Package 'mlr3measures'

May 9, 2026

Title Performance Measures for 'mlr3'

Version 1.3.0

Description Implements multiple performance measures for supervised learning. Includes over 40 measures for regression and classification. Additionally, meta information about the performance measures can be queried, e.g. what the best and worst possible performances scores are.

License LGPL-3

URL https://mlr3measures.mlr-org.com, https://github.com/mlr-org/mlr3measures

BugReports https://github.com/mlr-org/mlr3measures/issues

Depends R (>= 3.3.0)

Imports checkmate, mlr3misc, PRROC

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'assertions.R' 'bibentries.R' 'measures.R' 'binary_auc.R' 'binary_bbrier.R' 'binary_dor.R' 'binary_fbeta.R' 'binary_fdr.R' 'binary_fn.R' 'binary_fnr.R' 'binary_fomr.R' 'binary_fp.R' 'binary_fpr.R' 'binary_gmean.R' 'binary_gpr.R' 'binary_npv.R' 'binary_ppv.R' 'binary_prauc.R' 'binary_tn.R' 'binary_tnr.R' 'binary_tp.R' 'binary_tpr.R' 'classif_acc.R' 'classif_auc.R' 'classif_bacc.R' 'classif_ce.R' 'classif_logloss.R' 'classif_mbrier.R' 'classif_mcc.R' 'classif_zero_one.R' 'confusion_matrix.R' 'helper.R' 'regr_ae.R' 'regr_ape.R' 'regr_bias.R' 'regr_ktau.R' 'regr_linex.R' 'regr_mae.R' 'regr_mape.R' 'regr_maxae.R' 'regr_maxse.R' 'regr_medae.R' 'regr_medse.R' 'regr_mse.R' 'regr_msle.R' 'regr_pbias.R' 'regr_pinball.R' 'regr_rae.R' 'regr_rmse.R' 'regr_rmsle.R' 'regr_rrse.R' 'regr_rse.R'

'regr_rsq.R' 'regr_sae.R' 'regr_se.R' 'regr_sle.R'
'regr_smape.R' 'regr_srho.R' 'regr_sse.R' 'roxygen.R'
'similarity_jaccard.R' 'similarity_phi.R' 'zzz.R'

NeedsCompilation no

Author Michel Lang [aut] (ORCID: <https://orcid.org/0000-0001-9754-0393>),
Martin Binder [ctb],
Marc Becker [cre, aut] (ORCID: <https://orcid.org/0000-0002-8115-0400>),
Lona Koers [aut]

Maintainer Marc Becker <marcbecker@posteo.de>

Repository CRAN

Date/Publication 2026-04-17 13:20:02 UTC

Contents

mlr3measures-package . . . 3
acc . . . 4
ae . . . 5
ape . . . 6
auc . . . 7
bacc . . . 9
bbrier . . . 10
bias . . . 12
ce . . . 13
confusion_matrix . . . 14
dor . . . 15
fbeta . . . 17
fdr . . . 19
fn . . . 20
fnr . . . 22
fomr . . . 23
fp . . . 25
fpr . . . 26
gmean . . . 27
gpr . . . 29
jaccard . . . 30
ktau . . . 32
linex . . . 33
logloss . . . 34
mae . . . 36
mape . . . 37
mauc_aunu . . . 38
maxae . . . 40
maxse . . . 41
mbrier . . . 42
mcc . . . 44
measures . . . 46

medae . . . 47
medse . . . 48
mse . . . 49
msle . . . 50
npv . . . 51
pbias . . . 53
phi . . . 54
pinball . . . 56
ppv . . . 57
prauc . . . 59
rae . . . 60
rmse . . . 61
rmsle . . . 62
rrse . . . 64
rse . . . 65
rsq . . . 66
sae . . . 67
se . . . 69
sle . . . 70
smape . . . 71
srho . . . 72
sse . . . 73
tn . . . 74
tnr . . . 76
tp . . . 77
tpr . . . 79
zero_one . . . 81

Index **82**

`mlr3measures-package` *mlr3measures: Performance Measures for 'mlr3'*

Description

Implements multiple performance measures for supervised learning. Includes over 40 measures for regression and classification. Additionally, meta information about the performance measures can be queried, e.g. what the best and worst possible performances scores are.

Author(s)

Maintainer: Marc Becker <marcbecker@posteo.de> (ORCID)

Authors:

- Michel Lang <michellang@gmail.com> (ORCID)
- Lona Koers

Other contributors:

- Martin Binder <mlr.developer@mb706.com> [contributor]

See Also

Useful links:

- https://mlr3measures.mlr-org.com
- https://github.com/mlr-org/mlr3measures
- Report bugs at https://github.com/mlr-org/mlr3measures/issues

---

acc *Classification Accuracy*

Description

Measure to compare true observed labels with predicted labels in multiclass classification tasks.

Usage

```
acc(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the same levels and length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the same levels and length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Classification Accuracy is defined as

$$\sum_{i=1}^{n} w_i \mathbf{1}\left(t_i = r_i\right),$$

where w_i are weights normalized to sum to 1 for all observations x_i.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"classif"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

See Also

Other Classification Measures: `bacc()`, `ce()`, `logloss()`, `mauc_aunu()`, `mbrier()`, `mcc()`, `zero_one()`

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
acc(truth, response)
```

---

ae *Absolute Error (per observation)*

Description

Measure to compare true observed response with predicted response in regression tasks.

Note that this is an unaggregated measure, returning the losses per observation.

Usage

```
ae(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

Calculates the per-observation absolute error as

$$|t_i - r_i| .$$

Value

Performance value as `numeric(length(truth))`.

Meta Information

- Type: `"regr"`
- Range (per observation): $[0, \infty)$
- Minimize (per observation): `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

---

ape *Absolute Percentage Error (per observation)*

Description

Measure to compare true observed response with predicted response in regression tasks.

Note that this is an unaggregated measure, returning the losses per observation.

Usage

```
ape(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

Calculates the per-observation absolute percentage error as

$$\left| \frac{t_i - r_i}{t_i} \right|.$$

Value

Performance value as `numeric(length(truth))`.

Meta Information

- Type: "regr"
- Range (per observation): $[0, \infty)$
- Minimize (per observation): TRUE
- Required prediction: response

See Also

Other Regression Measures: ae(), bias(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

---

auc *Area Under the ROC Curve*

Description

Measure to compare true observed labels with predicted probabilities in binary classification tasks.

Usage

```
auc(truth, prob, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
prob	(numeric()) Predicted probability for positive class. Must have exactly same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

Computes the area under the Receiver Operator Characteristic (ROC) curve. The AUC can be interpreted as the probability that a randomly chosen positive observation has a higher predicted probability than a randomly chosen negative observation.

For n^+ positive and n^- negative observations with R_i^+ the rank of the i-th positive observation's predicted probability (average ranks for ties), the AUC is estimated as

$$\widehat{\text{AUC}} = \frac{1}{n^+ n^-} \left(\sum_{i=1}^{n^+} R_i^+ \; - \; \frac{n^+(n^+ + 1)}{2} \right).$$

If `sample_weights` are provided, let w_i^+ be the weight of the i-th positive observation with predicted probability p_i^+, $W^+ = \sum_i w_i^+$, and W^- the total weight of the negative observations. Define the weighted Mann-Whitney contribution of positive observation i as $U_i^w = W^-_{<p_i^+} + \frac{1}{2} W^-_{=p_i^+}$, i.e. the total weight of negative observations with a smaller predicted probability plus half the weight of negatives tied with p_i^+. The weighted AUC is then

$$\widehat{\text{AUC}}_w = \frac{1}{W^+ W^-} \sum_{i=1}^{n^+} w_i^+ U_i^w.$$

This measure is undefined if the true values are either all positive or all negative.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `prob`

References

Youden WJ (1950). "Index for rating diagnostic tests." *Cancer*, **3**(1), 32–35. doi:10.1002/1097-0142(1950)3:1<32::aidcncr2820030106>3.0.co;23.

See Also

Other Binary Classification Measures: `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `fpr()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tnr()`, `tp()`, `tpr()`

Examples

```
truth = factor(c("a", "a", "a", "b"))
prob = c(.6, .7, .1, .4)
auc(truth, prob, "a")
```

bacc *Balanced Accuracy*

Description

Measure to compare true observed labels with predicted labels in multiclass classification tasks.

Usage

```
bacc(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the same levels and length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the same levels and length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Balanced Accuracy computes the weighted balanced accuracy, suitable for imbalanced data sets. It is defined analogously to the definition in sklearn.

First, all sample weights w_i are normalized per class so that each class has the same influence:

$$\hat{w}_i = \frac{w_i}{\sum_{j=1}^{n} w_j \cdot \mathbf{1}(t_j = t_i)}.$$

The Balanced Accuracy is then calculated as

$$\frac{1}{\sum_{i=1}^{n} \hat{w}_i} \sum_{i=1}^{n} \hat{w}_i \cdot \mathbf{1}(r_i = t_i).$$

This definition is equivalent to `acc()` with class-balanced sample weights.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: "classif"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: response

References

Brodersen KH, Ong CS, Stephan KE, Buhmann JM (2010). "The Balanced Accuracy and Its Posterior Distribution." In *2010 20th International Conference on Pattern Recognition*. doi:10.1109/icpr.2010.764.

Guyon I, Bennett K, Cawley G, Escalante HJ, Escalera S, Ho TK, Macia N, Ray B, Saeed M, Statnikov A, Viegas E (2015). "Design of the 2015 ChaLearn AutoML challenge." In *2015 International Joint Conference on Neural Networks (IJCNN)*. doi:10.1109/ijcnn.2015.7280767.

See Also

Other Classification Measures: acc(), ce(), logloss(), mauc_aunu(), mbrier(), mcc(), zero_one()

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
bacc(truth, response)
```

---

bbrier *Binary Brier Score*

Description

Measure to compare true observed labels with predicted probabilities in binary classification tasks.

Usage

```
bbrier(truth, prob, positive, sample_weights = NULL, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
prob	(numeric()) Predicted probability for positive class. Must have exactly same length as truth.

positive (character(1))
Name of the positive class.

sample_weights (numeric())
Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

... (any)
Additional arguments. Currently ignored.

Details

The Binary Brier Score is defined as

$$\frac{1}{n}\sum_{i=1}^{n} w_i(I_i - p_i)^2,$$

where w_i are the sample weights, and I_i is 1 if observation x_i belongs to the positive class, and 0 otherwise.

Note that this (more common) definition of the Brier score is equivalent to the original definition of the multi-class Brier score (see mbrier()) divided by 2.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: TRUE
- Required prediction: prob

References

https://en.wikipedia.org/wiki/Brier_score

Brier GW (1950). "Verification of forecasts expressed in terms of probability." *Monthly Weather Review*, **78**(1), 1–3. doi:10.1175/15200493(1950)078<0001:vofeit>2.0.co;2.

See Also

Other Binary Classification Measures: auc(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
prob = runif(10)
bbrier(truth, prob, positive = "a")
```

---

bias *Bias*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
bias(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Bias is defined as

$$\frac{1}{n} \sum_{i=1}^{n} w_i \left(r_i - t_i \right),$$

where w_i are normalized sample weights. Good predictions score close to 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $(-\infty, \infty)$
- Minimize: `NA`
- Required prediction: `response`

See Also

Other Regression Measures: ae(), ape(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
bias(truth, response)
```

---

ce *Classification Error*

Description

Measure to compare true observed labels with predicted labels in multiclass classification tasks.

Usage

```
ce(truth, response, sample_weights = NULL, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the same levels and length as response.
response	(factor()) Predicted response labels. Must have the same levels and length as truth.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
...	(any) Additional arguments. Currently ignored.

Details

The Classification Error is defined as

$$\frac{1}{n} \sum_{i=1}^{n} w_i \mathbf{1} \left(t_i \neq r_i \right),$$

where w_i are normalized weights for each observation x_i.

Value

Performance value as numeric(1).

Meta Information

- Type: "classif"
- Range: $[0, 1]$
- Minimize: TRUE
- Required prediction: response

See Also

Other Classification Measures: acc(), bacc(), logloss(), mauc_aunu(), mbrier(), mcc(), zero_one()

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
ce(truth, response)
```

---

confusion_matrix *Calculate Binary Confusion Matrix*

Description

Calculates the confusion matrix for a binary classification problem once and then calculates all binary confusion measures of this package.

Usage

```
confusion_matrix(truth, response, positive, na_value = NaN, relative = FALSE)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
relative	(logical(1)) If TRUE, the returned confusion matrix contains relative frequencies instead of absolute frequencies.

Details

The binary confusion matrix is defined as

$$\begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix}.$$

If `relative = TRUE`, all values are divided by n.

Value

List with two elements:

- `matrix` stores the calculated confusion matrix.
- `measures` stores the metrics as named numeric vector.

Examples

```
set.seed(123)
lvls = c("a", "b")
truth = factor(sample(lvls, 20, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 20, replace = TRUE), levels = lvls)

confusion_matrix(truth, response, positive = "a")
confusion_matrix(truth, response, positive = "a", relative = TRUE)
confusion_matrix(truth, response, positive = "b")
```

---

dor *Diagnostic Odds Ratio*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
dor(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the exactly same two levels and the same length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the exactly same two levels and the same length as `truth`.
`positive`	(`character(1)`) Name of the positive class.

`sample_weights` (`numeric()`)
Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

`na_value` (`numeric(1)`)
Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.

`...` (`any`)
Additional arguments. Currently ignored.

Details

The Diagnostic Odds Ratio is defined as

$$\frac{\mathrm{TP}/\mathrm{FP}}{\mathrm{FN}/\mathrm{TN}}.$$

This measure is undefined if FP = 0 or FN = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, \infty)$
- Minimize: `FALSE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `fpr()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tnr()`, `tp()`, `tpr()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
dor(truth, response, positive = "a")
```

`fbeta` *F-beta Score*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fbeta(
  truth,
  response,
  positive,
  sample_weights = NULL,
  beta = 1,
  na_value = NaN,
  ...
)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the exactly same two levels and the same length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the exactly same two levels and the same length as `truth`.
`positive`	(`character(1)`) Name of the positive class.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`beta`	(`numeric(1)`) Parameter to give either precision or recall more weight. Default is `1`, resulting in balanced weights.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

With P as precision() and R as recall(), the F-beta Score is defined as

$$(1 + \beta^2) \frac{P \cdot R}{(\beta^2 P) + R}.$$

It measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision. For $\beta = 1$, this measure is called "F1" score.

This measure is undefined if precision or recall is undefined, i.e. TP + FP = 0 or TP + FN = 0.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: response

References

Rijsbergen, Van CJ (1979). *Information Retrieval*, 2nd edition. Butterworth-Heinemann, Newton, MA, USA. ISBN 408709294.

Goutte C, Gaussier E (2005). "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation." In *Lecture Notes in Computer Science*, 345–359. doi:10.1007/9783-540318651_25.

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fbeta(truth, response, positive = "a")
```

`fdr` *False Discovery Rate*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fdr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the exactly same two levels and the same length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the exactly same two levels and the same length as `truth`.
`positive`	(`character(1)`) Name of the positive class.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The False Discovery Rate is defined as

$$\frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}}.$$

This measure is undefined if TP + FP = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: TRUE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fdr(truth, response, positive = "a")
```

---

fn *False Negatives*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fn(truth, response, positive, sample_weights = NULL, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.

sample_weights (numeric())
Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

... (any)
Additional arguments. Currently ignored.

Details

This measure counts the false negatives (type 2 error), i.e. the number of predictions indicating a negative class label while in fact it is positive. This is sometimes also called a "miss" or an "underestimation".

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, \infty)$
- Minimize: TRUE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fn(truth, response, positive = "a")
```

fnr *False Negative Rate*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fnr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The False Negative Rate is defined as

$$\frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}.$$

Also know as "miss rate".

This measure is undefined if TP + FN = 0.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: TRUE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fnr(truth, response, positive = "a")
```

---

fomr *False Omission Rate*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fomr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.

`sample_weights` (`numeric()`)
Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

`na_value` (`numeric(1)`)
Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.

`...` (any)
Additional arguments. Currently ignored.

Details

The False Omission Rate is defined as

$$\frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TN}}.$$

This measure is undefined if FN + TN = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, 1]$
- Minimize: `TRUE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fp()`, `fpr()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tnr()`, `tp()`, `tpr()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fomr(truth, response, positive = "a")
```

fp *False Positives*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fp(truth, response, positive, sample_weights = NULL, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
...	(any) Additional arguments. Currently ignored.

Details

This measure counts the false positives (type 1 error), i.e. the number of predictions indicating a positive class label while in fact it is negative. This is sometimes also called a "false alarm".

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, \infty)$
- Minimize: TRUE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fp(truth, response, positive = "a")
```

---

fpr *False Positive Rate*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
fpr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The False Positive Rate is defined as

$$\frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}.$$

Also know as fall out or probability of false alarm.

This measure is undefined if FP + TN = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, 1]$
- Minimize: `TRUE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tnr()`, `tp()`, `tpr()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
fpr(truth, response, positive = "a")
```

---

gmean *Geometric Mean of Recall and Specificity*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
gmean(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

Calculates the geometric mean of recall() R and specificity() S as

$$\sqrt{\mathrm{R} \cdot \mathrm{S}}.$$

This measure is undefined if recall or specificity is undefined, i.e. if TP + FN = 0 or if FP + TN = 0.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: response

References

He H, Garcia EA (2009). "Learning from Imbalanced Data." *IEEE Transactions on knowledge and data engineering*, **21**(9), 1263–1284. doi:10.1109/TKDE.2008.239.

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
gmean(truth, response, positive = "a")
```

---

gpr *Geometric Mean of Precision and Recall*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
gpr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the exactly same two levels and the same length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the exactly same two levels and the same length as `truth`.
`positive`	(`character(1)`) Name of the positive class.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

Calculates the geometric mean of `precision()` P and `recall()` R as

$$\sqrt{\mathrm{P} \cdot \mathrm{R}}.$$

This measure is undefined if precision or recall is undefined, i.e. if TP + FP = 0 or if TP + FN = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

References

He H, Garcia EA (2009). "Learning from Imbalanced Data." *IEEE Transactions on knowledge and data engineering*, **21**(9), 1263–1284. doi:10.1109/TKDE.2008.239.

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `fpr()`, `gmean()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tnr()`, `tp()`, `tpr()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
gpr(truth, response, positive = "a")
```

---

jaccard *Jaccard Similarity Index*

Description

Measure to compare two or more sets w.r.t. their similarity.

Usage

```
jaccard(sets, na_value = NaN, ...)
```

Arguments

`sets`	(`list()`) List of character or integer vectors. `sets` must have at least 2 elements.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

For two sets A and B, the Jaccard Index is defined as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

If more than two sets are provided, the mean of all pairwise scores is calculated.

This measure is undefined if two or more sets are empty.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"similarity"`
- Range: $[0, 1]$
- Minimize: `FALSE`

References

Jaccard, Paul (1901). "Étude comparative de la distribution florale dans une portion des Alpes et du Jura." *Bulletin de la Société Vaudoise des Sciences Naturelles*, **37**, 547-579. doi:10.5169/SEALS-266450.

Bommert A, Rahnenführer J, Lang M (2017). "A Multicriteria Approach to Find Predictive and Sparse Models with Stable Feature Selection for High-Dimensional Data." *Computational and Mathematical Methods in Medicine*, **2017**, 1–18. doi:10.1155/2017/7907163.

Bommert A, Lang M (2021). "stabm: Stability Measures for Feature Selection." *Journal of Open Source Software*, **6**(59), 3010. doi:10.21105/joss.03010.

See Also

Package **stabm** which implements many more stability measures with included correction for chance.

Other Similarity Measures: `phi()`

Examples

```
set.seed(1)
sets = list(
  sample(letters[1:3], 1),
  sample(letters[1:3], 2)
)
jaccard(sets)
```

ktau *Kendall's tau*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
ktau(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

Kendall's tau is defined as Kendall's rank correlation coefficient between truth and response. It is defined as

$$\tau = \frac{(\text{numberofconcordantpairs}) - (\text{numberofdiscordantpairs})}{(\text{numberofpairs})}$$

Calls `stats::cor()` with `method` set to `"kendall"`.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[-1, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

References

Rosset S, Perlich C, Zadrozny B (2006). "Ranking-based evaluation of regression models." *Knowledge and Information Systems*, **12**(3), 331–353. doi:10.1007/s1011500600373.

See Also

Other Regression Measures: ae(), ape(), bias(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
ktau(truth, response)
```

---

linex *Linear-Exponential Loss (per observation)*

Description

Measure to compare true observed response with predicted response in regression tasks.

Note that this is an unaggregated measure, returning the losses per observation.

Usage

```
linex(truth, response, a = -1, b = 1, ...)
```

Arguments

truth	(numeric()) True (observed) values. Must have the same length as response.
response	(numeric()) Predicted response values. Must have the same length as truth.
a	(numeric(1)) Shape parameter controlling asymmetry. Negative values penalize overestimation more, positive values penalize underestimation more. As a approaches 0, the loss resembles squared error loss. Default is -1.
b	(numeric(1)) Positive scaling factor for the loss. Larger values increase the loss magnitude. Default is 1.
...	(any) Additional arguments. Currently ignored.

Details

The Linear-Exponential Loss is defined as

$$b(\exp(t_i - r_i) - a(t_i - r_i) - 1),$$

where $a \neq 0, b > 0$.

Value

Performance value as `numeric(length(truth))`.

Meta Information

- Type: `"regr"`
- Range (per observation): $[0, \infty)$
- Minimize (per observation): `TRUE`
- Required prediction: `response`

References

Varian, R. H (1975). "A Bayesian Approach to Real Estate Assessment." In Fienberg SE, Zellner A (eds.), *Studies in Bayesian Econometrics and Statistics: In Honor of Leonard J. Savage*, 195–208. North-Holland, Amsterdam.

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
linex(truth, response)
```

---

logloss *Log Loss*

Description

Measure to compare true observed labels with predicted probabilities in multiclass classification tasks.

Usage

```
logloss(truth, prob, sample_weights = NULL, eps = 1e-15, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the same levels and length as response.
prob	(matrix()) Matrix of predicted probabilities, each column is a vector of probabilities for a specific class label. Columns must be named with levels of truth.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
eps	(numeric(1)) Probabilities are clipped to max(eps, min(1 - eps, p)). Otherwise the measure would be undefined for probabilities p = 0 and p = 1.
...	(any) Additional arguments. Currently ignored.

Details

The Log Loss (a.k.a Bernoulli Loss, Logistic Loss, Cross-Entropy Loss) is defined as

$$-\frac{1}{n}\sum_{i=1}^{n} w_i \log\left(p_i\right)$$

where p_i is the probability for the true class of observation i, and w_i are normalized weights for each observation x_i.

Value

Performance value as numeric(1).

Meta Information

- Type: "classif"
- Range: $[0, \infty)$
- Minimize: TRUE
- Required prediction: prob

See Also

Other Classification Measures: acc(), bacc(), ce(), mauc_aunu(), mbrier(), mcc(), zero_one()

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
prob = matrix(runif(3 * 10), ncol = 3, dimnames = list(NULL, lvls))
prob = t(apply(prob, 1, function(x) x / sum(x)))
logloss(truth, prob)
```

mae *Mean Absolute Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
mae(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Mean Absolute Error is defined as

$$\frac{1}{n} \sum_{i=1}^{n} w_i \left| t_i - r_i \right|,$$

where w_i are normalized sample weights.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: ae(), ape(), bias(), ktau(), linex(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
mae(truth, response)
```

---

mape *Mean Absolute Percent Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
mape(truth, response, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(numeric()) True (observed) values. Must have the same length as response.
response	(numeric()) Predicted response values. Must have the same length as truth.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The Mean Absolute Percent Error is defined as

$$\frac{1}{n} \sum_{i=1}^{n} w_i \left| \frac{t_i - r_i}{t_i} \right|,$$

where w_i are normalized sample weights.

This measure is undefined if any element of t is 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

References

de Myttenaere, Arnaud, Golden, Boris, Le Grand, Bénédicte, Rossi, Fabrice (2016). "Mean Absolute Percentage Error for regression models." *Neurocomputing*, **192**, 38-48. ISSN 0925-2312, doi:10.1016/j.neucom.2015.12.114.

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
mape(truth, response)
```

---

mauc_aunu *Multiclass AUC Scores*

Description

Measure to compare true observed labels with predicted probabilities in multiclass classification tasks.

Usage

```
mauc_aunu(truth, prob, na_value = NaN, ...)

mauc_aunp(truth, prob, na_value = NaN, ...)

mauc_au1u(truth, prob, na_value = NaN, ...)

mauc_au1p(truth, prob, na_value = NaN, ...)

mauc_mu(truth, prob, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the same levels and length as response.
prob	(matrix()) Matrix of predicted probabilities, each column is a vector of probabilities for a specific class label. Columns must be named with levels of truth.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

Multiclass AUC measures.

- *AUNU*: AUC of each class against the rest, using the uniform class distribution. Computes the AUC treating a c-dimensional classifier as c two-dimensional 1-vs-rest classifiers, where classes are assumed to have uniform distribution, in order to have a measure which is independent of class distribution change (Fawcett 2001).
- *AUNP*: AUC of each class against the rest, using the a-priori class distribution. Computes the AUC treating a c-dimensional classifier as c two-dimensional 1-vs-rest classifiers, taking into account the prior probability of each class (Fawcett 2001).
- *AU1U*: AUC of each class against each other, using the uniform class distribution. Computes something like the AUC of c(c - 1) binary classifiers (all possible pairwise combinations). See Hand (2001) for details.
- *AU1P*: AUC of each class against each other, using the a-priori class distribution. Computes something like AUC of c(c - 1) binary classifiers while considering the a-priori distribution of the classes as suggested in Ferri (2009). Note we deviate from the definition in Ferri (2009) by a factor of c.
- *MU*: Multiclass AUC as defined in Kleinman and Page (2019). This measure is an average of the pairwise AUCs between all classes. The measure was tested against the Python implementation by Ross Kleinman.

Value

Performance value as numeric(1).

Meta Information

- Type: "classif"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: prob

References

Fawcett, Tom (2001). "Using rule sets to maximize ROC performance." In *Proceedings 2001 IEEE international conference on data mining*, 131–138. IEEE.

Ferri, César, Hernández-Orallo, José, Modroiu, R (2009). "An experimental comparison of performance measures for classification." *Pattern Recognition Letters*, **30**(1), 27–38. doi:10.1016/j.patrec.2008.08.010.

Hand, J D, Till, J R (2001). "A simple generalisation of the area under the ROC curve for multiple class classification problems." *Machine learning*, **45**(2), 171–186.

Kleiman R, Page D (2019). "AUC mu: A Performance Metric for Multi-Class Machine Learning Models." In Chaudhuri, Kamalika, Salakhutdinov, Ruslan (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 series Proceedings of Machine Learning Research, 3439–3447. PMLR.

See Also

Other Classification Measures: `acc()`, `bacc()`, `ce()`, `logloss()`, `mbrier()`, `mcc()`, `zero_one()`

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
prob = matrix(runif(3 * 10), ncol = 3)
colnames(prob) = levels(truth)
mauc_aunu(truth, prob)
```

---

maxae *Max Absolute Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
maxae(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Max Absolute Error is defined as

$$\max\left(|t_i - r_i|\right).$$

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
maxae(truth, response)
```

`maxse` *Max Squared Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
maxse(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Max Squared Error is defined as

$$\max\left(t_i - r_i\right)^2.$$

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
maxse(truth, response)
```

---

mbrier *Multiclass Brier Score*

Description

Measure to compare true observed labels with predicted probabilities in multiclass classification tasks.

Usage

```
mbrier(truth, prob, sample_weights = NULL, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the same levels and length as `response`.
`prob`	(`matrix()`) Matrix of predicted probabilities, each column is a vector of probabilities for a specific class label. Columns must be named with levels of `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

Brier score for multi-class classification problems with k labels defined as

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} (I_{ij} - p_{ij})^2.$$

I_{ij} is 1 if observation x_i has true label j, and 0 otherwise. p_{ij} is the probability that observation x_i belongs to class j.

Note that there also is the more common definition of the Brier score for binary classification problems in `bbrier()`.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"classif"`
- Range: $[0, 2]$
- Minimize: `TRUE`
- Required prediction: `prob`

References

Brier GW (1950). "Verification of forecasts expressed in terms of probability." *Monthly Weather Review*, **78**(1), 1–3. doi:10.1175/15200493(1950)078<0001:vofeit>2.0.co;2.

See Also

Other Classification Measures: `acc()`, `bacc()`, `ce()`, `logloss()`, `mauc_aunu()`, `mcc()`, `zero_one()`

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
prob = matrix(runif(3 * 10), ncol = 3)
colnames(prob) = levels(truth)
mbrier(truth, prob)
```

---

mcc *Matthews Correlation Coefficient*

Description

Measure to compare true observed labels with predicted labels in multiclass classification tasks.

Usage

```
mcc(truth, response, positive = NULL, sample_weights = NULL, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the same levels and length as response.
response	(factor()) Predicted response labels. Must have the same levels and length as truth.
positive	(character(1)) Name of the positive class in case of binary classification.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
...	(any) Additional arguments. Currently ignored.

Details

In the binary case, the Matthews Correlation Coefficient is defined as

$$\frac{\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FP} \cdot \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}},$$

where TP, FP, TN, TP are the number of true positives, false positives, true negatives, and false negatives respectively.

In the multi-class case, the Matthews Correlation Coefficient is defined for a multi-class confusion matrix C with K classes:

$$\frac{c \cdot s - \sum_k^K p_k \cdot t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) \cdot (s^2 - \sum_k^K t_k^2)}},$$

where

- $s = \sum_i^K \sum_j^K C_{ij}$: total number of samples
- $c = \sum_k^K C_{kk}$: total number of correctly predicted samples
- $t_k = \sum_i^K C_{ik}$: number of predictions for each class k
- $p_k = \sum_j^K C_{kj}$: number of true occurrences for each class k.

The above formula is undefined if any of the four sums in the denominator is 0 in the binary case, and more generally if either $s^2 - \sum_k^K p_k^2$ or $s^2 - \sum_k^K t_k^2$) is equal to 0. The denominator is then set to 1.

When there are more than two classes, the MCC will no longer range between -1 and +1. Instead, the minimum value will be between -1 and 0 depending on the true distribution. The maximum value is always +1.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"classif"`
- Range: $[-1, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Phi_coefficient

Matthews BW (1975). "Comparison of the predicted and observed secondary structure of T4 phage lysozyme." *Biochimica et Biophysica Acta (BBA) - Protein Structure*, **405**(2), 442–451. doi:10.1016/00052795(75)901099.

See Also

Other Classification Measures: `acc()`, `bacc()`, `ce()`, `logloss()`, `mauc_aunu()`, `mbrier()`, `zero_one()`

Examples

```
set.seed(1)
lvls = c("a", "b", "c")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
mcc(truth, response)
```

`measures` *Measure Registry*

Description

The `environment()` `measures` keeps track of all measures in this package. It stores meta information such as minimum, maximum or if the measure must be minimized or maximized. The following information is available for each measure:

- `id`: Name of the measure.
- `title`: Short descriptive title.
- `type`: `"binary"` for binary classification, `"classif"` for binary or multi-class classification, `"regr"` for regression and `"similarity"` for similarity measures.
- `lower`: lower bound.
- `upper`: upper bound.
- `predict_type`: prediction type the measure operates on. `"response"` corresponds to class labels for classification and the numeric response for regression. `"prob"` corresponds to class probabilities, provided as a matrix with class labels as column names. `"se"` corresponds to the vector of predicted standard errors for regression.
- `minimize`: If `TRUE` or `FALSE`, the objective is to minimize or maximize the measure, respectively. Can also be `NA`.
- `obs_loss`: Name of the function which is called to calculate the (unaggregated) loss per observation.
- `trafo`: Optional `list()` of length 2, containing a transformation `"fn"` and its derivative `"deriv"`.
- `aggregated`: If `TRUE`, this function aggregates the losses to a single numeric value. Otherwise, a vector of losses is returned.
- `sample_weights`: If `TRUE`, it is possible calculate a weighted measure.

Usage

```
measures
```

Format

An object of class `MeasureEnv` (inherits from `environment`) of length 65.

Examples

```
names(measures)
measures$tpr
as.data.frame(measures)
```

`medae` *Median Absolute Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
medae(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Median Absolute Error is defined as

$$\text{median}\,|t_i - r_i|\,.$$

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
medae(truth, response)
```

`medse` *Median Squared Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
medse(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(any) Additional arguments. Currently ignored.

Details

The Median Squared Error is defined as

$$\text{median}\left[(t_i - r_i)^2\right].$$

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
medse(truth, response)
```

`mse` *Mean Squared Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
mse(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Mean Squared Error is defined as

$$\frac{1}{n} \sum_{i=1}^{n} w_i \left(t_i - r_i\right)^2,$$

where w_i are normalized sample weights.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: "regr"
- Range: $[0, \infty)$
- Minimize: TRUE
- Required prediction: response

See Also

Other Regression Measures: ae(), ape(), bias(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
mse(truth, response)
```

---

msle *Mean Squared Log Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
msle(truth, response, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(numeric()) True (observed) values. Must have the same length as response.
response	(numeric()) Predicted response values. Must have the same length as truth.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The Mean Squared Log Error is defined as

$$\frac{1}{n} \sum_{i=1}^{n} w_i \left(\ln(1 + t_i) - \ln(1 + r_i) \right)^2 ,$$

where w_i are normalized sample weights. This measure is undefined if any element of t or r is less than or equal to -1.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
msle(truth, response)
```

---

npv *Negative Predictive Value*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
npv(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The Negative Predictive Value is defined as

$$\frac{\mathrm{TN}}{\mathrm{FN} + \mathrm{TN}}.$$

This measure is undefined if FN + TN = 0.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), ppv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
npv(truth, response, positive = "a")
```

---

pbias *Percent Bias*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
pbias(truth, response, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Percent Bias is defined as

$$\frac{1}{n}\sum_{i=1}^{n} w_i \frac{(r_i - t_i)}{|t_i|},$$

where w_i are normalized sample weights. Good predictions score close to 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: "regr"
- Range: $(-\infty, \infty)$
- Minimize: NA
- Required prediction: response

See Also

Other Regression Measures: ae(), ape(), bias(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
pbias(truth, response)
```

---

phi *Phi Coefficient Similarity*

Description

Measure to compare two or more sets w.r.t. their similarity.

Usage

```
phi(sets, p, na_value = NaN, ...)
```

Arguments

sets	(list()) List of character or integer vectors. sets must have at least 2 elements.
p	(integer(1)) Total number of possible elements.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The Phi Coefficient is defined as the Pearson correlation between the binary representation of two sets A and B. The binary representation for A is a logical vector of length p with the i-th element being 1 if the corresponding element is in A, and 0 otherwise.

If more than two sets are provided, the mean of all pairwise scores is calculated.

This measure is undefined if one set contains none or all possible elements.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"similarity"`
- Range: $[-1, 1]$
- Minimize: `FALSE`

References

Nogueira S, Brown G (2016). "Measuring the Stability of Feature Selection." In *Machine Learning and Knowledge Discovery in Databases*, 442–457. Springer International Publishing. doi:10.1007/9783319462271_28.

Bommert A, Rahnenführer J, Lang M (2017). "A Multicriteria Approach to Find Predictive and Sparse Models with Stable Feature Selection for High-Dimensional Data." *Computational and Mathematical Methods in Medicine*, **2017**, 1–18. doi:10.1155/2017/7907163.

Bommert A, Lang M (2021). "stabm: Stability Measures for Feature Selection." *Journal of Open Source Software*, **6**(59), 3010. doi:10.21105/joss.03010.

See Also

Package **stabm** which implements many more stability measures with included correction for chance.

Other Similarity Measures: `jaccard()`

Examples

```
set.seed(1)
sets = list(
  sample(letters[1:3], 1),
  sample(letters[1:3], 2)
)
phi(sets, p = 3)
```

`pinball` *Average Pinball Loss*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
pinball(truth, response, sample_weights = NULL, alpha = 0.5, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`alpha`	`numeric(1)` The quantile to compute the pinball loss.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The pinball loss for quantile regression is defined as

$$\text{Average Pinball Loss} = \frac{1}{n} \sum_{i=1}^{n} w_i \begin{cases} q \cdot (t_i - r_i) & \text{if } t_i \geq r_i \\ (1-q) \cdot (r_i - t_i) & \text{if } t_i < r_i \end{cases}$$

where q is the quantile and w_i are normalized sample weights.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $(-\infty, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: ae(), ape(), bias(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
pinball(truth, response)
```

---

ppv *Positive Predictive Value*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
ppv(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)

precision(
  truth,
  response,
  positive,
  sample_weights = NULL,
  na_value = NaN,
  ...
)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

na_value (numeric(1))
Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.

... (any)
Additional arguments. Currently ignored.

Details

The Positive Predictive Value is defined as

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}.$$

Also know as "precision".

This measure is undefined if TP + FP = 0.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

Goutte C, Gaussier E (2005). "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation." In *Lecture Notes in Computer Science*, 345–359. doi:10.1007/9783-540318651_25.

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), prauc(), tn(), tnr(), tp(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
ppv(truth, response, positive = "a")
```

prauc *Area Under the Precision-Recall Curve*

Description

Measure to compare true observed labels with predicted probabilities in binary classification tasks.

Usage

```
prauc(truth, prob, positive, na_value = NaN, ...)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
prob	(numeric()) Predicted probability for positive class. Must have exactly same length as truth.
positive	(character(1)) Name of the positive class.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

Computes the area under the Precision-Recall curve (PRC). The PRC can be interpreted as the relationship between precision and recall (sensitivity), and is considered to be a more appropriate measure for unbalanced datasets than the ROC curve. The AUC-PRC is computed by integration of the piecewise function.

This measure is undefined if the true values are either all positive or all negative.

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, 1]$
- Minimize: FALSE
- Required prediction: prob

References

Davis J, Goadrich M (2006). "The relationship between precision-recall and ROC curves." In *Proceedings of the 23rd International Conference on Machine Learning*. ISBN 9781595933836.

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), tn(), tnr(), tp(), tpr()

Examples

```
truth = factor(c("a", "a", "a", "b"))
prob = c(.6, .7, .1, .4)
prauc(truth, prob, "a")
```

---

rae *Relative Absolute Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
rae(truth, response, na_value = NaN, ...)
```

Arguments

truth	(numeric()) True (observed) values. Must have the same length as response.
response	(numeric()) Predicted response values. Must have the same length as truth.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The Relative Absolute Error is defined as

$$\frac{\sum_{i=1}^{n} |t_i - r_i|}{\sum_{i=1}^{n} |t_i - \bar{t}|},$$

where $\bar{t} = \sum_{i=1}^{n} t_i$. This measure is undefined for constant t.

Can be interpreted as absolute error of the predictions relative to a naive model predicting the mean.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
rae(truth, response)
```

---

rmse *Root Mean Squared Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
rmse(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Root Mean Squared Error is defined as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} w_i \left(t_i - r_i\right)^2},$$

where w_i are normalized sample weights.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
rmse(truth, response)
```

`rmsle` *Root Mean Squared Log Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
rmsle(truth, response, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Root Mean Squared Log Error is defined as

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n} w_i \left(\ln(1+t_i) - \ln(1+r_i)\right)^2},$$

where w_i are normalized sample weights.

This measure is undefined if any element of t or r is less than or equal to -1.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
rmsle(truth, response)
```

`rrse` *Root Relative Squared Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
rrse(truth, response, na_value = NaN, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Root Relative Squared Error is defined as

$$\sqrt{\frac{\sum_{i=1}^{n} (t_i - r_i)^2}{\sum_{i=1}^{n} (t_i - \bar{t})^2}},$$

where $\bar{t} = \sum_{i=1}^{n} t_i$.

Can be interpreted as root of the squared error of the predictions relative to a naive model predicting the mean.

This measure is undefined for constant t.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: ae(), ape(), bias(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rse(), rsq(), sae(), se(), sle(), smape(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
rrse(truth, response)
```

---

rse *Relative Squared Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
rse(truth, response, na_value = NaN, ...)
```

Arguments

truth	(numeric()) True (observed) values. Must have the same length as response.
response	(numeric()) Predicted response values. Must have the same length as truth.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The Relative Squared Error is defined as

$$\frac{\sum_{i=1}^{n} (t_i - r_i)^2}{\sum_{i=1}^{n} (t_i - \bar{t})^2},$$

where $\bar{t} = \sum_{i=1}^{n} t_i$.

Can be interpreted as squared error of the predictions relative to a naive model predicting the mean.

This measure is undefined for constant t.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
rse(truth, response)
```

---

rsq *R Squared*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
rsq(truth, response, na_value = NaN, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

R Squared is defined as

$$1 - \frac{\sum_{i=1}^{n} (t_i - r_i)^2}{\sum_{i=1}^{n} (t_i - \bar{t})^2},$$

where $\bar{t} = \sum_{i=1}^{n} t_i$.

Also known as coefficient of determination or explained variation. Subtracts the `rse()` from 1, hence it compares the squared error of the predictions relative to a naive model predicting the mean.

This measure is undefined for constant t.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $(-\infty, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
rsq(truth, response)
```

---

sae *Sum of Absolute Errors*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
sae(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. Weights for this function are not normalized. Defaults to sample weights 1.
`...`	(any) Additional arguments. Currently ignored.

Details

The Sum of Absolute Errors is defined as

$$\sum_{i=1}^{n} w_i \left| t_i - r_i \right| .$$

where w_i are unnormalized weights for each observation x_i, defaulting to 1.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `se()`, `sle()`, `smape()`, `srho()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
sae(truth, response)
```

se *Squared Error (per observation)*

Description

Measure to compare true observed response with predicted response in regression tasks.

Note that this is an unaggregated measure, returning the losses per observation.

Usage

```
se(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

Calculates the per-observation squared error as

$$(t_i - r_i)^2 .$$

Value

Performance value as `numeric(length(truth))`.

Meta Information

- Type: `"regr"`
- Range (per observation): $[0, \infty)$
- Minimize (per observation): `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `sle()`, `smape()`, `srho()`, `sse()`

`sle` *Squared Log Error (per observation)*

Description

Calculates the per-observation squared error as

$$(\ln(1 + t_i) - \ln(1 + r_i))^2 .$$

Measure to compare true observed response with predicted response in regression tasks.

Note that this is an unaggregated measure, returning the losses per observation.

Usage

```
sle(truth, response, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Value

Performance value as `numeric(length(truth))`.

Meta Information

- Type: `"regr"`
- Range (per observation): $[0, \infty)$
- Minimize (per observation): `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `smape()`, `srho()`, `sse()`

smape *Symmetric Mean Absolute Percent Error*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
smape(truth, response, sample_weights = NULL, na_value = NaN, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
`na_value`	(`numeric(1)`) Value that should be returned if the measure is not defined for the input (as described in the note). Default is `NaN`.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Symmetric Mean Absolute Percent Error is defined as

$$\frac{2}{n} \sum_{i=1}^{n} \frac{|t_i - r_i|}{|t_i| + |r_i|}.$$

This measure is undefined if any $|t| + |r|$ is equal to 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, 2]$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: ae(), ape(), bias(), ktau(), linex(), mae(), mape(), maxae(), maxse(), medae(), medse(), mse(), msle(), pbias(), pinball(), rae(), rmse(), rmsle(), rrse(), rse(), rsq(), sae(), se(), sle(), srho(), sse()

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
smape(truth, response)
```

---

srho *Spearman's rho*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
srho(truth, response, ...)
```

Arguments

truth	(numeric()) True (observed) values. Must have the same length as response.
response	(numeric()) Predicted response values. Must have the same length as truth.
...	(any) Additional arguments. Currently ignored.

Details

Spearman's rho is defined as Spearman's rank correlation coefficient between truth and response. Calls stats::cor() with method set to "spearman".

Value

Performance value as numeric(1).

Meta Information

- Type: "regr"
- Range: $[-1, 1]$
- Minimize: FALSE
- Required prediction: response

References

Rosset S, Perlich C, Zadrozny B (2006). "Ranking-based evaluation of regression models." *Knowledge and Information Systems*, **12**(3), 331–353. doi:10.1007/s1011500600373.

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `sse()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
srho(truth, response)
```

---

sse *Sum of Squared Errors*

Description

Measure to compare true observed response with predicted response in regression tasks.

Usage

```
sse(truth, response, sample_weights = NULL, ...)
```

Arguments

`truth`	(`numeric()`) True (observed) values. Must have the same length as `response`.
`response`	(`numeric()`) Predicted response values. Must have the same length as `truth`.
`sample_weights`	(`numeric()`) Vector of non-negative and finite sample weights. Must have the same length as `truth`. Weights for this function are not normalized. Defaults to sample weights 1.
`...`	(`any`) Additional arguments. Currently ignored.

Details

The Sum of Squared Errors is defined as

$$\sum_{i=1}^{n} w_i \left(t_i - r_i\right)^2 .$$

where w_i are unnormalized weights for each observation x_i, defaulting to 1.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"regr"`
- Range: $[0, \infty)$
- Minimize: `TRUE`
- Required prediction: `response`

See Also

Other Regression Measures: `ae()`, `ape()`, `bias()`, `ktau()`, `linex()`, `mae()`, `mape()`, `maxae()`, `maxse()`, `medae()`, `medse()`, `mse()`, `msle()`, `pbias()`, `pinball()`, `rae()`, `rmse()`, `rmsle()`, `rrse()`, `rse()`, `rsq()`, `sae()`, `se()`, `sle()`, `smape()`, `srho()`

Examples

```
set.seed(1)
truth = 1:10
response = truth + rnorm(10)
sse(truth, response)
```

---

tn *True Negatives*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
tn(truth, response, positive, sample_weights = NULL, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the exactly same two levels and the same length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the exactly same two levels and the same length as `truth`.
`positive`	(`character(1)`) Name of the positive class.

`sample_weights` (`numeric()`)
Vector of non-negative and finite sample weights. Must have the same length as `truth`. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

`...` (any)
Additional arguments. Currently ignored.

Details

This measure counts the true negatives, i.e. the number of predictions correctly indicating a negative class label. This is sometimes also called a "correct rejection".

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, \infty)$
- Minimize: `FALSE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `fpr()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tnr()`, `tp()`, `tpr()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
tn(truth, response, positive = "a")
```

tnr *True Negative Rate*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
tnr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)

specificity(
  truth,
  response,
  positive,
  sample_weights = NULL,
  na_value = NaN,
  ...
)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The True Negative Rate is defined as

$$\frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}}.$$

Also know as "specificity" or "selectivity".

This measure is undefined if FP + TN = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `fpr()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tp()`, `tpr()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
tnr(truth, response, positive = "a")
```

---

tp *True Positives*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
tp(truth, response, positive, sample_weights = NULL, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the exactly same two levels and the same length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the exactly same two levels and the same length as `truth`.

positive (character(1))
Name of the positive class.

sample_weights (numeric())
Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.

... (any)
Additional arguments. Currently ignored.

Details

This measure counts the true positives, i.e. the number of predictions correctly indicating a positive class label. This is sometimes also called a "hit".

Value

Performance value as numeric(1).

Meta Information

- Type: "binary"
- Range: $[0, \infty)$
- Minimize: FALSE
- Required prediction: response

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

See Also

Other Binary Classification Measures: auc(), bbrier(), dor(), fbeta(), fdr(), fn(), fnr(), fomr(), fp(), fpr(), gmean(), gpr(), npv(), ppv(), prauc(), tn(), tnr(), tpr()

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
tp(truth, response, positive = "a")
```

tpr *True Positive Rate*

Description

Measure to compare true observed labels with predicted labels in binary classification tasks.

Usage

```
tpr(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)

recall(truth, response, positive, sample_weights = NULL, na_value = NaN, ...)

sensitivity(
  truth,
  response,
  positive,
  sample_weights = NULL,
  na_value = NaN,
  ...
)
```

Arguments

truth	(factor()) True (observed) labels. Must have the exactly same two levels and the same length as response.
response	(factor()) Predicted response labels. Must have the exactly same two levels and the same length as truth.
positive	(character(1)) Name of the positive class.
sample_weights	(numeric()) Vector of non-negative and finite sample weights. Must have the same length as truth. The vector gets automatically normalized to sum to one. Defaults to equal sample weights.
na_value	(numeric(1)) Value that should be returned if the measure is not defined for the input (as described in the note). Default is NaN.
...	(any) Additional arguments. Currently ignored.

Details

The True Positive Rate is defined as

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

This is also know as "recall", "sensitivity", or "probability of detection".

This measure is undefined if TP + FN = 0.

Value

Performance value as `numeric(1)`.

Meta Information

- Type: `"binary"`
- Range: $[0, 1]$
- Minimize: `FALSE`
- Required prediction: `response`

References

https://en.wikipedia.org/wiki/Template:DiagnosticTesting_Diagram

Goutte C, Gaussier E (2005). "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation." In *Lecture Notes in Computer Science*, 345–359. doi:10.1007/9783-540318651_25.

See Also

Other Binary Classification Measures: `auc()`, `bbrier()`, `dor()`, `fbeta()`, `fdr()`, `fn()`, `fnr()`, `fomr()`, `fp()`, `fpr()`, `gmean()`, `gpr()`, `npv()`, `ppv()`, `prauc()`, `tn()`, `tnr()`, `tp()`

Examples

```
set.seed(1)
lvls = c("a", "b")
truth = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
response = factor(sample(lvls, 10, replace = TRUE), levels = lvls)
tpr(truth, response, positive = "a")
```

`zero_one` *Zero-One Classification Loss (per observation)*

Description

Calculates the per-observation 0/1 (zero-one) loss as

$$\mathbf{1}(t_i \neq r_1).$$

The 1/0 (one-zero) loss is equal to 1 - zero-one and calculated as

$$\mathbf{1}(t_i = r_i).$$

Measure to compare true observed labels with predicted labels in multiclass classification tasks.

Note that this is an unaggregated measure, returning the losses per observation.

Usage

```
zero_one(truth, response, ...)

one_zero(truth, response, ...)
```

Arguments

`truth`	(`factor()`) True (observed) labels. Must have the same levels and length as `response`.
`response`	(`factor()`) Predicted response labels. Must have the same levels and length as `truth`.
`...`	(`any`) Additional arguments. Currently ignored.

Value

Performance value as `numeric(length(truth))`.

Meta Information

- Type: `"classif"`
- Range (per observation): $[0, 1]$
- Minimize (per observation): `TRUE`
- Required prediction: `response`

See Also

Other Classification Measures: `acc()`, `bacc()`, `ce()`, `logloss()`, `mauc_aunu()`, `mbrier()`, `mcc()`

Index

∗ **Binary Classification Measures**
auc, 7
bbrier, 10
dor, 15
fbeta, 17
fdr, 19
fn, 20
fnr, 22
fomr, 23
fp, 25
fpr, 26
gmean, 27
gpr, 29
npv, 51
ppv, 57
prauc, 59
tn, 74
tnr, 76
tp, 77
tpr, 79
∗ **Classification Measures**
acc, 4
bacc, 9
ce, 13
logloss, 34
mauc_aunu, 38
mbrier, 42
mcc, 44
zero_one, 81
∗ **Regression Measures**
ae, 5
ape, 6
bias, 12
ktau, 32
linex, 33
mae, 36
mape, 37
maxae, 40
maxse, 41
medae, 47
medse, 48
mse, 49
msle, 50
pbias, 53
pinball, 56
rae, 60
rmse, 61
rmsle, 62
rrse, 64
rse, 65
rsq, 66
sae, 67
se, 69
sle, 70
smape, 71
srho, 72
sse, 73
∗ **Similarity Measures**
jaccard, 30
phi, 54
∗ **binary_classification_measure**
auc, 7
bbrier, 10
dor, 15
fbeta, 17
fdr, 19
fn, 20
fnr, 22
fomr, 23
fp, 25
fpr, 26
gmean, 27
gpr, 29
npv, 51
ppv, 57
prauc, 59
tn, 74
tnr, 76

tp, 77
tpr, 79
∗ **classification_measure**
acc, 4
bacc, 9
ce, 13
logloss, 34
mauc_aunu, 38
mbrier, 42
mcc, 44
zero_one, 81
∗ **datasets**
measures, 46
∗ **regression_measure**
ae, 5
ape, 6
bias, 12
ktau, 32
linex, 33
mae, 36
mape, 37
maxae, 40
maxse, 41
medae, 47
medse, 48
mse, 49
msle, 50
pbias, 53
pinball, 56
rae, 60
rmse, 61
rmsle, 62
rrse, 64
rse, 65
rsq, 66
sae, 67
se, 69
sle, 70
smape, 71
srho, 72
sse, 73
∗ **similarity_measure**
jaccard, 30
phi, 54

acc, 4, *10*, *14*, *35*, *40*, *43*, *45*, *81*
acc(), *9*
ae, 5, *7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
ape, *6*, 6, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
auc, 7, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*

bacc, *5*, 9, *14*, *35*, *40*, *43*, *45*, *81*
bbrier, *8*, 10, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
bbrier(), *43*
bias, *6, 7*, 12, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*

ce, *5*, *10*, 13, *35*, *40*, *43*, *45*, *81*
confusion_matrix, 14

dor, *8*, *11*, 15, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*

environment(), *46*

fbeta, *8*, *11*, *16*, 17, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
fdr, *8*, *11*, *16*, *18*, 19, *21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
fn, *8*, *11*, *16*, *18*, *20*, 20, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
fnr, *8*, *11*, *16*, *18*, *20, 21*, 22, *24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
fomr, *8*, *11*, *16*, *18*, *20, 21*, *23*, 23, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
fp, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, 25, *27, 28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
fpr, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26*, 26, *28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*

gmean, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26, 27*, 27, *30*, *52*, *58*, *60*, *75*, *77, 78*, *80*
gpr, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, 29, *52*, *58*, *60*, *75*, *77, 78*, *80*

jaccard, 30, *55*

ktau, *6, 7*, *13*, 32, *34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*

linex, *6, 7*, *13*, *33*, 33, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
logloss, *5*, *10*, *14*, 34, *40*, *43*, *45*, *81*

mae, *6, 7*, *13*, *33, 34*, 36, *38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*

mape, *6, 7*, *13*, *33, 34*, *37*, 37, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
mauc_au1p *(*mauc_aunu*)*, 38
mauc_au1u *(*mauc_aunu*)*, 38
mauc_aunp *(*mauc_aunu*)*, 38
mauc_aunu, *5*, *10*, *14*, *35*, 38, *43*, *45*, *81*
mauc_mu *(*mauc_aunu*)*, 38
maxae, *6, 7*, *13*, *33, 34*, *37, 38*, 40, *42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
maxse, *6, 7*, *13*, *33, 34*, *37, 38*, *41*, 41, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
mbrier, *5*, *10*, *14*, *35*, *40*, 42, *45*, *81*
mbrier(), *11*
mcc, *5*, *10*, *14*, *35*, *40*, *43*, 44, *81*
measures, 46
medae, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, 47, *48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
medse, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47*, 48, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72–74*
mlr3measures *(*mlr3measures-package*)*, 3
mlr3measures-package, 3
mse, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, 49, *51*, *54*, *57*, *61–63*, *65–70*, *72–74*
msle, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50*, 50, *54*, *57*, *61–63*, *65–70*, *72–74*

npv, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, 51, *58*, *60*, *75*, *77, 78*, *80*

one_zero *(*zero_one*)*, 81

pbias, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, 53, *57*, *61–63*, *65–70*, *72–74*
phi, *31*, 54
pinball, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, 56, *61–63*, *65–70*, *72–74*
ppv, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, 57, *60*, *75*, *77, 78*, *80*
prauc, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, 59, *75*, *77, 78*, *80*
precision, *18*
precision *(*ppv*)*, 57
precision(), *18*, *29*

rae, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, 60, *62, 63*, *65–70*, *72–74*
recall, *18*
recall *(*tpr*)*, 79
recall(), *18*, *28*, *29*
rmse, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61*, 61, *63*, *65–70*, *72–74*
rmsle, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61, 62*, 62, *65–70*, *72–74*
rrse, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, 64, *66–70*, *72–74*
rse, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65*, 65, *67–70*, *72–74*
rse(), *67*
rsq, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65, 66*, 66, *68–70*, *72–74*

sae, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–67*, 67, *69, 70*, *72–74*
se, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–68*, 69, *70*, *72–74*
sensitivity *(*tpr*)*, 79
sle, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–69*, 70, *72–74*
smape, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, 71, *73, 74*
specificity *(*tnr*)*, 76
specificity(), *28*
srho, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72*, 72, *74*
sse, *6, 7*, *13*, *33, 34*, *37, 38*, *41, 42*, *47, 48*, *50, 51*, *54*, *57*, *61–63*, *65–70*, *72, 73*, 73
stats::cor(), *32*, *72*

tn, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, 74, *77, 78*, *80*
tnr, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, 76, *78*, *80*
tp, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77*, 77, *80*
tpr, *8*, *11*, *16*, *18*, *20, 21*, *23, 24*, *26–28*, *30*, *52*, *58*, *60*, *75*, *77, 78*, 79

zero_one, *5*, *10*, *14*, *35*, *40*, *43*, *45*, 81