## S3 method for class 'mldr'
mldr1 == mldr2
```

Arguments

mldr1	First mldr object to compare
mldr2	Second mldr object to compare

Value

TRUE if the two mldr objects have the same structure, FALSE otherwise

Averaged metrics *Multi-label averaged evaluation metrics*

Description

Evaluation metrics based on simple metrics for the confusion matrix, averaged through several criteria.

Usage

```
accuracy(true_labels, predicted_labels, undefined_value = "diagnose")

precision(true_labels, predicted_labels, undefined_value = "diagnose")

micro_precision(true_labels, predicted_labels, ...)

macro_precision(true_labels, predicted_labels,
  undefined_value = "diagnose")

recall(true_labels, predicted_labels, undefined_value = "diagnose")

micro_recall(true_labels, predicted_labels, ...)

macro_recall(true_labels, predicted_labels, undefined_value = "diagnose")
```

```
fmeasure(true_labels, predicted_labels, undefined_value = "diagnose")

micro_fmeasure(true_labels, predicted_labels, ...)

macro_fmeasure(true_labels, predicted_labels,
  undefined_value = "diagnose")
```

Arguments

`true_labels`	Matrix of true labels, columns corresponding to labels and rows to instances.
`predicted_labels`	Matrix of predicted labels, columns corresponding to labels and rows to instances.
`undefined_value`	The value to be returned when a computation results in an undefined value due to a division by zero. See details.
`...`	Additional parameters for precision, recall and Fmeasure.

Details

Available metrics in this category

- `accuracy`: Bipartition based accuracy
- `fmeasure`: Example and binary partition F_1 measure (harmonic mean between precision and recall, averaged by instance)
- `macro_fmeasure`: Label and bipartition based F_1 measure (harmonic mean between precision and recall, macro-averaged by label)
- `macro_precision`: Label and bipartition based precision (macro-averaged by label)
- `macro_recall`: Label and bipartition based recall (macro-averaged by label)
- `micro_fmeasure`: Label and bipartition based F_1 measure (micro-averaged)
- `micro_precision`: Label and bipartition based precision (micro-averaged)
- `micro_recall`: Label and bipartition based recall (micro-averaged)
- `precision`: Example and bipartition based precision (averaged by instance)
- `recall`: Example and bipartition based recall (averaged by instance)

Deciding a value when denominators are zero

Parameter `undefined_value`: The value to be returned when a computation results in an undefined value due to a division by zero. Can be a single value (e.g. NA, 0), a function with the following signature:

`function(tp,fp,tn,fn)`

or a string corresponding to one of the predefined strategies. These are:

- `"diagnose"`: This strategy performs the following decision:
  - Returns 1 if there are no true labels and none were predicted

  - Returns 0 otherwise

  This is the default strategy, and the one followed by MULAN.

- `"ignore"`: Occurrences of undefined values will be ignored when averaging (averages will be computed with potentially less values than instances/labels). Undefined values in micro-averaged metrics cannot be ignored (will return `NA`).
- `"na"`: Will return `NA` (with class `numeric`) and it will be propagated when averaging (averaged metrics will potentially return `NA`).

Value

Atomical numeric vector containing the resulting value in the range [0, 1].

See Also

`mldr_evaluate`, `mldr_to_labels`

Other evaluation metrics: `Basic metrics`, `Ranking-based metrics`

Examples

```
true_labels <- matrix(c(
1,1,1,
0,0,0,
1,0,0,
1,1,1,
0,0,0,
1,0,0
), ncol = 3, byrow = TRUE)
predicted_labels <- matrix(c(
1,1,1,
0,0,0,
1,0,0,
1,1,0,
1,0,0,
0,1,0
), ncol = 3, byrow = TRUE)

precision(true_labels, predicted_labels, undefined_value = "diagnose")
macro_recall(true_labels, predicted_labels, undefined_value = 0)
macro_fmeasure(
  true_labels, predicted_labels,
  undefined_value = function(tp, fp, tn, fn) as.numeric(fp == 0 && fn == 0)
)
```

Basic metrics *Multi-label evaluation metrics*

Description

Several evaluation metrics designed for multi-label problems.

Usage

```
hamming_loss(true_labels, predicted_labels)

subset_accuracy(true_labels, predicted_labels)
```

Arguments

true_labels	Matrix of true labels, columns corresponding to labels and rows to instances.
predicted_labels	Matrix of predicted labels, columns corresponding to labels and rows to instances.

Details

Available metrics in this category

- hamming_loss: describes the average absolute distance between a predicted label and its true value.
- subset_accuracy: the ratio of correctly predicted labelsets.

Value

Resulting value in the range [0, 1]

See Also

mldr_evaluate, mldr_to_labels

Other evaluation metrics: Averaged metrics, Ranking-based metrics

Examples

```
true_labels <- matrix(c(
1,1,1,
0,0,0,
1,0,0,
1,1,1,
0,0,0,
1,0,0
), ncol = 3, byrow = TRUE)
predicted_labels <- matrix(c(
```

```
1,1,1,
0,0,0,
1,0,0,
1,1,0,
1,0,0,
0,1,0
), ncol = 3, byrow = TRUE)

hamming_loss(true_labels, predicted_labels)
subset_accuracy(true_labels, predicted_labels)
```

`birds` *birds*

Description

birds dataset.

Usage

```
birds
```

Format

An mldr object with 645 instances, 279 attributes and 19 labels

Source

F. Briggs, Yonghong Huang, R. Raich, K. Eftaxias, Zhong Lei, W. Cukierski, S. Hadley, A. Hadley, M. Betts, X. Fern, J. Irvine, L. Neal, A. Thomas, G. Fodor, G. Tsoumakas, Hong Wei Ng, Thi Ngoc Tho Nguyen, H. Huttunen, P. Ruusuvuori, T. Manninen, A. Diment, T. Virtanen, J. Marzat, J. Defretin, D. Callender, C. Hurlburt, K. Larrey, M. Milakov. "The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment", in proc. 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)

Examples

```
summary(birds)
birds$labels
```

`concurrenceReport` *Generates a label concurrence report*

Description

This function produces a label concurrence report, providing the average SCUMBLE, SCUMBLE by label, a list with the minority labels most affected by this problem indicating which majority labels they appear with, and a concurrence plot. The report output is written in the standard output by default, but it could be redirected to a PDF file.

Usage

```
concurrenceReport(mld, pdfOutput = FALSE, file = "Rconcurrence.pdf")
```

Arguments

`mld`	`mldr` object to analyze
`pdfOutput`	Boolean value indicating if the output has to be sent to a PDF file. Defaults to true, so the output is shown in the console.
`file`	If the `pdfOutput` parameter is `true` the output will be written in the file specified by this parameter. The default file name is `Rocurrence.pdf`.

Value

None

See Also

`remedial`, `labelInteractions`

Examples

```
library(mldr)
## Not run:
concurrenceReport(birds)

## End(Not run)
```

emotions *emotions*

Description

emotions dataset.

Usage

```
emotions
```

Format

An mldr object with 593 instances, 78 attributes and 6 labels

Source

K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas. "Multilabel Classification of Music into Emotions". Proc. 2008 International Conference on Music Information Retrieval (ISMIR 2008), pp. 325-330, Philadelphia, PA, USA, 2008

Examples

```
summary(emotions)
emotions$labels
```

genbase *genbase*

Description

genbase dataset.

Usage

```
genbase
```

Format

An mldr object with 662 instances, 1213 attributes and 27 labels

Source

S. Diplaris, G. Tsoumakas, P. Mitkas and I. Vlahavas. Protein Classification with Multiple Algorithms, Proc. 10th Panhellenic Conference on Informatics (PCI 2005), pp. 448-456, Volos, Greece, November 2005

Examples

```
summary(genbase)
genbase$labels
```

`labelInteractions` *Provides data about interactions between labels*

Description

This function facilitates a list with the minority labels most affected by the problem of concurrence with majority labels, provinding the indexes of the majority labels interacting with each minority and also the number of instances in which they appear together.

Usage

```
labelInteractions(mld, labelProportion)
```

Arguments

`mld`	`mldr` object to analyze
`labelProportion`	A value in the (0,1] range establishing the proportion of minority labels to be included as result. By default at least 3 or 10% of minority labels are included, or all of them if there are fewer than 3.

Value

A list with two slots, `indexes` and `interactions`. The former contains the indexes of the minority labels, sorted from higher to lower SCUMBLE metric. The latter will provide an element for each of the previous labels, communicating the indexes of the majority labels it interacts with and the number of samples in which they appear together.

See Also

`remedial`, `concurrenceReport`

Examples

```
library(mldr)
labelInteractions(birds)
```

mldr *Creates an object representing a multilabel dataset*

Description

Reads a multilabel dataset from a file and returns an `mldr` object containing the data and additional measures. The file has to be in ARFF format. The label information could be in a separate XML file (MULAN style) or in the the arff header (MEKA style)

Usage

```
mldr(filename, use_xml = TRUE, auto_extension = TRUE, xml_file,
  label_indices, label_names, label_amount,
  force_read_from_file = !all(c(missing(xml_file),
  missing(label_indices), missing(label_names), missing(label_amount),
  use_xml, auto_extension)), ...)
```

Arguments

`filename`	Name of the dataset
`use_xml`	Specifies whether to use an associated XML file to identify the labels. Defaults to TRUE
`auto_extension`	Specifies whether to add the '.arff' and '.xml' extensions to the filename where appropriate. Defaults to TRUE
`xml_file`	Path to the XML file. If not provided, the filename ending in ".xml" will be assumed
`label_indices`	Optional vector containing the indices of the attributes that should be read as labels
`label_names`	Optional vector containing the names of the attributes that should be read as labels
`label_amount`	Optional parameter indicating the number of labels in the dataset, which will be taken from the last attributes of the dataset
`force_read_from_file`	Set this parameter to TRUE to always read from a local file, or set it to FALSE to look for the dataset within the 'mldr.datasets' package
`...`	Extra parameters to be passed to 'read_arff'

Value

An mldr object containing the multilabel dataset

See Also

`mldr_from_dataframe`, `read.arff`, `summary.mldr`

Examples

```
library(mldr)
## Not run:
# Read "yeast.arff" and labels from "yeast.xml"
mymld <- mldr("yeast")

# Read "yeast.arff" and labels from "yeast.xml", converting categorical
# attributes to factors
mymld <- mldr("yeast", stringsAsFactors = TRUE)

# Read "yeast-tra.arff" and labels from "yeast.xml"
mymld <- mldr("yeast-tra", xml_file = "yeast.xml")

# Read "yeast.arff" specifying the amount of attributes to be used as labels
mymld <- mldr("yeast", label_amount = 14)

# Read MEKA style dataset, without XML file and giving extension
mymld <- mldr("IMDB.arff", use_xml = FALSE, auto_extension = FALSE)

## End(Not run)
```

---

mldrGUI *Launches the web-based GUI for mldr*

Description

Loads an interactive user interface in the web browser, built using R shiny.

Usage

```
mldrGUI()
```

Details

The **mldr** package provides a basic, Shiny-based GUI to work with multilabel datasets. You have to install the **shiny** package to be able to use this GUI.

The user interface allows working with any of the previous loaded datasets, as well as loading new ones. The GUI is structured into the following pages:

- **Main:** This page is divided into two sections. The one at the left can be used to choose a previously loaded dataset, as well as to load datasets from files. The right part shows some basic statistics about the selected multilabel dataset.
- **Labels:** This page shows a table containing for each label its name, index, count, relative frequency and imbalance ratio (IRLbl). The page also includes a bar plot of the label frequency. The range of labels in the plot can be customized.
- **Labelsets:** This page shows a table containing for each labelset its representation and a counter.

- **Attributes:** This page shows a table containing for each attribute its name, type and a summary of its values.
- **Concurrence:** This page shows for each label the number of instances in which it appears and its mean SCUMBLE measure, along with a plot that shows the level of concurrence among the selected labels. Clicking the labels in the table makes it possible to add/remove them from the plot.

The tables shown in these pages can be sorted by any of its fields, as well as filtered. The content of the tables can be copied to clipboard, printed and saved in CSV and Microsoft Excel format.

Value

Nothing

Examples

```
## Not run:
library(mldr)
mldrGUI()

## End(Not run)
```

`mldr_evaluate` *Evaluate predictions made by a multilabel classifier*

Description

Taking as input an `mldr` object and a matrix with the predictions given by a classifier, this function evaluates the classifier performance through several multilabel metrics.

Usage

```
mldr_evaluate(mldr, predictions, threshold = 0.5)
```

Arguments

`mldr`	Object of `"mldr"` class containing the instances to evaluate
`predictions`	Matrix with the labels predicted for each instance in the `mldr` parameter. Each element should be a value into [0,1] range
`threshold`	Threshold to use to generate bipartition of labels. By default the value 0.5 is used

Value

A list with multilabel predictive performance measures. The items in the list will be

- accuracy
- example_auc
- average_precision
- coverage
- fmeasure
- hamming_loss
- macro_auc
- macro_fmeasure
- macro_precision
- macro_recall
- micro_auc
- micro_fmeasure
- micro_precision
- micro_recall
- one_error
- precision
- ranking_loss
- recall
- subset_accuracy
- roc

The roc element corresponds to a roc object associated to the MicroAUC value. This object can be given as input to plot for plotting the ROC curve The example_auc, macro_auc, micro_auc and roc members will be NULL if the pROC package is not installed.

See Also

mldr, Basic metrics, Averaged metrics, Ranking-based metrics, roc.mldr

Examples

```
## Not run:
library(mldr)

# Get the true labels in emotions
predictions <- as.matrix(emotions$dataset[, emotions$labels$index])
# and introduce some noise (alternatively get the predictions from some classifier)
noised_labels <- cbind(sample(1:593, 200, replace = TRUE), sample(1:6, 200, replace = TRUE))
predictions[noised_labels] <- sample(0:1, 100, replace = TRUE)
# then evaluate predictive performance
res <- mldr_evaluate(emotions, predictions)
```

```
str(res)
plot(res$roc, main = "ROC curve for emotions")

## End(Not run)
```

`mldr_from_dataframe` *Generates an mldr object from a data.frame and a vector with label indices*

Description

This function creates a new `mldr` object from the data stored in a `data.frame`, taking as labels the columns pointed by the indexes given in a vector.

Usage

```
mldr_from_dataframe(dataframe, labelIndices, attributes, name)
```

Arguments

`dataframe`	The `data.frame` containing the dataset attributes and labels.
`labelIndices`	Vector containing the indices of attributes acting as labels. Usually the labels will be at the end (right-most columns) or the beginning (left-most columns) of the `data.frame`
`attributes`	Vector with the attributes type, as returned by the `attributes` member of an `mldr` object. By default the type of the data.frame columns will be used.
`name`	Name of the dataset. The name of the dataset given as first parameter will be used by default

Value

An mldr object containing the multilabel dataset

See Also

`mldr`, `summary.mldr`

Examples

```
library(mldr)

df <- data.frame(matrix(rnorm(1000), ncol = 10))
df$Label1 <- c(sample(c(0,1), 100, replace = TRUE))
df$Label2 <- c(sample(c(0,1), 100, replace = TRUE))
mymldr <- mldr_from_dataframe(df, labelIndices = c(11, 12), name = "testMLDR")

summary(mymldr)
```

mldr_to_labels *Label matrix of an MLD*

Description

Extracts a matrix with the true 0-1 assignment of labels of an "mldr" object.

Usage

```
mldr_to_labels(mldr)
```

Arguments

mldr	"mldr" object.

Value

Numeric matrix of labels.

See Also

Basic metrics, Averaged metrics, Ranking-based metrics.

mldr_transform *Transformns an MLDR into binary or multiclass datasets*

Description

Transforms an mldr object into one or serveral binary or multiclass datasets, returning them as data.frame objects

Usage

```
mldr_transform(mldr, type = "BR", labels)
```

Arguments

mldr	The mldr object to transform
type	Indicates the type of transformation to apply. Possible types are: • "BR" Produces one or more binary datasets, each one with one label • "LP" Produces a multiclass dataset using each labelset as class label
labels	Vector with the label indexes to include in the transformation. All labels will be used if not specified

Value

A list of data.frames containing the resulting datasets (for BR) or a data.frame with the dataset (for LP). The result is no longer an `mldr` object, but a plain `data.frame`

Examples

```
library(mldr)
emotionsbr <- mldr_transform(emotions, type = "BR")
emotionslp <- mldr_transform(emotions, type = "LP")
```

---

plot.mldr *Generates graphic representations of an mldr object*

Description

Generates graphic representations of an `mldr` object

Usage

```
## S3 method for class 'mldr'
plot(x, type = "LC", labelCount, labelIndices, title,
  ask = length(type) > prod(par("mfcol")), ...)
```

Arguments

`x`	The mldr object whose features are to be drawn
`type`	Indicates the type(s) of plot to be produced. Possible types are: • `"LC"` Draws a circular plot with sectors representing each label and links between them depicting label co-occurrences • `"LH"` for label histogram • `"LB"` for label bar plot • `"CH"` for cardinality histogram • `"AT"` for attributes by type pie chart • `"LSH"` for labelset histogram • `"LSB"` for labelset bar plot
`labelCount`	Samples the labels in the dataset to show information of only `labelCount` of them
`labelIndices`	Establishes the labels to be shown in the plot
`title`	A title to be shown above the plot. Defaults to the name of the dataset passed as first argument
`ask`	Specifies whether to pause the generation of plots after each one
`...`	Additional parameters to be given to barplot, hist, etc.

Examples

```
library(mldr)
## Not run:
# Label concurrence plot
plot(genbase, type = "LC") # Plots all labels
plot(genbase) # Same as above
plot(genbase, title = "genbase dataset", color.function = heat.colors) # Changes the title and color
plot(genbase, labelCount = 10) # Randomly selects 10 labels to plot
plot(genbase, labelIndices = genbase$labels$index[1:10]) # Plots info of first 10 labels

# Label bar plot
plot(emotions, type = "LB", col = terrain.colors(emotions$measures$num.labels))

# Label histogram plot
plot(emotions, type = "LH")

# Cardinality histogram plot
plot(emotions, type = "CH")

# Attributes by type
plot(emotions, type = "AT", cex = 0.85)

# Labelset histogram
plot(emotions, type = "LSH")

## End(Not run)
```

`print.mldr` *Prints the mldr content*

Description

Prints the `mldr` object data, including input attributs and output labels

Usage

```
## S3 method for class 'mldr'
print(x, ...)
```

Arguments

`x`	Object whose data are to be printed
`...`	Additional parameters to be given to print

See Also

`mldr`, `summary.mldr`

Examples

```
library(mldr)

emotions
print(emotions) # Same as above
```

`Ranking-based metrics` *Multi-label ranking-based evaluation metrics*

Description

Functions that compute ranking-based metrics, given a matrix of true labels and a matrix of predicted probabilities.

Usage

```
average_precision(true_labels, predictions, ...)

one_error(true_labels, predictions)

coverage(true_labels, predictions, ...)

ranking_loss(true_labels, predictions)

macro_auc(true_labels, predictions, undefined_value = 0.5,
  na.rm = FALSE)

micro_auc(true_labels, predictions)

example_auc(true_labels, predictions, undefined_value = 0.5,
  na.rm = FALSE)
```

Arguments

`true_labels`	Matrix of true labels, columns corresponding to labels and rows to instances.
`predictions`	Matrix of probabilities predicted by a classifier.
`...`	Additional parameters to be passed to the ranking function.
`undefined_value`	A default value for the cases when macro-averaged and example-averaged AUC encounter undefined (not computable) values, e.g. `0`, `0.5`, or `NA`.
`na.rm`	Logical specifying whether to ignore undefined values when `undefined_value` is set to `NA`.

Details

Available metrics in this category

- `average_precision`: Example and ranking based average precision (how many steps have to be made in the ranking to reach a certain relevant label, averaged by instance)
- `coverage`: Example and ranking based coverage (how many steps have to be made in the ranking to cover all the relevant labels, averaged by instance)
- `example_auc`: Example based Area Under the Curve ROC (averaged by instance)
- `macro_auc`: Label and ranking based Area Under the Curve ROC (macro-averaged by label)
- `micro_auc`: Label and ranking based Area Under the Curve ROC (micro-averaged)
- `one_error`: Example and ranking based one-error (how many times the top-ranked label is not a relevant label, averaged by instance)
- `ranking_loss`: Example and ranking based ranking-loss (how many times a non-relevant label is ranked above a relevant one, evaluated for all label pairs and averaged by instance)

Breaking ties in rankings

The additional `ties_method` parameter for the ranking function is passed to R's own `rank`. It accepts the following values:

- `"average"`
- `"first"`
- `"last"`
- `"random"`
- `"max"`
- `"min"`

See `rank` for information on the effect of each parameter. The default behavior in mldr corresponds to value `"last"`, since this is the behavior of the ranking method in MULAN, in order to facilitate fair comparisons among classifiers over both platforms.

Value

Atomical numeric vector specifying the resulting performance metric value.

See Also

`mldr_evaluate`, `mldr_to_labels`

Other evaluation metrics: `Averaged metrics`, `Basic metrics`

Examples

```
true_labels <- matrix(c(
1,1,1,
0,0,0,
1,0,0,
1,1,1,
```

```
0,0,0,
1,0,0
), ncol = 3, byrow = TRUE)
predicted_probs <- matrix(c(
.6,.5,.9,
.0,.1,.2,
.8,.3,.2,
.7,.9,.1,
.7,.3,.2,
.1,.8,.3
), ncol = 3, byrow = TRUE)

# by default, labels with same ranking are assigned ascending rankings
# in the order they are encountered
coverage(true_labels, predicted_probs)
# in the following, labels with same ranking will receive the same,
# averaged ranking
average_precision(true_labels, predicted_probs, ties_method = "average")

# the following will treat all undefined values as 0 (counting them
# for the average)
example_auc(true_labels, predicted_probs, undefined_value = 0)
# the following will ignore undefined values (not counting them for
# the average)
example_auc(true_labels, predicted_probs, undefined_value = NA, na.rm = TRUE)
```

`read.arff` *Read an ARFF file*

Description

Reads a multilabel dataset from an ARFF file in Mulan or MEKA and retrieves instances distinguishing attributes corresponding to labels

Usage

```
read.arff(filename, use_xml = TRUE, auto_extension = TRUE, xml_file,
  label_indices, label_names, label_amount, ...)
```

Arguments

`filename`	Name of the dataset
`use_xml`	Specifies whether to use an associated XML file to identify the labels. Defaults to TRUE
`auto_extension`	Specifies whether to add the '.arff' and '.xml' extensions to the filename where appropriate. Defaults to TRUE
`xml_file`	Path to the XML file. If not provided, the filename ending in ".xml" will be assumed

label_indices	Optional vector containing the indices of the attributes that should be read as labels
label_names	Optional vector containing the names of the attributes that should be read as labels
label_amount	Optional parameter indicating the number of labels in the dataset, which will be taken from the last attributes of the dataset
...	Extra parameters that will be passed to the parsers. Currently only the option stringsAsFactors is available

Value

A list containing four members: dataframe containing the dataset, labelIndices specifying the indices of the attributes that correspond to labels, attributes containing name and type of each attribute and name of the dataset.

See Also

mldr_from_dataframe, mldr

Examples

```
library(mldr)
## Not run:
# Read "yeast.arff" and labels from "yeast.xml"
mymld <- read.arff("yeast")

## End(Not run)
```

---

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

See Also

concurrenceReport, labelInteractions

Examples

```
library(mldr)
## Not run:
summary(birds)
summary(remedial(birds))

## End(Not run)
```

---

roc *ROC curve*

Description

Calculates the ROC (Receiver Operating Characteristic) curve for given true labels and predicted ones. The pROC package is needed for this functionality.

Usage

```
roc(...)

## S3 method for class 'mldr'
roc(mldr, predictions, ...)
```

Arguments

...	Additional parameters to be passed to the pROC::roc function. See roc for more information.
mldr	An "mldr" object. Its labels will be extracted via mldr_to_labels.
predictions	Matrix of predicted labels or probabilities, columns corresponding to labels and rows to instances.

Value

ROC object from pROC package.

See Also

mldr_evaluate roc

---

summary.mldr *Provides a summary of measures about the mldr*

Description

Prints a summary of the measures obtained from the mldr object

Usage

```
## S3 method for class 'mldr'
summary(object, ...)
```

Arguments

object	Object whose measures are to be printed
...	Additional parameters to be given to print

See Also

mldr, print.mldr

Examples

```
library(mldr)

summary(emotions)
```

write_arff *Write an "mldr" object to a file*

Description

Save the mldr content to an ARFF file and the label data to an XML file. If you need **faster write, more options and support for other formats**, please refer to the write.mldr function in package mldr.datasets.

Usage

```
write_arff(obj, filename, write.xml = FALSE)
```

Arguments

obj	The "mldr" object whose content is going to be written
filename	Base name for the files (without extension)
write.xml	TRUE or FALSE, stating if the XML file has to be written

See Also

mldr_from_dataframe, mldr

In mldr.datasets: write.mldr

Examples

```
dir <- tempdir()
write_arff(emotions, file.path(dir, "myemotions"))
file.remove(file.path(dir, "myemotions.arff"))
```

[.mldr *Filter rows in a* mldr *returning a new* mldr

Description

Generates a new mldr object containing the selected rows from an existent mldr

Usage

```
## S3 method for class 'mldr'
mldrObject[rowFilter = T]
```

Arguments

mldrObject	Original mldr object from which some rows are going to be selected
rowFilter	Expression to filter the rows

Value

A new mldr object with the selected rows

See Also

mldr_from_dataframe, ==.mldr, +.mldr

Examples

```
library(mldr)

highlycoupled <- genbase[.SCUMBLE > 0.05] # Select instances with highly imbalanced coupled labels
summary(highlycoupled)   # Compare the selected instances
summary(genbase)         # with the traits of the original MLD
```

Index