Package ‘modelStudio’

March 13, 2020

Title Interactive Studio for Explanatory Model Analysis

Version 1.0.1

Description Automate the explanatory analysis of machine learning predictive models. This package generates advanced interactive and animated model explanations in the form of a serverless HTML site with only one line of code. The main function computes various (instance and dataset level) model explanations and produces an interactive, customisable dashboard made with ‘D3.js’. It consists of multiple panels for plots with their short descriptions. Easily save and share the dashboard with others. Tools for model exploration unite with tools for EDA (Exploratory Data Analysis) to give a broad overview of the model behavior.

Depends R (>= 3.5)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Imports iBreakDown (>= 1.0), ingredients (>= 1.0), r2d3, jsonlite, progress

Suggests DALEX (>= 0.4.9), parallelMap, randomForest, xgboost, knitr, markdown, testthat, spelling

VignetteBuilder knitr


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

NeedsCompilation no

Author Hubert Baniecki [aut, cre] (<https://orcid.org/0000-0001-6661-5364>), Przemyslaw Biecek [aut] (<https://orcid.org/0000-0001-8423-1823>)

Maintainer Hubert Baniecki <hbaniecki@gmail.com>

Repository CRAN

Date/Publication 2020-03-13 11:40:02 UTC
R topics documented:

modelStudio ................................................................. 2
modelStudioOptions ...................................................... 5

Index

modelStudio Generate Interactive Studio for Explanatory Model Analysis

Description

This function computes various (instance and dataset level) model explanations and produces an interactive, customisable dashboard made with D3.js. It consists of multiple panels for plots with their short descriptions. Easily save and share the dashboard with others. Tools for model exploration unite with tools for EDA (Exploratory Data Analysis) to give a broad overview of the model behavior.

Find more details about the plots in Explanatory Model Analysis: Explore, Explain and Examine Predictive Models

Usage

modelStudio(explainer, ...)

## S3 method for class 'explainer'
modelStudio(
explainer,
new_observation = NULL,
new_observation_y = NULL,
facet_dim = c(2, 2),
time = 500,
max_features = 10,
N = 400,
B = 15,
eda = TRUE,
show_info = TRUE,
parallel = FALSE,
options = modelStudioOptions(),
viewer = "external",
...)

Arguments

explainer An explainer created with function DALEX::explain().

... Other parameters.

ew_observation A new observation with columns that correspond to variables used in the model.
new_observation_y
  True label for new_observation (optional).

facet_dim
  Dimensions of the grid. Default is c(2,2).

time
  Time in ms. Set animation length. Default is 500.

max_features
  Maximum number of features to be included in Break Down and Shapley Values plots. Default is 10.

N
  Number of observations used for calculation of partial dependence profiles. Default is 400.

B
  Number of random paths used for calculation of Shapley values. Default is 15.

eda
  Compute EDA plots. Default is TRUE.

show_info
  Verbose progress on the console. Default is TRUE.

parallel
  Speed up the computation using parallelMap::parallelMap(). See vignette. This might interfere with showing progress using show_info.

options
  Customize modelStudio. See modelStudioOptions and vignette.

viewer
  Default is external to display in an external RStudio window. Use browser to display in an external browser or internal to use the RStudio internal viewer pane for output.

Value

An object of the r2d3 class.

References

- Wrapper for the function is implemented in DALEX
- Feature Importance, Ceteris Paribus, Partial Dependence and Accumulated Dependence plots are implemented in ingredients
- Break Down and Shapley Values plots are implemented in iBreakDown

See Also

Python wrappers and more can be found in DALEXtra

Examples

```r
library("DALEX")
library("modelStudio")

# Create a model
model_titanic <- glm(survived ~ ., data = titanic_imputed, family = "binomial")

# Wrap it into an explainer
explainer_titanic <- explain(model_titanic,
```
```
data = titanic_imputed[,-8],
y = titanic_imputed[,8],
label = "glm",
verbose = FALSE)

# Pick some data points
new_observations <- titanic_imputed[1:2,]
rownames(new_observations) <- c("Lucas","James")

# Make a studio for the model
modelStudio(explainer_titanic, new_observations,
N = 100, B = 10, show_info = FALSE)

#:# ex2 regression on 'apartments' dataset
library("randomForest")

model_apartments <- randomForest(m2.price ~ ., data = apartments)

explainer_apartments <- explain(model_apartments,
data = apartments[,-1],
y = apartments[,1],
verbose = FALSE)

new_apartments <- apartments[1:2,]
rownames(new_apartments) <- c("ap1","ap2")

# change dashboard dimensions and animation length
modelStudio(explainer_apartments, new_apartments,
facet_dim = c(2, 3), time = 800,
show_info = FALSE)

# add information about true labels
modelStudio(explainer_apartments, new_apartments,
new_observation_y = apartments[1:2, 1],
show_info = FALSE)

# don't compute EDA plots
modelStudio(explainer_apartments, eda = FALSE,
show_info = FALSE)

#:# ex3 xgboost model on 'HR' dataset
library("xgboost")

model_matrix <- model.matrix(status == "fired" ~ . - 1, HR)
data <- xgb.DMatrix(model_matrix, label = HR$status == "fired")

params <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
objective = "binary:logistic", eval_metric = "auc")

model_HR <- xgb.train(params, data, nrounds = 50)
```
```r
explainer_HR <- explain(model_HR,
  data = model_matrix,
  y = HR$status == "fired",
  verbose = FALSE)

modelStudio(explainer_HR, show_info = FALSE)
```

---

### `modelStudioOptions` Modify default options and pass them to `modelStudio`

#### Description

This function returns default options for `modelStudio`. It is possible to modify values of this list and pass it to the `options` parameter in the main function. **WARNING:** Editing default options may cause unintended behavior.

#### Usage

```r
modelStudioOptions(...)
```

#### Arguments

... Options to change, `option_name = value`.

#### Value

list of options for `modelStudio`.

**Main options:**

- `scale_plot` TRUE Makes every plot the same height, ignores `bar_width`.
- `show_boxplot` TRUE Display boxplots in Feature Importance and Shapley Values plots.
- `show_subtitle` TRUE Should the subtitle be displayed?
- `subtitle` label parameter from explainer.
- `ms_title` Title of the dashboard.
- `margin_*` Plot margins. Change `margin_left` for longer/shorter axis labels.
- `w` 420 in px. Inner plot width.
- `h` 280 in px. Inner plot height.
- `bar_width` 16 in px. Default width of bars for all plots, ignored when `scale_plot = TRUE`.
- `line_size` 2 in px. Default width of lines for all plots.
- `point_size` 3 in px. Default point radius for all plots.
- `[bar,line,point]_color` ['#46bac2','#371ea3']
- `positive_color` #8bdcbe for Break Down and Shapley Values bars.
negative_color #f05a71 for Break Down and Shapley Values bars.
default_color #371ea3 for Break Down bar and highlighted line.

Plot specific options:: ** is a two letter code unique to each plot, might be one of [bd, sv, cp, fi, pd, ad, fd, tv, at].

**_title** Plot specific title. Default varies.
**_subtitle** Plot specific subtitle. Default is subtitle.
**_bar_width** Plot specific width of bars. Default is bar_width, ignored when scale_plot = TRUE.
**_line_size** line_size Plot specific width of lines. Default is line_size.
**_point_size** Plot specific point radius. Default is point_size.
**_*_color** Plot specific [bar, line, point] color. Default is [bar, line, point]_color.

Examples

library("DALEX")
library("modelStudio")

# Create a model
model_apartments <- glm(m2.price ~ . ,
  data = apartments)

# Wrap it into an explainer
explainer_apartments <- DALEX::explain(model_apartments,
  data = apartments[-1],
  y = apartments[,1],
  verbose = FALSE)

# Pick some data points
new_apartments <- apartments[1:2,]
rownames(new_apartments) <- c("ap1","ap2")

# Modify default options
op <- modelStudioOptions(
  show_subtitle = TRUE,
  bd_subtitle = "Hello World",
  line_size = 5,
  point_size = 9,
  line_color = "pink",
  point_color = "purple",
  bd_positive_color = "yellow",
  bd_negative_color = "orange"
)

# Make a studio for the model with modified option
modelStudio(explainer_apartments, new_apartments,
  N = 100, B = 10, options = op,
  show_info = FALSE)
Index

modelStudio, 2, 5
modelStudioOptions, 3, 5