Package: shapley (via r-universe)

October 24, 2024

Type Package

Title Weighted Mean SHAP for Feature Selection in ML Grid and Ensemble

Version 0.4

Depends R (>= 3.5.0)

Description This R package introduces Weighted Mean SHapley Additive exPlanations (WMSHAP), an innovative method for calculating SHAP values for a grid of fine-tuned base-learner machine learning models as well as stacked ensembles, a method not previously available due to the common reliance on single best-performing models. By integrating the weighted mean SHAP values from individual base-learners comprising the ensemble or individual base-learners in a tuning grid search, the package weights SHAP contributions according to each model's performance, assessed by multiple either R squared (for both regression and classification models). alternatively, this software also offers weighting SHAP values based on the area under the precision-recall curve (AUCPR), the area under the curve (AUC), and F2 measures for binary classifiers. It further extends this framework to implement weighted confidence intervals for weighted mean SHAP values, offering a more comprehensive and robust feature importance evaluation over a grid of machine learning models, instead of solely computing SHAP values for the best model. This methodology is particularly beneficial for addressing the severe class imbalance (class rarity) problem by providing a transparent, generalized measure of feature importance that mitigates the risk of reporting SHAP values for an overfitted or biased model and maintains robustness under severe class imbalance, where there is no universal criteria of identifying the absolute best model. Furthermore, the package implements hypothesis testing to ascertain the statistical significance of SHAP values for individual features, as well as comparative significance testing of SHAP contributions between features. Additionally, it tackles a critical gap in feature selection literature by presenting criteria for the automatic feature selection of the

most important features across a grid of models or stacked ensembles, eliminating the need for arbitrary determination of the number of top features to be extracted. This utility is invaluable for researchers analyzing feature significance, particularly within severely imbalanced outcomes where conventional methods fall short. Moreover, it is also expected to report democratic feature importance across a grid of models, resulting in a more comprehensive and generalizable feature selection. The package further implements a novel method for visualizing SHAP values both at subject level and feature level as well as a plot for feature selection based on the weighted mean SHAP ratios.

License MIT + file LICENSE

Encoding UTF-8

Imports ggplot2 (>= 3.4.2), h2o (>= 3.34.0.0), curl (>= 4.3.0), waffle (>= 1.0.2)

RoxygenNote 7.3.1

URL https://github.com/haghish/shapley, https://www.sv.uio.no/psi/english/people/academic/haghish/

BugReports https://github.com/haghish/shapley/issues

NeedsCompilation no

Author E. F. Haghish [aut, cre, cph]

Maintainer E. F. Haghish <haghish@hotmail.com>

Repository CRAN

Date/Publication 2024-10-23 03:40:02 UTC

Contents

h2o.get_ids *h2o.get_ids*

Description

extracts the model IDs from H2O AutoML object or H2O grid

Usage

```
h2o.get_ids(automl)
```

Arguments

automl	a h2o "AutoML" grid object

Value

a character vector of trained models' names (IDs)

Author(s)

E. F. Haghish

Examples

```
## Not run:
library(h2o)
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 30)

# get the model IDs
ids <- h2o.ids(aml)

## End(Not run)
```

`normalize` *Normalize a vector based on specified minimum and maximum values*

Description

This function normalizes a vector based on specified minimum and maximum values. If the minimum and maximum values are not specified, the function will use the minimum and maximum values of the vector.

Usage

```
normalize(x, min = NULL, max = NULL)
```

Arguments

`x`	numeric vector
`min`	minimum value
`max`	maximum value

Value

normalized numeric vector

Author(s)

E. F. Haghish

`shapley` *Weighted average of SHAP values and weighted SHAP confidence intervals for a grid of fine-tuned models or base-learners of a stacked ensemble model*

Description

Weighted average of SHAP values and weighted SHAP confidence intervals provide a measure of feature importance for a grid of fine-tuned models or base-learners of a stacked ensemble model. Instead of reporting relative SHAP contributions for a single model, this function takes the variability in feature importance of multiple models into account and computes weighted mean and confidence intervals for each feature, taking the performance metric of each model as the weight. The function also provides a plot of the weighted SHAP values and confidence intervals. Currently only models trained by h2o machine learning software or autoEnsemble package are supported.

Usage

```
shapley(
  models,
  newdata,
  plot = TRUE,
  performance_metric = "r2",
  standardize_performance_metric = FALSE,
  performance_type = "xval",
  minimum_performance = 0,
  method = c("lowerCI"),
  cutoff = 0,
  top_n_features = NULL,
  n_models = 10
)
```

Arguments

models	H2O search grid, AutoML grid, or a character vector of H2O model IDs. the "h2o.get_ids" function from "h2otools" can retrieve the IDs from grids.
newdata	h2o frame (data.frame). the data.frame must be already uploaded on h2o server (cloud). when specified, this dataset will be used for evaluating the models. if not specified, model performance on the training dataset will be reported.
plot	logical. if TRUE, the weighted mean and confidence intervals of the SHAP values are plotted. The default is TRUE.
performance_metric	character, specifying the performance metric to be used for weighting the SHAP values (mean and 95 "r2" (R Squared). For binary classification, other options include "aucpr" (area under the precision-recall curve), "auc" (area under the ROC curve), and "f2" (F2 score).
standardize_performance_metric	logical. if TRUE, performance_metric, which is used as weights vector is standardized such that the sum of the weights vector would be equal to the length of the vector. the default value is FALSE.
performance_type	character, specifying where the performance metric should be retrieved from. "train" means the performance of the training process should be reported, "valid" indicates that the performance of the validation process should be reported, and "xval" means the cross-validation performance to be retrieved.
minimum_performance	the minimum performance metric for a recognizable model. any model with performance equal or lower than this argument will have weight of zero in computing the weighted mean and CI SHAP values. the default value is zero.
method	character, specifying the method used for identifying the most important features according to their weighted SHAP values. The default selection method is "lowerCI", which includes features whose lower weighted confidence interval exceeds the predefined 'cutoff' value (default is 0). Alternatively, the "mean" option can be specified, indicating any feature with normalized weighted mean

SHAP contribution above the specified 'cutoff' should be selected. Another alternative options is "shapratio", a method that filters for features where the proportion of their relative weighted SHAP value exceeds the 'cutoff'. This approach calculates the relative contribution of each feature's weighted SHAP value against the aggregate of all features, with those surpassing the 'cutoff' being selected as top feature.

`cutoff` numeric, specifying the cutoff for the method used for selecting the top features. the default is zero, which means that all features with the "method" criteria above zero will be selected.

`top_n_features` integer. if specified, the top n features with the highest weighted SHAP values will be selected, overrullung the 'cutoff' and 'method' arguments. specifying top_n_feature is also a way to reduce computation time, if many features are present in the data set. The default is NULL, which means the shap values will be computed for all features.

`n_models` minimum number of models that should meet the 'minimum_performance' criterion in order to compute WMSHAP and CI. If the intention is to compute global summary SHAP values (at feature level) for a single model, set n_models to 1. The default is 10.

Value

a list including the GGPLOT2 object, the data frame of SHAP values, and performance metric of all models, as well as the model IDs.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

set.seed(10)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
```

```
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

               # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, performance_metric = "aucpr", plot = TRUE)

#########################################################
### PREPARE H2O Grid (takes a couple of minutes)
#########################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                  hyper_params = list(ntrees = seq(1,50,1)),
                  grid_id = "ensemble_grid",

               # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

result2 <- shapley(models = grid, newdata = prostate, performance_metric = "aucpr", plot = TRUE)

#########################################################
### PREPARE autoEnsemble STACKED ENSEMBLE MODEL
#########################################################

### get the models' IDs from the AutoML and grid searches.
### this is all that is needed before building the ensemble,
### i.e., to specify the model IDs that should be evaluated.
library(autoEnsemble)
ids    <- c(h2o.get_ids(aml), h2o.get_ids(grid))
autoSearch <- ensemble(models = ids, training_frame = prostate, strategy = "search")
result3 <- shapley(models = autoSearch, newdata = prostate,
                   performance_metric = "aucpr", plot = TRUE)



## End(Not run)
```

---

shapley.domain *compute and plot weighted mean SHAP contributions at group level (factors or domains)*

Description

This function applies different criteria to visualize SHAP contributions

Usage

```
shapley.domain(
  shapley,
  domains,
  plot = "bar",
  method = "AUTO",
  legendstyle = "continuous",
  scale_colour_gradient = NULL,
  print = FALSE
)
```

Arguments

shapley	object of class 'shapley', as returned by the 'shapley' function
domains	character list, specifying the domains for grouping the features' contributions. Domains are clusters of features' names, that can be used to compute WMSHAP at higher level, along with their 95 better understand how a cluster of features influence the outcome. Note that either of 'features' or 'domains' arguments can be specified at the time.
plot	character, specifying the type of the plot, which can be either 'bar', 'waffle', or 'shap'. The default is 'bar'.
method	character, specifying the method used for identifying the most important features according to their weighted SHAP values. The default selection method is "AUTO", which selects a method based on number of models that have been evaluated because lowerCI method is not applicable to SHAP values of a single model. If 'lowerCI' is specified, features whose lower weighted confidence interval exceeds the predefined 'cutoff' value would be reported. Alternatively, the "mean" option can be specified, indicating any feature with normalized weighted mean SHAP contribution above the specified 'cutoff' should be selected. Another alternative options is "shapratio", a method that filters for features where the proportion of their relative weighted SHAP value exceeds the 'cutoff'. This approach calculates the relative contribution of each feature's weighted SHAP value against the aggregate of all features, with those surpassing the 'cutoff' being selected as top feature.
legendstyle	character, specifying the style of the plot legend, which can be either 'continuous' (default) or 'discrete'. the continuous legend is only applicable to 'shap' plots and other plots only use 'discrete' legend.
scale_colour_gradient	character vector for specifying the color gradients for the plot.
print	logical. if TRUE, the WMSHAP summary table for the given row is printed

Value

ggplot object

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#########################################################
### PLOT THE WEIGHTED MEAN SHAP VALUES
#########################################################

shapley.plot(result, plot = "bar")
shapley.plot(result, plot = "waffle")

## End(Not run)
```

`shapley.feature.selection` *Selects the top features with highest weighted mean shap values based on the specified criteria*

Description

This function specifies the top features and prepares the data for plotting SHAP contributions for each row, or summary of absolute SHAP contributions for each feature.

Usage

```
shapley.feature.selection(
  shapley,
  method = "lowerCI",
  cutoff = 0,
  top_n_features = NULL,
  features = NULL
)
```

Arguments

`shapley`	shapley object
`method`	character, specifying the method used for identifying the most important features according to their weighted SHAP values. The default selection method is "lowerCI", which includes features whose lower weighted confidence interval exceeds the predefined 'cutoff' value (default is relative SHAP of 1 Alternatively, the "mean" option can be specified, indicating any feature with normalized weighted mean SHAP contribution above the specified 'cutoff' should be selected. Another alternative options is "shapratio", a method that filters for features where the proportion of their relative weighted SHAP value exceeds the 'cutoff'. This approach calculates the relative contribution of each feature's weighted SHAP value against the aggregate of all features, with those surpassing the 'cutoff' being selected as top feature.
`cutoff`	numeric, specifying the cutoff for the method used for selecting the top features. the default is zero, which means that all features with the "method" criteria above zero will be selected.
`top_n_features`	integer. if specified, the top n features with the highest weighted SHAP values will be selected, overrullung the 'cutoff' and 'method' arguments.
`features`	character vector, specifying the feature to be plotted.

Value

normalized numeric vector

Author(s)

E. F. Haghish

`shapley.plot` *Plot weighted SHAP contributions*

Description

This function applies different criteria to visualize SHAP contributions

Usage

```
shapley.plot(
  shapley,
  plot = "bar",
  method = "lowerCI",
  cutoff = 0,
  top_n_features = NULL,
  features = NULL,
  legendstyle = "continuous",
  scale_colour_gradient = NULL
)
```

Arguments

`shapley`	object of class 'shapley', as returned by the 'shapley' function
`plot`	character, specifying the type of the plot, which can be either 'bar', 'waffle', or 'shap'. The default is 'bar'.
`method`	character, specifying the method used for identifying the most important features according to their weighted SHAP values. The default selection method is "lowerCI", which includes features whose lower weighted confidence interval exceeds the predefined 'cutoff' value (default is relative SHAP of 1 Alternatively, the "mean" option can be specified, indicating any feature with normalized weighted mean SHAP contribution above the specified 'cutoff' should be selected. Another alternative options is "shapratio", a method that filters for features where the proportion of their relative weighted SHAP value exceeds the 'cutoff'. This approach calculates the relative contribution of each feature's weighted SHAP value against the aggregate of all features, with those surpassing the 'cutoff' being selected as top feature.
`cutoff`	numeric, specifying the cutoff for the method used for selecting the top features.
`top_n_features`	integer. if specified, the top n features with the highest weighted SHAP values will be selected, overrullung the 'cutoff' and 'method' arguments.
`features`	character vector, specifying the feature to be plotted.

legendstyle character, specifying the style of the plot legend, which can be either 'continuous' (default) or 'discrete'. the continuous legend is only applicable to 'shap' plots and other plots only use 'discrete' legend.

scale_colour_gradient
character vector for specifying the color gradients for the plot.

Value

ggplot object

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)
```

```
#######################################################
### PLOT THE WEIGHTED MEAN SHAP VALUES
#######################################################

shapley.plot(result, plot = "bar")
shapley.plot(result, plot = "waffle")

## End(Not run)
```

`shapley.row.plot` *Weighted mean SHAP values computed at subject level*

Description

Weighted mean of SHAP values and weighted SHAP confidence intervals provide a measure of feature importance for a grid of fine-tuned models or base-learners of a stacked ensemble model at subject level, showing that how each feature influences the prediction made for a row in the dataset and to what extend different models agree on that effect. If the 95 vertical line at 0.00, then it can be concluded that the feature does not significantly influences the subject, when variability across models is taken into consideration.

Usage

```
shapley.row.plot(
  shapley,
  row_index,
  features = NULL,
  plot = TRUE,
  print = FALSE
)
```

Arguments

`shapley`	object of class 'shapley', as returned by the 'shapley' function
`row_index`	subject or row number in a wide-format dataset to be visualized
`features`	character vector, specifying the feature to be plotted.
`plot`	logical. if TRUE, the plot is visualized.
`print`	logical. if TRUE, the WMSHAP summary table for the given row is printed

Value

a list including the GGPLOT2 object, the data frame of SHAP values, and performance metric of all models, as well as the model IDs.

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE,
         insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

set.seed(10)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate,
                   performance_metric = "aucpr", plot = TRUE)

#########################################################
### PREPARE H2O Grid (takes a couple of minutes)
#########################################################
# make sure equal number of "nfolds" is specified for different grids
grid <- h2o.grid(algorithm = "gbm", y = y, training_frame = prostate,
                  hyper_params = list(ntrees = seq(1,50,1)),
                  grid_id = "ensemble_grid",

               # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, fold_assignment = "Modulo", nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

result2 <- shapley(models = grid, newdata = prostate,
                    performance_metric = "aucpr", plot = TRUE)
```

```
#########################################################
### PREPARE autoEnsemble STACKED ENSEMBLE MODEL
#########################################################

### get the models' IDs from the AutoML and grid searches.
### this is all that is needed before building the ensemble,
### i.e., to specify the model IDs that should be evaluated.
library(autoEnsemble)
ids    <- c(h2o.get_ids(aml), h2o.get_ids(grid))
autoSearch <- ensemble(models = ids, training_frame = prostate, strategy = "search")
result3 <- shapley(models = autoSearch, newdata = prostate,
                   performance_metric = "aucpr", plot = TRUE)

#plot all important features
shapley.row.plot(shapley, row_index = 11)

#plot only the given features
shapPlot <- shapley.row.plot(shapley, row_index = 11, features = c("PSA", "AGE"))

# inspect the computed data for the row 11
ptint(shapPlot$rowSummarySHAP)

## End(Not run)
```

`shapley.test` *Normalize a vector based on specified minimum and maximum values*

Description

This function normalizes a vector based on specified minimum and maximum values. If the minimum and maximum values are not specified, the function will use the minimum and maximum values of the vector.

Usage

```
shapley.test(shapley, features, n = 5000)
```

Arguments

`shapley`	object of class 'shapley', as returned by the 'shapley' function
`features`	character, name of two features to be compared with permutation test
`n`	integer, number of permutations

Value

normalized numeric vector

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(autoEnsemble)  #autoEnsemble models, particularly useful under severe class imbalance
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.

set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#########################################################
### Significance testing of contributions of two features
#########################################################

shapley.test(result, features = c("GLEASON", "PSA"), n=5000)

## End(Not run)
```

shapley.top *Select top features in a model*

Description

This function applies different criteria simultaniously to identify the most important features in a model. The criteria include: 1) minimum limit of lower weighted confidence intervals of SHAP values relative to the feature with highest SHAP value. 2) minimum limit of percentage of weighted mean SHAP values relative to over all SHAP values of all features. These are specified with two different cutoff values.

Usage

```
shapley.top(shapley, lowerci = 0.01, shapratio = 0.005)
```

Arguments

shapley	object of class 'shapley', as returned by the 'shapley' function
lowerci	numeric, specifying the lower limit of weighted confidence intervals of SHAP values relative to the feature with highest SHAP value. the default is 0.01
shapratio	numeric, specifying the lower limit of percentage of weighted mean SHAP values relative to over all SHAP values of all features. the default is 0.005

Value

data.frame of selected features

Author(s)

E. F. Haghish

Examples

```
## Not run:
# load the required libraries for building the base-learners and the ensemble models
library(h2o)            #shapley supports h2o models
library(shapley)

# initiate the h2o server
h2o.init(ignore_config = TRUE, nthreads = 2, bind_to_localhost = FALSE, insecure = TRUE)

# upload data to h2o cloud
prostate_path <- system.file("extdata", "prostate.csv", package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)

### H2O provides 2 types of grid search for tuning the models, which are
### AutoML and Grid. Below, I demonstrate how weighted mean shapley values
### can be computed for both types.
```

```
set.seed(10)

#########################################################
### PREPARE AutoML Grid (takes a couple of minutes)
#########################################################
# run AutoML to tune various models (GBM) for 60 seconds
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])  #convert to factor for classification
aml <- h2o.automl(y = y, training_frame = prostate, max_runtime_secs = 120,
                  include_algos=c("GBM"),

              # this setting ensures the models are comparable for building a meta learner
                  seed = 2023, nfolds = 10,
                  keep_cross_validation_predictions = TRUE)

### call 'shapley' function to compute the weighted mean and weighted confidence intervals
### of SHAP values across all trained models.
### Note that the 'newdata' should be the testing dataset!
result <- shapley(models = aml, newdata = prostate, plot = TRUE)

#########################################################
### Significance testing of contributions of two features
#########################################################

shapley.top(result, lowerci = 0.01, shapratio = 0.005)

## End(Not run)
```

`test` *Weighted Permutation Test for Difference of Means*

Description

This function performs a weighted permutation test to determine if there is a significant difference between the means of two weighted numeric vectors. It tests the null hypothesis that the difference in means is zero against the alternative that it is not zero.

Usage

```
test(var1, var2, weights, n = 2000)
```

Arguments

`var1`	A numeric vector.
`var2`	A numeric vector of the same length as `var1`.
`weights`	A numeric vector of weights, assumed to be the same for both `var1` and `var2`.
`n`	The number of permutations to perform (default is 2000).

Value

A list containing the observed difference in means and the p-value of the test.

Index