Package 'vivo'

September 3, 2019

Title Local Variable Importance via Oscillations of Ceteris Paribus Profiles

Version 0.1.1

Description Provides an easy to calculate variable importance measure based on Ceteris Paribus plot and is calculated in eight variants. We obtain eight variants measure through the possible combinations of three parameters such as absolute_deviation, point and density.

Depends R (>= 3.0)

License GPL-2

Encoding UTF-8

LazyData true

Imports ggplot2, dplyr, ingredients, DALEX

Suggests knitr, rmarkdown, mlbench, randomForest, gridExtra, grid, lattice, testthat

VignetteBuilder knitr

RoxygenNote 6.1.1

URL https://github.com/ModelOriented/vivo

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

NeedsCompilation no

Author Anna Kozak [aut, cre], Przemyslaw Biecek [aut, ths]

Maintainer Anna Kozak <anna1993kozak@gmail.com>

Repository CRAN

Date/Publication 2019-09-03 17:20:03 UTC

R topics documented:

`calculate_variable_split` *Internal Function for Split Points for Selected Variables*

Description

This function calculate candidate splits for each selected variable. For numerical variables splits are calculated as percentiles (in general uniform quantiles of the length grid_points). For all other variables splits are calculated as unique values.

Usage

```
calculate_variable_split(data, variables = colnames(data),
  grid_points = 101)
```

Arguments

`data`	validation dataset. Is used to determine distribution of observations.
`variables`	names of variables for which splits shall be calculated
`grid_points`	number of points used for response path

Value

A named list with splits for selected variables

Note

This function is a copy of `calculate_varaible_split()` from `ingredients` package.

Author(s)

Przemyslaw Biecek

`calculate_weight` *Calculated empirical density and weight based on variable split.*

Description

This function calculate an empirical density of raw data based on variable split from Ceteris Paribus profiles. Then calculated weight for values generated by ingredients::ceteris_paribus().

Usage

```
calculate_weight(profiles, data, variable_split)
```

Arguments

profiles	data.frame generated by ingredients::ceteris_paribus()
data	data.frame with raw data to model
variable_split	list generated by vivo::calculate_variable_split()

Value

Return an weight based on empirical density.

Examples

```
library("DALEX", warn.conflicts = FALSE, quietly = TRUE)
data(apartments)

library("ingredients", warn.conflicts = FALSE, quietly = TRUE)
split <- vivo::calculate_variable_split(apartments,
                        variables = colnames(apartments))

library("randomForest", warn.conflicts = FALSE, quietly = TRUE)
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface +
                                    floor + no.rooms, data = apartments)

explainer_rf <- explain(apartments_rf_model, data = apartmentsTest[,2:5],
                        y = apartmentsTest$m2.price)

new_apartment <- data.frame(construction.year = 1998, surface = 88, floor = 2L, no.rooms = 3)

profiles <- ceteris_paribus(explainer_rf, new_apartment)

library("vivo")
calculate_weight(profiles, data = apartments[, 2:5], variable_split = split)
```

---

local_variable_importance

Local Variable Importance measure based on Ceteris Paribus profiles.

Description

This function calculate local importance measure in eight variants. We obtain eight variants measure through the possible options of three parameters such as ‘absolute_deviation‘, ‘point‘ and ‘density‘.

Usage

```
local_variable_importance(profiles, data, absolute_deviation = TRUE,
  point = TRUE, density = TRUE)
```

Arguments

profiles	data.frame generated by ingredients::ceteris_paribus()
data	data.frame with raw data to model
absolute_deviation	logical parameter, if 'absolute_deviation = TRUE' then measure is calculated as absolute deviation, else is calculated as a root from average squares
point	logical parameter, if 'point = TRUE' then measure is calculated as a distance from f(x), else measure is calculated as a distance from average profiles
density	logical parameter, if 'density = TRUE' then measure is weighted based on the density of variable, else is not weighted

Value

A data.frame of the class 'local_variable_importance'. It's a data.frame with calculated local variable importance measure.

Examples

```
library("DALEX")
data(apartments)

library("randomForest")
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface +
                                    floor + no.rooms, data = apartments)

explainer_rf <- explain(apartments_rf_model, data = apartmentsTest[,2:5],
                        y = apartmentsTest$m2.price)

new_apartment <- data.frame(construction.year = 1998, surface = 88, floor = 2L, no.rooms = 3)

library("ingredients")
profiles <- ceteris_paribus(explainer_rf, new_apartment)

library("vivo")
local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = TRUE, density = TRUE)

local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = TRUE, density = FALSE)

local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = FALSE, density = TRUE)
```

plot.local_importance *Plot Local Variable Importance measure*

Description

Function plot.local_importance plots local importance measure based on Ceteris Paribus profiles.

Usage

```
## S3 method for class 'local_importance'
plot(x, ...)
```

Arguments

x	object returned from local_variable_importance() function
...	other parameters

Value

a ggplot2 object

Examples

```
library("DALEX")
data(apartments)

library("randomForest")
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface +
                                    floor + no.rooms, data = apartments)

explainer_rf <- explain(apartments_rf_model, data = apartmentsTest[,2:5],
                        y = apartmentsTest$m2.price)

new_apartment <- data.frame(construction.year = 1998, surface = 88, floor = 2L, no.rooms = 3)

library("ingredients")
profiles <- ceteris_paribus(explainer_rf, new_apartment)

library("vivo")
measure <- local_variable_importance(profiles, apartments[,2:5],
                          absolute_deviation = TRUE, point = TRUE, density = FALSE)

plot(measure)
```

Index