Package ‘treeshap’
January 22, 2024

Title Compute SHAP Values for Your Tree-Based Models Using the 'TreeSHAP' Algorithm

Version 0.3.1

Description An efficient implementation of the 'TreeSHAP' algorithm introduced by Lundberg et al., (2020) <doi:10.1038/s42256-019-0138-9>. It is capable of calculating SHAP (SHapley Additive exPlanations) values for tree-based models in polynomial time. Currently supported models include 'gbm', 'randomForest', 'ranger', 'xgboost', 'lightgbm'.

License GPL-3


BugReports https://github.com/ModelOriented/treeshap/issues

Depends R (>= 2.10)

Imports data.table, ggplot2, Rcpp

Suggests gbm, jsonlite, lightgbm, randomForest, ranger, scales, survival, testthat, xgboost

LinkingTo Rcpp

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation yes

Author Konrad Komisarczyk [aut], Pawel Kozminski [aut], Szymon Maksymiuk [aut] (<https://orcid.org/0000-0002-3120-1601>), Lorenz A. Kapsner [ctb] (<https://orcid.org/0000-0003-1866-860X>), Mikolaj Spytek [ctb] (<https://orcid.org/0000-0001-7111-2286>), Mateusz Krzyzinski [ctb, cre] (<https://orcid.org/0000-0001-6143-488X>), Przemyslaw Biecek [aut, cph] (<https://orcid.org/0000-0001-8423-1823>)

Maintainer Mateusz Krzyzinski <krzyzinskimateusz23@gmail.com>

Repository CRAN

Date/Publication 2024-01-22 15:20:06 UTC
## Description
DrWhy color palettes for ggplot objects

## Usage
```r
colors_discrete_drwhy(n = 2)
colors_breakdown_drwhy()
```

## Arguments
- `n`: number of colors for color palette
Value

color palette as vector of characters

Description

Dataset consists of 56 columns, 55 numeric and one of type factor 'work_rate'. value_eur is a potential target feature.

Usage

fifa20

Format

A data frame with 18278 rows and 56 columns. Most of variables representing skills are in range from 0 to 100 and will not be described here. To list non obvious features:

- **overall**  Overall score of player's skills
- **potential**  Potential of a player, younger players tend to have higher level of potential
- **value_eur**  Market value of a player (in mln EUR)
- **international_reputation**  Range 1 to 5
- **weak_foot**  Range 1 to 5
- **skill_moves**  Range 1 to 5
- **work_rate**  Divided by slash levels of willingness to work in offense and defense respectively

Source

"Data has been scraped from the publicly available website [https://sofifa.com](https://sofifa.com) [https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset]"
gbm.unify  
*Unify GBM model*

**Description**

Convert your GBM model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in `treeshap()` function.

**Usage**

```r
gbm.unify(gbm_model, data)
```

**Arguments**

- `gbm_model`: An object of `gbm` class. At the moment, models built on data with categorical features are not supported - please encode them before training.
- `data`: Reference dataset. A `data.frame` or `matrix` with the same columns as in the training set of the model. Usually dataset used to train model.

**Value**

A unified model representation - a `model_unified.object` object

**See Also**

- `lightgbm.unify` for LightGBM models
- `xgboost.unify` for XGBoost models
- `ranger.unify` for ranger models
- `randomForest.unify` for randomForest models

**Examples**

```r
library(gbm)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
data['value_eur'] <- fifa20$target
gbm_model <- gbm::gbm(
  formula = value_eur ~ .,
  data = data,
  distribution = 'gaussian',
  n.trees = 20,
  interaction.depth = 4,
  n.cores = 1)
unified_model <- gbm.unify(gbm_model, data)
shaps <- treeshap(unified_model, data[1:2,])
plot_contribution(shaps, obs = 1)
```
**is.model_unified**

Check whether object is a valid `model_unified` object

**Description**

Does not check correctness of representation, only basic checks

**Usage**

```r
is.model_unified(x)
```

**Arguments**

- `x`: an object to check

**Value**

`boolean`

---

**is.treeshap**

Check whether object is a valid `treeshap` object

**Description**

Does not check correctness of result, only basic checks

**Usage**

```r
is.treeshap(x)
```

**Arguments**

- `x`: an object to check

**Value**

`boolean`
**Description**

Convert your LightGBM model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in `treeshap()` function.

**Usage**

```r
lightgbm.unify(lgb_model, data, recalculate = FALSE)
```

**Arguments**

- `lgb_model`: A lightgbm model - object of class `lgb.Booster`
- `data`: Reference dataset. A data.frame or matrix with the same columns as in the training set of the model. Usually dataset used to train model.
- `recalculate`: logical indicating if covers should be recalculated according to the dataset given in `data`. Keep it `FALSE` if training data are used.

**Value**

a unified model representation - a `model_unified.object` object

**See Also**

- `gbm.unify` for GBM models
- `xgboost.unify` for XGBoost models
- `ranger.unify` for ranger models
- `randomForest.unify` for randomForest models

**Examples**

```r
library(lightgbm)
param_lgbm <- list(objective = "regression", max_depth = 2,
                    force_row_wise = TRUE, num_iterations = 20)
data_fifa <- fifa20$data[, !colnames(fifa20$data) %in%
                          c("work_rate", "value_eur", "gk_diving", "gk_handling",
                          "gk_kicking", "gk_reflexes", "gk_speed", "gk_positioning")]
data <- na.omit(cbind(data_fifa, fifa20$target))sparse_data <- as.matrix(data[, -ncol(data)])x <- lightgbm::lgb.Dataset(sparse_data, label = as.matrix(data[, ncol(data)]))lgb_data <- lightgbm::lgb.Dataset.construct(x)lgb_model <- lightgbm::lightgbm(data = lgb_data, params = param_lgbm, verbose = -1, num_threads = 0)unified_model <- lightgbm.unify(lgb_model, sparse_data)
```
shaps <- treeshap(unified_model, data[, 1:2, ])
plot_contribution(shaps, obs = 1)

---

**model_unified.object**  
*Unified model representation*

**Description**

`model_unified` object produced by `*.unify` or `unify` function.

**Value**

List consisting of two elements:

- **model** - A `data.frame` representing model with following columns:
  - **Tree** - 0-indexed ID of a tree
  - **Node** - 0-indexed ID of a node in a tree. In a tree the root always has ID 0
  - **Feature** - In case of an internal node - name of a feature to split on. Otherwise - NA
  - **Decision.type** - A factor with two levels: "<" and "<=". In case of an internal node - predicate used for splitting observations. Otherwise - NA
  - **Split** - For internal nodes threshold used for splitting observations. All observations that satisfy the predicate `Decision.type(Split)` ("< Split" / "<= Split") are proceeded to the node marked as 'Yes'. Otherwise to the 'No' node. For leaves - NA
  - **Yes** - Index of a row containing a child Node. Thanks to explicit indicating the row it is much faster to move between nodes
  - **No** - Index of a row containing a child Node
  - **Missing** - Index of a row containing a child Node where are proceeded all observations with no value of the dividing feature
  - **Prediction** - For leaves: Value of prediction in the leaf. For internal nodes: NA
  - **Cover** - Number of observations seen by the internal node or collected by the leaf for the reference dataset

- **data** - Dataset used as a reference for calculating SHAP values. A dataset passed to the `*.unify`, `unify` or `set_reference_dataset` function with `data` argument. A `data.frame`.
  Object has two also attributes set:
  - **model** - A string. By what package the model was produced.
  - **missing_support** - A boolean. Whether the model allows missing values to be present in explained dataset.

**See Also**

`unify`
model_unified_multioutput.object

Unified model representations for multi-output model

Description

model_unified_multioutput object produced by *.unify or unify function.

Value

List consisting of model_unified objects, one for each individual output of a model. For survival models, the list is named using the time points, for which predictions are calculated.

See Also

unify

plot_contribution

SHAP value based Break-Down plot

Description

This function plots contributions of features into the prediction for a single observation.

Usage

plot_contribution(
  treeshap,  
  obs = 1,  
  max_vars = 5,  
  min_max = NA,  
  digits = 3,  
  explain_deviation = FALSE,  
  title = "SHAP Break-Down",  
  subtitle = ""  
)

Arguments

treeshap  
  A treeshap object produced with the treeshap function. treeshap.object.

obs  
  A numeric indicating which observation should be plotted. By default it's first observation.

max_vars  
  maximum number of variables that shall be presented. Variables with the highest importance will be presented. Remaining variables will be summed into one additional contribution. By default 5.
plot_feature_dependence

min_max a range of OX axis. By default NA, therefore it will be extracted from the contributions of x. But it can be set to some constants, useful if these plots are to be used for comparisons.
digits number of decimal places (round) to be used.
explain_deviation if TRUE then instead of explaining prediction and plotting intercept bar, only deviation from mean prediction of the reference dataset will be explained. By default FALSE.
title the plot's title, by default 'SHAP Break-Down'.
subtitle the plot's subtitle. By default no subtitle.

Value

a ggplot2 object

See Also

treeshap for calculation of SHAP values
plot_feature_importance, plot_feature_dependence, plot_interaction

Examples

library(xgboost)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
target <- fifa20$target
param <- list(objective = "reg:squarederror", max_depth = 3)
xgb_model <- xgboost::xgboost(as.matrix(data), params = param, label = target,
nrounds = 20, verbose = FALSE)
unified_model <- xgboost.unify(xgb_model, as.matrix(data))
x <- head(data, 1)
shap <- treeshap(unified_model, x)
plot_contribution(shap, 1, min_max = c(0, 12000000))
plot_feature_importance

Usage

plot_feature_dependence(
  treeshap,
  variable,
  title = "Feature Dependence",
  subtitle = NULL
)

Arguments

treeshap  A treeshap object produced with the `treeshap` function. `treeshap.object`.
variable  name or index of variable for which feature dependence will be plotted.
title     the plot's title, by default 'Feature Dependence'.
subtitle   the plot's subtitle. By default no subtitle.

Value

a ggplot2 object

See Also

`treeshap` for calculation of SHAP values

`plot_contribution, plot_feature_importance, plot_interaction`

Examples

```r
library(xgboost)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
target <- fifa20$target
param <- list(objective = "reg:squarederror", max_depth = 3)
xgb_model <- xgboost::xgboost(as.matrix(data), params = param, label = target, nrounds = 20, verbose = FALSE)
unified_model <- xgboost.unify(xgb_model, as.matrix(data))
x <- head(data, 100)
shaps <- treeshap(unified_model, x)
plot_feature_dependence(shaps, variable = "overall")
```

---

plot_feature_importance

SHAP value based Feature Importance plot

Description

This function plots feature importance calculated as means of absolute values of SHAP values of variables (average impact on model output magnitude).
Usage

plot_feature_importance(
  treeshap,
  desc_sorting = TRUE,
  max_vars = ncol(shaps),
  title = "Feature Importance",
  subtitle = NULL
)

Arguments

treeshap A treeshap object produced with the treeshap function. treeshap.object.
desc_sorting logical. Should the bars be sorted descending? By default TRUE.
max_vars maximum number of variables that shall be presented. By default all are presented.
title the plot’s title, by default ‘Feature Importance’.
subtitle the plot’s subtitle. By default no subtitle.

Value

a ggplot2 object

See Also

treeshap for calculation of SHAP values
plot_contribution, plot_feature_dependence, plot_interaction

Examples

library(xgboost)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
target <- fifa20$target
param <- list(objective = "reg:squarederror", max_depth = 3)
xgb_model <- xgboost::xgboost(as.matrix(data), params = param, label = target,
nrounds = 20, verbose = FALSE)
unified_model <- xgboost.unify(xgb_model, as.matrix(data))
shaps <- treeshap(unified_model, as.matrix(head(data, 3)))
plot_feature_importance(shaps, max_vars = 4)
plot_interaction

Description

This function plots SHAP Interaction value for two variables depending on the value of the first variable. Value of the second variable is marked with the color.

Usage

```
plot_interaction(
  treeshap,  # A treeshap object produced with `treeshap(interactions = TRUE)` function. treeshap.object.
  var1,  # name or index of the first variable - plotted on x axis.
  var2,  # name or index of the second variable - marked with color.
  title = "SHAP Interaction Value Plot",  # the plot's title, by default 'SHAP Interaction Value Plot'.
  subtitle = ""
)  # the plot's subtitle. By default no subtitle.
```

Arguments

- `treeshap`  
  A treeshap object produced with `treeshap(interactions = TRUE)` function. `treeshap.object`.
- `var1`  
  name or index of the first variable - plotted on x axis.
- `var2`  
  name or index of the second variable - marked with color.
- `title`  
  the plot's title, by default 'SHAP Interaction Value Plot'.
- `subtitle`  
  the plot's subtitle. By default no subtitle.

Value

a ggplot2 object

See Also

- `treeshap` for calculation of SHAP Interaction values
- `plot_contribution`, `plot_feature_importance`, `plot_feature_dependence`

Examples

```
data <- fifa20$data[!colnames(fifa20$data) == 'work_rate']  
target <- fifa20$target  
param2 <- list(objective = "reg:squarederror", max_depth = 5)  
xgb_model2 <- xgboost::xgboost(as.matrix(data), params = param2, label = target, nrounds = 10)  
unified_model2 <- xgboost.unify(xgb_model2, data)  
inters <- treeshap(unified_model2, as.matrix(data[1:50, ]), interactions = TRUE)  
plot_interaction(inters, "dribbling", "defending")
```
predict.model_unified

---

**predict.model_unified**  
*Predict*

---

### Description

Predict using unified_model representation.

### Usage

```r
## S3 method for class 'model_unified'
predict(object, x, ...)
```

### Arguments

- **object**
  - Unified model representation of the model created with a (model).unify function. `model_unified.object`
- **x**
  - Observations to predict. A `data.frame` or `matrix` with the same columns as in the training set of the model.
- **...**
  - other parameters

### Value

a vector of predictions.

### Examples

```r
library(gbm)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
data[,'value_eur'] <- fifa20$target
gbm_model <- gbm::gbm(
  formula = value_eur ~ .,
  data = data,
  distribution = "laplace",
  n.trees = 20,
  interaction.depth = 4,
  n.cores = 1)
unified <- gbm.unify(gbm_model, data)
predict(unified, data[2001:2005, ])
```
print.model_unified   Prints model_unified objects

Description
Prints model_unified objects

Usage
## S3 method for class 'model_unified'
print(x, ...)

Arguments
  x          a model_unified object
  ...        other arguments

Value
No return value, called for printing

print.model_unified_multioutput
   Prints model_unified_multioutput objects

Description
Prints model_unified_multioutput objects

Usage
## S3 method for class 'model_unified_multioutput'
print(x, ...)

Arguments
  x          a model_unified_multioutput object
  ...        other arguments

Value
No return value, called for printing
print.treeshap  Prints treeshap objects

Description
Prints treeshap objects

Usage

## S3 method for class 'treeshap'
print(x, ...)

Arguments

x  a treeshap object
...
other arguments

Value
No return value, called for printing

print.treeshap_multioutput  Prints treeshap_multioutput objects

Description
Prints treeshap_multioutput objects

Usage

## S3 method for class 'treeshap_multioutput'
print(x, ...)

Arguments

x  a treeshap_multioutput object
...
other arguments

Value
No return value, called for printing
randomForest.unify  Unify randomForest model

Description

Convert your randomForest model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in treeshap() function.

Usage

randomForest.unify(rf_model, data)

Arguments

rf_model  An object of randomForest class. At the moment, models built on data with categorical features are not supported - please encode them before training.
data  Reference dataset. A data.frame or matrix with the same columns as in the training set of the model. Usually dataset used to train model.

Details

Binary classification models with a target variable that is a factor with two levels, 0 and 1, are supported

Value

a unified model representation - a model_unified.object object

See Also

lightgbm.unify for LightGBM models
gbm.unify for GBM models
xgboost.unify for XGBoost models
ranger.unify for ranger models

Examples

library(randomForest)
data_fifa <- fifa20$data[!colnames(fifa20$data) %in%
c("work_rate", "value_eur", "gk_diving", "gk_handling",
   "gk_kicking", "gk_reflexes", "gk_speed", "gk_positioning")]
data <- na.omit(cbind(data_fifa, target = fifa20$target))

rf <- randomForest::randomForest(target~., data = data, maxnodes = 10, ntree = 10)
unified_model <- randomForest.unify(rf, data)
shaps <- treeshap(unified_model, data[1:2,])
ranger.unify

# plot_contribution(shaps, obs = 1)

---

**ranger.unify**

*Unify ranger model*

**Description**

Convert your ranger model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in `treeshap()` function.

**Usage**

```
ranger.unify(rf_model, data)
```

**Arguments**

- `rf_model` An object of `ranger` class. At the moment, models built on data with categorical features are not supported - please encode them before training.
- `data` Reference dataset. A `data.frame` or `matrix` with the same columns as in the training set of the model. Usually dataset used to train model.

**Value**

A unified model representation - a `model_unified.object` object

**See Also**

- `lightgbm.unify` for LightGBM models
- `gbm.unify` for GBM models
- `xgboost.unify` for XGBoost models
- `randomForest.unify` for randomForest models

**Examples**

```r
library(ranger)
data_fifa <- fifa20$data[!colnames(fifa20$data) %in%
c('work_rate', 'value_eur', 'gk_diving', 'gk_handling',
'gk_kicking', 'gk_reflexes', 'gk_speed', 'gk_positioning')]data <- na.omit(cbind(data_fifa, target = fifa20$target))

rf <- ranger::ranger(target~., data = data, max.depth = 10, num.trees = 10)unified_model <- ranger.unify(rf, data)shaps <- treeshap(unified_model, data[1:2,])plot_contribution(shaps, obs = 1)
```
ranger_surv.unify  

Unify ranger survival model

Description
Convert your ranger model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in `treeshap()` function.

Usage
```
ranger_surv.unify(
  rf_model, data,
  type = c("risk", "survival", "chf"),
  times = NULL
)
```

Arguments
- `rf_model`: An object of `ranger` class. At the moment, models built on data with categorical features are not supported - please encode them before training.
- `data`: Reference dataset. A `data.frame` or `matrix` with the same columns as in the training set of the model. Usually dataset used to train model.
- `type`: A character to define the type of model prediction to use. Either "risk" (default), which uses the risk score calculated as a sum of cumulative hazard function values, "survival", which uses the survival probability at certain time-points for each observation, or "chf", which used the cumulative hazard values at certain time-points for each observation.
- `times`: A numeric vector of unique death times at which the prediction should be evaluated. By default `unique.death.times` from model are used.

Details
The survival forest implemented in the `ranger` package stores cumulative hazard functions (CHFs) in the leaves of survival trees, as proposed for Random Survival Forests (Ishwaran et al. 2008). The final model prediction is made by averaging these CHFs from all the trees. To provide explanations in the form of a survival function, the CHFs from the leaves are converted into survival functions (SFs) using the formula $SF(t) = \exp(-\text{CHF}(t))$. However, it is important to note that averaging these SFs does not yield the correct model prediction as the model prediction is the average of CHFs transformed in the same way. Therefore, when you obtain explanations based on the survival function, they are only proxies and may not be fully consistent with the model predictions obtained using for example `predict` function.
Value

For type = "risk" a unified model representation is returned - a \texttt{model\_unified\_object} object. For type = "survival" or type = "chf" - a \texttt{model\_unified\_multioutput\_object} object is returned, which is a list that contains unified model representation (\texttt{model\_unified\_object} object) for each time point. In this case, the list names are time points at which the survival function was evaluated.

See Also

rn\_surv\_unify for regression and classification ranger models
lightgbm\_unify for LightGBM models
gbm\_unify for GBM models
xgboost\_unify for XGBoost models
randomForest\_unify for randomForest models

Examples

```r
library(ranger)
data_colon <- data.table::data.table(survival::colon)
data_colon <- na.omit(data_colon[get("etype") == 2, ])
surv_cols <- c("status", "time", "rx")

feature_cols <- colnames(data_colon)[3:(ncol(data_colon) - 1)]

train_x <- model.matrix(~ -1 + ., data_colon[, .SD, .SDcols = setdiff(feature_cols, surv_cols[1:2])])
train_y <- survival::Surv(event = (data_colon[, get("status")]) |> as.character() |> as.integer(),
                         time = data_colon[, get("time")], type = "right")

rf <- ranger::ranger(
x = train_x,
y = train_y,
data = data_colon,
max.depth = 10,
num.trees = 10)

unified_model_risk <- ranger_surv.unify(rf, train_x, type = "risk")
shaps <- treeshap(unified_model_risk, train_x[1:2,])

# compute shaps for 3 selected time points
unified_model_surv <- ranger_surv.unify(rf, train_x, type = "survival", times = c(23, 50, 73))
shaps_surv <- treeshap(unified_model_surv, train_x[1:2,])
```
Description

Change a dataset used as reference for calculating SHAP values. Reference dataset is initially set with `data` argument in unifying function. Usually reference dataset is dataset used to train the model. Important property of reference dataset is that SHAP values for each observation add up to its deviation from mean prediction for a reference dataset.

Usage

```r
set_reference_dataset(unified_model, x)
```

Arguments

- `x` Reference dataset. A `data.frame` or `matrix` with the same columns as in the training set of the model.

Value

`model_unified.object`. Unified representation of the model as created with a `(model).unify` function, but with changed reference dataset (Cover column containing updated values).

See Also

- `lightgbm.unify` for LightGBM models
- `gbm.unify` for GBM models
- `xgboost.unify` for XGBoost models
- `ranger.unify` for ranger models
- `randomForest.unify` for randomForest models

Examples

```r
library(gbm)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
data['value_eur'] <- fifa20$target
gbm_model <- gbm::gbm(
  formula = value_eur ~ .,
  data = data,
  distribution = "laplace",
  n.trees = 20,
)`
interaction.depth = 4,
n.cores = 1)
unified <- gbm.unify(gbm_model, data)
set_reference_dataset(unified, data[200:700, ])

---

**theme_drwhy**

*DrWhy Theme for ggplot objects*

**Description**

DrWhy Theme for ggplot objects

**Usage**

```r
theme_drwhy()
theme_drwhy_vertical()
```

**Value**

theme for ggplot2 objects

---

**treeshap**

*Calculate SHAP values of a tree ensemble model.*

**Description**

Calculate SHAP values and optionally SHAP Interaction values.

**Usage**

```r
treeshap(unified_model, x, interactions = FALSE, verbose = TRUE)
```

**Arguments**

- `unified_model`  
  Unified data.frame representation of the model created with a (model).unify function. A `model_unified.object` object.

- `x`  
  Observations to be explained. A `data.frame` or `matrix` object with the same columns as in the training set of the model. Keep in mind that objects different than `data.frame` or plain `matrix` will cause an error or unpredictable behavior.

- `interactions`  
  Whether to calculate SHAP interaction values. By default is `FALSE`. Basic SHAP values are always calculated.

- `verbose`  
  Whether to print progress bar to the console. Should be logical. Progress bar will not be displayed on Windows.
Value

A `treeshap.object` object (for single-output models) or `treeshap_multioutput.object`, which is a list of `treeshap.object` objects (for multi-output models). SHAP values can be accessed from `treeshap.object` with `$shaps`, and interaction values can be accessed with `$interactions`.

See Also

`xgboost.unify` for XGBoost models `lightgbm.unify` for LightGBM models `gbm.unify` for GBM models `randomForest.unify` for randomForest models `ranger.unify` for ranger models `ranger_surv.unify` for ranger survival models

Examples

```r
library(xgboost)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
target <- fifa20$target

# calculating simple SHAP values
param <- list(objective = "reg:squarederror", max_depth = 3)
xgb_model <- xgboost::xgboost(as.matrix(data), params = param, label = target, nrounds = 20, verbose = FALSE)
unified_model <- xgboost.unify(xgb_model, as.matrix(data))
treeshap1 <- treeshap(unified_model, head(data, 3))
plot_contribution(treeshap1, obs = 1)
treeshap1$shaps

# It's possible to calculate explanation over different part of the data set
unified_model_rec <- set_reference_dataset(unified_model, data[1:1000, ])
treeshap_rec <- treeshap(unified_model, head(data, 3))
plot_contribution(treeshap_rec, obs = 1)

treeshap2$interactions
```

treeshap.object  
treeshap results

Description

treeshap object produced by treeshap function.
treeshap_multioutput.object

Value

List consisting of four elements:

- **shaps** A data.frame with M columns, X rows (M - number of features, X - number of explained observations). Every row corresponds to SHAP values for an observation.

- **interactions** An array with dimensions (M, M, X) (M - number of features, X - number of explained observations). Every $[,] , i$ slice is a symmetric matrix - SHAP Interaction values for an observation. $[a, b, i]$ element is SHAP Interaction value of features a and b for observation i. Is NULL if interactions where not calculated (parameter interactions set FALSE.)

- **unified_model** An object of type model_unified.object. Unified representation of a model for which SHAP values were calculated. It is used by some of the plotting functions.

- **observations** Explained dataset. data.frame or matrix. It is used by some of the plotting functions.

See Also

treeshap,
plot_contribution, plot_feature_importance, plot_feature_dependence, plot_interaction

treeshap_multioutput.object

*treeshap results for multi-output model*

Description

treeshap_multioutput object produced by treeshap function.

Value

List consisting of treeshap objects, one for each individual output of a model. For survival models, the list is named using the time points, for which TreeSHAP values are calculated.

See Also

treeshap,
treeshap.object
unify

Unify tree-based model

Description

Convert your tree-based model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in `treeshap()` function.

Usage

```r
unify(model, data, ...)
```

Arguments

- **model**: A tree-based model object of any supported class (`gbm`, `lgb.Booster`, `randomForest`, `ranger`, or `xgb.Booster`).
- **data**: Reference dataset. A `data.frame` or `matrix` with the same columns as in the training set of the model. Usually dataset used to train model.
- **...**: Additional parameters passed to the model-specific unification functions.

Value

A unified model representation - a `model_unified.object` object (for single-output models) or `model_unified_multioutput.object`, which is a list of `model_unified.object` objects (for multi-output models).

See Also

- `lightgbm.unify` for LightGBM models
- `gbm.unify` for GBM models
- `xgboost.unify` for XGBoost models
- `ranger.unify` for ranger models
- `randomForest.unify` for randomForest models

Examples

```r
library(ranger)
data_fifa <- fifa20$data[!colnames(fifa20$data) %in%
  c('work_rate', 'value_eur', 'gk_diving', 'gk_handling',
  'gk_kicking', 'gk_reflexes', 'gk_speed', 'gk_positioning')]
data <- na.omit(cbind(data_fifa, target = fifa20$target))

rf1 <- ranger::ranger(target~., data = data, max.depth = 10, num.trees = 10)
unified_model1 <- unify(rf1, data)
shaps1 <- treeshap(unified_model1, data[1:2,])
plot_contribution(shaps1, obs = 1)
```
xgboost.unify

Unify XGBoost model

Description

Convert your XGBoost model into a standardized representation. The returned representation is easy to be interpreted by the user and ready to be used as an argument in `treeshap()` function.

Usage

```r
xgboost.unify(xgb_model, data, recalculate = FALSE)
```

Arguments

- `xgb_model`: A XGBoost model - object of class `xgb.Booster`
- `data`: Reference dataset. A `data.frame` or `matrix` with the same columns as in the training set of the model. Usually dataset used to train model.
- `recalculate`: logical indicating if covers should be recalculated according to the dataset given in `data`. Keep it `FALSE` if training data are used.

Value

- a unified model representation - a `model_unified.object` object

See Also

- `lightgbm.unify` for LightGBM models
- `gbm.unify` for GBM models
- `ranger.unify` for ranger models
- `randomForest.unify` for randomForest models

Examples

```r
library(xgboost)
data <- fifa20$data[colnames(fifa20$data) != 'work_rate']
target <- fifa20$target
param <- list(objective = "reg:squarederror", max_depth = 3)
xgb_model <- xgboost::xgboost(as.matrix(data), params = param, label = target,
    nrounds = 20, verbose = 0)
unified_model <- xgboost.unify(xgb_model, as.matrix(data))
```
shaps <- treeshap(unified_model, data[1:2,])
plot_contribution(shaps, obs = 1)
Index

* datasets
  fifa20, 3

  colors_breakdown_drwhy
    (colors_discrete_drwhy), 2
  colors_discrete_drwhy, 2

  fifa20, 3

  gbm.unify, 4, 6, 16, 17, 19, 20, 22, 24, 25
  is.model_unified, 5
  is.treeshap, 5

  lightgbm.unify, 4, 6, 16, 17, 19, 20, 22, 24, 25

  model_unified.object, 4, 6, 7, 13, 16, 17,
  19–21, 23–25
  model_unified_multioutput.object, 8, 19,
  24

  plot_contribution, 8, 10–12, 23
  plot_feature_dependence, 9, 9, 11, 12, 23
  plot_feature_importance, 9, 10, 10, 12, 23
  plot_interaction, 9–11, 12, 23
  predict.model_unified, 13
  print.model_unified, 14
  print.model_unified_multioutput, 14
  print.treeshap, 15
  print.treeshap_multioutput, 15

  randomForest.unify, 4, 6, 16, 17, 19, 20, 22,
  24, 25
  ranger.unify, 4, 6, 16, 17, 19, 20, 22, 24, 25
  ranger_surv.unify, 18, 22
  round, 9

  set_reference_dataset, 7, 20

  theme_drwhy, 21

theme_drwhy_vertical (theme_drwhy), 21
treeshap, 8–12, 21, 23
treeshap.object, 8, 10–12, 22, 22, 23
treeshap_multioutput.object, 22, 23

  unify, 7, 8, 24
xgboost.unify, 4, 6, 16, 17, 19, 20, 22, 24, 25

27