Package ‘dsos’

August 16, 2022

Title Dataset Shift with Outlier Scores

Version 0.1.1

Description Test for no adverse shift in two-sample comparison when we have a training set, the reference distribution, and a test set. The approach is flexible (extensible) and relies on a robust and powerful test statistic, the weighted AUC. See Kamulete, V. M. (2021) <arXiv:2107.02990> for details. Outlier scores such as trust scores and prediction uncertainty can be used as the basis for comparison for example.

License GPL (>= 3)

URL `https://github.com/vathymut/dsos`

BugReports `https://github.com/vathymut/dsos/issues`

Imports data.table (>= 1.14.0), future.apply (>= 1.9.0), ggplot2 (>= 3.3.3), scales (>= 1.1.1), simctest (>= 2.6), stats (>= 3.6.1)

Suggests fdrtool (>= 1.2.16), isotree (>= 0.2.7), ranger (>= 0.12.1), knitr (>= 1.33), rmarkdown (>= 2.7), testthat (>= 3.0.2)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.2.1

NeedsCompilation no

Author Vathy M. Kamulete [aut, cre] (<`https://orcid.org/0000-0002-4451-3743`>), Royal Bank of Canada (RBC) [cph] (Research supported by RBC)

Maintainer Vathy M. Kamulete <`vathymut@gmail.com`>

Repository CRAN

Date/Publication 2022-08-16 19:20:02 UTC

R topics documented:

---

at_from_os *Asymptotic Test from Outlier Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` or `os_train`, as the reference. The method checks whether the test set, `x_test` or `os_test`, is worse off relative to this reference set.

Usage

```
at_from_os(os_train, os_test)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.

Details

Li and Fine (2010) derives the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This approach does not use permutations and can, as a result, be much faster because it sidesteps the need to refit the scoring function `scorer`. This works well for large samples. The prefix *at* stands for asymptotic test to tell it apart from the prefix *pt*, the permutation test.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

These outlier scores should all be out-of-bag scores to mimic out-of-sample behaviour. Otherwise, the scores from the training (reference) distribution are biased (overfitted) whereas those from the test set are not. This mismatch – in-sample training scores versus out-of-sample (out-of-bag) test scores – would void the validity of the statistical test. A simple fix for this, without using resampling and/or permutations, is to get the training (reference) scores from a fresh (unused) validation set.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

Li, J., & Fine, J. P. (2010). *Weighted area under the receiver operating characteristic curve and its application to gene selection*. Journal of the Royal Statistical Society: Series C (Applied Statistics), 59(4), 673-692.

See Also

[at_oob()] for variant requiring a scoring function. [pt_from_os()] for permutation test with the outlier scores.

Other asymptotic-test: `at_oob()`

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
test_result <- at_from_os(os_train, os_test)
test_result
```

`at_oob` *Asymptotic Test With Out-Of-Bag Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` or `os_train`, as the reference. The method checks whether the test set, `x_test` or `os_test`, is worse off relative to this reference set.

Usage

```
at_oob(x_train, x_test, scorer)
```

Arguments

`x_train`	Training (reference/validation) sample.
`x_test`	Test sample.
`scorer`	Function which returns a named list with outlier scores from the training and test sample. The first argument to `scorer` must be `x_train`; the second, `x_test`. The returned named list contains two elements: *train* and *test*, each of which is a vector of (outlier) scores. See notes for more information.

Details

Li and Fine (2010) derives the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This approach does not use permutations and can, as a result, be much faster because it sidesteps the need to refit the scoring function `scorer`. This works well for large samples. The prefix *at* stands for asymptotic test to tell it apart from the prefix *pt*, the permutation test.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The scoring function, `scorer`, predicts out-of-bag scores to mimic out-of-sample behaviour. The suffix *oob* stands for out-of-bag to highlight this point. This out-of-bag variant avoids refitting the underlying algorithm from `scorer` at every permutation. It can, as a result, be computationally appealing.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

Li, J., & Fine, J. P. (2010). *Weighted area under the receiver operating characteristic curve and its application to gene selection*. Journal of the Royal Statistical Society: Series C (Applied Statistics), 59(4), 673-692.

See Also

[pt_oob()] for (faster) p-value approximation via out-of-bag predictions. [pt_refit()] for (slower) p-value approximation via refitting.

Other asymptotic-test: `at_from_os()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'

# Sample memberships with sample splitting
scorer_split <- function(x_train, x_test) split_cp(x_train, x_test)
cp_test <- at_oob(setosa, versicolor, scorer = scorer_split)
cp_test

# Sample memberships without sample splitting (out-of-bag predictions)
scorer_oob <- function(x_train, x_test) score_cp(x_train, x_test)
oob_test <- at_oob(setosa, versicolor, scorer = scorer_oob)
oob_test
```

`plot.outlier.test` *Plot result of test for no adverse shift.*

Description

Plot result of test for no adverse shift.

Usage

```
## S3 method for class 'outlier.test'
plot(x, ...)
```

Arguments

`x`	A `outlier.test` object from a D-SOS test.
`...`	Placeholder to be compatible with S3 ‘plot‘ generic.

Value

A **ggplot2** plot with outlier scores and p-value.

See Also

Other s3-method: `print.outlier.test()`

Examples

```
set.seed(12345)
os_train <- rnorm(n=3e2)
os_test <- rnorm(n=3e2)
test_to_plot <- at_from_os(os_train, os_test)
# Also: pt_from_os(os_train, os_test) for permutation test
plot(test_to_plot)
```

`print.outlier.test` *Print result of test for no adverse shift.*

Description

Print result of test for no adverse shift.

Usage

```
## S3 method for class 'outlier.test'
print(x, n = 5, ...)
```

Arguments

`x`	A `outlier.test` object from a D-SOS test.
`n`	The number of outlier scores to print for each sample.
`...`	Placeholder to be compatible with S3 ‘print‘ generic.

Value

A **ggplot2** plot with outlier scores and p-value.

See Also

Other s3-method: `plot.outlier.test()`

Examples

```
set.seed(12345)
os_train <- rnorm(n=3e2)
os_test <- rnorm(n=3e2)
test_to_print <- at_from_os(os_train, os_test)
# Also: pt_from_os(os_train, os_test) for permutation test
test_to_print
```

pt_from_os *Permutation Test from Outlier Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` or `os_train`, as the reference. The method checks whether the test set, `x_test` or `os_test`, is worse off relative to this reference set.

Usage

```
pt_from_os(os_train, os_test, n_pt = 2000)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.
`n_pt`	The number of permutations.

Details

The empirical null distribution uses `n_pt` permutations to estimate the p-value. For speed, this is implemented as a sequential Monte Carlo test with the **simctest** package. See Gandy (2009) for details. The prefix *pt* refers to permutation test. This approach does not use the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This is the recommended approach for small samples.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

These outlier scores should all be out-of-bag scores to mimic out-of-sample behaviour. Otherwise, the scores from the training (reference) distribution are biased (overfitted) whereas those from the test set are not. This mismatch – in-sample training scores versus out-of-sample (out-of-bag) test scores – would void the validity of the statistical test. A simple fix for this, without using resampling and/or permutations, is to get the training (reference) scores from a fresh (unused) validation set.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

Li, J., & Fine, J. P. (2010). *Weighted area under the receiver operating characteristic curve and its application to gene selection*. Journal of the Royal Statistical Society: Series C (Applied Statistics), 59(4), 673-692.

See Also

[pt_oob()] for variant requiring a scoring function. [at_from_os()] for asymptotic test with the outlier scores.

Other permutation-test: `pt_oob()`, `pt_refit()`

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
test_result <- pt_from_os(os_train, os_test)
test_result
```

`pt_oob` *Permutation Test With Out-Of-Bag Scores*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` or `os_train`, as the reference. The method checks whether the test set, `x_test` or `os_test`, is worse off relative to this reference set.

Usage

```
pt_oob(x_train, x_test, scorer, n_pt = 2000)
```

Arguments

`x_train`	Training (reference/validation) sample.
`x_test`	Test sample.

`scorer`	Function which returns a named list with outlier scores from the training and test sample. The first argument to `scorer` must be `x_train`; the second, `x_test`. The returned named list contains two elements: *train* and *test*, each of which is a vector of (outlier) scores. See notes below for more information.
`n_pt`	The number of permutations.

Details

The empirical null distribution uses `n_pt` permutations to estimate the p-value. For speed, this is implemented as a sequential Monte Carlo test with the **simctest** package. See Gandy (2009) for details. The prefix *pt* refers to permutation test. This approach does not use the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This is the recommended approach for small samples.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The scoring function, `scorer`, predicts out-of-bag scores to mimic out-of-sample behaviour. The suffix *oob* stands for out-of-bag to highlight this point. This out-of-bag variant avoids refitting the underlying algorithm from `scorer` at every permutation. It can, as a result, be computationally appealing.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

Li, J., & Fine, J. P. (2010). *Weighted area under the receiver operating characteristic curve and its application to gene selection*. Journal of the Royal Statistical Society: Series C (Applied Statistics), 59(4), 673-692.

See Also

[pt_refit()] for (slower) p-value approximation via refitting. [at_oob()] for p-value approximation from asymptotic null distribution.

Other permutation-test: `pt_from_os()`, `pt_refit()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
idx <- sample(nrow(iris), 2 / 3 * nrow(iris))
xy_train <- iris[idx, ]
xy_test <- iris[-idx, ]

# First example: residual diagnostics
scorer_1 <- function(x_train, x_test) score_rd(x_train, x_test, response_name = "Species")
rd_test <- pt_oob(xy_train, xy_test, scorer = scorer_1)
rd_test

# Second example: prediction uncertainty
scorer_2 <- function(x_train, x_test) score_rue(x_train, x_test, response_name = "Species")
rue_test <- pt_oob(xy_train, xy_test, scorer = scorer_2)
rue_test

# Third example: sample memberships (class probabilities)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'
scorer_3 <- function(x_train, x_test) score_cp(x_train, x_test)
cp_test <- pt_oob(setosa, versicolor, scorer = scorer_3)
cp_test
```

`pt_refit` *Permutation Test By Refitting*

Description

Test for no adverse shift with outlier scores. Like goodness-of-fit testing, this two-sample comparison takes the training set, `x_train` or `os_train`, as the reference. The method checks whether the test set, `x_test` or `os_test`, is worse off relative to this reference set.

Usage

```
pt_refit(x_train, x_test, scorer, n_pt = 2000)
```

Arguments

`x_train`	Training (reference/validation) sample.
`x_test`	Test sample.
`scorer`	Function which returns a named list with outlier scores from the training and test sample. The first argument to `scorer` must be `x_train`; the second, `x_test`. The returned named list contains two elements: *train* and *test*, each of which is a vector of (outlier) scores. See notes below for more information.
`n_pt`	The number of permutations.

Details

The empirical null distribution uses `n_pt` permutations to estimate the p-value. For speed, this is implemented as a sequential Monte Carlo test with the **simctest** package. See Gandy (2009) for details. The prefix *pt* refers to permutation test. This approach does not use the asymptotic null distribution for the weighted AUC (WAUC), the test statistic. This is the recommended approach for small samples.

Value

A named list of class `outlier.test` containing:

- `statistic`: observed WAUC statistic
- `seq_mct`: sequential Monte Carlo test, when applicable
- `p_value`: p-value
- `outlier_scores`: outlier scores from training and test set

Notes

The scoring function, `scorer`, predicts out-of-sample scores by refitting the underlying algorithm from `scorer` at every permutation The suffix *refit* emphasizes this point. This is in contrast to the out-of-bag variant, `pt_oob`, which only fits once. This method can be be computationally expensive.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts*. In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Gandy, A. (2009). *Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk*. Journal of the American Statistical Association, 104(488), 1504-1511.

Li, J., & Fine, J. P. (2010). *Weighted area under the receiver operating characteristic curve and its application to gene selection*. Journal of the Royal Statistical Society: Series C (Applied Statistics), 59(4), 673-692.

See Also

[pt_oob()] for (faster) p-value approximation via out-of-bag predictions. [at_oob()] for p-value approximation from asymptotic null distribution.

Other permutation-test: `pt_from_os()`, `pt_oob()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'
scorer <- function(x_train, x_test) score_od(x_train, x_test)
iris_test <- pt_refit(setosa, versicolor, scorer = scorer)
```

```
iris_test
```

`score_cp` *Predict Class Probability (Sample Membership)*

Description

Predict class probability using random forest with the **ranger** package. The prefix *cp* stands for class probability, which reflects sample membership between training and test set. This function is useful to test for dataset shift via classifier performance to mimic tests of equal distribution.

Usage

```
score_cp(x_train, x_test, n_trees = 500L, response_name = "label")
```

Arguments

`x_train`	Training (reference) sample.
`x_test`	Test sample.
`n_trees`	The number of trees in random forest.
`response_name`	The column name of the categorical outcome to predict.

Details

`score_cp` fits a classifier to discriminate between training and test sets. It uses out-of-bag predictions, namely class probabilities, to estimate sample memberships. As a result, estimating p-value via permutations does not require refitting the algorithm for every permutation.

Value

A named list or object of class `outlier.test` containing:

- `train`: vector of scores in training set
- `test`: vector of scores in test set

Notes

Kim et al. (2022) describes how a classifier can serve as a proxy for two-sample comparison. As in Hediger et al. (2022), we use random forest as the underlying classifier. The probability of belonging to the test set, as as opposed to the training set, is the outlier score. That is, the binary classifier assigns training and test set to different classes.

References

Hediger, S., Michel, L., & Näf, J. (2022). *On the use of random forest for two-sample testing*. Computational Statistics & Data Analysis, 170, 107435.

Kim, I., Ramdas, A., Singh, A., & Wasserman, L. (2021). *Classification accuracy as a proxy for two-sample testing*. The Annals of Statistics, 49(1), 411-434.

See Also

Other scoring: `score_od()`, `score_rd()`, `score_rue()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'
outlier_scores <- score_cp(setosa, versicolor, response_name = "label")
str(outlier_scores)
```

`score_od` *Predict Isolation Scores*

Description

Predict isolation scores using (extended) isolation forest with the **isotree** package. The prefix *od* stands for outlier detection, the relevant notion of outlyingness. This function is useful to test for dataset shift via density-based scores: isolation scores are inversely related, if not quite proportional, to densities.

Usage

```
score_od(x_train, x_test, n_trees = 500L, threshold = 0.6)
```

Arguments

`x_train`	Training (reference) sample.
`x_test`	Test sample.
`n_trees`	The number of trees in isolation forest.
`threshold`	Decision threshold. Set to a default value of 0.6, following Chabchoub et al. (2022). Outlier scores higher than the threshold are considered outliers, and lower values are inliers.

Value

A named list or object of class `outlier.test` containing:

- `train`: vector of scores in training set
- `test`: vector of scores in test set

Notes

Isolation forest detects *isolated* points that are typically out-of-distribution relative to the high-density regions of the data distribution. Any performant method for density-based out-of-distribution detection can replace isolation forest. The decision threshold, `threshold`, clips (winsorizes) the scores so that lower scores are set to the threshold value.

References

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). *Isolation forest*. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

Chabchoub, Y., Togbe, M. U., Boly, A., & Chiky, R. (2022). *An in-depth study and improvement of Isolation Forest.*. IEEE Access, 10, 10219-10237.

#' @details `score_od` first fits to the training data and then predict in-sample for this reference sample. Then it predicts out-of-sample for the test set. As a result, estimating p-value via permutations require refitting the algorithm for every permutation.

See Also

Other scoring: `score_cp()`, `score_rd()`, `score_rue()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'
score_od(setosa, versicolor)
```

`score_rd` *Predict Out-of-bag Errors (Residuals)*

Description

Predict out-of-bag errors (residuals) using random forest with the **ranger** package. The prefix *rd* stands for residual diagnostic, the relevant notion of outlyingness. This function is useful to test for dataset shift via prediction errors from the underlying supervised algorithm.

Usage

```
score_rd(x_train, x_test, n_trees = 500L, response_name = "label")
```

Arguments

`x_train`	Training (reference) sample.
`x_test`	Test sample.
`n_trees`	The number of trees in random forest.
`response_name`	The column name of the categorical outcome to predict.

Details

`score_rd` first fits to the training data and uses out-of-bag predictions to estimate errors (residuals) for this reference sample. Then it leverages out-of-sample predictions to calculate errors for the test set. As a result, estimating p-value via permutations does not require refitting the algorithm for every permutation. See Hediger et al. (2022) for details on this approach.

Value

A named list or object of class `outlier.test` containing:

- `train`: vector of scores in training set
- `test`: vector of scores in test set

Notes

Residuals traditionally underpin diagnostics (misspecification) tests in supervised learning. For a contemporaneous example of this approach using machine learning, see Janková et al. (2020) and references therein.

References

Janková, J., Shah, R. D., Bühlmann, P., & Samworth, R. J. (2020). *Goodness-of-fit testing in high dimensional generalized linear models*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(3), 773-795.

Hediger, S., Michel, L., & Näf, J. (2022). *On the use of random forest for two-sample testing*. Computational Statistics & Data Analysis, 170, 107435.

See Also

Other scoring: `score_cp()`, `score_od()`, `score_rue()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
idx <- sample(nrow(iris), 2 / 3 * nrow(iris))
```

```
xy_train <- iris[idx, ]
xy_test <- iris[-idx, ]
score_rd(xy_train, xy_test, n_trees = 500L, response_name = "Species")
```

score_rue *Predict Resampling Uncertainty (Prediction Confidence)*

Description

Estimate prediction uncertainty using random forest with the **ranger** package. The prefix *rue* stands for resampling uncertainty estimation, the relevant notion of outlyingness. This function is useful to test for dataset shift via prediction uncertainty from supervised algorithms.

Usage

```
score_rue(x_train, x_test, n_trees = 500L, response_name = "label")
```

Arguments

`x_train`	Training (reference) sample.
`x_test`	Test sample.
`n_trees`	The number of trees in random forest.
`response_name`	The column name of the categorical outcome to predict.

Details

`score_rue` first fits to the training data and uses out-of-bag predictions to estimate prediction uncertainty for this reference sample. Then it leverages out-of-sample predictions to do the same for the test set. As a result, estimating p-value via permutations does not require refitting the algorithm for every permutation.

Value

A named list or object of class `outlier.test` containing:

- `train`: vector of scores in training set
- `test`: vector of scores in test set

Notes

For prediction uncertainty, we essentially implement the approach in Schulam & Saria (2019) with random forest. The standard errors of the mean predictions are the outlier scores. Any performant method for confidence-based out-of-distribution (OOD) detection can replace random forest. Berger et al. (2021) compares methods for confidence-based OOD detection.

References

Schulam, P., & Saria, S. (2019, April). *Can you trust this prediction? Auditing pointwise reliability after learning*. In The 22nd International Conference on Artificial Intelligence and Statistics (pp. 1022-1031). PMLR.

Berger, C., Paschali, M., Glocker, B., & Kamnitsas, K. (2021). *Confidence-based out-of-distribution detection: a comparative study and analysis*. In Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Perinatal Imaging, Placental and Preterm Image Analysis (pp. 122-132). Springer, Cham.

See Also

Other scoring: `score_cp()`, `score_od()`, `score_rd()`

Examples

```
library(dsos)
set.seed(12345)
data(iris)
idx <- sample(nrow(iris), 2 / 3 * nrow(iris))
xy_train <- iris[idx, ]
xy_test <- iris[-idx, ]
outlier_scores <- score_rue(xy_train, xy_test, response_name = "Species")
str(outlier_scores)
```

`split_cp` *Split Samples And Predict Class Probability (Sample Membership)*

Description

Predict class probability using random forest with the **ranger** package. The prefix *cp* stands for class probability, which reflects sample membership between training and test set. This function is useful to test for dataset shift via classifier performance to mimic tests of equal distribution.

Usage

```
split_cp(x_train, x_test, n_trees = 500L, response_name = "label")
```

Arguments

`x_train`	Training (reference) sample.
`x_test`	Test sample.
`n_trees`	The number of trees in random forest.
`response_name`	The column name of the categorical outcome to predict.

Details

`split_cp` fits a classifier to discriminate between training and test sets. It splits training and test sets into half samples so that the first halves are for model fitting and the second halves, for out-of-sample predictions. As a result, estimating the p-value can take advantage of the asymptotic null distribution.

Value

A named list or object of class `outlier.test` containing:

- `train`: vector of scores in training set
- `test`: vector of scores in test set

Notes

See the docs for `score_cp` for more information. `split_cp` uses sample splitting (half samples), rather than out-of-bag predictions as in `score_cp`, for inference. Rinaldo et al. (2019) discusses how sample splitting can be used for valid inference (p-value estimation).

References

Rinaldo, A., Wasserman, L., & G'Sell, M. (2019). *Bootstrapping and sample splitting for high-dimensional, assumption-lean inference*. The Annals of Statistics, 47(6), 3438-3469.

See Also

[score_cp()] for the out-of-bag variant, rather than sample splitting.

Examples

```
library(dsos)
set.seed(12345)
data(iris)
setosa <- iris[1:50, 1:4] # Training sample: Species == 'setosa'
versicolor <- iris[51:100, 1:4] # Test sample: Species == 'versicolor'
outlier_scores <- split_cp(setosa, versicolor, response_name = "label")
str(outlier_scores)
```

`wauc_from_os` *Weighted AUC from Outlier Scores*

Description

Computes the weighted AUC with the weighting scheme described in Kamulete, V. M. (2021). This assumes that the training set is the reference distribution and specifies a particular functional form to derive weights from threshold scores.

Usage

```
wauc_from_os(os_train, os_test, weight = NULL)
```

Arguments

`os_train`	Outlier scores in training (reference) set.
`os_test`	Outlier scores in test set.
`weight`	Numeric vector of weights of length `length(os_train)` + `length(os_test)`. The first `length(os_train)` weights belongs to the training set, the rest is for the test set. If `NULL`, the default, all weights are set to 1.

Value

The value (scalar) of the weighted AUC given the weighting scheme.

References

Kamulete, V. M. (2022). *Test for non-negligible adverse shifts.* In The 38th Conference on Uncertainty in Artificial Intelligence. PMLR.

Examples

```
library(dsos)
set.seed(12345)
os_train <- rnorm(n = 100)
os_test <- rnorm(n = 100)
test_stat <- wauc_from_os(os_train, os_test)
```

Index