Package 'iBreakDown'

April 30, 2019

Title Model Agnostic Instance Level Variable Attributions

Version 0.9.6

Description Model agnostic tool for decomposition of predictions from black boxes.
Supports additive attributions and attributions with interactions.
The Break Down Table shows contributions of every variable to a final prediction.
The Break Down Plot presents variable contributions in a concise graphical way.
This package works for classification and regression models.
It is an extension of the 'breakDown' package (Staniak and Biecek 2018) <doi:10.32614/RJ-2018-072>,
with new and faster strategies for orderings.
It supports interactions in explanations and has interactive visuals (implemented with 'D3.js' library).
The methodology behind is described in the 'iBreakDown' article (Gosiewska and Biecek 2019) <arXiv:1903.11420>
This package is a part of the 'DrWhy.AI' universe (Biecek 2018) <arXiv:1806.08915>.

Depends R (>= 3.0)

Date 2019-04-01

License GPL-2

Encoding UTF-8

LazyData true

Imports ggplot2, DALEX

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, caret, randomForest, e1071, xgboost, ranger, nnet, testthat, r2d3

VignetteBuilder knitr

URL https://ModelOriented.github.io/iBreakDown/

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

NeedsCompilation no

Author Przemyslaw Biecek [aut, cre],
Alicja Gosiewska [aut],
Dariusz Komosinski [ctb]

Maintainer Przemyslaw Biecek <przemyslaw.biecek@gmail.com>

Repository CRAN

Date/Publication 2019-04-30 17:10:04 UTC

R topics documented:

`break_down` *Model Agnostic Sequential Variable Attributions*

Description

This function finds Variable Attributions via Sequential Variable Conditioning. It calls either `local_attributions` for additive attributions or `local_interactions` for attributions with interactions.

Usage

```
break_down(x, ..., interactions = FALSE)

## S3 method for class 'explainer'
break_down(x, new_observation, ...,
  interactions = FALSE)

## Default S3 method:
break_down(x, data, predict_function = predict,
  new_observation, keep_distributions = FALSE, order = NULL,
  label = class(x)[1], ..., interactions = interactions)
```

Arguments

`x`	a model to be explained, or an explainer created with function 'DALEX::explain()'.
`...`	parameters passed to 'local_*' functions.
`interactions`	shall interactions be included?
`new_observation`	a new observation with columns that correspond to variables used in the model.

`data` validation dataset, will be extracted from ‘x‘ if it is an explainer.

`predict_function` predict function, will be extracted from ‘x‘ if it’s an explainer.

`keep_distributions` if ‘TRUE‘, then distribution of partial predictions is stored and can be plotted with the generic ‘plot()‘.

`order` if not ‘NULL‘, then it will be a fixed order of variables. It can be a numeric vector or vector with names of variables.

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

Value

an object of the ‘break_down‘ class.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

See Also

local_attributions, local_interactions

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                      data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                          data = titanic_small[,-9],
                          y = titanic_small$survived == "yes")
bd_rf <- break_down(explain_titanic_glm, titanic_small[1, ])
bd_rf
plot(bd_rf, max_features = 3)


## Not run:
library("randomForest")
set.seed(1313)
# example with interaction
# classification for HR data
model <- randomForest(status ~ . , data = HR)
new_observation <- HR_test[1,]
```

```
explainer_rf <- explain(model,
                        data = HR[1:1000,1:5],
                        y = HR$status[1:1000])

bd_rf <- break_down(explainer_rf,
                           new_observation)
bd_rf
plot(bd_rf)
```

`break_down_uncertainty` *Explanation Level Uncertainty of Sequential Variable Attribution*

Description

The ‘break_down_uncertainty()‘ calles ‘B‘ times the break down algorithm for random orderings. Then it calculated distribution of attributions for these different orderings. Note that the ‘shap()‘ function is just a simplified interface to the ‘break_down_uncertainty()‘ function with by default ‘B=25‘ random draws.

Usage

```
break_down_uncertainty(x, ..., B = 10)

## S3 method for class 'explainer'
break_down_uncertainty(x, new_observation, ...,
  B = 10)

## Default S3 method:
break_down_uncertainty(x, data,
  predict_function = predict, new_observation, label = class(x)[1],
  ..., path = NULL, B = 10)

shap(x, ..., B = 25)
```

Arguments

`x`	a model to be explained, or an explainer created with function ‘DALEX::explain()‘.
`...`	other parameters.
`B`	number of random paths
`new_observation`	a new observation with columns that correspond to variables used in the model.
`data`	validation dataset, will be extracted from ‘x‘ if it is an explainer.
`predict_function`	predict function, will be extracted from ‘x‘ if it is an explainer.

label	name of the model. By default it's extracted from the 'class' attribute of the model.
path	if specified, then this path will be highlighed on the plot. Use 'average' in order to show an average effect

Value

an object of the 'break_down_uncertainty' class.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

See Also

break_down, local_attributions

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic_small[,-9],
                           y = titanic_small$survived == "yes")

# there is no explanation level uncertainty linked with additive models
bd_rf <- break_down_uncertainty(explain_titanic_glm, titanic_small[1, ])
bd_rf
plot(bd_rf)


## Not run:
library("randomForest")
set.seed(1313)
model <- randomForest(status ~ . , data = HR)
new_observation <- HR_test[1,]

explainer_rf <- explain(model,
                        data = HR[1:1000, 1:5])

bd_rf <- break_down_uncertainty(explainer_rf,
                           new_observation)
bd_rf
plot(bd_rf)
```

```
# example for regression - apartment prices
# here we do not have intreactions
model <- randomForest(m2.price ~ . , data = apartments)
explainer_rf <- explain(model,
                        data = apartments_test[1:1000, 2:6],
                        y = apartments_test$m2.price[1:1000])

bd_rf <- break_down_uncertainty(explainer_rf, apartments_test[1,])
bd_rf
plot(bd_rf)

bd_rf <- break_down_uncertainty(explainer_rf, apartments_test[1,], path = 1:5)
plot(bd_rf)

bd_rf <- break_down_uncertainty(explainer_rf,
                                    apartments_test[1,],
                                    path = c("floor", "no.rooms", "district",
                                        "construction.year", "surface"))
plot(bd_rf)

bd_rf <- shap(explainer_rf,
              apartments_test[1,])
bd_rf
plot(bd_rf)
plot(bd_rf, show_boxplots = FALSE)
```

`local_attributions` *Model Agnostic Sequential Variable attributions*

Description

This function finds Variable attributions via Sequential Variable Conditioning. The complexity of this function is O(2*p). This function works in a similar way to step-up and step-down greedy approximations in function 'breakDown::break_down()'. The main difference is that in the first step the order of variables is determined. And in the second step the impact is calculated.

Usage

```
local_attributions(x, ...)

## S3 method for class 'explainer'
local_attributions(x, new_observation,
  keep_distributions = FALSE, ...)

## Default S3 method:
local_attributions(x, data, predict_function = predict,
  new_observation, label = class(x)[1], keep_distributions = FALSE,
  order = NULL, ...)
```

Arguments

x	a model to be explained, or an explainer created with function 'DALEX::explain()'.
...	other parameters.
new_observation	a new observation with columns that correspond to variables used in the model.
keep_distributions	if 'TRUE', then distribution of partial predictions is stored and can be plotted with the generic 'plot()'.
data	validation dataset, will be extracted from 'x' if it is an explainer.
predict_function	predict function, will be extracted from 'x' if it is an explainer.
label	name of the model. By default it's extracted from the 'class' attribute of the model.
order	if not 'NULL', then it will be a fixed order of variables. It can be a numeric vector or vector with names of variables.

Value

an object of the 'break_down' class.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

See Also

break_down, local_interactions

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_small[,-9],
                               y = titanic_small$survived == "yes")
bd_rf <- local_attributions(explain_titanic_glm, titanic_small[1, ])
bd_rf
plot(bd_rf, max_features = 3)


## Not run:
```

```
library("randomForest")
set.seed(1313)
# example with interaction
# classification for HR data
model <- randomForest(status ~ . , data = HR)
new_observation <- HR_test[1,]

explainer_rf <- explain(model,
                        data = HR[1:1000,1:5],
                        y = HR$status[1:1000])

bd_rf <- local_attributions(explainer_rf,
                            new_observation)
bd_rf
plot(bd_rf)
plot(bd_rf, baseline = 0)

# example for regression - apartment prices
# here we do not have interactions
model <- randomForest(m2.price ~ . , data = apartments)
explainer_rf <- explain(model,
                        data = apartments_test[1:1000,2:6],
                        y = apartments_test$m2.price[1:1000])

bd_rf <- local_attributions(explainer_rf,
                            apartments_test[1,])
bd_rf
plot(bd_rf, digits = 1)

bd_rf <- local_attributions(explainer_rf,
                            apartments_test[1,],
                            keep_distributions = TRUE)
plot(bd_rf, plot_distributions = TRUE)
```

`local_interactions` *Model Agnostic Sequential Variable Attributions with Interactions*

Description

This function implements decomposition of model predictions with identification of interactions. The complexity of this function is O(2*p) for additive models and O(2*p^2) for interactions. This function works in a similar way to step-up and step-down greedy approximations in function 'breakDown::break_down()'. The main difference is that in the first step the order of variables and interactions is determined. And in the second step the impact is calculated.

Usage

```
local_interactions(x, ...)
```

```
## S3 method for class 'explainer'
local_interactions(x, new_observation,
  keep_distributions = FALSE, ...)

## Default S3 method:
local_interactions(x, data, predict_function = predict,
  new_observation, label = class(x)[1], keep_distributions = FALSE,
  order = NULL, interaction_preference = 1, ...)
```

Arguments

x	a model to be explained, or an explainer created with function 'DALEX::explain()'.
...	other parameters.
new_observation	a new observation with columns that correspond to variables used in the model.
keep_distributions	if 'TRUE', then the distribution of partial predictions is stored in addition to the average.
data	validation dataset, will be extracted from 'x' if it's an explainer.
predict_function	predict function, will be extracted from 'x' if it's an explainer.
label	character - the name of the model. By default it's extracted from the 'class' attribute of the model.
order	if not 'NULL', then it will be a fixed order of variables. It can be a numeric vector or vector with names of variables/interactions.
interaction_preference	a constant that set the preference for interactions. By default '1'. The larger the more frequently interactions will be presented in explanations.

Value

an object of the 'break_down' class.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

See Also

break_down, local_attributions

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
```

```
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic_small[,-9],
                           y = titanic_small$survived == "yes")

bd_rf <- local_interactions(explain_titanic_glm, titanic_small[1, ], interaction_preference = 500)
bd_rf
plot(bd_rf, max_features = 2)


library("DALEX")
# example with interaction
# classification for HR data
model <- randomForest(status ~ . , data = HR)
new_observation <- HR_test[1,]

explainer_rf <- explain(model,
                 data = HR[1:1000,1:5],
                 y = HR$status[1:1000])

bd_rf <- local_interactions(explainer_rf,
                 new_observation)

bd_rf
plot(bd_rf)

# example for regression - apartment prices
# here we do not have intreactions
model <- randomForest(m2.price ~ . , data = apartments)
explainer_rf <- explain(model,
         data = apartments_test[1:1000,2:6],
         y = apartments_test$m2.price[1:1000])

new_observation <- apartments_test[1,]

bd_rf <- local_interactions(explainer_rf,
                 new_observation,
                 keep_distributions = TRUE)

bd_rf
plot(bd_rf)
plot(bd_rf, plot_distributions = TRUE)
```

---

plot.break_down *Plot Generic for Break Down Objects*

Description

Plots waterfall break down for objects of the 'break_down' class. Usually executed after 'break_down()' function, 'local_attributions()' or 'local_interactions()'.

Usage

```
## S3 method for class 'break_down'
plot(x, ..., baseline = NA, max_features = 10,
  min_max = NA, vcolors = DALEX::theme_drwhy_colors_break_down(),
  digits = 3, rounding_function = round, add_contributions = TRUE,
  shift_contributions = 0.05, plot_distributions = FALSE)
```

Arguments

x	the model model of 'break_down' class.
...	other parameters.
baseline	if numeric then veritical line starts in 'baseline'.
max_features	maximal number of features to be included in the plot. default value is 4.
min_max	a range of OX axis. By deafult 'NA' therefore will be extracted from the contributions of 'x'. But can be set to some constants, usefull if these plots are used for comparisons.
vcolors	named vector with colors.
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.
add_contributions	shall variable contributions to be added the the plot?
shift_contributions	how much labels sholud be shifted right as a fraction of range. By default 0.05
plot_distributions	if 'TRUE' then distributions of conditional propotions will be plotted. This requires keep_distributions=TRUE in the break_down, local_attributions, or local_interactions.

Value

a 'ggplot2' object.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                            data = titanic_small[,-9],
                            y = titanic_small$survived == "yes")
bd_rf <- break_down(explain_titanic_glm, titanic_small[1, ])
bd_rf
plot(bd_rf, max_features = 3)


## Not run:
library("randomForest")
set.seed(1313)
# example with interaction
# classification for HR data
model <- randomForest(status ~ . , data = HR)
new_observation <- HR_test[1,]

explainer_rf <- explain(model,
                        data = HR[1:1000,1:5],
                        y = HR$status[1:1000])

bd_rf <- local_attributions(explainer_rf,
                            new_observation)
bd_rf
plot(bd_rf)
plot(bd_rf, baseline = 0)
plot(bd_rf, min_max = c(0,1))

bd_rf <- local_attributions(explainer_rf,
                            new_observation,
                            keep_distributions = TRUE)
bd_rf
plot(bd_rf, plot_distributions = TRUE)

bd_rf <- local_interactions(explainer_rf,
                 new_observation,
                 keep_distributions = TRUE)

bd_rf
plot(bd_rf)
plot(bd_rf, plot_distributions = TRUE)

# example for regression - apartment prices
# here we do not have intreactions
```

```
model <- randomForest(m2.price ~ . , data = apartments)
explainer_rf <- explain(model,
                        data = apartments_test[1:1000,2:6],
                        y = apartments_test$m2.price[1:1000])

bd_rf <- local_attributions(explainer_rf,
                            apartments_test[1,])
bd_rf
plot(bd_rf, digits = 1)
plot(bd_rf, digits = 1, baseline = 0)

bd_rf <- local_attributions(explainer_rf,
                            apartments_test[1,],
                            keep_distributions = TRUE)
plot(bd_rf, plot_distributions = TRUE)

bd_rf <- local_interactions(explainer_rf,
                 new_observation = apartments_test[1,],
                 keep_distributions = TRUE)

bd_rf
plot(bd_rf)
plot(bd_rf, plot_distributions = TRUE)
```

`plot.break_down_uncertainty` *Plot Generic for Break Down Uncertainty Objects*

Description

Plot Generic for Break Down Uncertainty Objects

Usage

```
## S3 method for class 'break_down_uncertainty'
plot(x, ...,
  vcolors = DALEX::theme_drwhy_colors_break_down(),
  show_boxplots = TRUE)
```

Arguments

`x`	the model model of ‘break_down_uncertainty‘ class.
`...`	other parameters.
`vcolors`	named vector with colors.
`show_boxplots`	logical if ‘TRUE‘ (default) boxplot will be plotted to show uncertanity of attributions

Value

a ‘ggplot2‘ object.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic_small[,-9],
                           y = titanic_small$survived == "yes")
bd_rf <- shap(explain_titanic_glm, titanic_small[1, ])
bd_rf
plot(bd_rf)


## Not run:
library("randomForest")
set.seed(1313)

model <- randomForest(status ~ . , data = HR)
new_observation <- HR_test[1,]

explainer_rf <- explain(model,
                        data = HR[1:1000,1:5],
                        y = HR$status[1:1000])

bd_rf <- break_down_uncertainty(explainer_rf,
                           new_observation,
                           path = c(3,2,4,1,5))
bd_rf
plot(bd_rf)

# example for regression - apartment prices
# here we do not have intreactions
model <- randomForest(m2.price ~ . , data = apartments)
explainer_rf <- explain(model,
                        data = apartments_test[1:1000,2:6],
                        y = apartments_test$m2.price[1:1000])

bd_rf <- break_down_uncertainty(explainer_rf,
                                    apartments_test[1,],
```

```
                                        path = c("floor", "no.rooms", "district",
                                            "construction.year", "surface"))
bd_rf
plot(bd_rf)

bd_rf <- shap(explainer_rf,
              apartments_test[1,])
bd_rf
plot(bd_rf)
plot(bd_rf, show_boxplots = FALSE)
```

---

plotD3 *Plot Break Down Objects in D3 with r2d3 package.*

Description

Experimental interactive explainer created with 'D3.js' library.

Usage

```
plotD3(x, ...)

## S3 method for class 'break_down'
plotD3(x, ..., baseline = NA, max_features = 10,
  min_max = NA, vcolors = DALEX::theme_drwhy_colors_break_down(),
  digits = 3, rounding_function = round)
```

Arguments

x	the model model of 'break_down' class.
...	other parameters.
baseline	if numeric then veritical line will start in baseline.
max_features	maximal number of features to be included in the plot. default value is 10.
min_max	a range of OX axis. By deafult 'NA' therefore will be extracted from the contributions of 'x'. But can be set to some constants, usefull if these plots are used for comparisons.
vcolors	named vector with colors.
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.

Value

an ‘r2d3‘ object.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic_small[,-9],
                           y = titanic_small$survived == "yes")
bd_rf <- local_attributions(explain_titanic_glm, titanic_small[1, ])
bd_rf
plotD3(bd_rf)


library("randomForest")
titanic <- na.omit(titanic)
model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                   fare + sibsp + parch,  data = titanic)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic[,-9],
                              y = titanic$survived == "yes",
                              label = "Random Forest v7")

new_passanger <- data.frame(
  class = factor("1st", levels = c("1st", "2nd", "3rd", "deck crew", "engineering crew",
                                   "restaurant staff", "victualling crew")),
  gender = factor("male", levels = c("female", "male")),
  age = 8,
  sibsp = 0,
  parch = 0,
  fare = 72,
  embarked = factor("Southampton",
                 levels = c("Belfast", "Cherbourg", "Queenstown", "Southampton")))

  rf_la <- local_attributions(explain_titanic_rf, new_passanger)
  rf_la

  plotD3(rf_la)
  plotD3(rf_la, max_features = 3)
```

```
plotD3(rf_la, max_features = 3, min_max = c(0,1))
```

`print.break_down` *Print Generic for Break Down Objects*

Description

Print Generic for Break Down Objects

Usage

```
## S3 method for class 'break_down'
print(x, ..., digits = 3,
  rounding_function = round)
```

Arguments

`x`	the model model of 'break_down' class.
`...`	other parameters.
`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.

Value

a data frame

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

`print.break_down_uncertainty` *Print Generic for Break Down Uncertainty Objects*

Description

Print Generic for Break Down Uncertainty Objects

Usage

```
## S3 method for class 'break_down_uncertainty'
print(x, ...)
```

Arguments

`x`	object of 'break_down_uncertainty' class.
`...`	other parameters.

Value

a data frame.

References

Predictive Models: Visual Exploration, Explanation and Debugging https://pbiecek.github.io/PM_VEE

Examples

```
library("DALEX")
library("iBreakDown")
# Toy examples, because CRAN angels ask for them
titanic <- na.omit(titanic)
set.seed(1313)
titanic_small <- titanic[sample(1:nrow(titanic), 500), c(1,2,6,9)]
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_small, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_small[,-9],
                               y = titanic_small$survived == "yes")
bd_rf <- break_down_uncertainty(explain_titanic_glm, titanic_small[1, ])
bd_rf
plot(bd_rf)


## Not run:
library("randomForest")
set.seed(1313)
model <- randomForest(status ~ . , data = HR)
```

```
new_observation <- HR_test[1,]

explainer_rf <- explain(model,
                        data = HR[1:1000,1:5],
                        y = HR$status[1:1000])

bd_rf <- break_down_uncertainty(explainer_rf,
                           new_observation)
bd_rf

# example for regression - apartment prices
# here we do not have intreactions
model <- randomForest(m2.price ~ . , data = apartments)
explainer_rf <- explain(model,
                        data = apartments_test[1:1000,2:6],
                        y = apartments_test$m2.price[1:1000])

bd_rf <- break_down_uncertainty(explainer_rf, apartments_test[1,])
bd_rf
```

Index