Package 'mshap'

June 17, 2021

Title Multiplicative SHAP Values for Two-Part Models

Version 0.1.0

Description Allows for the computation of mSHAP values on two-part models as proposed by Matthews, S. and Hartman, B. (2021) <arXiv:2106.08990>.
Also contains functions for simple plotting of the results (or any SHAP values). For information about the TreeSHAP algorithm that mSHAP builds on, see Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I. (2020) <doi:10.1038/s42256-019-0138-9>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Imports magrittr (>= 1.5), purrr (>= 0.3.4), dplyr (>= 1.0.4), forcats, stringr, ggplot2, ggbeeswarm, rlang, tidyr, tidyselect

Suggests rmarkdown, knitr, insuranceData, reticulate, caret, testthat (>= 3.0.0), covr

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 2.10)

NeedsCompilation no

Author Spencer Matthews [aut, cre]

Maintainer Spencer Matthews <srmatthews98@gmail.com>

Repository CRAN

Date/Publication 2021-06-17 08:40:02 UTC

R topics documented:

`mshap` *mSHAP*

Description

A function for calculation SHAP values of two-part models.

Usage

```
mshap(shap_1, shap_2, ex_1, ex_2, shap_1_names = NULL, shap_2_names = NULL)
```

Arguments

`shap_1, shap_2`	The SHAP values that will be multiplied together. They may be matrices or data frames, and up to one may be a list where each element is a matrix or data frame (this is necessary when one of the models is a multinomial classifier, for instance). Each data frame or matrix here must have the same number of rows, and if there are different numbers of columns or the columns are not the same, then shap_*_names must be specified.
`ex_1, ex_2`	The expected values of the models across the training set. If one of the arguments shap_* is a list, then the corresponding ex_* argument must be a vector (or array) of the same length as the list.
`shap_1_names, shap_2_names`	The character vector containing the names of the columns in `shap_1` and `shap_2`, respectively. These must be in the same order as the columns themselves. If a list is passed to one of the shap_* arguments, it does NOT affect the corresponding shap_*_names argument, which will still be a single character vector.

Details

This function allows the user to input the SHAP values for two separate models (along with the expected values), and mSHAP then outputs the SHAP values of the two model predictions multiplied together.

An included feature of the function is the ability to pass data frames that do not have the same number of columns. Say for instance that one model benefits from a certain variable but the other does not. As long as the shap_*_names arguments are supplied, the function will automatically add a column of 0's for missing variables in either data frame (matrix). This corresponds to a SHAP value of 0, which of course is accurate if the variable was not included in the model.

Value

A list containing the multiplied SHAP values and the expected value. Or, in the case of a list passed as one of the shap_* augments, a list of lists where each element corresponds to the same element in the list passed to shap_*.

Examples

```
if (interactive()) {
shap1 <- data.frame(
  age = runif(1000, -5, 5),
  income = runif(1000, -5, 5),
  married = runif(1000, -5, 5),
  sex = runif(1000, -5, 5)
)
shap2 <- list(
  data.frame(
    age = runif(1000, -5, 5),
    income = runif(1000, -5, 5),
    married = runif(1000, -5, 5),
    sex = runif(1000, -5, 5)
  ),
  data.frame(
    age = runif(1000, -5, 5),
    income = runif(1000, -5, 5),
    married = runif(1000, -5, 5),
    sex = runif(1000, -5, 5)
  ),
  data.frame(
    age = runif(1000, -5, 5),
    income = runif(1000, -5, 5),
    married = runif(1000, -5, 5),
    sex = runif(1000, -5, 5)
  )
)

ex1 <- 3
ex2 <- c(4, 5, 6)

# Case where both models have a single output
res1 <- mshap(
  shap_1 = shap1,
  shap_2 = shap2[[1]],
  ex_1 = ex1,
  ex_2 = ex2[1]
)
View(res1$shap_vals)
res1$expected_value

# Case where one of your models has multiple outputs that are explained
res2 <- mshap(
  shap_1 = shap1,
  shap_2 = shap2,
  ex_1 = ex1,
  ex_2 = ex2
)
View(res2[[1]]$shap_vals)
res2[[1]]$expected_value
```

```
# Case where the models have different variables
res3 <- mshap(
  shap_1 = shap1,
  shap_2 = shap2,
  ex_1 = ex1,
  ex_2 = ex2,
  shap_1_names = c("Age", "Income", "Married", "Sex"),
  shap_2_names = c("Age", "Income", "Children", "American")
)
# Note how there are now 6 columns of SHAP values, since there are 6
# distinct variables
View(res3[[1]]$shap_vals)
res3[[1]]$expected_value
}
```

`observation_plot` *SHAP Observation Plot*

Description

This Function plots the given contributions for a single observation, and demonstrates how the model arrived at the prediction for the given observation.

Usage

```
observation_plot(
  variable_values,
  shap_values,
  expected_value,
  names = NULL,
  num_vars = 10,
  fill_colors = c("#A54657", "#0D3B66"),
  connect_color = "#849698",
  expected_color = "#849698",
  predicted_color = "#EE964B",
  title = "Individual Observation Explanation",
  font_family = "Times New Roman"
)
```

Arguments

`variable_values`	A data frame of the values of the variables that caused the given SHAP values, generally will be the same data frame or matrix that was passed to the model for prediction.
`shap_values`	A data frame of shap values, either returned by `mshap()` or obtained from the python {shap} module.

expected_value	The expected value of the SHAP explainer, either returned by mshap() or obtained from the python {shap} module.
names	A character vector of variable names, corresponding to the order of the columns in both variable_values and shap_values. If NULL (default), then the column names of the variable_values are taken as names.
num_vars	An integer specifying the number of variables to show in the plot, defaults to the 10 most important.
fill_colors	A character vector of length 2. The first element specifies the fill of a negative SHAP value and the second element specifies the fill of a positive SHAP value.
connect_color	A string specifying the color of the line segment that connects the SHAP value bars
expected_color	A string specifying the color of the line that marks the baseline value, or the expected model output.
predicted_color	A string specifying the color of the line that marks the value predicted by the model.
title	A string specifying the title of the plot.
font_family	A string specifying the font family, defaults to Times New Roman.

Details

This function allows the user to pass a sing row from a data frame of SHAP values and variable values along with an expected model output and it returns a ggplot object displaying a specific map of the effect of Variable value on SHAP value. It is created with {ggplot2}, and the returned value is a {ggplot2} object that can be modified for given themes/colors.

Please note that for the variable_values and shap_values arguments, both of which are data frames, the columns must be in the same order. This is essential in assuring that the variable values and labels are matched to the correct shap values.

Value

A {ggplot2} object

Examples

```
if (interactive()) {
library(mshap)
library(ggplot2)

# Generate fake data
set.seed(18)
dat <- data.frame(
  age = runif(1000, min = 0, max = 20),
  prop_domestic = runif(1000),
  model = sample(c(0, 1), 1000, replace = TRUE),
  maintain = rexp(1000, .01) + 200
)
```

```
shap <- data.frame(
  age = rexp(1000, 1/dat$age) * (-1)^(rbinom(1000, 1, dat$prop_domestic)),
  prop_domestic = -200 * rnorm(100, dat$prop_domestic, 0.02) + 100,
  model = ifelse(dat$model == 0, rnorm(1000, -50, 30), rnorm(1000, 50, 30)),
  maintain = (rnorm(1000, dat$maintain, 100) - 400) * 0.2
)
expected_value <- 1000

# A Basic sumary plot
summary_plot(
  variable_values = dat,
  shap_values = shap
)

# A Customized summary plot
summary_plot(
  variable_values = dat,
  shap_values = shap,
  legend.position = "bottom",
  names = c("Age", "% Domestic", "Model", "Maintenence Hours"),
  colorscale = c("blue", "purple", "red"),
  font_family = "Arial",
  title = "A Custom Title"
)

# A basic observation plot
observation_plot(
  variable_values = dat[1,],
  shap_values = shap[1,],
  expected_value = expected_value
)

# A Customized Observation plot
observation_plot(
  variable_values = dat[1,],
  shap_values = shap[1,],
  expected_value = expected_value,
  names = c("Age", "% Domestic", "Model", "Maintenence Hours"),
  font_family = "Arial",
  title = "A Custom Title",
  fill_colors = c("red", "blue"),
  connect_color = "black",
  expected_color = "purple",
  predicted_color = "yellow"
)

# Add elements to the returned object
# see vignette("mshap_plots") for more information
observation_plot(
  variable_values = dat[1,],
  shap_values = shap[1,],
  expected_value = expected_value,
  names = c("Age", "% Domestic", "Model", "Maintenence Hours"),
```

```
  font_family = "Arial",
  title = "A Custom Title"
) +
  geom_label(
    aes(y = 950, x = 4, label = "This is a really big bar!"),
    color = "#FFFFFF",
    fill = NA
  ) +
  theme(
    plot.background = element_rect(fill = "grey"),
    panel.background = element_rect(fill = "lightyellow")
  )
}
```

---

summary_plot *SHAP Summary Plot*

Description

A Function for obtaining a beeswarm plot, similar to the summary plot in the {shap} python package.

Usage

```
summary_plot(
  variable_values,
  shap_values,
  names = NULL,
  num_vars = 10,
  colorscale = c("#A54657", "#FAF0CA", "#0D3B66"),
  legend.position = c(0.8, 0.2),
  font_family = "Times New Roman",
  title = "SHAP Value Summary"
)
```

Arguments

variable_values	A data frame of the values of the variables that caused the given SHAP values, generally will be the same data frame or matrix that was passed to the model for prediction.
shap_values	A data frame of shap values, either returned by mshap() or obtained from the python {shap} module.
names	A character vector of variable names, corresponding to the order of the columns in both variable_values and shap_values. If NULL (default), then the column names of the variable_values are taken as names.
num_vars	An integer specifying the number of variables to show in the plot, defaults to the 10 most important.

`colorscale` The color scale used for the color of the plot. It should be a character vector of length three, with the low color first, the middle color second, and the high color third. These can be hex color codes or colors recognized by {ggplot2}.

`legend.position` The position of the legend. See `?ggplot2::theme` for more information.

`font_family` A character string specifying the family of the text on the plot. Defaults to Times New Roman.

`title` A character string specifying the title of the plot.

Details

This function allows the user to pass a data frame of SHAP values and variable values and returns a ggplot object displaying a general summary of the effect of Variable level on SHAP value by variable. It is created with {ggbeeswarm}, and the returned value is a {ggplot2} object that can be modified for given themes/colors.

Please note that for the `variable_values` and `shap_values` arguments, both of which are data frames, the columns must be in the same order. This is essential in assuring that the variable values and labels are matched to the correct shap values.

Value

A {ggplot2} object

Examples

```
if (interactive()) {
library(mshap)
library(ggplot2)

# Generate fake data
set.seed(18)
dat <- data.frame(
  age = runif(1000, min = 0, max = 20),
  prop_domestic = runif(1000),
  model = sample(c(0, 1), 1000, replace = TRUE),
  maintain = rexp(1000, .01) + 200
)
shap <- data.frame(
  age = rexp(1000, 1/dat$age) * (-1)^(rbinom(1000, 1, dat$prop_domestic)),
  prop_domestic = -200 * rnorm(100, dat$prop_domestic, 0.02) + 100,
  model = ifelse(dat$model == 0, rnorm(1000, -50, 30), rnorm(1000, 50, 30)),
  maintain = (rnorm(1000, dat$maintain, 100) - 400) * 0.2
)
expected_value <- 1000

# A Basic sumary plot
summary_plot(
  variable_values = dat,
  shap_values = shap
```

```
)

# A Customized summary plot
summary_plot(
  variable_values = dat,
  shap_values = shap,
  legend.position = "bottom",
  names = c("Age", "% Domestic", "Model", "Maintenence Hours"),
  colorscale = c("blue", "purple", "red"),
  font_family = "Arial",
  title = "A Custom Title"
)

# A basic observation plot
observation_plot(
  variable_values = dat[1,],
  shap_values = shap[1,],
  expected_value = expected_value
)

# A Customized Observation plot
observation_plot(
  variable_values = dat[1,],
  shap_values = shap[1,],
  expected_value = expected_value,
  names = c("Age", "% Domestic", "Model", "Maintenence Hours"),
  font_family = "Arial",
  title = "A Custom Title",
  fill_colors = c("red", "blue"),
  connect_color = "black",
  expected_color = "purple",
  predicted_color = "yellow"
)

# Add elements to the returned object
# see vignette("mshap_plots") for more information
observation_plot(
  variable_values = dat[1,],
  shap_values = shap[1,],
  expected_value = expected_value,
  names = c("Age", "% Domestic", "Model", "Maintenence Hours"),
  font_family = "Arial",
  title = "A Custom Title"
) +
  geom_label(
    aes(y = 950, x = 4, label = "This is a really big bar!"),
    color = "#FFFFFF",
    fill = NA
  ) +
  theme(
    plot.background = element_rect(fill = "grey"),
    panel.background = element_rect(fill = "lightyellow")
  )
```

```
}
```

`where` *Select variables with a function*

Description

This selection helper selects the variables for which a function returns `TRUE`.

Usage

```
where(fn)
```

Arguments

`fn`	A function that returns `TRUE` or `FALSE` (technically, a *predicate* function). Can also be a purrr-like formula.

Value

A selection of columns

Index