Package ‘shapper’

October 14, 2022

Title   Wrapper of Python Library 'shap'
Version 0.1.3
Description Provides SHAP explanations of machine learning models. In applied machine learning, there is a strong belief that we need to strike a balance between interpretability and accuracy. However, in field of the Interpretable Machine Learning, there are more and more new ideas for explaining black-box models. One of the best known method for local explanations is SHapley Additive exPlanations (SHAP) introduced by Lundberg, S., et al., (2016) <arXiv:1705.07874> The SHAP method is used to calculate influences of variables on the particular observation. This method is based on Shapley values, a technique used in game theory. The R package ‘shapper’ is a port of the Python library 'shap'.

License GPL
Encoding UTF-8
LazyData true
URL https://github.com/ModelOriented/shapper
BugReports https://github.com/ModelOriented/shapper/issues
RoxygenNote 7.1.1
Imports reticulate, DALEX, ggplot2
Suggests covr, knitr, randomForest, rpart, testthat, markdown, qpdf
VignetteBuilder knitr
NeedsCompilation no
Author Szymon Maksymiuk [aut, cre], Alicja Gosiewska [aut], Przemyslaw Biecek [aut], Mateusz Staniak [ctb], Michal Burdukiewicz [ctb]
Maintainer Szymon Maksymiuk <sz.maksymiuk@gmail.com>
Repository CRAN
Date/Publication 2020-08-28 09:00:03 UTC
R topics documented:

individual_variable_effect ........................................... 2
install_shap ................................................................. 4
plot.individual_variable_effect ........................................ 5
print.individual_variable_effect ........................................ 6
theme_drwhy_colors ....................................................... 7

Index 8

individual_variable_effect

Individual Variable Effect

Description

Individual Variable Effect

Usage

individual_variable_effect(x, ...)

## S3 method for class 'explainer'
individual_variable_effect(
  x,
  new_observation,
  method = "KernelSHAP",
  nsamples = "auto",
  ...
)

## Default S3 method:
individual_variable_effect(
  x,
  data,
  predict_function = predict,
  new_observation,
  label = tail(class(x), 1),
  method = "KernelSHAP",
  nsamples = "auto",
  ...
)

shap(x, ...)
individual_variable_effect

Arguments

x a model to be explained, or an explainer created with function explain.

... other parameters.

new_observation an observation/observations to be explained. Required for local/instance level explainers. Columns in should correspond to columns in the data argument. Data set should not contain any additional columns.

method an estimation method of SHAP values. Currently the only available is ‘KernelSHAP’.

nsamples number of samples or "auto". Note that number must be as integer. Use ‘as.integer()’.

data validation dataset. Used to determine univariate distributions, calculation of quantiles, correlations and so on. It will be extracted from ‘x’ if it’s an explainer.

predict_function predict function that operates on the model ‘x’. Since the model is a black box, the ‘predict_function’ is the only interface to access values from the model. It should be a function that takes at least a model ‘x’ and data and returns vector of predictions. If model response has more than a single number (like multiclass models) then this function should return a matrix/data.frame of the size ‘m’ x ‘d’, where ‘m’ is the number of observations while ‘d’ is the dimensionality of model response. It will be extracted from ‘x’ if it’s an explainer.

label name of the model. By default it’s extracted from the class attribute of the model

Value

an object of class individual_variable_effect with shap values of each variable for each new observation. Columns:

- first d columns contains variable values.
- _id_ - id of observation, number of row in ‘new_observation’ data.
- _ylevel_ - level of y
- _yhat_ - predicted value for level of y
- _yhat_mean_- expected value of prediction, mean of all predictions
- _vname_- variable name
- _attribution_- attribution of variable
- _sign_- a sign of attribution
- _label_- a label

In order to use shapper with other python virtual environment following R command are required to execute reticulate::use_virtualenv("path_to_your_env") or for conda reticulate::use_conda("name_of_conda_env") before attaching shapper.
Examples

```r
have_shap <- reticulate::py_module_available("shap")

if(have_shap){
  library("shapper")
  library("DALEX")
  library("randomForest")
  Y_train <- HR$status
  x_train <- HR[, -6]
  set.seed(123)
  model_rf <- randomForest(x = x_train, y = Y_train, ntree= 50)
  p_function <- function(model, data) predict(model, newdata = data, type = "prob")
  ive_rf <- individual_variable_effect(model_rf, data = x_train, predict_function = p_function,
                                           new_observation = x_train[1:2,], nsamples = 50)
  ive_rf
} else{
  print('Python testing environment is required.')
}
```

install_shap

Install shap Python library

Description

Install shap Python library

Usage

```r
install_shap(method = "auto", conda = "auto", envname = NULL)
```

Arguments

- **method**: Installation method. By default, "auto". It is passed to the `py_install` function from package 'reticulate'.
- **conda**: Path to conda executable. It is passed to the `py_install` function from package 'reticulate'.
- **envname**: Name of environment to install shapp package into. If NULL it will install into default It is passed to the `py_install` function from package 'reticulate'.

To use conda installation execute `install_shap(method = "conda", envname = nameofenv)` Please keep in mind that winodws accepts only conda instalations.
Examples

```r
## Not run:
install_shap((method = "auto", conda = "auto")

## End(Not run)
```

## Description

Function `plot.individual_variable_effect` plots variables effects plots.

## Usage

```r
## S3 method for class 'individual_variable_effect'
plot(
  x,
  ..., id = 1,
  digits = 2,
  rounding_function = round,
  show_predicted = TRUE,
  show_attributions = TRUE,
  cols = c("label", "id"),
  rows = "ylevel",
  selected = NULL,
  bar_width = 8,
  vcolors = c("#f05a71", "#371ea3", "#8bdcbe", X = "#371ea3", pred = "#371ea3")
)
```

## Arguments

- `x`: an individual variable effect explainer produced with function `individual_variable_effect()`
- `...`: other explainers that shall be plotted together
- `id`: of observation. By default first observation is taken.
- `digits`: number of decimal places (round) or significant digits (signif) to be used. See the `rounding_function` argument.
- `rounding_function`: function that is to used for rounding numbers. It may be `signif()` which keeps a specified number of significant digits. Or the default `round()` to have the same precision for all components
- `show_predicted`: show arrows for predicted values.
Print Individual Variable Effects

Description

Print Individual Variable Effects

show_attributions

- show attributions values.

cols
- A vector of characters defining faceting groups on columns dimension. Possible values: 'label', 'id', 'ylevel'.

rows
- A vector of characters defining faceting groups on rows dimension. Possible values: 'label', 'id', 'ylevel'.

selected
- A vector of characters. If specified, then only selected classes are presented

bar_width
- width of bars. By default 8

vcolors
- named vector with colors

Value

- a ggplot2 object

Examples

```r
have_shap <- reticulate::py_module_available("shap")

if(have_shap){
  library("shapper")
  library("DALEX")
  library("randomForest")
  Y_train <- HR$status
  x_train <- HR[, -6]
  set_seed(123)
  model_rf <- randomForest(x = x_train, y = Y_train, ntree = 50)
  p_function <- function(model, data) predict(model, newdata = data, type = "prob")
  ive_rf <- individual_variable_effect(model_rf, data = x_train, predict_function = p_function,
    new_observation = x_train[1:2,], nsamples = 50)
  pl1 <- plot(ive_rf, bar_width = 4)
  pl2 <- plot(ive_rf, bar_width = 4, show_predicted = FALSE)
  pl3 <- plot(ive_rf, bar_width = 4, show_predicted = FALSE,
    cols = c("id","ylevel"), rows = "label")
  print(pl1)
  print(pl2)
  print(pl3)
} else {
  print("Python testing environment is required.")
}
```
### Theme Drwhy_colors

**DrWhy Theme for ggplot objects**

#### Description

DrWhy Theme for ggplot objects

#### Usage

```r
theme_drwhy_colors(n = 2)
```

#### Arguments

- `n`: number of colors for color palette

#### Value

theme for ggplot2 objects
Index

explain, 3

individual_variable_effect, 2, 7
install_shap, 4

plot.individual_variable_effect, 5
print.individual_variable_effect, 6
py_install, 4

shap(individual_variable_effect), 2

theme_drwhy_colors, 7