Package 'shapley'

September 18, 2025

Type Package

Title Weighted Mean SHAP and CI for Robust Feature Assessment in ML Grid

Version 0.5.1

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), pander (>= 0.6.5)

RoxygenNote 7.3.2

URL https://github.com/haghish/shapley

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 2025-09-18 13:00:02 UTC

Contents

`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

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

Arguments

shapley	shapley object
method	Character. The column name in summaryShaps used for feature selection. Default is "mean", which selects important features which have weighted mean shap ratio (WMSHAP) higher than the specified cutoff. Other alternative is "lowerCI", which selects features which their lower bound of confidence interval is higher than the cutoff.
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

---

feature.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

```
feature.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.

---

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 Mean SHAP Ratio and Confidence Interval for a ML Grid of Fine-Tuned Models or Base-Learners of a Stacked Ensemble Model*

Description

Calculates weighted mean SHAP ratios and confidence intervals to assess feature importance across a collection of models (e.g., a grid of fine-tuned models or base-learners in a stacked ensemble). Rather than reporting relative SHAP contributions for only a single model, this function accounts for variability in feature importance across multiple models. Each model's performance metric is used as a weight. The function also provides a plot of weighted SHAP values with confidence intervals. Currently, only models trained by the `h2o` machine learning platform, `autoEnsemble`, and the `HMDA` R packages are supported.

Usage

```
shapley(
  models,
  newdata,
  plot = TRUE,
  performance_metric = "r2",
  standardize_performance_metric = FALSE,
  performance_type = "xval",
  minimum_performance = 0,
  method = "mean",
  cutoff = 0.01,
  top_n_features = NULL,
  n_models = 10,
  sample_size = nrow(newdata)
)
```

Arguments

`models`	h2o search grid, autoML grid, or a character vector of H2O model IDs.
`newdata`	An `h2o` frame (or `data.frame`) already uploaded to the `h2o` server. This data will be used for computing SHAP contributions for each model, alongside model's performance weights.
`plot`	logical. if TRUE, the weighted mean and confidence intervals of the SHAP values are plotted. The default is TRUE.
`performance_metric`	Character specifying which performance metric to use as weights. The default is `"r2"`, which can be used for both regression and classification. 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, indicating whether to standardize the performance metric used as weights so their sum equals the number of models. The default is `FALSE`.
`performance_type`	Character. Specify which performance metric should be reported: `"train"` for training data, `"valid"` for validation, or `"xval"` for cross-validation (default).
`minimum_performance`	Numeric. Specify the minimum performance metric for a model to be included in calculating weighted mean SHAP ratio Models below this threshold receive zero weight. The default is `0`.
`method`	Character. Specify the method for selecting important features based on their weighted mean SHAP ratios. The default is `"mean"`, which selects features whose weighted mean shap ratio (WMSHAP) exceeds the `cutoff`. The alternative is `"lowerCI"`, which selects features whose lower bound of confidence interval exceeds the `cutoff`.
`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. 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.

`sample_size` integer. number of rows in the `newdata` that should be used for SHAP assessment. By default, all rows are used, which is the recommended procedure for scientific analyses. However, SHAP analysis is time consuming and in the process of code development, lower values can be used for quicker shapley analyses.

Details

The function works as follows:

1. SHAP contributions are computed at the individual level (row) for each model for the given "newdata".
2. Each model's feature-level SHAP ratios (i.e., share of total SHAP) are computed.
3. The performance metrics of the models are used as weights.
4. Using the weights vector and shap ratio of features for each model, the weighted mean SHAP ratios and their confidence intervals are computed.

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",
  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'.
`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")

#########################################################
### DEFINE DOMAINS (GROUPS OF FEATURES OR FACTORS)
#########################################################
shapley.domain(shapley = shapley, plot = "bar",
               domains = list(Demographic = c("RACE", "AGE"),
                              Cancer = c("VOL", "PSA", "GLEASON"),
                              Tests = c("DPROS", "DCAPS")),
                              print = TRUE
```

```
## End(Not run)
```

---

shapley.feature.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.feature.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.plot *Plot weighted SHAP contributions*

Description

This function applies different criteria to visualize SHAP contributions

Usage

```
shapley.plot(
  shapley,
  plot = "bar",
  method = "mean",
  cutoff = 0.01,
  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. The column name in summaryShaps used for feature selection. Default is "mean", which selects important features which have weighted mean shap ratio (WMSHAP) higher than the specified cutoff. Other alternative is "lowerCI", which selects features which their lower bound of confidence interval is higher than the cutoff.
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")

## 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.table` *Create SHAP Summary Table Based on the Given Criterion*

Description

Generates a summary table of weighted mean SHAP (WMSHAP) values and confidence intervals for each feature based on a weighted SHAP analysis. The function filters the SHAP summary table

(from a wmshap object) by selecting features that meet or exceed a specified cutoff using a selection method (default "mean", which is weighted mean shap ratio). It then sorts the table by the mean SHAP value, formats the SHAP values along with their 95% confidence intervals into a single string, and optionally adds human-readable feature descriptions from a provided dictionary. The output is returned as a markdown table using the **pander** package, or as a data frame if requested.

Usage

```
shapley.table(
  wmshap,
  method = "mean",
  cutoff = 0.01,
  round = 3,
  exclude_features = NULL,
  dict = NULL,
  markdown.table = TRUE,
  split.tables = 120,
  split.cells = 50
)
```

Arguments

wmshap	A wmshap object, returned by the shapley function containing a data frame summaryShaps.
method	Character. The column name in summaryShaps used for feature selection. Default is "mean", which selects important features which have weighted mean shap ratio (WMSHAP) higher than the specified cutoff. Other alternative is "lowerCI", which selects features which their lower bound of confidence interval is higher than the cutoff.
cutoff	Numeric. The threshold cutoff for the selection method; only features with a value in the method column greater than or equal to this value are retained. Default is 0.01.
round	Integer. The number of decimal places to round the SHAP mean and confidence interval values. Default is 3.
exclude_features	Character vector. A vector of feature names to be excluded from the summary table. Default is NULL.
dict	A data frame containing at least two columns named "name" and "description". If provided, the function uses this dictionary to add human-readable feature descriptions. Default is NULL.
markdown.table	Logical. If TRUE, the output is formatted as a markdown table using the **pander** package; otherwise, a data frame is returned. Default is TRUE.
split.tables	Integer. Controls table splitting in pander(). Default is 120.
split.cells	Integer. Controls cell splitting in pander(). Default is 50.

Value

If `markdown.table = TRUE`, returns a markdown table (invisibly) showing two columns: `"Description"` and `"WMSHAP"`. If `markdown.table = FALSE`, returns a data frame with these columns.

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)

# get the output as a Markdown table:
md_table <- shapley.table(wmshap = result2,
                          method = "mean",
                          cutoff = 0.01,
                          round = 3,
                          markdown.table = TRUE)
head(md_table)

## 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, mean = 0.01, lowerCI = 0.01)
```

Arguments

`shapley`	object of class 'shapley', as returned by the 'shapley' function
`mean`	Numeric. specifying the cutoff of weighted mean SHAP ratio (WMSHAP). The default is 0.01. Lower values will be more generous in defining "importance", while higher values are more restrictive. However, these default values are not generalizable to all situations and algorithms.
`lowerCI`	numeric. Specifying the limit of lower bound of 95% WMSHAP The default is 0.01. Lower values will be more generous in defining "importance", while higher values are more restrictive. However, these default values are not generalizable to all situations and algorithms.

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, mean = 0.005, lowerCI = 0.01)

## End(Not run)
```

Index