Package 'SHAPBoost'

January 19, 2026

Title The SHAPBoost Feature Selection Algorithm

Version 1.0.3

Description The implementation of SHAPBoost, a boosting-based feature selection technique that ranks features iteratively based on Shapley values.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports xgboost (>= 3.1), SHAPforxgboost, methods, caret, Matrix

Suggests flare, survival

URL <https://github.com/O-T-O-Z/SHAPBoost-R>

BugReports <https://github.com/O-T-O-Z/SHAPBoost-R/issues>

NeedsCompilation no

Author Ömer Tarik Özyilmaz [aut, cre, cph] (ORCID:
    <https://orcid.org/0009-0005-3911-5520>),
    Tamas Szili-Török [aut, cph] (ORCID:
    <https://orcid.org/0000-0001-8675-9234>)

Maintainer Ömer Tarik Özyilmaz <o.t.ozyilmaz@umcg.nl>

Repository CRAN

Date/Publication 2026-01-19 15:50:02 UTC

Contents

1

```
SHAPBoostEstimator-class
```
                                    SHAPBoostEstimator Class

Description

This class implements the SHAPBoost algorithm for feature selection. It is designed to be extended by specific implementations such as SHAPBoostRegressor and SHAPBoostSurvival. Any new method should implement the abstract methods defined in this class.

Fields

evaluator The model that is used to evaluate each additional feature.

metric A character string representing the evaluation metric.

xgb_params A list of parameters for the XGBoost model.

number_of_folds The number of folds for cross-validation.

epsilon A small value to determine convergence.

max_number_of_features The maximum number of features to select.

siso_ranking_size The number of features to consider in the SISO ranking.

siso_order The order of combinations to consider in SISO.

reset A logical indicating whether to reset the weights.

num_resets The number of resets allowed.

fold_random_state The random state for reproducibility in cross-validation.

verbose The verbosity level of the output.

fixed_variables A character vector of variable names to be always included.

stratification A logical indicating whether to use stratified sampling. Only applicable for c-index metric.

collinearity_check A logical indicating whether to check for collinearity.

correlation_threshold The threshold for correlation to consider features as collinear.

Examples

```
if (requireNamespace("flare", quietly = TRUE)) {
  data("eyedata", package = "flare")
  shapboost <- SHAPBoostRegressor$new(
    max_number_of_features = 1,
    evaluator = "lr",
    metric = "mae",
    siso_ranking_size = 10,
    verbose = 0
  )
  X <- as.data.frame(x)
  y <- as.data.frame(y)
  subset <- shapboost$fit(X, y)
}
```

```
SHAPBoostRegressor-class
```
SHAPBoostRegressor is a reference class for regression feature selection through gradient boosting.

Description

This class extends the SHAPBoostEstimator class and implements methods for initializing, updating weights, scoring, and fitting estimators.

Fields

evaluator The model that is used to evaluate each additional feature. Choice between "lr" and "xgb".

metric The metric used for evaluation, such as "mae", "mse", or "r2".

xgb_params A list of parameters for the XGBoost model.

number_of_folds The number of folds for cross-validation.

epsilon A small value to prevent division by zero.

max_number_of_features The maximum number of features to consider.

siso_ranking_size The size of the SISO ranking.

siso_order The order of the SISO ranking.

reset A boolean indicating whether to reset the model.

xgb_importance The importance type for XGBoost.

num_resets The number of resets for the model.

fold_random_state The random state for folds.

verbose The verbosity level for logging.

stratification A boolean indicating whether to use stratification. Only applicable for c-index metric.

use_shap A boolean indicating whether to use SHAP values.

collinearity_check A boolean indicating whether to check for collinearity.

correlation_threshold The threshold for correlation to consider features as collinear.

Examples

```
if (requireNamespace("flare", quietly = TRUE)) {
  data("eyedata", package = "flare")
  shapboost <- SHAPBoostRegressor$new(
    max_number_of_features = 1,
    evaluator = "lr",
    metric = "mae",
    siso_ranking_size = 10,
    verbose = 0
  )
```

```
  X <- as.data.frame(x)
  y <- as.data.frame(y)
  subset <- shapboost$fit(X, y)
}
```

SHAPBoostSurvival-class

> *SHAPBoostSurvival is a reference class for survival analysis feature selection through gradient boosting.*

Description

This class extends the SHAPBoostEstimator class and implements methods for initializing, updating weights, scoring, and fitting estimators.

Fields

evaluator The model that is used to evaluate each additional feature. Choice between "coxph" and "xgb".

metric The metric used for evaluation, such as "mae", "mse", or "r2".

xgb_params A list of parameters for the XGBoost model.

number_of_folds The number of folds for cross-validation.

epsilon A small value to prevent division by zero.

max_number_of_features The maximum number of features to consider.

siso_ranking_size The size of the SISO ranking.

siso_order The order of the SISO ranking.

reset A boolean indicating whether to reset the model.

xgb_importance The importance type for XGBoost.

num_resets The number of resets for the model.

fold_random_state The random state for folds.

verbose The verbosity level for logging.

stratification A boolean indicating whether to use stratification. Only applicable for c-index metric.

use_shap A boolean indicating whether to use SHAP values.

collinearity_check A boolean indicating whether to check for collinearity.

correlation_threshold The threshold for correlation to consider features as collinear.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  shapboost <- SHAPBoostSurvival$new(
    max_number_of_features = 1,
    evaluator = "coxph",
    metric = "c-index",
    verbose = 0,
    xgb_params = list(
      objective = "survival:cox",
      eval_metric = "cox-nloglik"
    )
  )

  X <- as.data.frame(survival::gbsg[, -c(1, 10, 11)])
  y <- as.data.frame(survival::gbsg[, c(10, 11)])
  subset <- shapboost$fit(X, y)
}
```

Index