Package: PerfMeas (via r-universe)

August 24, 2024

Type Package

Title Performance Measures for Ranking and Classification Tasks

Version 1.2.5

Date 2022-09-09

Author Giorgio Valentini [aut, cre]

Maintainer Giorgio Valentini <valentini@di.unimi.it>

Description Implementation of different performance measures for classification and ranking tasks including Area Under the Receiving Characteristic Curve (AUROC) and Area Under the Precision Recall Curve (AUPRC), precision at a given recall, F-score for single and multiple classes.

License GPL (>= 2)

LazyLoad yes

Depends limma, graph, RBGL

NeedsCompilation yes

Date/Publication 2022-09-14 06:10:02 UTC

Repository https://gvalentini58.r-universe.dev

RemoteUrl https://github.com/cran/PerfMeas

RemoteRef HEAD

RemoteSha 038339e0053cc716ca4970f20c9abb608fc8e236

Contents

`PerfMeas-package` *Performance Measures for Ranking and Classification Tasks*

Description

Metrics for ranking and classification tasks: Area Under the Receiving Operating Characteristic Curve (AUROC), Area Under the Precision and Recall Curve (AUPRC), F-scores, and precision at given recall level are implemented.

Details

Package:	PerfMeas
Type:	Package
Version:	1.2.5
Date:	2022-09-09
License:	GPL (>= 2)
LazyLoad:	yes

This package implements a set of functions to estimate the AUC, F-score, precision, recall, specificity, accuracy according to the 0/1 loss, and precision at given recall level for ranking and classification problems.

Functions to compute the above measures for single classes or for sets of classes are provided.

Author(s)

Giorgio Valentini

DI, Dipartimento di Informatica

Universita' degli Studi di Milano

`<{valentini}@di.unimi.it>`

Maintainer: *Giorgio Valentini*

AUC.measures	*AUC measures*

Description

Set of functions to compute the Area Under the ROC Curve (AUC)

Usage

```
AUC.single(pred, labels)
AUC.single.over.classes(target, predicted, g, root = "00")
AUC.n.single(pred, labels, n=50)
AUC.n.single.over.classes(target, predicted, g, n=50, root = "00")
compute.mean.AUC.single.over.classes(y)
```

Arguments

`pred`	numeric vector (scores) of the values of the predicted labels
`labels`	numeric vector of the true labels (0 negative, 1 positive examples)
`target`	matrix with the target multilabels: rows correspond to examples and columns to classes. target[i,j] = 1 if example i belongs to class j, target[i,j] = 0 otherwise.
`predicted`	a numeric matrix with predicted values (scores): rows correspond to examples and columns to classes.
`g`	a graph of class *graphNEL* (package **graph**) of the classes. If g is missing no per.level results are computed
`n`	number of negatives (def=50)
`root`	the name of the root node (def. "00")
`y`	a list of lists. The components of the outer list is a list returned from the function `AUC.single.over.classes`

Details

`AUC.single` computes the AUC for a single class.

`AUC.single.over.classes` computes AUC for a set of classes, including their average values across classes and the average values across the levels of the hierarchy (if any); level 1 classes are at distance 1 from the root, level 2 the second level, till to last level correponding to the leaves. Note that if the argument g is missing no per-level values are computed.

`AUC.n.single` computes the AUCn for a single class, i.e. the AUC by considering only the first n top ranked negatives, where n is the absolute number of negative examples receiving the highest scores.

`AUC.n.single.over.classes` computes AUCn for a set of classes, including their average values across classes and the average values across the levels of the hierarchy (if any); level 1 classes are at distance 1 from the root, level 2 the second level, till to last level correponding to the leaves. Note that if the argument g is missing no per-level values are computed.

compute.mean.AUC.single.over.classes compute means across folds of AUC.single.over.classes. It can be used to automatically computed average values (for each class, level, or average across classes) across folds.

Value

AUC.single returns a numeric value corresponding to the AUC.

AUC.single.over.classes returns a list with three elements:

- average	the average AUC across classes
- per.level	a named vector with average AUC for each level of the hierarchy; names correspond to levels
- per.class	a named vector with AUC for each class; names correspond to classes

AUC.n.single returns a numeric value corresponding to the AUCn.

AUC.n.single.over.classes returns a list with three elements:

- average	the average AUCn across classes
- per.level	a named vector with average AUCn for each level of the hierarchy; names correspond to levels
- per.class	a named vector with AUCn for each class; names correspond to classes

compute.mean.AUC.single.over.classes returns a list obtained by averaging the results across folds of the input y. The components are:

- average	the average AUC across classes
- per.level	a named vector with average AUC for each level of the hierarchy; names correspond to levels
- per.class	a named vector with AUC for each class; names correspond to classes

See Also

F.measures, PXR

Examples

```
# preparing pseudo.random scores and target-labels for examples: 100 examples
# and 10 classes
Scores <- matrix(runif(1000),nrow=100);
Targets <- matrix(integer(1000),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(1000, sd=0.3),nrow=100);
colnames(Scores) <-colnames(Targets) <- LETTERS[1:10];
# getting scores and labels of class "A"
scores <- Scores[,"A"];
labels <- Targets[,"A"];
# AUC for a single class
AUC.single(scores,labels);
# AUC for the 10 classes
```

```
AUC.single.over.classes(Targets, Scores);
# AUCn for a single class considering only the first top scored negatives
AUC.n.single(scores,labels, n=20);
# AUCn for the 10 classes considering only the first top scored negatives
AUC.n.single.over.classes(Targets, Scores, n=20);
```

AUPRC *Area Under the Precision Recall Curve*

Description

Functions to compute the Area Under the Precision Recall Curve (AUPRC) and the Area Under the F-score Recall Curve (AUFRC)

Usage

```
AUPRC(z, comp.precision=TRUE)
trap.rule.integral(x,y)
```

Arguments

`z`	a list of lists. The components of the outer list is a list returned from the function `precision.at.all.recall.levels` that reports precision, recall and f-score results at different levels for different methods or tasks.
`comp.precision`	boolean. It TRUE (default) the AUPRC is computed otherwise the area under the F-score curve is computed
`x`	vector of the x values in increasing order
`y`	vector of the corresponding y=f(x) values

Details

`AUPRC` computes the Area Under the Precision Recall Curve or the Area Under the F-score Recall Curve (AUFRC) for multiple curves by using the output of the function `precision.at.all.recall.levels`.

The function `trap.rule.integral` implements the trapezoidal rule of integration and can be used to compute the integral of any empirical function expressed as a set of pair values (a vector of `x` values and a vector of `y = f(x)` values). In particular if `x` is the recall (with values in ascending order) and `y` the corresponding precision, `trap.rule.integral` copmutes the AUPRC.

Value

`AUPRC` returns the value of the AUPRC (if the argument `comp.precision = TRUE`), otherwise the value of the AUFRC.

`trap.rule.integral` returns the value of the integral.

See Also

`AUC.measures`, `PXR`

Examples

```
# loading matrices of scores an correponding table of classes
data(T);
data(Scores);
res=list();
classes=1:10
# computing precision recall values
for (j in classes) res=c(res, list(precision.at.all.recall.levels(Scores[,j], T[,j])));
names(res)<-seq(0.1, 1, by=0.1);
# computing AUPRC
AUPRC (res, comp.precision=TRUE);
# computing AU F-score recall curve
AUPRC (res, comp.precision=TRUE);

# Loading precision at given recall levels for different methods
data(PrecRec);
# computing AUPRC for different methods
x <- seq(0.1, 1, by=0.1);
res <- numeric(nrow(PrecRec));
names(res) <- rownames(PrecRec);
for (i in 1:nrow(PrecRec))
  res[i] <- trap.rule.integral(x, PrecRec[i,]);
print(res);
```

`example.data` *Datasets used in the examples of the package*

Description

Collection of datasets used in the examples of the package

Usage

```
data(Scores)
data(T)
data(PrecRec)
```

Details

The `T` data is a named 1901 X 10 matrix whose rows correspondes to yest genes, while columns correspond to 10 FunCat (Functional Categories) classes. If T_ij = 1 gene i belong to class j, if T_ij = 0 gene i does not belong to class j. The `Scores` data is a named 1901 X 10 matrix representing scores (likelihood) that a given gene belongs to a given class: higher the value higher the likelihood. `PrecRec` is a matrix representing precision at 10 different recall values of 7 methods for gene function prediction.

`F.measures` *F-measures*

Description

Set of functions to compute the F-measure, precision, recall, specificity and 0/1 loss accuracy.

Usage

```
F.measure.single(pred, labels)
F.measure.single.over.classes(target, predicted, g, root = "00")
compute.mean.F.measure.single.over.classes(y)
```

Arguments

`pred`	vector of the predicted labels. 0 stands for negative and 1 for positive
`labels`	vector of the true labels. 0 stands for negative and 1 for positive
`target`	matrix with the target multilabels. 0 stands for negative and 1 for positive. Rows correspond to examples and columns to classes.
`predicted`	matrix with the predicted multilabels. 0 stands for negative and 1 for positive. Rows correspond to examples and columns to classes.
`g`	graph of the classes (object of class graphNEL, package **graph**). If missing, no per level results are computed.
`root`	the name of the root node (def. "00") of the graph g.
`y`	a list of lists. The components of the outer list is a list returned from the function `F.measure.single.over.classes`

Details

`F.measure.single` computes the F.score, precision, recall, specificity and accuracy for a single class.

`F.measure.single.over.classes` computes precision, recall, specificity, accuracy and F-measure for a set of classes. In particualr it computes the correponding average values across classes, the average values across levels of the hierarchy of the classes (if any), and the values of the measures for each class. Note that if there is no hierarchy between classes (represented by the graph g), you can miss the g parameter and no per-level values are computed.

`compute.mean.F.measure.single.over.classes` computes means across folds
of F.measure.single.over.classes. This function could be useful in cross-validated or multiple hold-out experimental settings.

Value

`F.measure.single` returns a named numeric vector with six elements:

`- P`	precision

- `R`	recall (sensitivity)
- `S`	specificity
- `F`	F measure
- `A`	0/1 loss accuracy
- `npos`	number of positive examples

`F.measure.single.over.classes` returns a list with three elements:

- `average`	a named vector with the average precision, recall, specificity, F-measure, accuracy and average number of positive examples across classes.
- `per.level`	a named matrix with average precision, recall, specificity, F-measure and accuracy for each level of the hierarchy. Named rows correspond to levels, named columns correspond respectively to precision, recall, specificity, F-measure, accuracy and number of positive examples.
- `per.class`	a named matrix with precision, recall, specificity, F-measure, accuracy and number of positive examples for each class. Named rows correspond to classes, named columns correspond respectively to precision, recall, specificity, F-measure, accuracy and and number of positive examples.

`compute.mean.F.measure.single.over.classes` returns a list obtained by averaging the results across folds of the input y. The components are:

- `average`	a named vector with the average precision, recall, specificity, F-measure and accuracy across classes across folds.
- `per.level`	a named matrix with average precision, recall, specificity, F-measure and accuracy for each level of the hierarchy across folds. Named rows correspond to levels, named columns correspond respectively to precision, recall, specificity, F-measure and accuracy
- `per.class`	a named matrix with precision, recall, specificity, F-measure and accuracy for each class across folds. Named rows correspond to classes, named columns correspond respectively to precision, recall, specificity, F-measure and accuracy.

See Also

`AUC.measures`, `PXR`

Examples

```
# preparing pseudo-random predictions and target-labels for examples: 100 examples
# and 10 classes
Scores <- matrix(runif(1000),nrow=100);
Targets <- Pred <- matrix(integer(1000),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(1000, sd=0.3),nrow=100);
Pred[Scores>0.5] <- 1;
colnames(Pred) <-colnames(Targets) <- LETTERS[1:10];
# getting predictions and labels of class "A"
pred <- Pred[,"A"];
```

```
labels <- Targets[,"A"];
# F.score and other metrics for a single class
F.measure.single(pred,labels);
# F.score and other metrics for the 10 classes
F.measure.single.over.classes(Targets, Pred);
```

`get.all.nodes.by.depth` *Getting nodes by their depth*

Description

Grouping classes by level in a given hierarchy.

Usage

```
get.all.nodes.by.depth(g, root = "00")
```

Arguments

`g`	graph of the classes (object of class graphNEL, package **graph**).
`root`	name of the root node (def. 00)

Details

The minimum paths between the "root" and all the other classes/nodes are computed. Levels are numbered from 1 in increasing order by their distance from the "root" class.

Value

a list of the nodes, grouped w.r.t. the distance from the root. The first element of the list corresponds to the nodes at distance 1, the second to nodes at distance 2 and so on.

`graphics` *Graphics function to plot precision/recall or f.score/recall curves*

Description

Function to plot multiple precision/recall or f.score/recall curves

Usage

```
precision.recall.curves.plot(y, range=seq(from=0, to=1, by=0.1),
curve.names=1:length(y), cex.val=0.6, f="", height=9, width=11,
col=c("black","red1","blue1","green1","darkgrey","brown1","yellow1","orange1",
"red4","blue4","green4","lightgrey","brown4","yellow4","orange4"),
line.type=1, leg=TRUE, pos=c(range[length(range)-2], range[length(range)]), do.grid=TRUE,
plot.precision=TRUE, trap.rule=TRUE)

performance.curves.plot(m, x.range=seq(from=0.1, to=1, by=0.1),
y.range=c(0,1), curve.names=1:nrow(m), cex.val=0.6, f="", height=9, width=11,
col=c("black","red1","blue1","green1","darkgrey","brown1","yellow1","orange1",
"red4","blue4","green4","lightgrey","brown4","yellow4","orange4"), line.type=1,
patch.type=1:16, leg=TRUE, pos=c(x.range[length(x.range)-2], y.range[2]), do.grid=TRUE,
x.label="Recall", y.label="Precision")
```

Arguments

`y`	a list of lists. Each component list is a list returned from `precision.at.all.recall.levels` that reports precision and recall results at different levels for different methods or tasks
`range`	numeric vector of the precision/recall values to be represented (def: values between 0 and 1 step 0.1)
`curve.names`	names of the compared methods to be reported in the legenda (def: numbers)
`cex.val`	magnification value for characters (def. 0.6)
`f`	file name. If is given, an encapsulated postscript file is created, otherwise the output is rendered on a window.
`height`	relative heigth of the graph (def. 9)
`width`	relative width of the graph (def. 11)
`col`	colors of the lines. 14 different colors are given as default, but any vector of color from colors() (package **graphics**) can be used. Colors are recycled if length(col) < length(y).
`line.type`	type of the line. Any valid vector of integer can be assigned (values between 1 and 6, see lty in par(), package **graphics** for details). Values are recycled if length(line.type) < length(y). Def.: 1 (solid lines).
`leg`	boolean: if TRUE (def.) a legend is depicted.
`pos`	coordinates of the position of the legend.
`plot.precision`	boolean: if TRUE (def.) precision/recall curves are plotted, othewise f-score/recall curves.
`trap.rule`	boolean: if TRUE (def.) the integral of the curves are computed.
`m`	a numeric matrix. Rows correspond to different methods and columns to precision or f-score at given recall values
`x.range`	vector of the recall values to be represented
`y.range`	vector with 2 elements: range of the precision/f-score to be represented

`patch.type`	numeric vector corresponding to the symbols to be plotted for different recall values (def. 1:16)
`do.grid`	a bollen value indicating whether a grid will be plotted or not (def: TRUE)
`x.label`	label of the abscissa (def: Recall)
`y.label`	label of the ordinate (def: Precision)

Details

The function `precision.recall.curves.plot` plots multiple precision/recall curves (or f-score/recall curves, if the argument `plot.precision=FALSE`) by using the output of the function `precision.at.all.recall.levels`, that compute several precison/recall pairs by moving the threshold from the lowest to the highest score achieved by each example.

The function `performance.curves.plot` plots precision of F-score/recall curves ofr a predefined set of recall levels. Thsi function can be useful to plot and to compare the average results between methods across multiple classes.

The curves can differ by color, type of line and for `performance.curves.plot` for each recall value a symbol can be also plotted. A legend can be automatically constructed.

Value

The functions output a graphic file either on a window or on an encapsulated postscript file. The function `precision.recall.curves.plot`, if the parameter `trap.rule = TRUE` (default), outputs a vector with the AUPRC (or the Area Under the F-score Curve if the parameter `plot.precision=FALSE`) for each curve.

Examples

```
# loading  an example matrix of scores an the correponding table of classes
data(T);
data(Scores);
res=list();
classes=c(1,2,7,8)
# computing precison recall values
for (j in classes) res=c(res, list(precision.at.all.recall.levels(Scores[,j], T[,j])));
# plotting precision/recall curves
precision.recall.curves.plot(res,curve.names=colnames(T)[classes],
pos=c(0.7,1), plot.precision=TRUE);
# black and white version
precision.recall.curves.plot(res,curve.names=colnames(T)[classes], pos=c(0.7,1),
plot.precision=TRUE, line.type=1:4, col=1);
# plotting f-score/recall curves
precision.recall.curves.plot(res,curve.names=colnames(T)[classes], pos=c(0.7,1),
plot.precision=FALSE);
# black and white version
precision.recall.curves.plot(res,curve.names=colnames(T)[classes], pos=c(0.7,1),
plot.precision=TRUE, line.type=1:4, col=1);
```

PXR *Precision at a given recall level measures*

Description

Set of functions to compute the precision at fixed recall levels.

Usage

```
precision.at.recall.level(scores, labels, rec.level = 0.2)
precision.at.recall.level.over.classes(target, predicted,
                          g, rec.level = 0.2, root = "00")
precision.at.multiple.recall.level(scores, labels,
           rec.levels = seq(from = 0.1, to = 1, by = 0.1))
precision.at.multiple.recall.level.over.classes(target,
   predicted, rec.levels = seq(from = 0.1, to = 1, by = 0.1))
precision.at.all.recall.levels(scores, labels, resolution=1)
```

Arguments

`scores`	vector of the predicted scores in [0,1]
`labels`	0/1 vector of the true labels
`rec.level`	rec.level: the desired recall level (def: 0.2)
`target`	matrix with the target multilabels; rows correspond to examples, columns to classes
`predicted`	matrix with the predicted multilabels; rows correspond to examples, columns to classes
`g`	graph of the classes (object of class graphNEL, package graph). If missing, no per level results are computed.
`root`	the name of the root node (def. "00") of the graph g.
`rec.levels`	a vector with the desired recall levels (def. 0.1 to 1 by 0.1 step)
`resolution`	a number between 0 and 1 (def. 1). This represents the fraction of precision, recall and f-score values returned.

Details

`precision.at.recall.level` computes the precision at a given recall level for a single class.

`precision.at.recall.level.over.classes` computes precision at a given recall level for a set of classes.

`precision.at.multiple.recall.level` computes the precision at multiple levels of recall for a single class.

`precision.at.multiple.recall.level.over.classes` computes the precision at multiple levels of recall for multiple classes.

`precision.at.all.recall.levels` compute the precision at all recall levels for a single class. It returns a pair of precision and recall values by moving a threshold from the lowest to the highest score: a number of precision and recall values equal to the number n of available examples is returned if resolution=1, otherwise a number of values equal to n * resolution.

Value

`precision.at.recall.level` returns the precision at the requested recall

`precision.at.recall.level.over.classes` a list with three elements:

- `average`	the average precision at a given recall level across classes.
- `per.level`	a named vector with average precision at a given recall level for each level of the hierarchy; names correspond to levels
- `per.class`	a named vector with precision at a given recall level for each class. Names correspond to classes

`precision.at.multiple.recall.level` a list with 2 elements:

- `precisions`	a vector with the precision at different recall levels
- `f.score`	a vector with the f-score at different recall levels

`precision.at.multiple.recall.level.over.classes`

- `PXR`	a matrix with the precisions at different recall levels: rows are classes, columns precisions at different recall levels
- `avgPXR`	a vector with the the average precisions at different recall levels across classes

`precision.at.all.recall.levels` a list with 3 elements:

- `precision`	precision at different thresholds
- `recall`	recall at different thresholds
- `f.score`	f.score at different thresholds

See Also

`AUC.measures`, `F.measures`

Examples

```
# preparing pseudo-random predictions and target-labels for examples:
# 100 examples and 10 classes
Scores <- matrix(runif(1000),nrow=100);
Targets <- matrix(integer(1000),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(1000, sd=0.3),nrow=100);
colnames(Scores) <-colnames(Targets) <- LETTERS[1:10];
# getting scores and labels of class "A"
scores <- Scores[,"A"];
labels <- Targets[,"A"];
```

```
# precsion at 0.4 recall level for class A
precision.at.recall.level(scores, labels, rec.level=0.4);
# precision at 0.4 recall level for all the 10 classes
precision.at.recall.level.over.classes(Targets, Scores, rec.level=0.4);
# precision at multiple recall levels for class A
levels <- seq(from=0.1, to=1, by=0.1);
precision.at.multiple.recall.level(scores, labels, rec.levels=levels);
# precision at multiple recall levels for all the 10 classes
precision.at.multiple.recall.level.over.classes(Targets, Scores);
# precision, recall and f-score for a single class obtained
# by moving the threshold across the examples
precision.at.all.recall.levels(scores, labels);
```

Index