Package ‘ingredients’

January 15, 2023

Title Effects and Importances of Model Ingredients

Version 2.3.0

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_dependence() for partial dependence plots,
conditional_dependence() for conditional dependence plots,
accumulated_dependence() 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 (>= 3.5)

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Imports ggplot2, scales, gridExtra, methods

Suggests DALEX, gower, ranger, testthat, r2d3, jsonlite, knitr,
rmarkdown, covr

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] (<`https://orcid.org/0000-0001-6661-5364`>),
Adam Izdebski [ctb]

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

Repository CRAN

Date/Publication 2023-01-15 11:40:02 UTC

R topics documented:

`accumulated_dependence` *Accumulated Local Effects Profiles aka ALEPlots*

Description

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

Usage

```
accumulated_dependence(x, ...)

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

## Default S3 method:
accumulated_dependence(
  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_dependence(x, ..., variables = NULL)

accumulated_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 dependence 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 Dependence 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`, Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. `https://ema.drwhy.ai/`

Examples

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

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

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

adp_glm <- accumulated_dependence(explain_titanic_glm,
                                  N = 25, variables = c("age", "fare"))
head(adp_glm)
plot(adp_glm)


library("ranger")

model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)
```

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

adp_rf <- accumulated_dependence(explain_titanic_rf, N = 200, variable_type = "categorical")
plotD3(adp_rf, 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 Dependence Profile (average across Ceteris Paribus Profiles), Conditional Dependence Profile (local weighted average across Ceteris Paribus Profiles) or Accumulated Local Dependence Profile (cummulated average local changes in Ceteris Paribus Profiles).

Usage

```
aggregate_profiles(
  x,
  ...,
  variable_type = "numerical",
  groups = NULL,
  type = "partial",
  variables = NULL,
  span = 0.25,
  center = FALSE
)
```

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 partial dependence, conditional profiles of accumulated local effects
variables	if not NULL then aggregate only for selected variables will be calculated
span	smoothing coefficient, by default 0.25. It's the sd for gaussian kernel
center	by default accumulated profiles start at 0. If center=TRUE, then they are centered around mean prediction, which is calculated on the observations used in ceteris_paribus.

Value

an object of the class `aggregated_profiles_explainer`

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
library("ranger")
head(titanic_imputed)

model_titanic_rf <- ranger(survived ~.,  data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

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

# continuous 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()
```

`bind_plots` *Bind Multiple ggplot Objects*

Description

This is an aesthetically efficient implementation of the `grid.arrange`

Usage

```
bind_plots(..., byrow = FALSE)
```

Arguments

`...`	(`ggplot`) ggplot objects to combine.
`byrow`	(`logical`) if `FALSE` (the default) the plots are bind by columns, otherwise the plots are bind by rows.

Value

(`gtable`) A plottable object with `plot()`.

Author(s)

`https://github.com/harell`

Examples

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

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

explain_glm <- explain(titanic_glm,
                       data = titanic_imputed,
                       y = titanic_imputed$survived,
                       verbose = FALSE)

pdp_numerical <- partial_dependence(explain_glm, N = 50, variable_type = "numerical")
pdp_categorical <- partial_dependence(explain_glm, N = 50, variable_type = "categorical")

# Bind plots by rows
bind_plots(plot(pdp_numerical), plot(pdp_categorical), byrow = TRUE)

# Bind plots by columns
bind_plots(plot(pdp_numerical), plot(pdp_categorical), byrow = FALSE)
```

`calculate_oscillations` *Calculate Oscillations for Ceteris Paribus Explainer*

Description

Oscillations are proxies for local feature importance at the instance level. Find more details 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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

titanic_small <- select_sample(titanic_imputed, n = 500, seed = 1313)

# build a model
model_titanic_glm <- glm(survived ~ gender + age + fare,
                         data = titanic_small, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_small[,-8],
                               y = titanic_small[,8])

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

calculate_oscillations(cp_rf)


library("ranger")

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

explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,-1],
                        y = apartments_test$m2.price,
                        label = "ranger forest",
                        verbose = FALSE)

apartment <- apartments_test[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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

`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,
  variable_splits_type = "quantiles",
  new_observation = NA
)

## Default S3 method:
calculate_variable_split(
  data,
  variables = colnames(data),
  grid_points = 101,
  variable_splits_type = "quantiles",
  new_observation = NA
)
```

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
`variable_splits_type`	how variable grids shall be calculated? Use "quantiles" (default) for percentiles or "uniform" to get uniform grid of points
`new_observation`	if specified (not `NA`) then all values in `new_observation` will be included in `variable_splits`

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,
  variable_splits_type = "quantiles",
  ...
)

## Default S3 method:
ceteris_paribus(
  x,
  data,
  predict_function = predict,
  new_observation,
  y = NULL,
  variables = NULL,
  variable_splits = NULL,
  grid_points = 101,
  variable_splits_type = "quantiles",
  variable_splits_with_obs = FALSE,
  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`	maximum number of points for profile calculations. Note that the finaln number of points may be lower than `grid_points`, eg. if there is not enough unique values for a given variable. Will be passed to `calculate_variable_split`.
`variable_splits_type`	how variable grids shall be calculated? Use "quantiles" (default) for percentiles or "uniform" to get uniform grid of points
`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
`variable_splits_with_obs`	if `TRUE` then all values in `new_observation` will be included in `variable_splits`
`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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. `https://ema.drwhy.ai/`

Examples

```
library("DALEX")
library("ingredients")
titanic_small <- select_sample(titanic_imputed, n = 500, seed = 1313)

# build a model
model_titanic_glm <- glm(survived ~ gender + age + fare,
                         data = titanic_small,
                         family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_small[,-8],
                               y = titanic_small[,8])

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

plot(cp_rf, variables = "age")


library("ranger")
model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)


explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = 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

Value

an object of the class `ceteris_paribus_2d_explainer`.

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

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


explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8])

cp_rf <- ceteris_paribus_2d(explain_titanic_glm, titanic_imputed[1,],
                       variables = c("age", "fare", "sibsp"))
head(cp_rf)

plot(cp_rf)

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

apartments_rf_model <- ranger(m2.price ~., data = apartments)

explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,-1],
                        y = apartments_test[,1],
                        label = "ranger forest",
                        verbose = FALSE)

new_apartment <- apartments_test[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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

selected_passangers <- select_sample(titanic_imputed, n = 100)
model_titanic_glm <- glm(survived ~ gender + age + fare,
                         data = titanic_imputed, family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8])

cp_rf <- ceteris_paribus(explain_titanic_glm, selected_passangers)
clust_rf <- cluster_profiles(cp_rf, k = 3, variables = "age")
plot(clust_rf)


library("ranger")
model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

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_dependence

Conditional Dependence Profiles

Description

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

Usage

```
conditional_dependence(x, ...)

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

## Default S3 method:
conditional_dependence(
  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_dependence(x, ..., variables = NULL)

local_dependency(x, ...)

conditional_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 dependence 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 details in the Accumulated Local Dependence Chapter.

Value

an object of the class `aggregated_profile_explainer`

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. `https://ema.drwhy.ai/`

Examples

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

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

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8],
                               verbose = FALSE)
```

```
cdp_glm <- conditional_dependence(explain_titanic_glm,
                                  N = 150, variables = c("age", "fare"))
head(cdp_glm)
plot(cdp_glm)


library("ranger")

model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

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

cdp_rf <- conditional_dependence(explain_titanic_rf, N = 200, variable_type = "categorical")
plotD3(cdp_rf, label_margin = 100, scale_plot = TRUE)
```

describe.partial_dependence_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_dependence_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.

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
library("ranger")


model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

selected_passangers <- select_sample(titanic_imputed, 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("ranger")


model_titanic_rf <- ranger(survived ~.,  data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

selected_passanger <- select_sample(titanic_imputed, 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[,-1], y = apartments[,1])

fi_lm <- feature_importance(explainer_lm, loss_function = DALEX::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 = DALEX::loss_root_mean_square,
  ...,
  type = c("raw", "ratio", "difference"),
  n_sample = NULL,
  B = 10,
  variables = NULL,
  variable_groups = NULL,
  N = n_sample,
  label = NULL
)

## Default S3 method:
feature_importance(
  x,
  data,
  y,
  predict_function = predict,
  loss_function = DALEX::loss_root_mean_square,
  ...,
  label = class(x)[1],
  type = c("raw", "ratio", "difference"),
  n_sample = NULL,
  B = 10,
  variables = NULL,
  N = n_sample,
  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 losses, "ratio" returns `drop_loss/drop_loss_full_model` while "difference" returns `drop_loss - drop_loss_full_model`
`n_sample`	alias for `N` held for backwards compatibility. number of observations that should be sampled for calculation of variable importance.
`B`	integer, number of permutation rounds to perform on each variable. By default it's `10`.
`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`
`N`	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).
`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 details in the Feature Importance Chapter.

Value

an object of the class `feature_importance`

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. `https://ema.drwhy.ai/`

Examples

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

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

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8])

fi_glm <- feature_importance(explain_titanic_glm, B = 1)
plot(fi_glm)



fi_glm_joint1 <- feature_importance(explain_titanic_glm,
                   variable_groups = list("demographics" = c("gender", "age"),
                   "ticket_type" = c("fare")),
                   label = "lm 2 groups")

plot(fi_glm_joint1)

fi_glm_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(fi_glm_joint2, fi_glm_joint1)

library("ranger")
model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

fi_rf <- feature_importance(explain_titanic_rf)
plot(fi_rf)

fi_rf <- feature_importance(explain_titanic_rf, B = 6) # 6 replications
plot(fi_rf)

fi_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(fi_rf_group, fi_rf)

HR_rf_model <- ranger(status ~., data = HR, probability = TRUE)

explainer_rf  <- explain(HR_rf_model, data = HR, y = HR$status,
                         model_info = list(type = 'multiclass'))

fi_rf <- feature_importance(explainer_rf, type = "raw",
                            loss_function = DALEX::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 = as.numeric(HR$status == "fired"))
fi_glm <- feature_importance(explainer_glm, type = "raw",
                             loss_function = DALEX::loss_root_mean_square)
head(fi_glm)
plot(fi_glm)
```

partial_dependence *Partial Dependence Profiles*

Description

Partial Dependence Profiles are averages from Ceteris Paribus Profiles. Function `partial_dependence` calls `ceteris_paribus` and then `aggregate_profiles`.

Usage

```
partial_dependence(x, ...)

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

## Default S3 method:
partial_dependence(
  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_dependence(x, ..., variables = NULL)

partial_dependency(x, ...)
```

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 dependence 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 details in the Partial Dependence Profiles Chapter.

Value

an object of the class `aggregated_profiles_explainer`

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")

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

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

pdp_glm <- partial_dependence(explain_titanic_glm,
                              N = 25, variables = c("age", "fare"))
head(pdp_glm)
plot(pdp_glm)


library("ranger")

model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

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

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

---

plot.aggregated_profiles_explainer
Plots Aggregated Profiles

Description

Function `plot.aggregated_profiles_explainer` plots partial dependence 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,
  facet_scales = "free_x",
  variables = NULL,
  title = NULL,
  subtitle = 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 hex code for a color, or `_label_` if models shall be colored, or `_ids_` if instances shall be colored
`facet_ncol`	number of columns for the `facet_wrap`
`facet_scales`	a character value for the `facet_wrap`. Default is `"free_x"`.
`variables`	if not `NULL` then only `variables` will be presented
`title`	a character. Partial and accumulated dependence explainers have deafult value.
`subtitle`	a character. If `NULL` value will be dependent on model usage.

Value

a ggplot2 object

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

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

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

pdp_rf_p <- partial_dependence(explain_titanic_glm, N = 50)
pdp_rf_p$`_label_` <- "RF_partial"
pdp_rf_l <- conditional_dependence(explain_titanic_glm, N = 50)
pdp_rf_l$`_label_` <- "RF_local"
pdp_rf_a<- accumulated_dependence(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("ranger")

model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

selected_passangers <- select_sample(titanic_imputed, 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.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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
library("ranger")
```

```
apartments_rf_model <- ranger(m2.price ~., data = apartments)

explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,-1],
                        y = apartments_test[,1],
                        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 <- ranger(status ~ gender + age + hours + evaluation + salary, data = HR,
                probability = TRUE)

pred1 <- function(m, x)   predict(m, x)$predictions[,1]

explainer_rf_fired <- explain(model,
                              data = HR[,1:5],
                              y = as.numeric(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 details 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,
  facet_scales = NULL,
  variables = NULL,
  title = "Ceteris Paribus profile",
  subtitle = NULL,
  categorical_type = "profiles"
)
```

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`
`facet_scales`	a character value for the `facet_wrap`. Default is `"free_x"`, but `"free_y"` if `categorical_type="bars"`.
`variables`	if not `NULL` then only `variables` will be presented
`title`	a character. Plot title. By default "Ceteris Paribus profile".
`subtitle`	a character. Plot subtitle. By default `NULL` - then subtitle is set to "created for the XXX, YYY model", where XXX, YYY are labels of given explainers.
`categorical_type`	a character. How categorical variables shall be plotted? Either `"profiles"` (default) or `"bars"` or `"lines"`.

Value

a `ggplot2` object

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. `https://ema.drwhy.ai/`

Examples

```
library("DALEX")

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

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

cp_glm <- ceteris_paribus(explain_titanic_glm, titanic_imputed[1,])
cp_glm

plot(cp_glm, variables = "age")


library("ranger")
model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

selected_passangers <- select_sample(titanic_imputed, 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_imputed, 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"), facet_ncol = 1, categorical_type = "bars")
plotD3(cp_rf, variables = c("class", "embarked", "gender"),
              variable_type = "categorical", scale_plot = TRUE,
              label_margin = 70)
```

`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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ranger")

apartments_rf_model <- ranger(m2.price ~., data = apartments)

explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,-1],
                        y = apartments_test[,1],
                        label = "ranger forest",
                        verbose = FALSE)

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,
  show_boxplots = TRUE,
  bar_width = 10,
  desc_sorting = TRUE,
  title = "Feature Importance",
  subtitle = NULL
)
```

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
`show_boxplots`	logical if `TRUE` (default) boxplot will be plotted to show permutation data.
`bar_width`	width of bars. By default `10`
`desc_sorting`	logical. Should the bars be sorted descending? By default TRUE
`title`	the plot's title, by default `'Feature Importance'`
`subtitle`	the plot's subtitle. By default - `NULL`, which means the subtitle will be 'created for the XXX model', where XXX is the label of explainer(s)

Details

Find more details in the Feature Importance Chapter.

Value

a `ggplot2` object

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

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

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8])

fi_rf <- feature_importance(explain_titanic_glm, B = 1)
plot(fi_rf)


library("ranger")
model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

fi_rf <- feature_importance(explain_titanic_rf)
plot(fi_rf)

HR_rf_model <- ranger(status~., data = HR, probability = TRUE)

explainer_rf  <- explain(HR_rf_model, data = HR, y = HR$status,
                         verbose = FALSE, precalculate = FALSE)

fi_rf <- feature_importance(explainer_rf, type = "raw", max_vars = 3,
                            loss_function = DALEX::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 = as.numeric(HR$status == "fired"))
```

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

---

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 details 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.

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
library("ranger")


model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

selected_passangers <- select_sample(titanic_imputed, 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_imputed, 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 details in Ceteris Paribus Chapter.

Usage

```
## S3 method for class 'aggregated_profiles_explainer'
plotD3(
  x,
  ...,
  size = 2,
  alpha = 1,
  color = "#46bac2",
  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
`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

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
library("ranger")

# smaller data, quicker example
titanic_small <- select_sample(titanic_imputed, n = 500, seed = 1313)


# build a model
model_titanic_rf <- ranger(survived ~., data = titanic_small, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_small[,-8],
                              y = titanic_small[,8],
                              label = "ranger forest",
                              verbose = FALSE)

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, scale_plot = TRUE)

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

plotD3(pdp, variables = c("gender","class"), 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,
  show_boxplots = TRUE,
  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
`show_boxplots`	logical if `TRUE` (default) boxplot will be plotted to show permutation data.
`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.

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

lm_model <- lm(m2.price ~., data = apartments)
explainer_lm <- explain(lm_model,
                        data = apartments[,-1],
                        y = apartments[,1],
                        verbose = FALSE)

fi_lm <- feature_importance(explainer_lm,
      loss_function = DALEX::loss_root_mean_square, B = 1)

head(fi_lm)
plotD3(fi_lm)


library("ranger")

rf_model <- ranger(m2.price~., data = apartments)

explainer_rf <- explain(rf_model,
                        data = apartments[,-1],
                        y = apartments[,1],
                        label = "ranger forest",
                        verbose = FALSE)

fi_rf <- feature_importance(explainer_rf, loss_function = DALEX::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)
```

`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")
library("ingredients")

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

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8])

selected_passangers <- select_sample(titanic_imputed, 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("ranger")

model_titanic_rf <- ranger(survived ~.,  data = titanic_imputed,
                           probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = 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("ingredients")
titanic_small <- select_sample(titanic_imputed, n = 500, seed = 1313)

# build a model
model_titanic_glm <- glm(survived ~ gender + age + fare,
                         data = titanic_small,
                         family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_small[,-8],
                               y = titanic_small[,8])

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


library("ranger")

apartments_rf_model <- ranger(m2.price ~., data = apartments)

explainer_rf <- explain(apartments_rf_model,
                        data = apartments_test[,-1],
                        y = apartments_test[,1],
                        label = "ranger forest",
                        verbose = FALSE)

apartments_small <- select_sample(apartments_test, 10)
```

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

---

print.feature_importance_explainer

Print Generic for Feature Importance Object

Description

Print Generic for Feature Importance Object

Usage

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

Arguments

`x`	an explanation created with `feature_importance`
`...`	other parameters.

Value

a data frame.

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

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

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

fi_glm <- feature_importance(explain_titanic_glm)

fi_glm
```

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 neighbors 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("ingredients")

new_apartment <- DALEX::apartments[1,]
small_apartments <- select_neighbours(new_apartment, DALEX::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("ingredients")

small_apartments <- select_sample(DALEX::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

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

selected_passangers <- select_sample(titanic_imputed, n = 100)

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

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8])

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("ranger")

model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = 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

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

library("ranger")

rf_model <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explainer_rf <- explain(rf_model,
                        data = titanic_imputed[,-8],
                        y = titanic_imputed[,8],
                        label = "ranger forest",
                        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

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

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

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

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

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_imputed[,-8],
                               y = titanic_imputed[,8],
                               label = "glm", 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("ranger")

model_titanic_rf <- ranger(survived ~.,  data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = 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` = "#8bdcbe", `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

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
library("ranger")

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"))
)


model_titanic_rf <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explain_titanic_rf <- explain(model_titanic_rf,
                              data = titanic_imputed[,-8],
                              y = titanic_imputed[,8],
                              label = "ranger forest",
                              verbose = FALSE)

johny_neighbours <- select_neighbours(data = titanic_imputed,
                                      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

References

Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models. https://ema.drwhy.ai/

Examples

```
library("DALEX")
library("ingredients")
titanic_small <- select_sample(titanic_imputed, n = 500, seed = 1313)

# build a model
model_titanic_glm <- glm(survived ~ gender + age + fare,
                         data = titanic_small,
                         family = "binomial")

explain_titanic_glm <- explain(model_titanic_glm,
                               data = titanic_small[,-8],
                               y = titanic_small[,8])

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


library("ranger")

rf_model <- ranger(survived ~., data = titanic_imputed, probability = TRUE)

explainer_rf <- explain(rf_model,
                        data = titanic_imputed[,-8],
                        y = titanic_imputed[,8],
                        label = "ranger forest",
                        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