Package 'HNPclassifier'

February 8, 2026

Title Hierarchical Neyman-Pearson Classification for Ordered Classes

Version 0.1.0

Description The Hierarchical Neyman-Pearson (H-NP) classification framework extends the Neyman-Pearson classification paradigm to multi-class settings where classes have a natural priority ordering. This is particularly useful for classification in unbalanced dataset, for example, disease severity classification, where under-classification errors (misclassifying patients into less severe categories) are more consequential than other misclassifications. The package implements H-NP umbrella algorithms that controls under-classification errors under user specified control levels with high probability. It supports the creation of H-NP classifiers using scoring functions based on built-in classification methods (including logistic regression, support vector machines, and random forests), as well as user-trained scoring functions. For theoretical details, please refer to Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li & Xin Tong (2024) <doi:10.1080/01621459.2023.2270657>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, e1071, nnet, randomForest

NeedsCompilation no

Author Che Shen [aut, cre] (Implementation and maintenance),
Lujia Yang [aut] (Testing and debugging),
Lijia Wang [aut] (Original theory and supervision),
Shunan Yao [aut] (Supervision and debugging)

Maintainer Che Shen <chshen3-c@my.cityu.edu.hk>

Repository CRAN

Date/Publication 2026-02-08 16:40:07 UTC

Contents

`base_function` *Base Classifier Training function Train a base multi-class model (RF / SVM / Multinomial Logistic)*

Description

Fit one of the supported classifiers for ternary classification: Random Forest, SVM (with probabilities), or multinomial logistic regression via `nnet::multinom`.

Usage

```
base_function(x, y, method = "randomforest")
```

Arguments

`x`	A data.frame of predictors/features.
`y`	A factor response with levels "1","2","3".
`method`	Character string: one of 'randomforest', 'svm', or 'logistic'.

Value

A trained model object compatible with the downstream scoring functions.

Examples

```
set.seed(123)
x <- data.frame(a = rnorm(20), b = rnorm(20))
y <- factor(sample(c("1","2","3"), 20, TRUE))
model <- base_function(x, y, method = 'randomforest')
```

hnp_box_plot *HNP Box Plot Experiment*

Description

Runs multiple iterations of HNP experiment on a dataset (with random 7:3 splits) and generates a PDF with 15 boxplots comparing Before vs After NP performance.

Usage

```
hnp_box_plot(
  data,
  class_col,
  method = "logistic",
  n_runs = 100,
  levels = c(0.05, 0.05),
  tolerances = c(0.05, 0.05),
  output_file = NULL,
  hnp_split = NULL,
  split_ratio = c(0.7, 0.3)
)
```

Arguments

data	A data.frame containing features and class label.
class_col	Character. Name of the class column (must be mapped to "1","2","3").
method	Character. Base classifier method ('randomforest', 'svm', 'logistic').
n_runs	Integer. Number of iterations to run.
levels	Numeric vector. Alpha levels (constraints) for classes (e.g., c(0.05, 0.1)).
tolerances	Numeric vector. Delta tolerances for classes (e.g., c(0.01, 0.02)).
output_file	Character. Path to save the PDF output.
hnp_split	List. Split configuration for HNP internal validation.
split_ratio	Numeric vector. Ratio of data used for training and testing (e.g., c(0.7, 0.3)).

Value

No return value, called for side effects.

Examples

```
set.seed(123)
n <- 2000
features <- data.frame(
  x1 = rnorm(n),
  x2 = rnorm(n)
```

```
)
y <- factor(sample(c("1", "2", "3"), n, replace = TRUE, prob = c(0.2, 0.3, 0.5)))
data <- cbind(features, y)
hnp_box_plot(
  data = data,
  class_col = "y",
  method = "logistic",
  n_runs = 2,
  levels = c(0.05, 0.05),
  tolerances = c(0.05, 0.05),
  output_file = tempfile(fileext = ".pdf")
)
```

`hnp_delta_search` *Delta search*

Description

Calculate the order k of the statistic that satisfies the given confidence requirements for determining classification thresholds.

Usage

```
hnp_delta_search(n, level, delta)
```

Arguments

`n`	Integer specifying the cardinality of the grid set Tau (size of `S_it`).
`level`	Numeric between 0 and 1 representing the desired control level (alpha) for the ith under-classification error.
`delta`	Numeric tolerance parameter for the confidence bound.

Value

An integer k representing the order of the statistic that meets the confidence requirements. Returns `NA` if no valid solution exists.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
k <- hnp_delta_search(n = 100, level = 0.05, delta = 0.01)
```

hnp_map_classes *Classes Mapping function for HNP Algorithm Map class labels to canonical levels "1", "2", "3"*

Description

Validate the class column and re-label provided class names to canonical factor levels "1", "2", and "3". Useful for preparing datasets before training and evaluation in the HNP Umbrella pipeline.

Usage

```
hnp_map_classes(data, class_col, class_1, class_2, class_3)
```

Arguments

data	A data.frame or data.table containing the dataset.
class_col	Character scalar. Name of the class/label column in data.
class_1	Character. Original label that should map to level "1" (most severe with most attentions).
class_2	Character. Original label that should map to level "2" (median severe).
class_3	Character. Original label that should map to level "3" (normal or less important).

Value

The input data with class_col converted to a factor with levels c("1","2","3").

Examples

```
df <- data.frame(y = c("low","mid","high","mid"), x1 = rnorm(4))
df2 <- hnp_map_classes(df, class_col = "y", class_1 = "low", class_2 = "mid", class_3 = "high")
table(df2$y)
```

hnp_summary *hnp_summary Summarize a ternary classifier's performance*

Description

Compute confusion matrix, class-wise false positive/negative rates, over- and under-classification errors, overall accuracy, and a normalized error table for a ternary classifier produced by the HNP pipeline.

Usage

```
hnp_summary(classifier, data, class_col, class_number = NULL)
```

Arguments

classifier	A function function(X) { ... } that returns class labels 1/2/3 for a single-row data.frame or vectorized over rows.
data	A data.frame containing features and the true class column.
class_col	Character scalar. Name of the true class/label column.
class_number	Optional integer. Number of classes; if NULL, inferred from the data.

Value

A list with components: confusion_matrix, false_positive_rate, false_negative_rate, overall_accuracy, predictions, under_classification_error, over_classification_error, total_over_classification_error, total_under_classification_error, and error_table.

Examples

```
set.seed(123)
n <- 50
x <- data.frame(a = rnorm(n), b = rnorm(n))
y <- factor(sample(c("1","2","3"), n, TRUE))
df <- cbind(x, y)
clf <- function(X) sample(c(1,2,3), nrow(X), replace=TRUE)
res <- hnp_summary(clf, data = df, class_col = "y")
```

---

hnp_umbrella *HNP Umbrella Algorithm*

Description

Implementation of the HNP Umbrella algorithm for ternary classification

Usage

```
hnp_umbrella(
  S,
  levels,
  tolerances,
  A1 = NULL,
  method = "randomforest",
  hnp_split = NULL,
  class_col
)
```

Arguments

S	Training dataset
levels	Confidence levels (alpha) for each class
tolerances	Tolerance parameters (delta) for each class
A1	Candidate thresholds for class 1
method	Classification method to use ('randomforest', 'svm', 'logistic')
hnp_split	Data splitting ratios for each class
class_col	Character scalar. Name of the class column in the dataset (must be mapped to levels "1","2","3").

Value

A classifier function that takes new data and classifies it into a class with controlled type-one error rate

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
n <- 500
features <- data.frame(
  x1 = rnorm(n),
  x2 = rnorm(n)
)
y <- factor(sample(c("1", "2", "3"), n, replace = TRUE, prob = c(0.2, 0.3, 0.5)))
data <- cbind(features, y)
clf <- hnp_umbrella(
  S = data,
  levels = c(0.1, 0.1),
  tolerances = c(0.1, 0.1),
  class_col = "y",
  method = "randomforest"
)
```

---

hnp_umbrella_flex *HNP Umbrella (flex): use custom score functions and pre-split data*

Description

Flexible variant of the HNP Umbrella algorithm that accepts user-provided scoring functions and explicit data splits for thresholding and error estimation. This bypasses model training inside and focuses on threshold selection with confidence controls.

Usage

```
hnp_umbrella_flex(
  score_data,
  threshold_data,
  error_data,
  levels,
  tolerances,
  A1 = NULL,
  score_functions = NULL,
  class_col
)
```

Arguments

score_data	A data.frame for fitting/deriving scoring behavior.
threshold_data	A data.frame used to compute thresholds.
error_data	A data.frame used to estimate empirical errors.
levels	Numeric vector of length 2. Confidence levels (alpha) for class 1 and class 2 under-classification controls.
tolerances	Numeric vector of length 2. Tolerance (delta) values for the corresponding classes.
A1	Optional numeric vector of candidate thresholds for class 1.
score_functions	A list with at least two functions: T1, T2. Each must accept a data.frame and return numeric scores.
class_col	Character scalar. Name of the class column in the data.

Value

A classifier function `function(new_data) data.frame(result=...)`, or `NULL` if no valid classifier is found.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
n <- 500
score_data <- data.frame(x=rnorm(n), y=factor(sample(1:3, n, replace=TRUE)))
threshold_data <- data.frame(x=rnorm(n), y=factor(sample(1:3, n, replace=TRUE)))
error_data <- data.frame(x=rnorm(n), y=factor(sample(1:3, n, replace=TRUE)))
T1 <- function(d) as.numeric(d$x > 0)
T2 <- function(d) as.numeric(d$x > 0.5)
```

```
clf <- hnp_umbrella_flex(score_data, threshold_data, error_data,
                         levels = c(0.05, 0.05), tolerances = c(0.01, 0.01),
                         score_functions = list(T1, T2), class_col = 'y')
preds <- clf(score_data)
```

---

hnp_upper_bound *Upper Bound of the ith Threshold (Optimal ith Threshold)*

Description

Compute the optimal threshold for class i using score functions and confidence bounds, given tolerance and under classification error level.

Usage

```
hnp_upper_bound(S_it, level, delta_i, score_functions, thresholds, i)
```

Arguments

`S_it`	The left-out class-i samples.
`level`	(alpha) desired control level for the ith under classification error.
`delta_i`	ith tolerance parameter.
`score_functions`	A list of score functions (T_1, ..., T_i).
`thresholds`	Numeric vector of length `i - 1` with thresholds for previously evaluated classes; ignored when `i == 1`.
`i`	Class-i.

Value

t_i_bar Optimal ith threshold.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
n <- 200
S_it <- data.frame(
  feature1 = rnorm(n, mean = 2, sd = 1),
  feature2 = runif(n, min = 0, max = 5)
)
level <- 0.05
```

```
delta_i <- 0.01
score_functions <- list(
  function(data) runif(nrow(data)),
  function(data) runif(nrow(data))
)
thresholds <- c(2.5, NA)
i <- 1
t_i_bar <- hnp_upper_bound(S_it, level, delta_i, score_functions, thresholds, i)
```

`probability_to_score_1` *T1 Calculation Create T1 scoring function from a fitted model*

Description

Return a function that takes new data and outputs the score for class 1, typically the predicted probability P(Y=1|X). Works with the supported `methods` used by `base_function`.

Usage

```
probability_to_score_1(model, method)
```

Arguments

`model`	A fitted model returned by `base_function` or equivalent.
`method`	Character string specifying the model family used: one of 'svm', 'randomforest', or 'logistic'.

Value

A function of the form `function(X) numeric`, where `X` is a data.frame of features and the returned numeric vector are scores for class 1.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
x <- data.frame(a = rnorm(20), b = rnorm(20))
y <- factor(sample(c("1","2","3"), 20, TRUE))
model <- base_function(x, y, method = 'randomforest')
T1 <- probability_to_score_1(model, method = 'randomforest')
newx <- data.frame(a = rnorm(5), b = rnorm(5))
scores <- T1(newx)
```

probability_to_score_2 *T2 Calculation Create T2 scoring function as ratio P(class 2)/P(class 3)*

Description

Return a function that produces the ratio of predicted probabilities P(Y=2|X) / P(Y=3|X), with safeguards for zeros/NA and infinite values. Works with the supported `methods` used by `base_function`.

Usage

```
probability_to_score_2(model, method)
```

Arguments

`model`	A fitted model returned by `base_function` or equivalent.
`method`	Character string specifying the model family used: one of 'svm', 'randomforest', or 'logistic'.

Value

A function of the form `function(X) numeric`, where `X` is a data.frame of features and the returned numeric vector are T2 scores.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
x <- data.frame(a = rnorm(20), b = rnorm(20))
y <- factor(sample(c("1","2","3"), 20, TRUE))
model <- base_function(x, y, method = 'randomforest')
T2 <- probability_to_score_2(model, method = 'randomforest')
newx <- data.frame(a = rnorm(5), b = rnorm(5))
scores <- T2(newx)
```

Index