Package 'SHAPforxgboost'

August 28, 2019

Title SHAP Plots for 'XGBoost'

Version 0.0.2

Description The aim of 'SHAPforxgboost' is to aid in visual data investigations
using SHAP (SHapley Additive exPlanation) visualization plots for 'XGBoost'.
It provides summary plot, dependence plot, interaction plot, and force plot.
It relies on the 'dmlc/xgboost' package to produce SHAP values.
Please refer to 'slundberg/shap' for the original implementation of SHAP in 'Python'.

License MIT + file LICENSE

URL `https://github.com/liuyanguu/SHAPforxgboost`

BugReports `https://github.com/liuyanguu/SHAPforxgboost/issues`

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0)

Imports ggplot2 (>= 3.0.0), xgboost (>= 0.81.0.0), data.table (>=
1.12.0), ggforce (>= 0.2.1.9000), ggExtra (>= 0.8),
RColorBrewer (>= 1.1.2), ggpubr, BBmisc

Suggests gridExtra (>= 2.3), here, parallel

RoxygenNote 6.1.1.9000

NeedsCompilation no

Author Yang Liu [aut, cre] (<https://orcid.org/0000-0001-6557-6439>),
Allan Just [ctb] (<https://orcid.org/0000-0003-4312-5957>)

Maintainer Yang Liu `<lyhello@gmail.com>`

Repository CRAN

Date/Publication 2019-08-28 13:50:02 UTC

R topics documented:

`dataXY_df` *Terra satellite data (X,Y) for running the xgboost model .*

Description

Data.table, contains 9 features, and about 10,000 observations

Usage

```
dataXY_df
```

Format

An object of class `data.table` (inherits from `data.frame`) with 10148 rows and 10 columns.

References

http://doi.org/10.5281/zenodo.3334713

`label.feature` *helper function to modify labels for features under plotting*

Description

If a list is created in the environment named **new_labels** (`!is.null(new_labels)`, the plots will use that list to replace default list of labels `labels_within_package`.

Usage

```
label.feature(x)
```

Arguments

`x`	variable names

Value

a character, e.g. "date", "Time Trend", etc.

`labels_within_package` *labels_within_package: Some labels package auther defined to make his plot, mainly serve the paper publication.*

Description

It contains a list that match each feature to its labels. It is used in the function `label.feature`.

Usage

```
labels_within_package
```

Format

An object of class `list` of length 20.

Details

labels_within_package <- list(dayint = "Time trend", diffcwv = "delta CWV (cm)", date = "", Column_WV = "MAIAC CWV (cm)", AOT_Uncertainty = "Blue band uncertainty", elev = "Elevation (m)", aod = "Aerosol optical depth", RelAZ = "Relative azimuth angle", DevAll_P1km = expression(paste("Proportion developed area in 1",km^2)), dist_water_km = "Distance to water (km)", forestProp_1km = expression(paste("Proportion of forest in 1",km^2)), Aer_optical_depth = "DSCOVR EPIC MAIAC AOD400nm", aer_aod440 = "AERONET AOD440nm", aer_aod500 = "AERONET AOD500nm", diff440 = "DSCOVR MAIAC - AERONET AOD", diff440_pred = "Predicted Error", aer_aod440_hat = "Predicted AERONET AOD440nm", AOD_470nm = "AERONET AOD470nm", Optical_Depth_047_t = "MAIAC AOD470nm (Terra)", Optical_Depth_047_a = "MAIAC AOD470nm (Aqua)")

References

http://doi.org/10.5281/zenodo.3334713

`new_labels` *new_labels: a place holder default to NULL.*

Description

if supplied as a list, it offers user to rename labels

Usage

```
new_labels
```

Format

An object of class `NULL` of length 0.

`plot.label` *internal-function to revise axis label for each feature*

Description

This function further fine-tune the format of each feature

Usage

```
## S3 method for class 'label'
plot(plot1, show_feature)
```

Arguments

`plot1`	ggplot2 object
`show_feature`	feature to plot

Value

returns ggplot2 object with further mordified layers based on the feature

`scatter.plot.diagonal` *make customized scatter plot with diagonal line and R2 printed.*

Description

make customized scatter plot with diagonal line and R2 printed.

Usage

```
scatter.plot.diagonal(data, x, y, size0 = 0.2, alpha0 = 0.3,
  dilute = FALSE, add_abline = FALSE, add_hist = TRUE)
```

Arguments

`data`	dataset
`x`	x
`y`	y
`size0`	point size, default to 1 of nobs<1000, 0.4 if nobs>1000
`alpha0`	alpha of point
`dilute`	a number or logical, dafault to TRUE, will plot `nrow(data_long)/dilute` data. For example, if dilute = 5 will plot 1/5 of the data. if dilute = TRUE will plot half of the data.
`add_abline`	default to FALSE, add a diagonal line `ggExtra::ggMarginal` but notice if add histogram, what is returned is no longer a ggplot2 object
`add_hist`	optional to add marginal histogram using `ggExtra::ggMarginal` but notice if add histogram, what is returned is no longer a ggplot2 object

Value

ggplot2 object if `add_hist = FALSE`

Examples

```
scatter.plot.diagonal(data = iris, x = "Sepal.Length", y = "Petal.Length")
```

scatter.plot.simple *simple scatter plot, adding marginal histogram by default.*

Description

simple scatter plot, adding marginal histogram by default.

Usage

```
scatter.plot.simple(data, x, y, size0 = 0.2, alpha0 = 0.3,
  dilute = FALSE, add_hist = TRUE)
```

Arguments

`data`	dataset
`x`	x
`y`	y
`size0`	point size, default to 1 of nobs<1000, 0.4 if nobs>1000
`alpha0`	alpha of point
`dilute`	a number or logical, dafault to TRUE, will plot `nrow(data_long)/dilute` data. For example, if dilute = 5 will plot 1/5 of the data. if dilute = TRUE will plot half of the data.
`add_hist`	optional to add marginal histogram using `ggExtra::ggMarginal` but notice if add histogram, what is returned is no longer a ggplot2 object

Value

ggplot2 object if `add_hist = FALSE`

Examples

```
scatter.plot.simple(data = shap_score, x = "dayint", y = "AOT_Uncertainty")
```

shap.plot.dependence *SHAP dependence plot and interaction plot, optional to be colored by a selected feature*

Description

This function makes the simple dependence plot with SHAP values on the y axis, optional to add color by another feature, optional to use a different y variable for SHAP values Not colored if `color_feature` is not supplied. If `data_int` (the SHAP interaction values dataset) is supplied, it will plot the interaction effect between `y` and `x` on the y axis.

Usage

```
shap.plot.dependence(data_long, x, y = NULL, color_feature = NULL,
  data_int = NULL, dilute = FALSE, smooth = TRUE, size0 = NULL,
  add_hist = FALSE)
```

Arguments

`data_long`	the long format SHAP values from `shap.prep`
`x`	which feature to show on x axis, it will plot the feature value.
`y`	which shap values to show on y axis, it will plot the SHAP value of that feature. y is default to x, if y is not provided, just plot the SHAP values of x on the y axis
`color_feature`	which feature value to use for coloring, color by the feature value.
`data_int`	the 3-dimention SHAP interaction values array. if `data_int` is supplied, y axis will plot the interaction values of y (vs. x)
`dilute`	a number or logical, dafault to TRUE, will plot `nrow(data_long)/dilute` data. For example, if dilute = 5 will plot 20 As long as dilute != FALSE, will plot at most half the data. from `predict.xgb.Booster` or `shap.prep.interaction`.
`smooth`	optional to add *loess* smooth line, default to TRUE.
`size0`	point size, default to 1 of nobs<1000, 0.4 if nobs>1000.
`add_hist`	whether to add histogram using `ggMarginal`, default to TRUE. But notice the plot after adding histogram it is `ggExtraPlot` object, cannot add geom to that anymore. If wish to add more ggplot layers, turn the histogram off

Details

Dependence plot is very easy to make if you have the SHAP values dataset from `predict.xgb.Booster` It is not necessary to start with the long-format data, but since I used that for the summary plot, I just continue to use the long dataset

Value

returns a ggplot2 object, based on which you could add more geom layers.

Examples

```
# **SHAP dependence plot**

# 1. simple dependence plot with SHAP values of x on the y axis
shap.plot.dependence(data_long = shap_long_iris, x="Petal.Length", add_hist = TRUE)

# 2. can choose a different SHAP values on the y axis
shap.plot.dependence(data_long = shap_long_iris, x="Petal.Length",
                           y = "Petal.Width")

# 3. color by another feature's feature values
shap.plot.dependence(data_long = shap_long_iris, x="Petal.Length",
                           color_feature = "Petal.Width")

# 4. choose 3 different variables for x, y, and color
shap.plot.dependence(data_long = shap_long_iris, x="Petal.Length",
                           y = "Petal.Width", color_feature = "Petal.Width")

# Optional to add hist or remove smooth line, optional to plot fewer data (make plot quicker)
shap.plot.dependence(data_long = shap_long_iris, x="Petal.Length",
                     y = "Petal.Width", color_feature = "Petal.Width",
                     add_hist = TRUE, smooth = FALSE, dilute = 3)

# to make a list of plot
plot_list <- lapply(names(iris)[2:3], shap.plot.dependence, data_long = shap_long_iris)

# **SHAP interaction effect plot **

# To get the interaction SHAP dataset for plotting, need to get `shap_int` first:
mod1 = xgboost::xgboost(
  data = as.matrix(iris[,-5]), label = iris$Species,
  gamma = 0, eta = 1, lambda = 0,nrounds = 1, verbose = FALSE)
# Use either:
data_int <- shap.prep.interaction(xgb_mod = mod1,
                                  X_train = as.matrix(iris[,-5]))
# or:
shap_int <- predict(mod1, as.matrix(iris[,-5]),
                    predinteraction = TRUE)

# if data_int is supplied, y axis will plot the interaction values of y (vs. x)
shap.plot.dependence(data_long = shap_long_iris,
                           data_int = shap_int_iris,
                           x="Petal.Length",
                           y = "Petal.Width",
                           color_feature = "Petal.Width")
```

shap.plot.force_plot *make the SHAP force plot*

Description

The force/stack plot, optional to zoom in at certain x or certain cluster.

Usage

```
shap.plot.force_plot(shapobs, id = "id", zoom_in_location = NULL,
  y_parent_limit = NULL, y_zoomin_limit = NULL, zoom_in = TRUE,
  zoom_in_group = NULL)
```

Arguments

`shapobs`	The dataset obtained by `shap.prep.stack.data`.
`id`	the id variable.
`zoom_in_location`	where to zoom in, default at place of 60 percent of the data.
`y_parent_limit`	set y axis limits.
`y_zoomin_limit`	`c(a,b)` to limit the y-axis in zoom-in.
`zoom_in`	default to TRUE, zoom in by `ggforce::facet_zoom`.
`zoom_in_group`	optional to zoom in certain cluster.

Examples

```
# **SHAP force plot**
plot_data <- shap.prep.stack.data(shap_contrib = shap_values_iris,
                                  n_groups = 4)
shap.plot.force_plot(plot_data)
shap.plot.force_plot(plot_data,  zoom_in_group = 2)

# plot all the clusters:
shap.plot.force_plot_bygroup(plot_data)
```

`shap.plot.force_plot_bygroup`

make the stack plot, optional to zoom in at certain x or certain cluster

---

Description

A collective display of zoom in plot: one plot of every group of clustered observations.

Usage

```
shap.plot.force_plot_bygroup(shapobs, id = "id", y_parent_limit = NULL)
```

Arguments

`shapobs`	The dataset obtained by `shap.prep.stack.data`.
`id`	the id variable.
`y_parent_limit`	set y axis limits.

Examples

```
# **SHAP force plot**
plot_data <- shap.prep.stack.data(shap_contrib = shap_values_iris,
                                  n_groups = 4)
shap.plot.force_plot(plot_data)
shap.plot.force_plot(plot_data,  zoom_in_group = 2)

# plot all the clusters:
shap.plot.force_plot_bygroup(plot_data)
```

`shap.plot.summary` *SHAP summary plot core function using the long-format SHAP values*

Description

The summary plot (sina plot) uses a long-format data of SHAP values. The long-format data could be obtained from either xgboost model or a SHAP matrix using `shap.values`. If you want to start with xgbmodel and data_X, use `shap.plot.summary.wrap1`. If you want to use self-derived SHAP matrix, use `shap.plot.summary.wrap2`. If a global list named **new_labels** is provided (`!is.null(new_labels)`, the plots will use that list to replace default labels `labels_within_package`.

Usage

```
shap.plot.summary(data_long, x_bound = NULL, dilute = FALSE,
  scientific = FALSE, my_format = NULL)
```

Arguments

`data_long`	a long format data of SHAP values from `shap.prep`
`x_bound`	in case need to limit x_axis_limit
`dilute`	a number or logical, dafault to TRUE, will plot `nrow(data_long)/dilute` data. for example, if dilute = 5 will plot 1/5 of the data. If dilute = TRUE or a number, we will plot at most half points per feature, so the plot won't be too slow. If you put dilute too high, at least 10 points per feature would be kept. If the dataset is even smaller than that, will just plot all the data.
`scientific`	show the mean|SHAP| in scientific format or not default to F, label format is 0.000, If true, label format is 0.0E-0,
`my_format`	supply your own number format if you really want to do so

Value

returns a ggplot2 object, could add further layers.

Examples

```
data("iris")
X1 = as.matrix(iris[,-5])
mod1 = xgboost::xgboost(
  data = X1, label = iris$Species, gamma = 0, eta = 1,
  lambda = 0,nrounds = 1, verbose = FALSE)


# shap.values(model, X_dataset) returns the SHAP
# data matrix and ranked features by mean|SHAP|
shap_values <- shap.values(xgb_model = mod1, X_train = X1)
shap_values$mean_shap_score
shap_values_iris <- shap_values$shap_score

# shap.prep() returns the long-format SHAP data from either model or
shap_long_iris <- shap.prep(xgb_model = mod1, X_train = X1)
# is the same as: using given shap_contrib
shap_long_iris <- shap.prep(shap_contrib = shap_values_iris, X_train = X1)

# **SHAP summary plot**
shap.plot.summary(shap_long_iris, scientific = TRUE)
shap.plot.summary(shap_long_iris, x_bound  = 1.5, dilute = 10)

# Alternatives options to make the same plot:
# option 1: from the xgboost model
shap.plot.summary.wrap1(mod1, X = as.matrix(iris[,-5]), top_n = 3)

# option 2: supply a self-made SHAP values dataset
# (e.g. sometimes as output from cross-validation)
shap.plot.summary.wrap2(shap_values_iris, X1, top_n = 3)
```

`shap.plot.summary.wrap1` *A wrapped function to make summary plot from xgb model object and predictors*

Description

wraps up function `shap.prep` and `shap.plot.summary` If a global list named **new_labels** is provided (`!is.null(new_labels)`, the plots will use that list to replace default labels `labels_within_package`.

Usage

```
shap.plot.summary.wrap1(model, X, top_n, dilute = FALSE)
```

Arguments

`model`	the xgboost model
`X`	the dataset of predictors used for the xgboost model

`top_n`	how many predictors you want to show in the plot (ranked)
`dilute`	a number or logical, dafault to TRUE, will plot `nrow(data_long)/dilute` data. for example, if dilute = 5 will plot 1/5 of the data. If dilute = TRUE or a number, we will plot at most half points per feature, so the plot won't be too slow. If you put dilute too high, at least 10 points per feature would be kept. If the dataset is even smaller than that, will just plot all the data.

Examples

```
data("iris")
X1 = as.matrix(iris[,-5])
mod1 = xgboost::xgboost(
  data = X1, label = iris$Species, gamma = 0, eta = 1,
  lambda = 0,nrounds = 1, verbose = FALSE)


# shap.values(model, X_dataset) returns the SHAP
# data matrix and ranked features by mean|SHAP|
shap_values <- shap.values(xgb_model = mod1, X_train = X1)
shap_values$mean_shap_score
shap_values_iris <- shap_values$shap_score

# shap.prep() returns the long-format SHAP data from either model or
shap_long_iris <- shap.prep(xgb_model = mod1, X_train = X1)
# is the same as: using given shap_contrib
shap_long_iris <- shap.prep(shap_contrib = shap_values_iris, X_train = X1)

# **SHAP summary plot**
shap.plot.summary(shap_long_iris, scientific = TRUE)
shap.plot.summary(shap_long_iris, x_bound  = 1.5, dilute = 10)

# Alternatives options to make the same plot:
# option 1: from the xgboost model
shap.plot.summary.wrap1(mod1, X = as.matrix(iris[,-5]), top_n = 3)

# option 2: supply a self-made SHAP values dataset
# (e.g. sometimes as output from cross-validation)
shap.plot.summary.wrap2(shap_values_iris, X1, top_n = 3)
```

`shap.plot.summary.wrap2` *A wrapped function to make summary plot from given SHAP values matrix*

Description

Sometimes the SHAP matrix is returned from cross-validation. This function wraps up function `shap.prep` and `shap.plot.summary`.

Usage

```
shap.plot.summary.wrap2(shap_score, X, top_n, dilute = FALSE)
```

Arguments

`shap_score`	the SHAP values dataset, could be obtained by `shap.prep`.
`X`	the dataset of predictors used for the xgboost model
`top_n`	how many predictors you want to show in the plot (ranked)
`dilute`	a number or logical, dafault to TRUE, will plot `nrow(data_long)/dilute` data. for example, if dilute = 5 will plot 1/5 of the data. If dilute = TRUE or a number, we will plot at most half points per feature, so the plot won't be too slow. If you put dilute too high, at least 10 points per feature would be kept. If the dataset is even smaller than that, will just plot all the data.

Details

If a global list named **new_labels** is provided (`!is.null(new_labels)`, the plots will use that list to replace default labels `labels_within_package`.

Examples

```
data("iris")
X1 = as.matrix(iris[,-5])
mod1 = xgboost::xgboost(
  data = X1, label = iris$Species, gamma = 0, eta = 1,
  lambda = 0,nrounds = 1, verbose = FALSE)


# shap.values(model, X_dataset) returns the SHAP
# data matrix and ranked features by mean|SHAP|
shap_values <- shap.values(xgb_model = mod1, X_train = X1)
shap_values$mean_shap_score
shap_values_iris <- shap_values$shap_score

# shap.prep() returns the long-format SHAP data from either model or
shap_long_iris <- shap.prep(xgb_model = mod1, X_train = X1)
# is the same as: using given shap_contrib
shap_long_iris <- shap.prep(shap_contrib = shap_values_iris, X_train = X1)

# **SHAP summary plot**
shap.plot.summary(shap_long_iris, scientific = TRUE)
shap.plot.summary(shap_long_iris, x_bound  = 1.5, dilute = 10)

# Alternatives options to make the same plot:
# option 1: from the xgboost model
shap.plot.summary.wrap1(mod1, X = as.matrix(iris[,-5]), top_n = 3)

# option 2: supply a self-made SHAP values dataset
# (e.g. sometimes as output from cross-validation)
shap.plot.summary.wrap2(shap_values_iris, X1, top_n = 3)
```

shap.prep *prep SHAP values into long format for plotting*

Description

prep SHAP values into long format for plotting

Usage

```
shap.prep(xgb_model = NULL, shap_contrib = NULL, X_train,
  top_n = NULL)
```

Arguments

`xgb_model`	a xgboost model object
`shap_contrib`	optional to supply SHAP values dataset, default to NULL
`X_train`	the dataset of predictors used for the xgboost model if not NULL, will be taken as SHAP values,
`top_n`	to choose top_n variables ranked by mean|SHAP| if needed

Value

a long-format data.table, named as `shap_long` in other functions

Examples

```
data("iris")
X1 = as.matrix(iris[,-5])
mod1 = xgboost::xgboost(
  data = X1, label = iris$Species, gamma = 0, eta = 1,
  lambda = 0,nrounds = 1, verbose = FALSE)


# shap.values(model, X_dataset) returns the SHAP
# data matrix and ranked features by mean|SHAP|
shap_values <- shap.values(xgb_model = mod1, X_train = X1)
shap_values$mean_shap_score
shap_values_iris <- shap_values$shap_score

# shap.prep() returns the long-format SHAP data from either model or
shap_long_iris <- shap.prep(xgb_model = mod1, X_train = X1)
# is the same as: using given shap_contrib
shap_long_iris <- shap.prep(shap_contrib = shap_values_iris, X_train = X1)

# **SHAP summary plot**
shap.plot.summary(shap_long_iris, scientific = TRUE)
shap.plot.summary(shap_long_iris, x_bound  = 1.5, dilute = 10)
```

```
# Alternatives options to make the same plot:
# option 1: from the xgboost model
shap.plot.summary.wrap1(mod1, X = as.matrix(iris[,-5]), top_n = 3)

# option 2: supply a self-made SHAP values dataset
# (e.g. sometimes as output from cross-validation)
shap.plot.summary.wrap2(shap_values_iris, X1, top_n = 3)
```

`shap.prep.interaction` *prepare the interaction SHAP values from predict.xgb.Booster*

Description

This function just runs `shap_int <-predict(xgb_mod,as.matrix(X_train),predinteraction = TRUE)`, may not be necessary, maybe just use xgboost::predict.xgb.Booster directly,

Usage

```
shap.prep.interaction(xgb_model, X_train)
```

Arguments

`xgb_model`	a xgboost model object
`X_train`	the dataset of predictors used for the xgboost model

Value

a 3-dimention array: #obs x #features x #features

Examples

```
# To get the interaction SHAP dataset for plotting:
# fit the xgboost model
mod1 = xgboost::xgboost(
  data = as.matrix(iris[,-5]), label = iris$Species,
  gamma = 0, eta = 1, lambda = 0,nrounds = 1, verbose = FALSE)
# Use either:
data_int <- shap.prep.interaction(xgb_mod = mod1,
                                  X_train = as.matrix(iris[,-5]))
# or:
shap_int <- predict(mod1, as.matrix(iris[,-5]),
                    predinteraction = TRUE)

# **SHAP interaction effect plot **
shap.plot.dependence(data_long = shap_long_iris,
                           data_int = shap_int_iris,
                           x="Petal.Length",
                           y = "Petal.Width",
                           color_feature = "Petal.Width")
```

shap.prep.stack.data *Prepare data for SHAP force plot (stack plot)*

Description

Make force plot for top_n features, option to randomly plot certain portion of the data in case the dataset is large.

Usage

```
shap.prep.stack.data(shap_contrib, top_n = NULL, data_percent = 1,
  cluster_method = "ward.D", n_groups = 10L)
```

Arguments

shap_contrib	shap_contrib is the SHAP value data returned from predict.xgb.booster
top_n	integer, optional to show only top_n features, combine the rest
data_percent	what percent of data to plot (to speed up), in the range of (0,1]
cluster_method	default to ward.D
n_groups	a integer, how many groups to plot in shap.plot.force_plot_bygroup

Value

a dataset for stack plot

Examples

```
# **SHAP force plot**
plot_data <- shap.prep.stack.data(shap_contrib = shap_values_iris,
                                  n_groups = 4)
shap.plot.force_plot(plot_data)
shap.plot.force_plot(plot_data,  zoom_in_group = 2)

# plot all the clusters:
shap.plot.force_plot_bygroup(plot_data)
```

shap.values *return SHAP contribution from xgboost model*

Description

`shap.values` returns from xgboost model a list of 1.the matrix of shap score and 2. the ranked variable vector by each variable's mean absolute SHAP value

Usage

```
shap.values(xgb_model, X_train)
```

Arguments

`xgb_model`	a xgboost model object
`X_train`	the dataset of predictors used for the xgboost model

Value

a list of three elements, the SHAP values as data.table, ranked mean|SHAP|, BIAS

Examples

```
data("iris")
X1 = as.matrix(iris[,-5])
mod1 = xgboost::xgboost(
  data = X1, label = iris$Species, gamma = 0, eta = 1,
  lambda = 0,nrounds = 1, verbose = FALSE)


# shap.values(model, X_dataset) returns the SHAP
# data matrix and ranked features by mean|SHAP|
shap_values <- shap.values(xgb_model = mod1, X_train = X1)
shap_values$mean_shap_score
shap_values_iris <- shap_values$shap_score

# shap.prep() returns the long-format SHAP data from either model or
shap_long_iris <- shap.prep(xgb_model = mod1, X_train = X1)
# is the same as: using given shap_contrib
shap_long_iris <- shap.prep(shap_contrib = shap_values_iris, X_train = X1)

# **SHAP summary plot**
shap.plot.summary(shap_long_iris, scientific = TRUE)
shap.plot.summary(shap_long_iris, x_bound  = 1.5, dilute = 10)

# Alternatives options to make the same plot:
# option 1: from the xgboost model
shap.plot.summary.wrap1(mod1, X = as.matrix(iris[,-5]), top_n = 3)
```

```
# option 2: supply a self-made SHAP values dataset
# (e.g. sometimes as output from cross-validation)
shap.plot.summary.wrap2(shap_values_iris, X1, top_n = 3)
```

`shap_int_iris` *The interaction effect SHAP values example using iris dataset.*

Description

The interaction effect SHAP values example using iris dataset.

Usage

```
shap_int_iris
```

Format

An object of class `array` of dimension 150 x 5 x 5.

`shap_long_iris` *The long-format SHAP values example using iris dataset.*

Description

The long-format SHAP values example using iris dataset.

Usage

```
shap_long_iris
```

Format

An object of class `data.table` (inherits from `data.frame`) with 600 rows and 5 columns.

`shap_score` *SHAP values example from dataXY_df .*

Description

SHAP values example from dataXY_df .

Usage

```
shap_score
```

Format

An object of class `data.table` (inherits from `data.frame`) with 10148 rows and 9 columns.

References

http://doi.org/10.5281/zenodo.3334713

`shap_values_iris` *SHAP values example using iris dataset.*

Description

SHAP values example using iris dataset.

Usage

```
shap_values_iris
```

Format

An object of class `data.table` (inherits from `data.frame`) with 150 rows and 4 columns.

Index