Package ‘ingredients’

August 27, 2019

Title Effects and Importances of Model Ingredients

Version 0.3.9

Description Collection of tools for assessment of feature importance and feature effects.
Key functions are:
feature_importance() for assessment of global level feature importance,
ceteris_paribus() for calculation of the what-if plots,
partial_dependency() for partial dependency plots,
conditional_dependency() for conditional dependency plots,
accumulated_dependency() for accumulated local effects plots,
aggregate_profiles() and cluster_profiles() for aggregation of ceteris paribus profiles,
generic print() and plot() for better usability of selected explainers,
generic plotD3() for interactive, D3 based explanations, and
generic describe() for explanations in natural language.
The package 'ingredients' is a part of the 'DrWhy.AI' universe (Biecek 2018) <arXiv:1806.08915>.

Depends R (>= 3.0)

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports DALEX, ggplot2, glmnet, scales

Suggests gower, randomForest, xgboost, testthat, r2d3, ggpubr,
jsonlite, knitr, rmarkdown

URL `https://ModelOriented.github.io/ingredients/`,
`https://github.com/ModelOriented/ingredients`

BugReports `https://github.com/ModelOriented/ingredients/issues`

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2019-08-26 22:10:06 UTC

R topics documented:

accumulated_dependency

Accumulated Local Effects Profiles aka ALEPlots

Description

Accumulated Local Effects Profiles accumulate local changes in Ceteris Paribus Profiles. Function `accumulated_dependency` calls `ceteris_paribus` and then `aggregate_profiles`.

Usage

```
accumulated_dependency(x, ...)

## S3 method for class 'explainer'
accumulated_dependency(x, variables = NULL,
  N = 500, variable_splits = NULL, grid_points = 101, ...,
  variable_type = "numerical")

## Default S3 method:
accumulated_dependency(x, data,
  predict_function = predict, label = class(x)[1], variables = NULL,
  N = 500, variable_splits = NULL, grid_points = 101, ...,
  variable_type = "numerical")

## S3 method for class 'ceteris_paribus_explainer'
accumulated_dependency(x, ...,
  variables = NULL)
```

Arguments

`x`	an explainer created with function `DALEX::explain()`, an object of the class `ceteris_paribus_explainer` or a model to be explained.
`...`	other parameters
`variables`	names of variables for which profiles shall be calculated. Will be passed to `calculate_variable_split`. If `NULL` then all variables from the validation data will be used.
`N`	number of observations used for calculation of partial dependency profiles. By default, 500 observations will be chosen randomly.
`variable_splits`	named list of splits for variables, in most cases created with `calculate_variable_split`. If `NULL` then it will be calculated based on validation data avaliable in the `explainer`.
`grid_points`	number of points for profile. Will be passed to`calculate_variable_split`.
`variable_type`	a character. If "numerical" then only numerical variables will be calculated. If "categorical" then only categorical variables will be calculated.

`data`	validation dataset Will be extracted from `x` if it's an explainer NOTE: It is best when target variable is not present in the `data`
`predict_function`	predict function 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

Details

Find more detailes in the Accumulated Local Dependency Chapter.

Value

an object of the class `aggregated_profiles_explainer`

References

ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots `https://cran.r-project.org/package=ALEPlot`, Predictive Models: Visual Exploration, Explanation and Debugging `https://pbiecek.github.io/PM_VEE`

Examples

```
library("DALEX")

titanic_imputed$country <- NULL

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_imputed, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed$survived == "yes",
                               verbose = FALSE)

adp_glm <- accumulated_dependency(explain_titanic_glm,
                                  N = 50, variables = c("age", "fare"))
head(adp_glm)
plot(adp_glm)


library("randomForest")

model_titanic_rf <- randomForest(survived == "yes" ~.,  data = titanic_imputed)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed$survived == "yes",
                              verbose = FALSE)

adp_rf <- accumulated_dependency(explain_titanic_rf, N = 200, variable_type = "numerical")
```

```
plot(adp_rf)

adp_rf <- accumulated_dependency(explain_titanic_rf, N = 200, variable_type = "categorical")
plotD3(adp_rf, variable_type = "categorical", label_margin = 80, scale_plot = TRUE)
```

aggregate_profiles *Aggregates Ceteris Paribus Profiles*

Description

The function aggregate_profiles() calculates an aggregate of ceteris paribus profiles. It can be: Partial Dependency Profile (average across Ceteris Paribus Profiles), Conditional Dependency Profile (local weighted average across Ceteris Paribus Profiles) or Accumulated Local Dependency Profile (cummulated average local changes in Ceteris Paribus Profiles).

Usage

```
aggregate_profiles(x, ..., variable_type = "numerical", groups = NULL,
  type = "partial", variables = NULL)
```

Arguments

x	a ceteris paribus explainer produced with function ceteris_paribus()
...	other explainers that shall be calculated together
variable_type	a character. If "numerical" then only numerical variables will be calculated. If "categorical" then only categorical variables will be calculated.
groups	a variable name that will be used for grouping. By default NULL which means that no groups shall be calculated
type	either "partial"/"conditional"/"accumulated" for parital dependence, conditional profiles of accumulated local effects
variables	if not NULL then aggregate only for selected variables will be calculated

Value

an object of the class aggregated_profiles_explainer

References

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

Examples

```
library("DALEX")
library("randomForest")

titanic_imputed$country <- NULL

model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic_imputed)

explain_titanic_rf <- explain(model_titanic_rf,
                           data = titanic_imputed[,-8],
                           y = titanic_imputed$survived == "yes")

selected_passangers <- select_sample(titanic_imputed, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
head(cp_rf)

# continouse variable
pdp_rf_p <- aggregate_profiles(cp_rf, variables = "age", type = "partial")
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_c <- aggregate_profiles(cp_rf, variables = "age", type = "conditional")
pdp_rf_c$`_label_` <- "RF_conditional"
pdp_rf_a <- aggregate_profiles(cp_rf, variables = "age", type = "accumulated")
pdp_rf_a$`_label_` <- "RF_accumulated"

plot(pdp_rf_p, pdp_rf_c, pdp_rf_a, color = "_label_")


pdp_rf <- aggregate_profiles(cp_rf, variables = "age",
                             groups = "gender")
head(pdp_rf)
plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggregated_profiles(pdp_rf, size = 3, color = "_label_")

# categorical variable
pdp_rf_p <- aggregate_profiles(cp_rf, variables = "class",
                               variable_type = "categorical",  type = "partial")
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_c <- aggregate_profiles(cp_rf, variables = "class",
                               variable_type = "categorical", type = "conditional")
pdp_rf_c$`_label_` <- "RF_conditional"
pdp_rf_a <- aggregate_profiles(cp_rf, variables = "class",
                               variable_type = "categorical", type = "accumulated")
pdp_rf_a$`_label_` <- "RF_accumulated"
plot(pdp_rf_p, pdp_rf_c, pdp_rf_a, color = "_label_")

# or maybe flipped?
library(ggplot2)
plot(pdp_rf_p, pdp_rf_c, pdp_rf_a, color = "_label_") + coord_flip()
```

```
pdp_rf <- aggregate_profiles(cp_rf, variables = "class", variable_type = "categorical",
                             groups = "gender")
head(pdp_rf)
plot(pdp_rf, variables = "class")
# or maybe flipped?
plot(pdp_rf, variables = "class") + coord_flip()
```

---

aspect_importance

Calculates the feature groups importance (called aspects importance) for a selected observation

---

Description

Aspect Importance function takes a sample from a given dataset and modifies it. Modification is made by replacing part of its aspects by values from the observation. Then function is calculating the difference between the prediction made on modified sample and the original sample. Finally, it measures the impact of aspects on the change of prediction by using the linear model or lasso.

Usage

```
aspect_importance(x, ...)

## S3 method for class 'explainer'
aspect_importance(x, new_observation, aspects,
  N = 100, sample_method = "default", n_var = 0, f = 2,
  show_cor = FALSE, ...)

## Default S3 method:
aspect_importance(x, data, predict_function = predict,
  new_observation, aspects, N = 100, sample_method = "default",
  n_var = 0, f = 2, show_cor = FALSE, ...)

lime(x, ...)
```

Arguments

`x`	an explainer created with the `DALEX::explain()` function or a model to be explained.
`...`	other parameters
`new_observation`	selected observation with columns that corresponds to variables used in the model
`aspects`	list containting grouping of features into aspects
`N`	number of observarions to be sampled from data

`sample_method`	sampling method in `get_sample`
`n_var`	how many non-zero coefficients should be after lasso fitting, if zero than linear regression is used
`f`	frequency in `get_sample`
`show_cor`	show if all features in aspect are pairwise positivly correlated, works only if dataset contains solely numeric values
`data`	dataset, it will be extracted from `x` if it's an explainer NOTE: It is best when target variable is not present in the `data`
`predict_function`	predict function, it will be extracted from `x` if it's an explainer

Value

An object of the class `aspect_importance`. Contains dataframe that describes aspects' importance.

Examples

```
library("DALEX")

titanic_imputed$country <- NULL

model_titanic_glm <- glm(survived == "yes" ~
                         class+gender+age+sibsp+parch+fare+embarked,
                         data = titanic_imputed,
                         family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed$survived == "yes")

aspects <- list(wealth = c("class", "fare"),
                family = c("sibsp", "parch"),
                personal = c("gender", "age"),
                embarked = "embarked")

aspect_importance(explain_titanic_glm,
                  new_observation = titanic_imputed[1,],
                  aspects = aspects)


library("randomForest")
model_titanic_rf <- randomForest(survived ~ class + gender + age + sibsp +
                                 parch + fare + embarked,
                                 data = titanic_imputed)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed$survived == "yes")

aspect_importance(explain_titanic_rf,
                  new_observation = titanic_imputed[1,],
                  aspects = aspects)
```

`aspect_importance_single` *Aspects importance for single aspects*

Description

Calculates aspect_importance for single aspects (every aspect contains only one feature).

Usage

```
aspect_importance_single(x, data, predict_function = predict,
  new_observation, N = 100, sample_method = "default", n_var = 0,
  f = 2, response_variable = "")
```

Arguments

`x`	a model to be explained
`data`	dataset
`predict_function`	predict function
`new_observation`	selected observation with columns that corresponds to variables used in the model
`N`	number of rows to be sampled from data
`sample_method`	sampling method in `get_sample`
`n_var`	how many non-zero coefficients for lasso fitting, if zero than linear regression is used
`f`	frequency in in `get_sample`
`response_variable`	name of response variable, should be provided if it is included in data

Value

An object of the class 'aspect_importance'. Contains dataframe that describes aspects' importance.

Examples

```
library("DALEX")
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~
                           class+gender+age+sibsp+parch+fare+embarked,
                         data = titanic, family = "binomial")

aspect_importance_single(model_titanic_glm, titanic, new_observation = titanic[1,],
                  response_variable = "survived")
```

`calculate_oscillations`

Calculate Oscillations for Ceteris Paribus Explainer

Description

Oscillations are proxies for local feature importance at the instance level. Find more detailes in Ceteris Paribus Oscillations Chapter.

Usage

```
calculate_oscillations(x, sort = TRUE, ...)
```

Arguments

`x`	a ceteris_paribus explainer produced with the `ceteris_paribus()` function
`sort`	a logical value. If TRUE then rows are sorted along the oscillations
`...`	other arguments

Value

an object of the class `ceteris_paribus_oscillations`

References

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

Examples

```
library("DALEX")

titanic_small <- titanic_imputed[1:500, 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[,-4],
                               y = titanic_small$survived == "yes")

cp_rf <- ceteris_paribus(explain_titanic_glm, titanic_small[1,])

calculate_oscillations(cp_rf)


library("randomForest")

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

explainer_rf <- explain(apartments_rf_model,
                        data = apartmentsTest,
                         y = apartmentsTest$m2.price)

apartment <- apartmentsTest[1,]

cp_rf <- ceteris_paribus(explainer_rf, apartment)

calculate_oscillations(cp_rf)
```

calculate_variable_profile

Internal Function for Individual Variable Profiles

Description

This function calculates individual variable profiles (ceteris paribus profiles), i.e. series of predictions from a model calculated for observations with altered single coordinate.

Usage

```
calculate_variable_profile(data, variable_splits, model,
  predict_function = predict, ...)

## Default S3 method:
calculate_variable_profile(data, variable_splits, model,
  predict_function = predict, ...)
```

Arguments

`data`	set of observations. Profile will be calculated for every observation (every row)
`variable_splits`	named list of vectors. Elements of the list are vectors with points in which profiles should be calculated. See an example for more details.

`model`	a model that will be passed to the `predict_function`
`predict_function`	function that takes data and model and returns numeric predictions. Note that the ... arguments will be passed to this function.
`...`	other parameters that will be passed to the `predict_function`

Details

Note that `calculate_variable_profile` function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

a data frame with profiles for selected variables and selected observations

References

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

`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)

## Default S3 method:
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

Details

Note that calculate_variable_split function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

A named list with splits for selected variables

`ceteris_paribus` *Ceteris Paribus Profiles aka Individual Variable Profiles*

Description

This explainer works for individual observations. For each observation it calculates Ceteris Paribus Profiles for selected variables. Such profiles can be used to hypothesize about model results if selected variable is changed. For this reason it is also called 'What-If Profiles'.

Usage

```
ceteris_paribus(x, ...)

## S3 method for class 'explainer'
ceteris_paribus(x, new_observation, y = NULL,
  variables = NULL, variable_splits = NULL, grid_points = 101, ...)

## Default S3 method:
ceteris_paribus(x, data, predict_function = predict,
  new_observation, y = NULL, variables = NULL,
  variable_splits = NULL, grid_points = 101, label = class(x)[1],
  ...)
```

Arguments

`x`	an explainer created with the `DALEX::explain()` function, or a model to be explained.
`...`	other parameters
`new_observation`	a new observation with columns that corresponds to variables used in the model
`y`	true labels for `new_observation`. If specified then will be added to ceteris paribus plots. NOTE: It is best when target variable is not present in the `new_observation`
`variables`	names of variables for which profiles shall be calculated. Will be passed to `calculate_variable_split`. If NULL then all variables from the validation data will be used.
`variable_splits`	named list of splits for variables, in most cases created with `calculate_variable_split`. If NULL then it will be calculated based on validation data available in the `explainer`.

`grid_points`	number of points for profile. Will be passed to `calculate_variable_split`.
`data`	validation dataset. It will be extracted from `x` if it's an explainer NOTE: It is best when target variable is not present in the `data`
`predict_function`	predict function. 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

Details

Find more details in Ceteris Paribus Chapter.

Value

an object of the class `ceteris_paribus_explainer`.

References

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

Examples

```
library("DALEX")

titanic_small <- titanic_imputed[, 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[,-4],
                               y = titanic_small$survived == "yes",
                               verbose = FALSE)

cp_rf <- ceteris_paribus(explain_titanic_glm, titanic_small[1,])
cp_rf

plot(cp_rf, variables = "age")


library("randomForest")
model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                 fare + sibsp + parch,  data = titanic_imputed)


explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-9],
                              y = titanic_imputed$survived,
                              label = "Random Forest v7",
                                  verbose = FALSE,
                                  precalculate = FALSE)

# select few passangers
selected_passangers <- select_sample(titanic_imputed, n = 20)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red")
```

`ceteris_paribus_2d` *Ceteris Paribus 2D Plot*

Description

This function calculates ceteris paribus profiles for grid of values spanned by two variables. It may be useful to identify or present interactions between two variables.

Usage

```
ceteris_paribus_2d(explainer, observation, grid_points = 101,
  variables = NULL)
```

Arguments

`explainer`	a model to be explained, preprocessed by the `DALEX::explain()` function
`observation`	a new observation for which predictions need to be explained
`grid_points`	number of points used for response path. Will be used for both variables
`variables`	if specified, then only these variables will be explained

Details

Find more details in Ceteris Paribus 2D.

Value

an object of the class `ceteris_paribus_2d_explainer`.

Examples

```
library("DALEX")
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
cp_rf <- ceteris_paribus_2d(explain_titanic_glm, titanic[1,])
head(cp_rf)
plot(cp_rf)


library("randomForest")
set.seed(59)

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

explainer_rf <- explain(apartments_rf_model,
      data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)

new_apartment <- apartmentsTest[1, ]
new_apartment

wi_rf_2d <- ceteris_paribus_2d(explainer_rf, observation = new_apartment,
         variables = c("surface", "floor", "no.rooms"))
head(wi_rf_2d)
plot(wi_rf_2d)
```

`cluster_profiles` *Cluster Ceteris Paribus Profiles*

Description

This function calculates aggregates of ceteris paribus profiles based on hierarchical clustering.

Usage

```
cluster_profiles(x, ..., aggregate_function = mean,
  variable_type = "numerical", center = FALSE, k = 3,
  variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`
`...`	other explainers that shall be plotted together
`aggregate_function`	a function for profile aggregation. By default it's `mean`
`variable_type`	a character. If "numerical" then only numerical variables will be computed. If "categorical" then only categorical variables will be computed.
`center`	shall profiles be centered before clustering
`k`	number of clusters for the hclust function
`variables`	if not `NULL` then only `variables` will be presented

Details

Find more detailes in the Clustering Profiles Chapter.

Value

an object of the class `aggregated_profiles_explainer`

References

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

Examples

```
library("DALEX")
titanic <- na.omit(titanic)
selected_passangers <- select_sample(titanic, n = 100)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")
cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
clust_rf <- cluster_profiles(cp_rf, k = 3, variables = "age")
plot(clust_rf)


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

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

```
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)
clust_rf <- cluster_profiles(cp_rf, k = 3, variables = "age")
head(clust_rf)

plot(clust_rf, color = "_label_") +
  show_aggregated_profiles(pdp_rf, color = "black", size = 3)

plot(cp_rf, color = "grey", variables = "age") +
  show_aggregated_profiles(clust_rf, color = "_label_", size = 2)

clust_rf <- cluster_profiles(cp_rf, k = 3, center = TRUE, variables = "age")
head(clust_rf)
```

`conditional_dependency` *Conditional Dependency Profiles*

Description

Conditional Dependency Profiles (aka Local Profiles) average localy Ceteris Paribus Profiles. Function 'conditional_dependency' calls 'ceteris_paribus' and then 'aggregate_profiles'.

Usage

```
conditional_dependency(x, ...)

## S3 method for class 'explainer'
conditional_dependency(x, variables = NULL,
  N = 500, variable_splits = NULL, grid_points = 101, ...,
  variable_type = "numerical")

## Default S3 method:
conditional_dependency(x, data,
  predict_function = predict, label = class(x)[1], variables = NULL,
  N = 500, variable_splits = NULL, grid_points = 101, ...,
  variable_type = "numerical")

## S3 method for class 'ceteris_paribus_explainer'
conditional_dependency(x, ...,
  variables = NULL)

local_dependency(x, ...)
```

Arguments

`x`	an explainer created with function `DALEX::explain()`, an object of the class `ceteris_paribus_explainer` or a model to be explained.
`...`	other parameters
`variables`	names of variables for which profiles shall be calculated. Will be passed to `calculate_variable_split`. If `NULL` then all variables from the validation data will be used.
`N`	number of observations used for calculation of partial dependency profiles. By default 500.
`variable_splits`	named list of splits for variables, in most cases created with `calculate_variable_split`. If `NULL` then it will be calculated based on validation data avaliable in the `explainer`.
`grid_points`	number of points for profile. Will be passed to `calculate_variable_split`.
`variable_type`	a character. If "numerical" then only numerical variables will be calculated. If "categorical" then only categorical variables will be calculated.
`data`	validation dataset, will be extracted from `x` if it's an explainer NOTE: It is best when target variable is not present in the `data`
`predict_function`	predict function, 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

Details

Find more detailes in Local Dependency Profiles Chapter.

Value

an object of the class `aggregated_profile_explainer`

References

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

Examples

```
library("DALEX")

titanic_imputed$country <- NULL

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_imputed, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed$survived == "yes",
                              verbose = FALSE)

cdp_glm <- conditional_dependency(explain_titanic_glm,
                                 N = 50, variables = c("age", "fare"))
head(cdp_glm)
plot(cdp_glm)


library("randomForest")

model_titanic_rf <- randomForest(survived == "yes" ~.,  data = titanic_imputed)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed$survived == "yes",
                              verbose = FALSE)

cdp_rf <- conditional_dependency(explain_titanic_rf, N = 200, variable_type = "numerical")
plot(cdp_rf)

cdp_rf <- conditional_dependency(explain_titanic_rf, N = 200, variable_type = "categorical")
plotD3(cdp_rf, variable_type = "categorical", label_margin = 80, scale_plot = TRUE)
```

describe.partial_dependency_explainer

Natural language description of feature importance explainer

Description

Generic function `describe` generates a natural language description of `ceteris_paribus()`, `aggregated_profiles()` and `feature_importance()` explanations what enchaces their interpretability.

Usage

```
## S3 method for class 'partial_dependency_explainer'
describe(x,
  nonsignificance_treshold = 0.15, ..., display_values = FALSE,
  display_numbers = FALSE, variables = NULL, label = "prediction")

describe(x, ...)

## S3 method for class 'ceteris_paribus_explainer'
describe(x,
  nonsignificance_treshold = 0.15, ..., display_values = FALSE,
  display_numbers = FALSE, variables = NULL, label = "prediction")

## S3 method for class 'feature_importance_explainer'
```

```
describe(x,
  nonsignificance_treshold = 0.15, ...)
```

Arguments

`x`	a ceteris paribus explanation produced with function `ceteris_paribus()`
`nonsignificance_treshold`	a parameter specifying a treshold for variable importance
`...`	other arguments
`display_values`	allows for displaying variable values
`display_numbers`	allows for displaying numerical values
`variables`	a character of a single variable name to be described
`label`	label for model's prediction

Details

Function `describe.ceteris_paribus()` generates a natural language description of ceteris paribus profile. The description summarizes variable values, that would change model's prediction at most. If a ceteris paribus profile for multiple variables is passed, `variables` must specify a single variable to be described. Works only for a ceteris paribus profile for one observation. In current version only categorical values are discribed. For `display_numbers = TRUE` three most important variable values are displayed, while `display_numbers = FALSE` displays all the important variables, however without further details.

Function `describe.ceteris_paribus()` generates a natural language description of ceteris paribus profile. The description summarizes variable values, that would change model's prediction at most. If a ceteris paribus profile for multiple variables is passed, `variables` must specify a single variable to be described. Works only for a ceteris paribus profile for one observation. For `display_numbers = TRUE` three most important variable values are displayed, while `display_numbers = FALSE` displays all the important variables, however without further details.

Function `describe.feature_importance_explainer()` generates a natural language description of feature importance explanation. It prints the number of important variables, that have significant dropout difference from the full model, depending on `nonsignificance_treshold`. The description prints the three most important variables for the model's prediction. The current design of DALEX explainer does not allow for displaying variables values.

Examples

```
library("DALEX")
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 = "rf")

selected_passangers <- select_sample(titanic, n = 10)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
pdp <- aggregate_profiles(cp_rf, type = "partial", variable_type = "categorical")
describe(pdp, variables = "gender")

library("DALEX")
library("ingredients")
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 = "rf")

selected_passanger <- select_sample(titanic, n = 1, seed = 123)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passanger)

plot(cp_rf, variable_type = "categorical")
describe(cp_rf, variables = "class", label = "the predicted probability")

library("DALEX")
library("ingredients")

lm_model <- lm(m2.price~., data = apartments)
explainer_lm <- explain(lm_model, data = apartments[,2:6],
                        y = apartments$m2.price, label="lm")
fi_lm <- feature_importance(explainer_lm, loss_function = loss_root_mean_square)

plot(fi_lm)
describe(fi_lm)
```

feature_importance *Feature Importance*

Description

This function calculates permutation based feature importance. For this reason it is also called the Variable Dropout Plot.

Usage

```
feature_importance(x, ...)

## S3 method for class 'explainer'
```

```
feature_importance(x,
  loss_function = loss_root_mean_square, ..., type = c("raw", "ratio",
  "difference"), n_sample = NULL, B = 1,
  keep_raw_permutations = NULL, variables = NULL,
  variable_groups = NULL, label = NULL)

## Default S3 method:
feature_importance(x, data, y,
  predict_function = predict, loss_function = loss_root_mean_square,
  ..., label = class(x)[1], type = c("raw", "ratio", "difference"),
  n_sample = NULL, B = 1, keep_raw_permutations = NULL,
  variables = NULL, variable_groups = NULL)
```

Arguments

`x`	an explainer created with function `DALEX::explain()`, or a model to be explained.
`...`	other parameters
`loss_function`	a function thet will be used to assess variable importance
`type`	character, type of transformation that should be applied for dropout loss. "raw" results raw drop lossess, "ratio" returns `drop_loss/drop_loss_full_model` while "difference" returns `drop_loss -drop_loss_full_model`
`n_sample`	number of observations that should be sampled for calculation of variable importance. If `NULL` then variable importance will be calculated on whole dataset (no sampling).
`B`	integer, number of permutation rounds to perform on each variable
`keep_raw_permutations`	logical or `NULL`, determines if output retains information for individual permutations; default is to omit for `B=1` and keep otherwise
`variables`	vector of variables. If `NULL` then variable importance will be tested for each variable from the `data` separately. By default `NULL`
`variable_groups`	list of variables names vectors. This is for testing joint variable importance. If `NULL` then variable importance will be tested separately for `variables`. By default `NULL`. If specified then it will override `variables`
`label`	name of the model. By default it's extracted from the `class` attribute of the model
`data`	validation dataset, will be extracted from `x` if it's an explainer NOTE: It is best when target variable is not present in the `data`
`y`	true labels for `data`, will be extracted from `x` if it's an explainer
`predict_function`	predict function, will be extracted from `x` if it's an explainer

Details

Find more detailes in the Feature Importance Chapter.

Value

an object of the class `feature_importance`

References

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

Examples

```
library("DALEX")
titanic <- na.omit(titanic)
model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                       data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes")

vd_rf <- feature_importance(explain_titanic_glm)
plot(vd_rf)

vd_rf_joint1 <- feature_importance(explain_titanic_glm,
                   variable_groups = list("demographics" = c("gender", "age"),
                   "ticket_type" = c("fare")),
                   label = "lm 2 groups",
)

plot(vd_rf_joint1)

vd_rf_joint2 <- feature_importance(explain_titanic_glm,
                   variable_groups = list("demographics" = c("gender", "age"),
                   "wealth" = c("fare", "class"),
                   "family" = c("sibsp", "parch"),
                   "embarked" = "embarked"),
                   label = "lm 5 groups",
)

plot(vd_rf_joint2, vd_rf_joint1)

explain_titanic_glm


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")
```

```
vd_rf <- feature_importance(explain_titanic_rf)
plot(vd_rf)

vd_rf <- feature_importance(explain_titanic_rf, B = 5) # 5 replications
plot(vd_rf)

vd_rf_group <- feature_importance(explain_titanic_rf,
                   variable_groups = list("demographics" = c("gender", "age"),
                   "wealth" = c("fare", "class"),
                   "family" = c("sibsp", "parch"),
                   "embarked" = "embarked"),
                   label = "rf 4 groups",
)
plot(vd_rf_group, vd_rf)

HR_rf_model <- randomForest(status~., data = HR, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR, y = HR$status,
                         verbose = FALSE, precalculate = FALSE)

vd_rf <- feature_importance(explainer_rf, type = "raw",
                            loss_function = loss_cross_entropy)
head(vd_rf)
plot(vd_rf)

HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired")
vd_glm <- feature_importance(explainer_glm, type = "raw",
                        loss_function = loss_root_mean_square)
head(vd_glm)
plot(vd_glm)

library("xgboost")
model_martix_train <- model.matrix(status == "fired" ~ . -1, HR)
data_train <- xgb.DMatrix(model_martix_train, label = HR$status == "fired")
param <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
              objective = "binary:logistic", eval_metric = "auc")
HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)

explainer_xgb <- explain(HR_xgb_model, data = model_martix_train,
                     y = HR$status == "fired", label = "xgboost")

vd_xgb <- feature_importance(explainer_xgb, type = "raw")
head(vd_xgb)
plot(vd_xgb, vd_glm)
```

get_sample *Function for getting binary matrix*

Description

Function creates binary matrix, to be used in aspect_importance method. It starts with a zero matrix. Then it replaces some zeros with ones. It either randomly replaces one or two zeros per row. Or replace random number of zeros per row - average number of replaced zeros can be controled by parameter f. Function doesn't allow the returned matrix to have rows with only zeros.

Usage

```
get_sample(n, p, sample_method = c("default", "binom"), f = 2)
```

Arguments

`n`	number of rows
`p`	number of columns
`sample_method`	sampling method
`f`	frequency for binomial sampling

Value

a binary matrix

Examples

```
 ## Not run:
 get_sample(100,6,"binom",3)

## End(Not run)
```

`group_variables` *Groups numeric features into aspects*

Description

Divides correlated features into groups, called aspects. Division is based on correlation cutoff level.

Usage

```
group_variables(x, p = 0.5, clust_method = "complete",
  draw_tree = FALSE)
```

Arguments

`x`	dataframe with only numeric columns
`p`	correlation value for cut-off level
`clust_method`	the agglomeration method to be used, see `hclust` methods
`draw_tree`	if TRUE, function plots tree that illustrates grouping

Value

list of aspects

Examples

```
library("DALEX")
dragons_data <- dragons[,c(2,3,4,7,8)]
group_variables(dragons_data, p = 0.7, clust_method = "complete")
```

partial_dependency *Partial Dependency Profiles*

Description

Partial Dependency Profiles are averages from Ceteris Paribus Profiles. Function 'partial_dependency' calls 'ceteris_paribus' and then 'aggregate_profiles'.

Usage

```
partial_dependency(x, ...)

## S3 method for class 'explainer'
partial_dependency(x, variables = NULL, N = 500,
  variable_splits = NULL, grid_points = 101, ...,
  variable_type = "numerical")

## Default S3 method:
partial_dependency(x, data, predict_function = predict,
  label = class(x)[1], variables = NULL, grid_points = 101,
  variable_splits = NULL, N = 500, ..., variable_type = "numerical")

## S3 method for class 'ceteris_paribus_explainer'
partial_dependency(x, ...,
  variables = NULL)
```

Arguments

`x`	an explainer created with function `DALEX::explain()`, an object of the class `ceteris_paribus_explainer` or or a model to be explained.
`...`	other parameters
`variables`	names of variables for which profiles shall be calculated. Will be passed to `calculate_variable_split`. If `NULL` then all variables from the validation data will be used.
`N`	number of observations used for calculation of partial dependency profiles. By default 500.

variable_splits	named list of splits for variables, in most cases created with calculate_variable_split. If NULL then it will be calculated based on validation data avaliable in the explainer.
grid_points	number of points for profile. Will be passed to calculate_variable_split.
variable_type	a character. If "numerical" then only numerical variables will be calculated. If "categorical" then only categorical variables will be calculated.
data	validation dataset, will be extracted from x if it's an explainer NOTE: It is best when target variable is not present in the data
predict_function	predict function, 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

Details

Find more detailes in the Partial Dependence Profiles Chapter.

Value

an object of the class aggregated_profiles_explainer

References

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

Examples

```
library("DALEX")

titanic_imputed$country <- NULL

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic_imputed, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed$survived == "yes",
                               verbose = FALSE)

pdp_glm <- partial_dependency(explain_titanic_glm,
                             N = 50, variables = c("age", "fare"))
head(pdp_glm)
plot(pdp_glm)


library("randomForest")

model_titanic_rf <- randomForest(survived == "yes" ~.,  data = titanic_imputed)
```

```
explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed$survived == "yes",
                              verbose = FALSE)

pdp_rf <- partial_dependency(explain_titanic_rf, variable_type = "numerical")
plot(pdp_rf)

pdp_rf <- partial_dependency(explain_titanic_rf, variable_type = "categorical")
plotD3(pdp_rf, variable_type = "categorical", label_margin = 80, scale_plot = TRUE)
```

plot.aggregated_profiles_explainer
Plots Aggregated Profiles

Description

Function `plot.aggregated_profiles_explainer` plots partial dependency plot or accumulated effect plot. It works in a similar way to `plot.ceteris_paribus`, but instead of individual profiles show average profiles for each variable listed in the `variables` vector.

Usage

```
## S3 method for class 'aggregated_profiles_explainer'
plot(x, ..., size = 1,
  alpha = 1, color = "_label_", facet_ncol = NULL,
  variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `aggregate_profiles()`
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`facet_ncol`	number of columns for the `facet_wrap`
`variables`	if not `NULL` then only `variables` will be presented

Value

a `ggplot2` object

References

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

Examples

```
library("DALEX")

titanic <- na.omit(titanic)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes",
                               verbose = FALSE)

pdp_rf_p <- partial_dependency(explain_titanic_glm, N = 50)
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_l <- conditional_dependency(explain_titanic_glm, N = 50)
pdp_rf_l$`_label_` <- "RF_local"
pdp_rf_a<- accumulated_dependency(explain_titanic_glm, N = 50)
pdp_rf_a$`_label_` <- "RF_accumulated"
head(pdp_rf_p)
plot(pdp_rf_p, pdp_rf_l, pdp_rf_a, color = "_label_")


library("randomForest")

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",
                              verbose = FALSE)

selected_passangers <- select_sample(titanic, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf_p <- aggregate_profiles(cp_rf, variables = "age", type = "partial")
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_c <- aggregate_profiles(cp_rf, variables = "age", type = "conditional")
pdp_rf_c$`_label_` <- "RF_conditional"
pdp_rf_a <- aggregate_profiles(cp_rf, variables = "age", type = "accumulated")
pdp_rf_a$`_label_` <- "RF_accumulated"

head(pdp_rf_p)
plot(pdp_rf_p)
plot(pdp_rf_p, pdp_rf_c, pdp_rf_a)

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggregated_profiles(pdp_rf_p, size = 2)
```

plot.aspect_importance

Function for plotting aspect_importance results

Description

This function plots the results of aspect_importance.

Usage

```
## S3 method for class 'aspect_importance'
plot(x, bar_width = 10, ...)
```

Arguments

`x`	object of aspect_importance class
`bar_width`	bar width
`...`	other parameters

Value

a ggplot2 object

plot.ceteris_paribus_2d_explainer

Plot Ceteris Paribus 2D Explanations

Description

This function plots What-If Plots for a single prediction / observation.

Usage

```
## S3 method for class 'ceteris_paribus_2d_explainer'
plot(x, ..., facet_ncol = NULL,
  add_raster = TRUE, add_contour = TRUE, bins = 3,
  add_observation = TRUE, pch = "+", size = 6)
```

Arguments

`x`	a ceteris paribus explainer produced with the `ceteris_paribus_2d()` function
`...`	currently will be ignored
`facet_ncol`	number of columns for the `facet_wrap`
`add_raster`	if `TRUE` then `geom_raster` will be added to present levels with diverging colors
`add_contour`	if `TRUE` then `geom_contour` will be added to present contours
`bins`	number of contours to be added
`add_observation`	if `TRUE` then `geom_point` will be added to present observation that is explained
`pch`	character, symbol used to plot observations
`size`	numeric, size of individual datapoints

Value

a `ggplot2` object

References

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

Examples

```
library("DALEX")

library("randomForest")

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

explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,2:6],
                        y = apartments_test$m2.price,
                        verbose = FALSE)

new_apartment <- apartments_test[1, ]
new_apartment

wi_rf_2d <- ceteris_paribus_2d(explainer_rf, observation = new_apartment)
head(wi_rf_2d)

plot(wi_rf_2d)
plot(wi_rf_2d, add_contour = FALSE)
plot(wi_rf_2d, add_observation = FALSE)
plot(wi_rf_2d, add_raster = FALSE)

# HR data
model <- randomForest(status ~ gender + age + hours + evaluation + salary, data = HR)
```

```
pred1 <- function(m, x)   predict(m, x, type = "prob")[,1]

explainer_rf_fired <- explain(model,
                              data = HR[,1:5],
                              y = HR$status == "fired",
                              predict_function = pred1,
                              label = "fired")
new_emp <- HR[1, ]
new_emp

wi_rf_2d <- ceteris_paribus_2d(explainer_rf_fired, observation = new_emp)
head(wi_rf_2d)

plot(wi_rf_2d)
```

---

plot.ceteris_paribus_explainer

Plots Ceteris Paribus Profiles

Description

Function `plot.ceteris_paribus_explainer` plots Individual Variable Profiles for selected observations. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Find more detailes in Ceteris Paribus Chapter.

Usage

```
## S3 method for class 'ceteris_paribus_explainer'
plot(x, ..., size = 1, alpha = 1,
  color = "#46bac2", variable_type = "numerical", facet_ncol = NULL,
  variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`variable_type`	a character. If "numerical" then only numerical variables will be plotted. If "categorical" then only categorical variables will be plotted.
`facet_ncol`	number of columns for the `facet_wrap`
`variables`	if not `NULL` then only `variables` will be presented

Value

a ggplot2 object

References

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

Examples

```
library("DALEX")

titanic <- na.omit(titanic)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes",
                               verbose = FALSE)

cp_rf <- ceteris_paribus(explain_titanic_glm, titanic[1,])
cp_rf

plot(cp_rf, variables = "age")


library("randomForest")
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",
                              verbose = FALSE)

selected_passangers <- select_sample(titanic, n = 100)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red")

selected_passangers <- select_sample(titanic, n = 1)
selected_passangers

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
```

```
plot(cp_rf) +
  show_observations(cp_rf)

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age")

plot(cp_rf, variables = "class")
plot(cp_rf, variables = c("class", "embarked"), facet_ncol = 1)
plot(cp_rf, variables = c("class", "embarked", "gender", "sibsp"), variable_type = "categorical")
```

`plot.ceteris_paribus_oscillations`
Plot Ceteris Paribus Oscillations

Description

This function plots local variable importance plots calculated as oscillations in the Ceteris Paribus Profiles.

Usage

```
## S3 method for class 'ceteris_paribus_oscillations'
plot(x, ..., bar_width = 10)
```

Arguments

`x`	a ceteris paribus oscillation explainer produced with function `calculate_oscillations()`
`...`	other explainers that shall be plotted together
`bar_width`	width of bars. By default 10

Value

a ggplot2 object

References

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

Examples

```
library("DALEX")

library("randomForest")

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

```
explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,2:6],
                        y = apartments_test$m2.price)

apartment <- apartments_test[1:2,]

cp_rf <- ceteris_paribus(explainer_rf, apartment)
plot(cp_rf, color = "_ids_")

vips <- calculate_oscillations(cp_rf)
vips

plot(vips)
```

plot.feature_importance_explainer

Plots Feature Importance

Description

This function plots variable importance calculated as changes in the loss function after variable drops. It uses output from feature_importance function that corresponds to permutation based measure of variable importance. Variables are sorted in the same order in all panels. The order depends on the average drop out loss. In different panels variable contributions may not look like sorted if variable importance is different in different in different models.

Usage

```
## S3 method for class 'feature_importance_explainer'
plot(x, ..., max_vars = NULL,
  bar_width = 10)
```

Arguments

x	a feature importance explainer produced with the feature_importance() function
...	other explainers that shall be plotted together
max_vars	maximum number of variables that shall be presented for for each model. By default NULL what means all variables
bar_width	width of bars. By default 10

Details

Find more details in the Feature Importance Chapter.

Value

a ggplot2 object

References

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

Examples

```
library("DALEX")

titanic <- na.omit(titanic)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")

fi_rf <- feature_importance(explain_titanic_glm)
plot(fi_rf)


library("randomForest")

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")

fi_rf <- feature_importance(explain_titanic_rf)
plot(fi_rf)

HR_rf_model <- randomForest(status~., data = HR, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR, y = HR$status,
                         verbose = FALSE, precalculate = FALSE)

fi_rf <- feature_importance(explainer_rf, type = "raw",
                            loss_function = loss_cross_entropy)
head(fi_rf)
plot(fi_rf)

HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired")

fi_glm <- feature_importance(explainer_glm, type = "raw",
                             loss_function = loss_root_mean_square)
head(fi_glm)
plot(fi_glm)
```

```
library("xgboost")

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

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

HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)

explainer_xgb <- explain(HR_xgb_model, data = model_martix_train,
                         y = HR$status == "fired", label = "xgboost")

fi_xgb <- feature_importance(explainer_xgb, type = "raw")

head(fi_xgb)
plot(fi_glm, fi_xgb, bar_width = 5)
```

`plotD3` *Plots Ceteris Paribus Profiles in D3 with r2d3 Package.*

Description

Function `plotD3.ceteris_paribus_explainer` plots Individual Variable Profiles for selected observations. It uses output from `ceteris_paribus` function. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Find more detailes in Ceteris Paribus Chapter.

Usage

```
plotD3(x, ...)

## S3 method for class 'ceteris_paribus_explainer'
plotD3(x, ..., size = 2, alpha = 1,
  color = "#46bac2", variable_type = "numerical", facet_ncol = 2,
  scale_plot = FALSE, variables = NULL,
  chart_title = "Ceteris Paribus Profiles", label_margin = 60,
  show_observations = TRUE, show_rugs = TRUE)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`
`...`	other explainers that shall be plotted together
`size`	a numeric. Set width of lines
`alpha`	a numeric between 0 and 1. Opacity of lines

color	a character. Set line color
variable_type	a character. If "numerical" then only numerical variables will be plotted. If "categorical" then only categorical variables will be plotted.
facet_ncol	number of columns for the facet_wrap
scale_plot	a logical. If TRUE, the height of plot scales with window size. By default it's FALSE
variables	if not NULL then only variables will be presented
chart_title	a character. Set custom title
label_margin	a numeric. Set width of label margins in "categorical" type
show_observations	a logical. Adds observations layer to a plot. By default it's TRUE
show_rugs	a logical. Adds rugs layer to a plot. By default it's TRUE

Value

a r2d3 object.

Examples

```
library("DALEX")
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 = "rf")

selected_passangers <- select_sample(titanic, n = 10)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)

plotD3(cp_rf, variables = c("age","parch","fare","sibsp"),
     facet_ncol = 2, scale_plot = TRUE)

selected_passanger <- select_sample(titanic, n = 1)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passanger)

plotD3(cp_rf, variables = c("class", "embarked", "gender", "sibsp"),
    facet_ncol = 2, variable_type = "categorical", label_margin = 100, scale_plot = TRUE)
```

plotD3.aggregated_profiles_explainer

Plots Aggregated Ceteris Paribus Profiles in D3 with r2d3 Package.

Description

Function plotD3.aggregated_profiles_explainer plots an aggregate of ceteris paribus profiles. It works in a similar way to plotD3.ceteris_paribus_explainer but, instead of individual profiles, show average profiles for each variable listed in the variables vector.

Find more detailes in Ceteris Paribus Chapter.

Usage

```
## S3 method for class 'aggregated_profiles_explainer'
plotD3(x, ..., size = 2,
  alpha = 1, color = "#46bac2", variable_type = "numerical",
  facet_ncol = 2, scale_plot = FALSE, variables = NULL,
  chart_title = "Aggregated Profiles", label_margin = 60)
```

Arguments

x	a aggregated profiles explainer produced with function aggregate_profiles()
...	other explainers that shall be plotted together
size	a numeric. Set width of lines
alpha	a numeric between 0 and 1. Opacity of lines
color	a character. Set line/bar color
variable_type	a character. If "numerical" then only numerical variables will be plotted. If "categorical" then only categorical variables will be plotted.
facet_ncol	number of columns for the facet_wrap
scale_plot	a logical. If TRUE, the height of plot scales with window size. By default it's FALSE
variables	if not NULL then only variables will be presented
chart_title	a character. Set custom title
label_margin	a numeric. Set width of label margins in "categorical" type

Value

a r2d3 object.

References

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

Examples

```
library("DALEX")
library("randomForest")

titanic_small <- na.omit(titanic[1:500,-5])
model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + embarked + class +
                                   fare + sibsp + parch,  data = titanic_small)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_small[,-8],
                              y = titanic_small$survived == "yes",
                              label = "Random Forest v7")

selected_passangers <- select_sample(titanic_small, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)

pdp_rf_p <- aggregate_profiles(cp_rf, type = "partial", variable_type = "numerical")
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_c <- aggregate_profiles(cp_rf, type = "conditional", variable_type = "numerical")
pdp_rf_c$`_label_` <- "RF_conditional"
pdp_rf_a <- aggregate_profiles(cp_rf, type = "accumulated", variable_type = "numerical")
pdp_rf_a$`_label_` <- "RF_accumulated"

plotD3(pdp_rf_p, pdp_rf_c, pdp_rf_a, variable_type = "numerical", scale_plot = TRUE)

pdp <- aggregate_profiles(cp_rf, type = "partial", variable_type = "categorical")
pdp$`_label_` <- "RF_partial"

plotD3(pdp, variables = c("gender","class"), variable_type = "categorical", label_margin = 70)
```

`plotD3.feature_importance_explainer`
Plot Feature Importance Objects in D3 with r2d3 Package.

Description

Function `plotD3.feature_importance_explainer` plots dropouts for variables used in the model. It uses output from `feature_importance` function that corresponds to permutation based measure of feature importance. Variables are sorted in the same order in all panels. The order depends on the average drop out loss. In different panels variable contributions may not look like sorted if variable importance is different in different models.

Usage

```
## S3 method for class 'feature_importance_explainer'
plotD3(x, ..., max_vars = NULL,
  bar_width = 12, split = "model", scale_height = FALSE,
  margin = 0.15, chart_title = "Feature importance")
```

Arguments

`x`	a feature importance explainer produced with the `feature_importance()` function
`...`	other explainers that shall be plotted together
`max_vars`	maximum number of variables that shall be presented for for each model. By default `NULL` which means all variables
`bar_width`	width of bars in px. By default 12px
`split`	either "model" or "feature" determines the plot layout
`scale_height`	a logical. If `TRUE`, the height of plot scales with window size. By default it's `FALSE`
`margin`	extend x axis domain range to adjust the plot. Usually value between 0.1 and 0.3, by default it's 0.15
`chart_title`	a character. Set custom title

Value

a `r2d3` object.

Examples

```
library("DALEX")

lm_model <- lm(m2.price~., data = apartments)
explainer_lm <- explain(lm_model,
                        data = apartments[,2:6],
                        y = apartments$m2.price,
                        label = "lm", verbose = FALSE)

fi_lm <- feature_importance(explainer_lm, loss_function = loss_root_mean_square)

head(fi_lm)
plotD3(fi_lm)

## Not run:
library("randomForest")

rf_model <- randomForest(m2.price~., data = apartments)
explainer_rf <- explain(rf_model,
                        data = apartments[,2:6],
                        y = apartments$m2.price,
                        label = "rf", verbose = FALSE)

fi_rf <- feature_importance(explainer_rf, loss_function = loss_root_mean_square)

head(fi_rf)
plotD3(fi_lm, fi_rf)

plotD3(fi_lm, fi_rf, split = "feature")
```

```
plotD3(fi_lm, fi_rf, max_vars = 3, bar_width = 16, scale_height = TRUE)
plotD3(fi_lm, fi_rf, max_vars = 3, bar_width = 16, split = "feature", scale_height = TRUE)
plotD3(fi_lm, margin = 0.2)

## End(Not run)
```

print.aggregated_profiles_explainer
Prints Aggregated Profiles

Description

Prints Aggregated Profiles

Usage

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

Arguments

x	an individual variable profile explainer produced with the `aggregate_profiles()` function
...	other arguments that will be passed to `head()`

Examples

```
library("DALEX")
titanic <- na.omit(titanic)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")
selected_passangers <- select_sample(titanic, n = 100)
cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)

head(cp_rf)

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)


library("randomForest")

model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                    fare + sibsp + parch,  data = titanic)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic[,-9],
                              y = titanic$survived,
                              verbose = FALSE, precalculate = FALSE)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)
```

print.ceteris_paribus_explainer

Prints Individual Variable Explainer Summary

Description

Prints Individual Variable Explainer Summary

Usage

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

Arguments

x	an individual variable profile explainer produced with the `ceteris_paribus()` function
...	other arguments that will be passed to `head()`

Examples

```
library("DALEX")
library("randomForest")

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

explainer_rf <- explain(apartments_rf_model,
                        data = apartmentsTest[,2:6],
                        y = apartmentsTest$m2.price)

apartments_small <- select_sample(apartmentsTest, 10)

cp_rf <- ceteris_paribus(explainer_rf, apartments_small)
```

```
cp_rf
```

select_neighbours *Select Subset of Rows Closest to a Specified Observation*

Description

Function `select_neighbours` selects subset of rows from data set. This is useful if data is large and we need just a sample to calculate profiles.

Usage

```
select_neighbours(observation, data, variables = NULL,
  distance = gower::gower_dist, n = 20, frac = NULL)
```

Arguments

`observation`	single observation
`data`	set of observations
`variables`	names of variables that shall be used for calculation of distance. By default these are all variables present in `data` and `observation`
`distance`	the distance function, by default the `gower_dist()` function.
`n`	number of neighbours to select
`frac`	if `n` is not specified (NULL), then will be calculated as `frac` * number of rows in `data`. Either `n` or `frac` need to be specified.

Details

Note that `select_neighbours()` function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

a data frame with selected rows

Examples

```
library("DALEX")

new_apartment <- apartments[1,2:6]
small_apartments <- select_neighbours(new_apartment, apartments_test, n = 10)

new_apartment
small_apartments
```

`select_sample` *Select Subset of Rows*

Description

Function `select_sample` selects subset of rows from data set. This is useful if data is large and we need just a sample to calculate profiles.

Usage

```
select_sample(data, n = 100, seed = 1313)
```

Arguments

`data`	set of observations. Profile will be calculated for every observation (every row)
`n`	number of observations to select.
`seed`	seed for random number generator.

Details

Note that `select_subsample()` function is S3 generic. If you want to work on non standard data sources (like H2O ddf, external databases) you should overload it.

Value

a data frame with selected rows

Examples

```
library("DALEX")

small_apartments <- select_sample(apartments_test)
head(small_apartments)
```

`show_aggregated_profiles` *Adds a Layer with Aggregated Profiles*

Description

Function `show_aggregated_profiles` adds a layer to a plot created with `plot.ceteris_paribus_explainer`.

Usage

```
show_aggregated_profiles(x, ..., size = 0.5, alpha = 1,
  color = "#371ea3", variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`variables`	if not `NULL` then only `variables` will be presented

Value

a `ggplot2` layer

Examples

```
library("DALEX")

titanic <- na.omit(titanic)

selected_passangers <- select_sample(titanic, n = 100)

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic[,-9],
                               y = titanic$survived == "yes")

cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)

pdp_rf <- aggregate_profiles(cp_rf, type = "partial", variables = "age")

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_aggregated_profiles(pdp_rf, size = 3)


library("randomForest")

model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                 fare + sibsp + parch, data = titanic)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic[,-9],
                              y = titanic$survived,
                              verbose = FALSE, precalculate = FALSE)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf
```

```
pdp_rf <- aggregate_profiles(cp_rf, type = "partial", variables = "age")
head(pdp_rf)

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_aggregated_profiles(pdp_rf, size = 3)
```

`show_observations` *Adds a Layer with Observations to a Profile Plot*

Description

Function `show_observations` adds a layer to a plot created with `plot.ceteris_paribus_explainer` for selected observations. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Usage

```
show_observations(x, ..., size = 2, alpha = 1, color = "#371ea3",
  variable_type = "numerical", variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`variable_type`	a character. If "numerical" then only numerical variables will be plotted. If "categorical" then only categorical variables will be plotted.
`variables`	if not `NULL` then only `variables` will be presented

Value

a `ggplot2` layer

Examples

```
library("DALEX")
library("randomForest")

rf_model <- randomForest(survived == "yes" ~.,
                         data = titanic_imputed)

explainer_rf <- explain(rf_model, data = titanic_imputed,
                        y = titanic_imputed$survived == "yes",
                        label = "RF", verbose = FALSE)

selected_passangers <- select_sample(titanic_imputed, n = 100)
cp_rf <- ceteris_paribus(explainer_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "age", color = "grey") +
show_observations(cp_rf, variables = "age", color = "black") +
  show_rugs(cp_rf, variables = "age", color = "red")
```

`show_profiles` *Adds a Layer with Profiles*

Description

Function `show_profiles` adds a layer to a plot created with `plot.ceteris_paribus_explainer`.

Usage

```
show_profiles(x, ..., size = 0.5, alpha = 1, color = "#371ea3",
  variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`variables`	if not `NULL` then only `variables` will be presented

Value

a `ggplot2` layer

Examples

```
library("DALEX")

titanic <- na.omit(titanic)

selected_passangers <- select_sample(titanic, n = 100)
selected_john <- titanic[1,]

model_titanic_glm <- glm(survived == "yes" ~ gender + age + fare,
                         data = titanic, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                           data = titanic[,-9],
                           y = titanic$survived == "yes",
                           label = "cool_model", verbose = FALSE)

cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
cp_rf_john <- ceteris_paribus(explain_titanic_glm, selected_john)
plot(cp_rf, variables = "age") +
  show_profiles(cp_rf_john, variables = "age", size = 2)


library("randomForest")

model_titanic_rf <- randomForest(survived ~ gender + age + class + embarked +
                                 fare + sibsp + parch,  data = titanic)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic[,-9],
                              y = titanic$survived,
                              verbose = FALSE, precalculate = FALSE)

cp_rf <- ceteris_paribus(explain_titanic_rf, selected_passangers)
cp_rf_john <- ceteris_paribus(explain_titanic_rf, selected_john)

cp_rf

pdp_rf <- aggregate_profiles(cp_rf, variables = "age")
head(pdp_rf)

plot(cp_rf, variables = "age") +
  show_observations(cp_rf, variables = "age") +
  show_rugs(cp_rf, variables = "age", color = "red") +
  show_profiles(cp_rf_john, variables = "age", color = "red", size = 2)
```

---

show_residuals *Adds a Layer with Residuals to a Profile Plot*

Description

Function `show_residuals` adds a layer to a plot created with `plot.ceteris_paribus_explainer` for selected observations. Note that the `y` argument has to be specified in the `ceteris_paribus` function.

Usage

```
show_residuals(x, ..., size = 0.75, alpha = 1, color = c(`TRUE` =
  "#371ea3", `FALSE` = "#f05a71"), variables = NULL)
```

Arguments

`x`	a ceteris paribus explainer produced with function `ceteris_paribus()`. Note that `y` parameter shall be supplied in this function.
`...`	other explainers that shall be plotted together
`size`	a numeric. Size of lines to be plotted
`alpha`	a numeric between 0 and 1. Opacity of lines
`color`	a character. Either name of a color or name of a variable that should be used for coloring
`variables`	if not `NULL` then only `variables` will be presented

Value

a ggplot2 layer

Examples

```
library("DALEX")
library("randomForest")

titanic <- na.omit(titanic)
model_titanic_rf <- randomForest(survived == "yes" ~ gender + age + class + embarked +
                                   fare + sibsp + parch,  data = titanic, ntree = 500)

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

johny_d <- 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"))
)
```

```
johny_neighbours <- select_neighbours(data = titanic,
                                      observation = johny_d,
                                      variables = c("age", "gender", "class",
                                                    "fare", "sibsp", "parch"),
                                      n = 10)

cp_neighbours <- ceteris_paribus(explain_titanic_rf,
                                 johny_neighbours,
                                 y = johny_neighbours$survived == "yes",
                              variable_splits = list(age = seq(0,70, length.out = 1000)))

plot(cp_neighbours, variables = "age") +
  show_observations(cp_neighbours, variables = "age")


cp_johny <- ceteris_paribus(explain_titanic_rf, johny_d,
                            variable_splits = list(age = seq(0,70, length.out = 1000)))

plot(cp_johny, variables = "age", size = 1.5, color = "#8bdcbe") +
  show_profiles(cp_neighbours, variables = "age", color = "#ceced9") +
  show_observations(cp_johny, variables = "age", size = 5, color = "#371ea3") +
  show_residuals(cp_neighbours, variables = "age")
```

show_rugs — *Adds a Layer with Rugs to a Profile Plot*

Description

Function show_rugs adds a layer to a plot created with plot.ceteris_paribus_explainer for selected observations. Various parameters help to decide what should be plotted, profiles, aggregated profiles, points or rugs.

Usage

```
show_rugs(x, ..., size = 0.5, alpha = 1, color = "#371ea3",
  variable_type = "numerical", sides = "b", variables = NULL)
```

Arguments

x	a ceteris paribus explainer produced with function ceteris_paribus()
...	other explainers that shall be plotted together
size	a numeric. Size of lines to be plotted
alpha	a numeric between 0 and 1. Opacity of lines
color	a character. Either name of a color or name of a variable that should be used for coloring

`variable_type`	a character. If "numerical" then only numerical variables will be plotted. If "categorical" then only categorical variables will be plotted.
`sides`	a string containing any of "trbl", for top, right, bottom, and left. Passed to geom rug.
`variables`	if not NULL then only `variables` will be presented

Value

a `ggplot2` layer

Examples

```
library("DALEX")
library("randomForest")

rf_model <- randomForest(survived == "yes" ~.,
                         data = titanic_imputed)

explainer_rf <- explain(rf_model, data = titanic_imputed,
                        y = titanic_imputed$survived == "yes",
                        label = "RF", verbose = FALSE)

selected_passangers <- select_sample(titanic_imputed, n = 100)
cp_rf <- ceteris_paribus(explainer_rf, selected_passangers)
cp_rf

plot(cp_rf, variables = "age", color = "grey") +
show_observations(cp_rf, variables = "age", color = "black") +
  show_rugs(cp_rf, variables = "age", color = "red")
```

Index