Package ‘modelStudio’

May 9, 2020

Title Interactive Studio for Explanatory Model Analysis

Version 1.1.0

Description Automate the explanatory analysis of machine learning predictive models. Generate advanced interactive and animated model explanations in the form of a serverless HTML site with only one line of code. This tool is model agnostic, therefore compatible with most of the black box predictive models and frameworks. The main function computes various (instance and dataset level) model explanations and produces an interactive, customisable dashboard. 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 7.1.0

Imports iBreakDown (>= 1.1.0), ingredients (>= 1.0.0), r2d3, jsonlite, progress, digest

Suggests DALEX (>= 1.0), parallelMap, ranger, xgboost, knitr, rmarkdown, testthat, spelling

VignetteBuilder knitr

URL https://modelstudio.drwhy.ai,
    https://github.com/ModelOriented/modelStudio

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

Language en-US

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 2020-05-08 22:50:10 UTC
modelStudio

R topics documented:

- modelStudio .................................................. 2
- ms_options ..................................................... 5
- ms_update_observations ................................. 7
- ms_update_options ................................. 9

Index 12

---

modelStudio
Interactive Studio for Explanatory Model Analysis

Description

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

Theoretical introduction to the plots: 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 = 300,
    B = 10,
    eda = TRUE,
    show_info = TRUE,
    parallel = FALSE,
    options = ms_options(),
    viewer = "external",
    ...
)

Arguments

- explainer An explainer created with DALEX::explain().
- ... Other parameters.
new_observation
New observations 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 the animation length. Default is 500.

max_features
Maximum number of features to be included in BD and SV plots. Default is 10.

N
Number of observations used for the calculation of PD and AD. 10*N is a number of observations used for the calculation of FI. Default N is 300. See vignette

B
Number of permutation rounds used for calculation of SV and FI. Default is 10. See vignette

eda
Compute EDA plots. Default is TRUE.

show_info
Verbose a 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 ms_options 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,htmlwidget,modelStudio class.

References
• The input object 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
Vignettes: modelStudio - R & Python examples and modelStudio - perks and features

Examples

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

# ex1 classification on 'titanic' data

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

# create an explainer for the model
explainer_titanic <- explain(model_titanic, 
  data = titanic_imputed, 
  y = titanic_imputed$survived, 
  label = "Titanic GLM")

# pick observations
new_observations <- titanic_imputed[1:2,]
rownames(new_observations) <- c("Lucas","James")

# make a studio for the model
modelStudio(explainer_titanic, 
  new_observations)

#: # ex2 regression on 'apartments' data
library("ranger")

model_apartments <- ranger(m2.price ~ . ,data = apartments)
explainer_apartments <- explain(model_apartments, 
  data = apartments, 
  y = apartments$m2.price)

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)

# add information about true labels
modelStudio(explainer_apartments, 
  new_apartments, 
  new_observation_y = new_apartments$m2.price)

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

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

HR_matrix <- model.matrix(status == "fired" ~ . -1, HR)

# fit a model
xgb_matrix <- xgb.DMatrix(HR_matrix, label = HR$status == "fired")
params <- list(max_depth = 3, objective = "binary:logistic", eval_metric = "auc")
model_HR <- xgb.train(params, xgb_matrix, nrounds = 300)
# create an explainer for the model
explainer_HR <- explain(model_HR,
  data = HR_matrix,
  y = HR$status == "fired",
  label = "xgboost")

# pick observations
new_observation <- HR_matrix[1:2, , drop=FALSE]
rownames(new_observation) <- c("id1", "id2")
# make a studio for the model
modelStudio(explainer_HR,
  new_observation)

---

**ms_options**

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
ms_options(...)  
modelStudioOptions(...)  
```

**Arguments**

`...` Options to change in the form `option_name = value`.

**Value**

list of options for `modelStudio`.

**Options**

**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.
**ms_options**

- **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,#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,sv,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.

References
---
- The input object is implemented in [DALEX](https://github.com/ModelOriented/DALEX)
- Feature Importance, Ceteris Paribus, Partial Dependence and Accumulated Dependence plots are implemented in [ingredients](https://github.com/ModelOriented/ingredients)
- Break Down and Shapley Values plots are implemented in [iBreakDown](https://github.com/ModelOriented/iBreakDown)

See Also
---
Vignettes: [modelStudio - R & Python examples](https://github.com/ModelOriented/modelStudio) and [modelStudio - perks and features](https://github.com/ModelOriented/modelStudio)

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

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

# create an explainer for the model
explainer_apartments <- explain(model_apartments,
data = apartments,
y = apartments$m2.price)

# pick observations
new_observation <- apartments[1:2,]
rownames(new_observation) <- c("ap1","ap2")
```
# modify default options
new_options <- ms_options(
  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
modelStudio(explainer_apartments,
             new_observation,
             options = new_options,
             N = 200, B = 5) # faster example

ms_update_observations

Update the observations of a modelStudio object

Description

This function calculates local explanations on new observations and adds them to a modelStudio object.

Usage

ms_update_observations(
  object,
  explainer,
  new_observation = NULL,
  new_observation_y = NULL,
  max_features = 10,
  B = 10,
  show_info = TRUE,
  parallel = FALSE,
  overwrite = FALSE,
  ...
)

Arguments

  object A modelStudio created with modelStudio().
  explainer An explainer created with DALEX::explain().
new_observation
   New observations with columns that correspond to variables used in the model.
new_observation_y
   True label for new_observation (optional).
max_features
   Maximum number of features to be included in BD and SV plots. Default is 10.
B
   Number of permutation rounds used for calculation of SV and FI. Default is 10.
   See vignette
show_info
   Verbose a 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.
overwrite
   Overwrite existing observations and their explanations. Default is FALSE which
   means add new observations to the existing ones.
   ...
   Other parameters.

Value
   An object of the r2d3,htmlwidget,modelStudio class.

References
   • The input object 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
   Vignettes: modelStudio - R & Python examples and modelStudio - perks and features

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

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

   # create an explainer for the model
   explainer_titanic <- explain(model_titanic,
                                data = titanic_imputed,
                                y = titanic_imputed$survived)

   # make a studio for the model
   ms <- modelStudio(explainer_titanic)

   # add new observations
```r
ms <- ms_update_observations(ms,
    explainer_titanic,
    new_observation = titanic_imputed[100:101,],
    new_observation_y = titanic_imputed$survived[100:101])

ms

# overwrite the observations with new ones
ms <- ms_update_observations(ms,
    explainer_titanic,
    new_observation = titanic_imputed[100:101,],
    overwrite = TRUE)

ms
```

---

**ms_update_options**

Update the options of a modelStudio object

**Description**

This function updates the options of a modelStudio object. **WARNING: Editing default options may cause unintended behavior.**

**Usage**

```r
ms_update_options(object, ...)
```

**Arguments**

- `object` A modelStudio created with `modelStudio()`.
- `...` Options to change in the form `option_name = value`, e.g. `time = 0`, `facet_dim = c(1,2)`.

**Value**

An object of the `r2d3,htmlwidget,modelStudio` class.

**Options**

- **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,#46bac2,#371ea3]
positive_color #8bdcbef 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.

References

- The input object 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

Vignettes: modelStudio - R & Python examples and modelStudio - perks and features

Examples

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

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

# create an explainer for the model
explainer_titanic <- explain(model_titanic,
data = titanic_imputed,
y = titanic_imputed$survived)

# make a studio for the model
ms <- modelStudio(explainer_titanic)
# update the options
new_ms <- ms_update_options(ms,
    time = 0,
    facet_dim = c(1,2),
    margin_left = 150)

new_ms
Index

modelStudio, 2, 5, 9
modelStudioOptions (ms_options), 5
ms_options, 3, 5
ms_update_observations, 7
ms_update_options, 9