Package: softclassval (via r-universe)

May 14, 2026

Type Package

Title Soft Classification Performance Measures

Description An extension of sensitivity, specificity, positive and negative predictive value to continuous predicted and reference memberships in [0, 1].

Maintainer Claudia Beleites <claudia.beleites@chemometrix.eu>

Version 1.1.0

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

Depends arrayhelpers (>= 0.76)

Imports svUnit

URL http://softclassval.r-forge.r-project.org/

Collate 'softclassval-package.R' 'make01.R' 'hardclasses.R' 'unittestdata.R' 'dev.R' 'factor2matrix.R' 'init.R' 'nsamples.R' 'postproc.R' 'operators.R' 'performance.R' 'unittests.R'

RoxygenNote 7.3.2

NeedsCompilation no

Author Claudia Beleites [aut, cre]

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

Date/Publication 2026-05-14 14:02:00 UTC

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/softclassval

RemoteSha 4253f137b4b05e9991f0083e0e0b260a010236c1

RemoteSubdir softclassval

Contents

`checkrp` *Input checks and reference preparation for performance calculation*

Description

Checks whether `r` and `p` are valid reference and predictions. If `p` is a multiple of `r`, recycles `r` to the size and shape of `p`. If `r` has additional length 1 dimensions (usually because dimensions were dropped from `p`), it is shortend to the shape of `p`.

Usage

```
checkrp(r, p)
```

Arguments

`r`	reference
`p`	prediction

Details

In addition, any `NA`s in `p` are transferred to `r` so that these samples are excluded from counting in `nsamples`.

`checkrp` is automatically called by the performance functions, but doing so beforehand and then setting `.checked = TRUE` can save time when several performance measures are to be calculated on the same results.

Value

`r`, possibly recycled to length of `p` or with dimensions shortened to `p`.

Author(s)

Claudia Beleites

Examples

```
ref <- softclassval:::ref
ref

pred <- softclassval:::pred
pred

ref <- checkrp (r = ref, p = pred)
sens (r = ref, p = pred, .checked = TRUE)
```

---

confusion *Performance calculation for soft classification*

Description

These performance measures can be used with prediction and reference being continuous class memberships in [0, 1].

Usage

```
confusion(
  r = stop("missing reference"),
  p = stop("missing prediction"),
  groups = NULL,
  operator = "prd",
  drop = FALSE,
  .checked = FALSE
)

confmat(r = stop("missing reference"), p = stop("missing prediction"), ...)

sens(
  r = stop("missing reference"),
  p = stop("missing prediction"),
  groups = NULL,
  operator = "prd",
  op.dev = dev(match.fun(operator)),
  op.postproc = postproc(match.fun(operator)),
  eps = 1e-08,
  drop = FALSE,
  .checked = FALSE
)

spec(r = stop("missing reference"), p = stop("missing prediction"), ...)

ppv(
  r = stop("missing reference"),
```

```
  p = stop("missing prediction"),
  ...,
  .checked = FALSE
)

npv(
  r = stop("missing reference"),
  p = stop("missing prediction"),
  ...,
  .checked = FALSE
)
```

Arguments

r	vector, matrix, or array with reference.
p	vector, matrix, or array with predictions
groups	grouping variable for the averaging by rowsum. If NULL, all samples (rows) are averaged.
operator	the operators to be used
drop	should the results possibly be returned as vector instead of 1d array? (Note that levels of groups are never dropped, you need to do that e.g. by factor.)
.checked	for internal use: the inputs are guaranteed to be of same size and shape. If TRUE, confusion omits input checking
...	handed to sens
op.dev	does the operator measure deviation?
op.postproc	if a post-processing function is needed after averaging, it can be given here. See the example.
eps	limit below which denominator is considered 0

Details

The rows of r and p are considered the samples, columns will usually hold the classes, and further dimensions are preserved but ignored.

r must have the same number of rows and columns as p, all other dimensions may be filled by recycling.

spec, ppv, and npv use the symmetry between the performance measures as described in the article and call sens.

Value

numeric of size (ngroups x dim(p)[-1]) with the respective performance measure

Author(s)

Claudia Beleites

References

see the literature in `citation ("softclassval")`

See Also

Operators: `prd`

For the complete confusion matrix, `confmat`

Examples

```
ref <- softclassval:::ref
ref

pred <- softclassval:::pred
pred

## Single elements or diagonal of confusion matrix
confusion (r = ref, p = pred)

## complete confusion matrix
cm <- confmat (r = softclassval:::ref, p = pred) [1,,]
cm

## Sensitivity-Specificity matrix:
cm / rowSums (cm)

## Matrix with predictive values:
cm / rep (colSums (cm), each = nrow (cm))

## sensitivities
sens (r = ref, p = pred)

## specificities
spec (r = ref, p = pred)

## predictive values
ppv (r = ref, p = pred)
npv (r = ref, p = pred)
```

---

dev *Mark operator as deviation measure*

Description

The operators measure either a performance (i.e. accordance between reference and prediction) or a deviation. `dev (op) == TRUE` marks operators measuring deviation.

Usage

```
dev(op)

dev (op) <- value
```

Arguments

op	the operator (function)
value	logical indicating the operator type

Value

logical indicating the type of operator. NULL if the attribute is missing.

Author(s)

Claudia Beleites

See Also

sens postproc

Examples

```
dev (wRMSE)
myop <- function (r, p) p * (r == 1)
dev (myop) <- TRUE
```

---

factor2matrix *Convert hard class labels to membership matrix*

Description

Converts a factor with hard class memberships into a membership matrix

Usage

```
factor2matrix(f)
```

Arguments

f	factor with class labels

Value

matrix of size length (f) x nlevels (f)

Author(s)

Claudia Beleites

See Also

hardclasses for the inverse

`hard` *Mark operator as hard measure*

Description

The operators may work only for hard classes (see and). `hard (op) == TRUE` marks hard operators.

Usage

```
hard(op)

hard (op) <- value
```

Arguments

`op`	the operator (function)
`value`	logical indicating the operator type

Value

logical indicating the type of operator. `NULL` if the attribute is missing.

Author(s)

Claudia Beleites

See Also

sens and

Examples

```
hard (and)
myop <- function (r, p) p * (r == 1)
hard (myop) <- TRUE
```

`hardclasses` *Convert to hard class labels*

Description

`hardclasses` converts the soft class labels in `x` into a factor with hard class memberships and `NA` for soft samples.

Usage

```
hardclasses(x, classdim = 2L, soft.name = NA, tol = 1e-05, drop = TRUE)

harden(x, classdim = 2L, tol = 1e-06, closed = TRUE)
```

Arguments

`x`	matrix or array holding the class memberships
`classdim`	dimension that holds the classes, default columns
`soft.name`	level for soft samples
`tol`	tolerance: samples with membership >= 1 - tol are considered to be hard samples of the respective class.
`drop`	see `drop1d`
`closed`	logical indicating whether the system should be treated as closed-world (i.e. all memberships add to 1)

Details

`harden` hardens the soft

Value

factor array of shape `dim(x)[-classdim]`

Author(s)

Claudia Beleites

See Also

`factor2matrix` for the inverse

Examples

```
softclassval:::pred
harden (softclassval:::pred)
harden (softclassval:::pred, closed = FALSE)

## classical threshold at 0.5
harden (softclassval:::pred, tol = 0.5)

## grey zone: NA for memberships between 0.25 and 0.75
harden (softclassval:::pred, tol = 0.25)

## threshold at 0.7 = 0.5 + 0.2:
harden (softclassval:::pred - 0.2, tol = 0.5)
harden (softclassval:::pred - 0.2, tol = 0.5, closed = FALSE)
```

---

nsamples *Number of samples*

Description

Count number of samples

Usage

```
nsamples(r = r, groups = NULL, operator = "prd", hard.operator)
```

Arguments

r	reference class labels with samples in rows.
groups	grouping variable for the averaging by rowsum. If NULL, all samples (rows) are averaged.
operator	the operator to be used
hard.operator	optional: a logical determining whether only hard samples should be counted

Details

Basically, the reference is summed up. For hard operators, the reference is hardened first: soft values, i.e. r in (0, 1) are set to NA.

Value

number of samples in each group (rows) for each class (columns) and all further dimensions of ref.

Author(s)

Claudia Beleites

Examples

```
ref <- softclassval:::ref
ref
nsamples (ref)
nsamples (ref, hard.operator = TRUE)
```

---

postproc *Attach postprocessing function to operator*

Description

The postprocessing function is applied during performance calculation after averaging but before dev is applied. This is the place where the root is taken of root mean squared errors.

Usage

```
postproc(op)

postproc (op) <- value
```

Arguments

op	the operator (function)
value	function (or its name or symbol) to do the post-processing. NULL deletes the postprocessing function.

Details

postproc (op) retrieves the postprocessing function (or NULL if none is attached)

Value

logical indicating the type of operator. NA if the attribute is missing.

Author(s)

Claudia Beleites

See Also

sens dev

Examples

```
postproc (wRMSE)
myop <- function (r, p) p * (r == 1)
postproc (myop) <- `sqrt`
```

softclassval.unittest *Run the unit tests*

Description

Run the unit tests attached to the functions via svUnit

Usage

```
softclassval.unittest()
```

Value

invisibly TRUE if the tests pass, NA if svUnit is not available. Stops if errors are encountered.

Author(s)

Claudia Beleites

See Also

svUnit

strong *And (conjunction) operators*

Description

And operators for the soft performance calculation. The predefined operators are:

Name	Definition	dev?	postproc?	hard?	Explanation
gdl	pmin (r, p)	FALSE		FALSE	the Gödel-operator (weak conjunction)
luk	pmax (r + p - 1, 0)	FALSE		FALSE	Łukasiewicz-operator (strong conjunction)
prd	r * p	FALSE		FALSE	product operator
and	r * p	FALSE		TRUE	Boolean conjunction: accepts only 0 or 1, otherwise yiel
wMAE	r * abs (r - p)	TRUE		FALSE	for weighted mean absolute error
wRMAE	r * abs (r - p)	TRUE	sqrt	FALSE	for weighted root mean absolute error (bound for RMSE)
##' wMSE	r * (r - p)^2	TRUE		FALSE	for weighted mean squared error
wRMSE	r * (r - p)^2	TRUE	sqrt	FALSE	for root weighted mean squared error

Usage

```
strong(r, p)

luk(r, p)

weak(r, p)

gdl(r, p)

prd(r, p)

and(r, p)

wMAE(r, p)

wRMAE(r, p)

wMSE(r, p)

wRMSE(r, p)
```

Arguments

r	reference vector, matrix, or array with numeric values in [0, 1], for and in {0, 1}
p	prediction vector, matrix, or array with numeric values in [0, 1], for and in {0, 1}

Value

numeric of the same size as p

Author(s)

Claudia Beleites

References

see the literature in `citation("softclassval")`

See Also

Performance measures: sens

Examples

```
ops <- c ("luk", "gdl", "prd", "and", "wMAE", "wRMAE", "wMSE", "wRMSE")

## make a nice table
```

```
lastline <- function (f){
  body <- body (get (f))    ## function body
  body <- deparse (body)
  body [length (body) - 1]  ## last line is closing brace
}

data.frame (source = sapply (ops, lastline),
            dev = sapply (ops, function (f) dev (get (f))),
            hard = sapply (ops, function (f) hard (get (f))),
            postproc = I (lapply (ops, function (f) postproc (get (f))))
            )

x <- softclassval:::v
x

luk (0.7, 0.8)

## The behaviour of the operators
## op (x, 1)
cbind (x, sapply (c ("luk", "gdl", "prd", "wMAE", "wRMAE", "wMSE", "wRMSE"),
                  function (op, x) get (op) (x, 1), x))

## op (x, 0)
cbind (x, sapply (c ("luk", "gdl", "prd", "wMAE", "wRMAE", "wMSE", "wRMSE"),
                  function (op, x) get (op) (x, 0), x))

## op (x, x)
cbind (x, sapply (c ("luk", "gdl", "prd", "wMAE", "wRMAE", "wMSE", "wRMSE"),
                  function (op, x) get (op) (x, x), x))


## Note that the deviation operators are not commutative
## (due to the weighting by reference)
zapsmall (
cbind (sapply (c ("luk", "gdl", "prd", "wMAE", "wRMAE", "wMSE", "wRMSE"),
                  function (op, x) get (op) (1, x), x)) -
cbind (sapply (c ("luk", "gdl", "prd", "wMAE", "wRMAE", "wMSE", "wRMSE"),
                  function (op, x) get (op) (x, 1), x))
)
```

Index