Package ‘DALEX’

October 24, 2019

**Title**  Descriptive mAchine Learning EXplanations

**Version**  0.4.9

**Description**  
Machine Learning (ML) models are widely used and have various applications in classification or regression. Models created with boosting, bagging, stacking or similar techniques are often used due to their high performance, but such black-box models usually lack of interpretability. DALEX package contains various explainers that help to understand the link between input variables and model output.  
The single_variable() explainer extracts conditional response of a model as a function of a single selected variable.  
The single_prediction() explainer attributes parts of a model prediction to particular variables used in the model.  
It is a wrapper over 'breakDown' package (Staniak and Biecek 2018) <doi:10.32614/RJ-2018-072>.  
The variable_dropout() explainer calculates variable importance scores based on variable shuffling (Fisher at al. 2018) <arXiv:1801.01489>.  
All these explainers can be plotted with generic plot() function and compared across different models.  
‘DALEX’ is a part of the ‘DrWhy.AI’ universe (Biecek 2018) <arXiv:1806.08915>.

**Depends**  R (>= 3.5)

**License**  GPL

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  6.1.1

**Imports**  ggplot2

**Suggests**  ALEPlot, breakDown, factorMerger, gbm, ggpubr, iBreakDown, ingredients, pdp, randomForest, testthat

**URL**  https://ModelOriented.github.io/DALEX/, https://github.com/ModelOriented/DALEX
BugReports: https://github.com/ModelOriented/DALEX/issues

NeedsCompilation: no

Author: Przemyslaw Biecek [aut, cre] (<https://orcid.org/0000-0001-8423-1823>),
Szymon Maksymiuk [ctb]

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

Repository: CRAN

Date/Publication: 2019-10-24 19:20:06 UTC

R topics documented:

- apartments .................................................. 3
- colors_discrete_drwhy ...................................... 3
- dragons ...................................................... 4
- explain.default ............................................ 5
- feature_response .......................................... 7
- HR ............................................................. 9
- install_dependencies ....................................... 10
- loss_cross_entropy ....................................... 10
- model_info .................................................. 11
- model_performance ......................................... 13
- plot.feature_response_explainer .......................... 14
- plot.model_performance_explainer ........................ 15
- plot.prediction_breakdown_explainer ....................... 16
- plot.variable_importance_explainer ....................... 18
- plot.variable_response_explainer ........................ 19
- predict.explainer .......................................... 20
- prediction_breakdown ...................................... 21
- print.description .......................................... 23
- print.explainer ............................................. 23
- print.model_info ........................................... 24
- print.model_performance_explainer ....................... 24
- theme_drwhy ............................................... 25
- titanic ..................................................... 26
- update_data ............................................... 27
- update_label ............................................... 28
- variable_importance ....................................... 28
- variable_response ......................................... 30
- yhat ......................................................... 31

Index: 34
apartments

Description

Datasets `apartments` and `apartments_test` are artificial, generated from the same model. Structure of the dataset is copied from real dataset from PBImisc package, but they were generated in a way to mimic effect of Anscombe quartet for complex black box models.

Usage

data(apartments)

Format

a data frame with 1000 rows and 6 columns

Details

- `m2.price` - price per square meter
- `surface` - apartment area in square meters
- `n.rooms` - number of rooms (correlated with surface)
- `district` - district in which apartment is located, factor with 10 levels
- `floor` - floor
- `construction.date` - construction year

colors_discrete_drwhy    DrWhy color palettes for ggplot objects

Description

DrWhy color palettes for ggplot objects

Usage

colors_discrete_drwhy(n = 2)
colors_diverging_drwhy()
colors_breakdown_drwhy()
theme_drwhy_colors(n = 2)
theme_drwhy_colors_break_down()
Arguments

n  number of colors for color palette

Value

color palette as vector of characters

---

dragons  

*Dragon Data*

Description

Datasets dragons and dragons_test are artificial, generated form the same ground truth model, but with sometimes different data distribution.

Usage

data(dragons)

Format

a data frame with 2000 rows and 8 columns

Details

Values are generated in a way to: - have nonlinearity in year_of_birth and height - have concept drift in the test set

- year_of_birth - year in which the dragon was born. Negative year means year BC, eg: -1200 = 1201 BC
- year_of_discovery - year in which the dragon was found.
- height - height of the dragon in yards.
- weight - weight of the dragon in tons.
- scars - number of scars.
- colour - colour of the dragon.
- number_of_lost_teeth - number of teeth that the dragon lost.
- life_length - life length of the dragon.
**explain.default**  

Create Model Explainer

**Description**

Black-box models may have very different structures. This function creates a unified representation of a model, which can be further processed by various explainers.

**Usage**

```r
explain.default(model, data = NULL, y = NULL, predict_function = NULL, residual_function = NULL, weights = NULL, ..., label = NULL, verbose = TRUE, precalculate = TRUE, colorize = TRUE, model_info = NULL)
```

```r
explain(model, data = NULL, y = NULL, predict_function = NULL, residual_function = NULL, weights = NULL, ..., label = NULL, verbose = TRUE, precalculate = TRUE, colorize = TRUE, model_info = NULL)
```

**Arguments**

- **model** object - a model to be explained
- **data** data.frame or matrix - data that was used for fitting. If not provided then will be extracted from the model. Data should be passed without target column (this shall be provided as the y argument). NOTE: If target variable is present in the data, some of the functionalities may not work properly.
- **y** numeric vector with outputs / scores. If provided then it shall have the same size as data
- **predict_function** function that takes two arguments: model and new data and returns numeric vector with predictions
- **residual_function** function that takes three arguments: model, data and response vector y. It should return a numeric vector with model residuals for given data. If not provided, response residuals ($y - \hat{y}$) are calculated.
- **weights** numeric vector with sampling weights. By default it’s `NULL`. If provided then it shall have the same length as data
- **...** other parameters
- **label** character - the name of the model. By default it’s extracted from the ’class’ attribute of the model
- **verbose** if TRUE (default) then diagnostic messages will be printed
- **precalculate** if TRUE (default) then predicted_values and residual are calculated when explainer is created. This will happen also if verbose is TRUE. Set both verbose and precalculate to FALSE to omit calculations.
colorize  if TRUE (default) then WARNINGS, ERRORS and NOTES are colorized. Will work only in the R console.

model_info  a named list (package, version, type) containing information about model. If NULL, DALEX will seek for information on its own.

Details

Please NOTE, that the model is the only required argument. But some explainers may require that other arguments will be provided too.

Value

An object of the class explainer.

It’s a list with following fields:

• model the explained model.
• data the dataset used for training.
• y response for observations from data.
• weights sample weights for data. NULL if weights are not specified.
• y_hat calculated predictions.
• residuals calculated residuals.
• predict_function function that may be used for model predictions, shall return a single numerical value for each observation.
• residual_function function that returns residuals, shall return a single numerical value for each observation.
• class class/classes of a model.
• label label of explainer.
• model_info named list containing basic information about model, like package, version of package and type.

Examples

# simple explainer for regression problem
aps_lm_model4 <- lm(m2.price ~ ., data = apartments)
aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v")
aps_lm_explainer4

# various parameters for the explain function
# all defaults
aps_lm <- explain(aps_lm_model4)

# silent execution
aps_lm <- explain(aps_lm_model4, verbose = FALSE)

# user provided predict_function
aps_lm <- explain(aps_lm_model4, data = apartments, label = "model_4v", predict_function = predict)
# set target variable
aps_lm <- explain(aps_lm_model4, data = apartments, label = "model_4v", y = apartments$m2.price)
aps_lm <- explain(aps_lm_model4, data = apartments, label = "model_4v", y = apartments$m2.price,
predict_function = predict)

# set model_info
model_info <- list(package = "stats", ver = "3.6.1", type = "regression")
aps_lm_model4 <- lm(m2.price ~., data = apartments)
aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v",
                         model_info = model_info)

## Not run:
# set model_info
model_info <- list(package = "stats", ver = "3.6.1", type = "regression")
aps_lm_model4 <- lm(m2.price ~., data = apartments)
aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v",
                         model_info = model_info)

aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v",
                         weights = as.numeric(apartments$construction.year > 2000))

# more complex model
library("randomForest")
aps_rf_model4 <- randomForest(m2.price ~., data = apartments)
aps_rf_explainer4 <- explain(aps_rf_model4, data = apartments, label = "model_rf")
aps_rf_explainer4

## End(Not run)

---

**feature_response**

**Calculate Marginal Response for a Single Feature**

**Description**

Calculates the average model response as a function of a single selected variable. Use the `type` parameter to select the type of marginal response to be calculated. Currently for numeric variables we have Partial Dependency and Accumulated Local Effects implemented. Current implementation uses the `pdp` package (Brandon M. Greenwell (2017). pdp: An R Package for Constructing Partial Dependence Plots. The R Journal, 9(1), 421–436.) and `ALEPlot` (Dan Apley (2017). ALEPlot: Accumulated Local Effects Plots and Partial Dependence Plots.)

**Usage**

feature_response(x, ...)

## S3 method for class 'explainer'
feature_response(x, feature, type = "pdp",
                which_class = NULL, ...)
# Default S3 method:

```r
default::feature_response(x, data, predict_function, feature,
    type = "pdp", label = class(x)[1], which_class = NULL, ...)
```

## Arguments

- **x**: a model to be explained, or an explainer created with function `DALEX::explain()`.
- **feature**: character - name of a single variable
- **type**: character - type of the response to be calculated. Currently following options are implemented: 'pdp' for Partial Dependency and 'ale' for Accumulated Local Effects
- **which_class**: character, for multilabel classification you can restrict results to selected classes. By default 'NULL' which means that all classes are considered.
- **data**: validation dataset, will be extracted from 'x' if it's an explainer
- **predict_function**: predict function, will be extracted from 'x' if it's an explainer
- **label**: name of the model. By default it's extracted from the 'class' attribute of the model

## Details

This function is set deprecated. It is suggested to use `partial_dependency, accumulated_dependency` instead. Find information how to use these functions here: [https://pbiecek.github.io/PM_VEE/partialDependenceProfiles.html](https://pbiecek.github.io/PM_VEE/partialDependenceProfiles.html) and [https://pbiecek.github.io/PM_VEE/accumulatedLocalProfiles.html](https://pbiecek.github.io/PM_VEE/accumulatedLocalProfiles.html).

For factor variables we are using the 'factorMerger' package. Please note that the argument `type` must be set to 'factor' to use this method.

## Value

An object of the class `feature_response_explainer`. It's a data frame with calculated average response.

## References


## Examples

```r
HR_glm_model <- glm(status == "fired" ~ ., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
expl_glm <- feature_response(explainer_glm, "age", "pdp")
head(expl_glm)
plot(expl_glm)
```
## Not run:
library("randomForest")
HR_rf_model <- randomForest(status ~ ., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR)
expl_rf <- feature_response(explainer_rf, feature = "age", type = "pdp")
head(expl_rf)
plot(expl_rf)
expl_rf <- feature_response(explainer_rf, feature = "age", type = "pdp",
    which_class = 2)
plot(expl_rf)

## End(Not run)

### HR

#### Human Resources Data

**Description**

Datasets HR and HR_test are artificial, generated form the same model. Structure of the dataset is based on a real data, from Human Resources department with information which employees were promoted, which were fired.

**Usage**

data(HR)

**Format**

a data frame with 10000 rows and 6 columns

**Details**

Values are generated in a way to: - have interaction between age and gender for the 'fired' variable - have non monotonic relation for the salary variable - have linear effects for hours and evaluation.

- gender - gender of an employee.
- age - age of an employee in the moment of evaluation.
- hours - average number of working hours per week.
- evaluation - evaluation in the scale 2 (bad) - 5 (very good).
- salary - level of salary in the scale 0 (lowest) - 5 (highest).
- status - target variable, either 'fired' or 'promoted' or 'ok'.


install_dependencies  Install all dependencies for the DALEX package

Description
By default 'heavy' dependencies are not installed along DALEX. This function silently install all required packages.

Usage
install_dependencies(packages = c("ingredients", "iBreakDown", "pdp", "ALEPlot", "breakDown", "ggpubr", "factorMerger"))

Arguments
packages which packages shall be installed?

loss_cross_entropy  Calculate Loss Functions

Description
Calculate Loss Functions

Usage
loss_cross_entropy(observed, predicted, p_min = 1e-04, na.rm = TRUE)
loss_sum_of_squares(observed, predicted, na.rm = TRUE)
loss_root_mean_square(observed, predicted, na.rm = TRUE)
loss_accuracy(observed, predicted, na.rm = TRUE)
loss_one_minus_auc(observed, predicted)

Arguments
observed observed scores or labels, these are supplied as explainer specific y
predicted predicted scores, either vector of matrix, these are returned from the model specific predict_function()
p_min for cross entropy, minimal value for probability to make sure that log will not explode
na.rm logical, should missing values be removed?
model_info

Value

numeric - value of the loss function

Examples

```r
## Not run:
library("randomForest")
HR_rf_model <- randomForest(as.factor(status == "fired") ~ ., data = HR, ntree = 100)
loss_sum_of_squares(as.numeric(HR$status == "fired"), yhat(HR_rf_model))
## End(Not run)
```

Description

This generic function let user extract base information about model. The function returns a named list of class model_info that contain about package of model, version and task type. For wrappers like mlr or caret both, package and wrapper information are stored

Usage

```r
model_info(model, ...)
```

## S3 method for class 'lm'
```r
model_info(model, ...)
```

## S3 method for class 'randomForest'
```r
model_info(model, ...)
```

## S3 method for class 'svm'
```r
model_info(model, ...)
```

## S3 method for class 'glm'
```r
model_info(model, ...)
```

## S3 method for class 'glmnet'
```r
model_info(model, ...)
```

## S3 method for class 'cv.glmnet'
```r
model_info(model, ...)
```

## S3 method for class 'ranger'
```r
model_info(model, ...)
```

## S3 method for class 'gbm'
```r
```
model_info(model, ...)

## S3 method for class 'model_fit'
model_info(model, ...)

## S3 method for class 'train'
model_info(model, ...)

## Default S3 method:
model_info(model, ...)

Arguments

model - model object
...
- another arguments

Currently supported packages are:

- class 'cv.glmnet' and 'glmnet' - models created with 'glmnet' package
- class 'glm' - generalized linear models
- class 'model_fit' - models created with 'parsnip' package
- class 'lm' - linear models created with 'stats::lm'
- class 'ranger' - models created with 'ranger' package
- class 'randomForest' - random forest models created with 'randomForest' package
- class 'svm' - support vector machines models created with the 'e1071' package
- class 'train' - models created with 'caret' package
- class 'gbm' - models created with 'gbm' package

Value

A named list of class model_info

Examples

aps_lm_model4 <- lm(m2.price ~., data = apartments)
model_info(aps_lm_model4)

model_regr_rf <- randomForest::randomForest(m2.price ~., data = apartments, ntree = 50)
model_info(model_regr_rf)
Calculate Model Performance

Description
Prepare a data frame with model residuals.

Usage
model_performance(explainer, ...)

Arguments
explainer a model to be explained, preprocessed by the explain function
...
other parameters

Value
An object of the class model_performance_explainer.

References

Examples
## Not run:
library("randomForest")
HR_rf_model <- randomForest(as.factor(status == "fired")~-., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR, y = HR$status == "fired")
# resulting dataframe has predicted values and residuals
model_performance(explainer_rf)

HR_glm_model <- glm(status == "fired"~-., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired",
                         predict_function = function(m,x) predict.glm(m,x,type = "response"))
mp_ex_glm <- model_performance(explainer_glm)
mp_ex_glm
plot(mp_ex_glm)

HR_lm_model <- lm(status == "fired"~-., data = HR)
explainer_lm <- explain(HR_lm_model, data = HR, y = HR$status == "fired")
model_performance(explainer_lm)

## End(Not run)
plot.feature_response_explainer

Plot Marginal Model Explanations (Single Variable Responses)

Description

Function plot_variable_response_explainer plots marginal responses for one or more explainers.

Usage

## S3 method for class 'feature_response_explainer'
plot(x, ..., use_facets = FALSE)

Arguments

x  a single variable explainer produced with the single_feature function
...
other explainers that shall be plotted together
use_facets logical. If TRUE then separate models are on different facets

Value

a ggplot2 object

Examples

HR_glm_model <- glm(status == "fired" ~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
expl_glm <- feature_response(explainer_glm, "hours", "pdp")
head(expl_glm)
plot(expl_glm)

## Not run:
library("randomForest")
HR_rf_model <- randomForest(as.factor(status == "fired" )~., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR)
expl_rf <- feature_response(explainer_rf, feature = "hours", type = "pdp")
head(expl_rf)
plot(expl_rf)

plot(expl_rf, expl_glm)
plot(expl_rf, expl_glm, use_facets = TRUE)

## End(Not run)
**plot.model_performance_explainer**

*Plot Model Performance Explanations*

Description

Plot Model Performance Explanations

Usage

```r
## S3 method for class 'model_performance_explainer'
plot(x, ..., geom = "ecdf",
     show_outliers = 0, ptlabel = "name", lossFunction = function(x)
     sqrt(mean(x^2)))
```

Arguments

- **x**: a model to be explained, preprocessed by the `explain` function
- **...**: other parameters
- **geom**: either "ecdf" or "boxplot" determines how residuals shall be summarized
- **show_outliers**: number of largest residuals to be presented (only when geom = boxplot).
- **ptlabel**: either "name" or "index" determines the naming convention of the outliers
- **lossFunction**: function that calculates the loss for a model based on model residuals. By default it's the root mean square.

Value

An object of the class `model_performance_explainer`.

Examples

```r
## Not run:
library("randomForest")
HR_rf_model <- randomForest(as.factor(status == "fired")~., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR, y = HR$status == "fired")
mp_rf <- model_performance(explainer_rf)
plot(mp_rf)
plot(mp_rf, geom = "boxplot", show_outliers = 1)

HR_rf_model2 <- randomForest(as.factor(status == "fired")~age + hours, data = HR, ntree = 100)
explainer_rf2 <- explain(HR_rf_model2, data = HR, y = HR$status == "fired")
mp_rf2 <- model_performance(explainer_rf2)
plot(mp_rf, mp_rf2)

HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired", label = "glm",
  predict_function = function(m,x) predict.glm(m,x,type = "response"))
```
plot.prediction_breakdown_explainer

## End(Not run)

---

plot.prediction_breakdown_explainer

### Plot Break Down Explanations (Single Prediction)

#### Description

Function `plot.single_prediction_explainer` plots break down plots for a single prediction.

#### Usage

```r
## S3 method for class 'prediction_breakdown_explainer'
plot(x, ..., 
    add_contributions = TRUE, vcolors = c(`-1` = `#f05a71`, `0` = `#371ea3`, `1` = `#8bdcbe`, X = `#371ea3`), digits = 3, 
    rounding_function = round)
```

#### Arguments

- **x**: a single prediction explainer produced with the `single_prediction` function
- **...**: other explainers that shall be plotted together
- **add_contributions**: shall variable contributions to be added on plot?
- **vcolors**: named vector with colors
- **digits**: number of decimal places round or significant digits `signif` to be used. See the `rounding_function` argument
- **rounding_function**: function that is to used for rounding numbers. It may be `signif()` which keeps a specified number of significant digits. Or the default `round()` to have the same precision for all components

#### Value

a ggplot2 object
Examples

```r
## Not run:

new_dragon <- data.frame(year_of_birth = 200,
                          height = 80,
                          weight = 12.5,
                          scars = 0,
                          number_of_lost_teeth = 5)

dragon_lm_model4 <- lm(life_length ~ year_of_birth + height +
                        weight + scars + number_of_lost_teeth,
                        data = dragons)

dragon_lm_explainer4 <- explain(dragon_lm_model4, data = dragons, y = dragons$year_of_birth,
                                 label = "model_4v")

dragon_lm_predict4 <- prediction_breakdown(dragon_lm_explainer4, observation = new_dragon)

plot(dragon_lm_predict4)

library("randomForest")

dragon_rf_model4 <- randomForest(life_length ~ year_of_birth + height +
                                  weight + scars + number_of_lost_teeth,
                                  data = dragons)

dragon_rf_explainer4 <- explain(dragon_rf_model4, data = dragons, y = dragons$year_of_birth,
                                 label = "model_rf")

dragon_rf_predict4 <- prediction_breakdown(dragon_rf_explainer4, observation = new_dragon)

plot(dragon_rf_predict4)

# both models
plot(dragon_rf_predict4, dragon_lm_predict4)

library("gbm")

# create a gbm model

model <- gbm(life_length ~ year_of_birth + height + weight + scars + number_of_lost_teeth,
             data = dragons,
             distribution = "gaussian",
             n.trees = 1000,
             interaction.depth = 4,
             shrinkage = 0.01,
             n.minobsinnode = 10,
             verbose = FALSE)

# make an explainer for the model

explainer_gbm <- explain(model, data = dragons, predict_function =
                          function(model, x) predict(model, x, n.trees = 1000))

# create a new observation

exp_sgn <- prediction_breakdown(explainer_gbm, observation = new_dragon)

head(exp_sgn)

plot(exp_sgn)

plot(exp_sgn)

exp_sgn <- prediction_breakdown(explainer_gbm, observation = new_dragon, baseline = 0)

plot(exp_sgn)

## End(Not run)
```
Function `plot.variable_dropout_explainer` plots dropouts for variables used in the model. It uses output from `variable_dropout` 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 models.

### Usage

```r
## S3 method for class 'variable_importance_explainer'
plot(x, ..., max_vars = 10,
     bar_width = 10, show_baseline = FALSE, desc_sorting = TRUE)
```

### Arguments

- `x`: a variable dropout explainer produced with the `variable_dropout` function
- `...`: other explainers that shall be plotted together
- `max_vars`: maximum number of variables that shall be presented for for each model
- `bar_width`: width of bars. By default 10
- `show_baseline`: logical. Should the baseline be included?
- `desc_sorting`: logical. Should the bars be sorted descending? By default TRUE

### Value

a ggplot2 object

### Examples

```r
## Not run:
library("randomForest")
HR_rf_model <- randomForest(as.factor(status == "fired")~., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR, y = HR$status == "fired")
vdrf <- variable_importance(explainer_rf, type = "raw")
head(vdrf)
plot(vdrf)

HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired")
logit <- function(x) exp(x)/(1+exp(x))
vdlg <- variable_importance(explainer_glm, type = "raw",
                           loss_function = function(observed, predicted)
```
```r
library("xgboost")
model_matrix_train <- model.matrix(status == "fired"~.-1, HR)
data_train <- xgb.DMatrix(model_matrix_train, label = HR$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_matrix_train,
y = HR$status == "fired", label = "xgboost")
vd_xgb <- variable_importance(explainer_xgb, type = "raw")
head(vd_xgb)
plot(vd_xgb)
plot(vd_rf, vd_glm, vd_xgb, bar_width = 4)
```

**plot.variable_response_explainer**

*Plot Marginal Model Explanations (Single Variable Responses)*

**Description**

Function `plot.variable_response_explainer` plots marginal responses for one or more explainers.

**Usage**

```r
## S3 method for class 'variable_response_explainer'
plot(x, ..., use_facets = FALSE)
```

**Arguments**

- `x`  
  a single variable explainer produced with the `single_variable` function
- `...`  
  other explainers that shall be plotted together
- `use_facets`  
  logical. If TRUE then separate models are on different facets
predict.explainer  

**Value**

a ggplot2 object

**Examples**

```r
HR$evaluation <- factor(HR$evaluation)

HR_glm_model <- glm(status == "fired" ~ ., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
expl_glm <- variable_response(explainer_glm, "age", "pdp")
plot(expl_glm)

## Not run:
library("randomForest")
HR_rf_model <- randomForest(status == "fired" ~ ., data = HR)
explainer_rf <- explain(HR_rf_model, data = HR)
expl_rf <- variable_response(explainer_rf, variable = "age", type = "pdp")
plot(expl_rf)
plot(expl_rf, expl_glm)

# Example for factor variable (with factorMerger)
expl_rf <- variable_response(explainer_rf, variable = "evaluation", type = "factor")
plot(expl_rf)

expl_glm <- variable_response(explainer_glm, variable = "evaluation", type = "factor")
plot(expl_glm)

# both models
plot(expl_rf, expl_glm)

## End(Not run)
```

**predict.explainer**  

*Calculate Predictions for Explainer*

**Description**

This is a generic `predict()` function works for explainer objects.

**Usage**

```r
## S3 method for class 'explainer'
predict(object, newdata, ...)
```
prediction_breakdown

Arguments

- object: a model to be explained, object of the class explainer
- newdata: data.frame or matrix - observations for prediction
- ...: other parameters that will be passed to the predict function

Value

An numeric matrix of predictions

Examples

```r
HR_glm_model <- glm(status == "fired"~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
predict(explainer_glm, HR[1:3,])

## Not run:
library("randomForest")
HR_rf_model <- randomForest(status == "fired" ~., data = HR)
explainer_rf <- explain(HR_rf_model, data = HR)
predict(explainer_rf, HR[1:3,])

## End(Not run)
```

---

prediction_breakdown  Calculate Break Down Explanations

Description

This function is set deprecated. It is suggested to use break_down instead. Find information how to use these functions here: https://pbiecek.github.io/PM_VEE/breakDown.html.

Usage

```r
prediction_breakdown(explainer, observation, ...)
```

Arguments

- explainer: a model to be explained, preprocessed by the `explain` function
- observation: a new observation for which predictions need to be explained
- ...: other parameters that will be passed to breakDown::broken.default()

Value

An object of the class `single_prediction_explainer`. It’s a data frame with calculated average response.
References


Examples

```r
new_dragon <- data.frame(year_of_birth = 200,
                        height = 80,
                        weight = 12.5,
                        scars = 0,
                        number_of_lost_teeth = 5)

dragon_lm_model4 <- lm(life_length ~ year_of_birth + height +
                        weight + scars + number_of_lost_teeth,
                        data = dragons)
dragon_lm_explainer4 <- explain(dragon_lm_model4, data = dragons, y = dragons$year_of_birth,
                        label = "model_4v")
dragon_lm_predict4 <- prediction_breakdown(dragon_lm_explainer4, observation = new_dragon)
head(dragon_lm_predict4)
plot(dragon_lm_predict4)

## Not run:
library("randomForest")
dragon_rf_model4 <- randomForest(life_length ~ year_of_birth + height +
                        weight + scars + number_of_lost_teeth,
                        data = dragons)
dragon_rf_explainer4 <- explain(dragon_rf_model4, data = dragons, y = dragons$year_of_birth,
                        label = "model_rf")
dragon_rf_predict4 <- prediction_breakdown(dragon_rf_explainer4, observation = new_dragon)
head(dragon_rf_predict4)
plot(dragon_rf_predict4)

library("gbm")
# create a gbm model
model <- gbm(life_length ~ year_of_birth + height + weight + scars +
             number_of_lost_teeth, data = dragons,
             distribution = "gaussian",
             n.trees = 1000,
             interaction.depth = 4,
             shrinkage = 0.01,
             n.minobsinnode = 10,
             verbose = FALSE)
# make an explainer for the model
explainer_gbm <- explain(model, data = dragons, predict_function =
                         function(model, x) predict(model, x, n.trees = 1000))
# create a new observation
exp_sgn <- prediction_breakdown(explainer_gbm, observation = new_dragon)
head(exp_sgn)
plot(exp_sgn)

exp_sgn <- prediction_breakdown(explainer_gbm, observation = new_dragon, baseline = 0)
plot(exp_sgn)
```
print.description  
*Print Natural Language Descriptions*

## Description

Generic function

## Usage

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

## Arguments

- `x`: an individual explainer produced with the `describe()` function
- `...`: other arguments

---

print.explainer  
*Print Explainer Summary*

## Description

Print Explainer Summary

## Usage

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

## Arguments

- `x`: a model explainer created with the `explain` function
- `...`: other parameters
Examples

```r
aps_lm_model4 <- lm(m2.price~., data = apartments)
aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, y = apartments$m2.price,
                             label = "model_4v")
aps_lm_explainer4
```

```r
## Not run:
library("randomForest")
HR_rf_model4 <- randomForest(as.factor(status == "fired")~., data = HR, ntree = 100)
HR_rf_explainer4 <- explain(HR_rf_model4, data = HR, label = "model_rf")
HR_rf_explainer4
## End(Not run)
```

---

**print.model_info**

*Print model_info*

**Description**

Function prints object of class `model_info` created with `model_info`

**Usage**

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

**Arguments**

- `x` - an object of class `model_info`
- `...` - other parameters

---

**print.model_performance_explainer**

*Print Model Performance Summary*

**Description**

Print Model Performance Summary

**Usage**

```r
## S3 method for class 'model_performance_explainer'
print(x, ...)
```
Arguments

x  a model to be explained, object of the class 'model_performance_explainer'
...
other parameters

Examples

```r
## Not run:
library("breakDown")
library("randomForest")
HR_rf_model <- randomForest(status == "fired" ~ ., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR, y = HR$status == "fired")
mp_ex_rf <- model_performance(explainer_rf)
mp_ex_rf
plot(mp_ex_rf)
## End(Not run)
```

**theme_drwhy**  
*DrWhy Theme for ggplot objects*

Description

DrWhy Theme for ggplot objects

MI² Theme

Usage

theme_drwhy()

theme_drwhy_vertical()

theme_mi2()

Value

theme for ggplot2 objects
Passengers and Crew on the RMS Titanic Data

Description

The titanic data is a complete list of passengers and crew members on the RMS Titanic. It includes a variable indicating whether a person did survive the sinking of the RMS Titanic on April 15, 1912.

Usage

data(titanic)
data(titanic_imputed)

Format

a data frame with 2207 rows and 9 columns

Details

This dataset was copied from the stablelearner package and went through few variable transformations. Levels in embarked was replaced with full names, sibsp, parch and fare were converted to numerical variables and values for crew were replaced with 0. If you use this dataset please cite the original package.

From stablelearner: The website https://www.encyclopedia-titanica.org offers detailed information about passengers and crew members on the RMS Titanic. According to the website 1317 passengers and 890 crew member were abord. 8 musicians and 9 employees of the shipyard company are listed as passengers, but travelled with a free ticket, which is why they have NA values in fare. In addition to that, fare is truely missing for a few regular passengers.

• gender a factor with levels male and female.
• age a numeric value with the persons age on the day of the sinking.
• class a factor specifying the class for passengers or the type of service aboard for crew members.
• embarked a factor with the persons place of of embarkment (Belfast/Cherbourg/Queenstown/Southampton).
• country a factor with the persons home country.
• fare a numeric value with the ticket price (0 for crew members, musicians and employees of the shipyard company).
• sibsp an ordered factor specifying the number if siblings/spouses aboard; adopted from Vanderbild data set (see below).
• parch an ordered factor specifying the number of parents/children aboard; adopted from Vanderbild data set (see below).
• survived a factor with two levels (no and yes) specifying whether the person has survived the sinking.
NOTE: The titanic_imputed dataset use following imputation rules.

- Missing 'age' is replaced with the mean of the observed ones, i.e., 30.
- For sibsp and parch, missing values are replaced by the most frequently observed value, i.e., 0.
- For fare, mean fare for a given class is used, i.e., 0 pounds for crew, 89 pounds for the 1st, 22 pounds for the 2nd, and 13 pounds for the 3rd class.

Source

This dataset was copied from the stablelearner package and went through few variable transformations. The complete list of persons on the RMS titanic was downloaded from https://www.encyclopedia-titanica.org on April 5, 2016. The information given in sibsp and parch was adopted from a data set obtained from http://biostat.mc.vanderbilt.edu/DataSets.

References


---

### update_data

**Update data of an explainer object**

**Description**

Function allows users to update data an y of any explainer in a unified way. It doesn’t require knowledge about structre of an explainer.

**Usage**

```r
update_data(explainer, data, y = NULL, verbose = TRUE)
```

**Arguments**

- `explainer` - explainer object that is supposed to be updated.
- `data` - new data, is going to be passed to an explainer
- `y` - new y, is going to be passed to an explainer
- `verbose` - logical, indicates if information about update should be printed

**Value**

updated explainer object

**Examples**

```r
aps_lm_model4 <- lm(m2.price ~., data = apartments)
aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v")
explainer <- update_data(aps_lm_explainer4, data = apartmentsTest, y = apartmentsTest$m2.price)
```
update_label  
Update label of explainer object

Description
Function allows users to update label of any explainer in a unified way. It doesn’t require knowledge about structure of an explainer.

Usage
update_label(explainer, label, verbose = TRUE)

Arguments
- explainer: explainer object that is supposed to be updated.
- label: new label, is going to be passed to an explainer
- verbose: logical, indicates if information about update should be printed

Value
updated explainer object

Examples
aps_lm_model4 <- lm(m2.price ~ ., data = apartments)
aps_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v")
explainer <- update_label(aps_lm_explainer4, label = "lm")

variable_importance  
Calculate Feature Importance Explanations as Loss from Feature Dropout

Description
This function is set deprecated. It is suggested to use feature_importance instead. Find information how to use these functions here: https://pbiecek.github.io/PM_VEE/featureImportance.html.

Usage
variable_importance(explainer, loss_function = loss_sum_of_squares, ..., type = "raw", n_sample = 1000)
variable_importance

Arguments

explainer  a model to be explained, preprocessed by the 'explain' function
loss_function a function that will be used to assess variable importance
... other parameters
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 negative then variable importance will be calculated on whole dataset (no sampling).

Value

An object of the class 'variable_leverage_explainer'. It's a data frame with calculated average response.

References


Examples

```r
## Not run:
library(DALEX)
library("randomForest")
HR_rf_model <- randomForest(as.factor(status == "fired")~., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR, y = HR$status == "fired")
vd_rf <- variable_importance(explainer_rf, type = "raw")

HR_glm_model <- glm(as.factor(status == "fired")~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR, y = HR$status == "fired")
logit <- function(x) exp(x)/(1+exp(x))
vd_glm <- variable_importance(explainer_glm, type = "raw",
loss_function = function(observed, predicted)
sum((observed - logit(predicted))^2))

library("xgboost")
model_matrix_train <- model.matrix(status == "fired" ~ .-1, HR)
data_train <- xgb.DMatrix(model_matrix_train, label = HR$status == "fired")
param <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
objective = "binary:logistic", eval_metric = "auc")
HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)
explainer_xgb <- explain(HR_xgb_model, data = model_matrix_train,
y = HR$status == "fired", label = "xgboost")
vd_xgb <- variable_importance(explainer_xgb, type = "raw")
```

variable_response  Calculate Marginal Response Explanations for a Single Variable

Description

Calculates the average model response as a function of a single selected variable. Use the 'type' parameter to select the type of marginal response to be calculated. Currently for numeric variables we have Partial Dependency and Accumulated Local Effects implemented. Current implementation uses the 'pdp' package (Brandon M. Greenwell (2017). pdp: An R Package for Constructing Partial Dependence Plots. The R Journal, 9(1), 421–436.) and 'ALEPlot' (Dan Apley (2017). ALEPlot: Accumulated Local Effects Plots and Partial Dependence Plots.)

Usage

variable_response(explainer, variable, type = "pdp", trans = I, ...)

Arguments

explainer  a model to be explained, preprocessed by the 'explain' function
variable  character - name of a single variable
type  character - type of the response to be calculated. Currently following options are implemented: 'pdp' for Partial Dependency and 'ale' for Accumulated Local Effects
trans  function - a transformation/link function that shall be applied to raw model predictions. This will be inherited from the explainer.
...  other parameters

Details

This function is set deprecated. It is suggested to use partial_dependency, accumulated_dependency instead. Find information how to use these functions here: https://pbiecek.github.io/PM_VEE/partialDependenceProfiles.html and https://pbiecek.github.io/PM_VEE/accumulatedLocalProfiles.html.

For factor variables we are using the 'factorMerger' package. Please note that the argument type must be set to 'factor' to use this method.

Value

An object of the class 'variable_response_explainer'. It’s a data frame with calculated average response.
References


Examples

```r
HR$evaluation <- factor(HR$evaluation)
HR_glm_model <- glm(status == "fired" ~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
expl_glm <- variable_response(explainer_glm, "age", "pdp")
plot(expl_glm)
```

```r
## Not run:
library("randomForest")
HR_rf_model <- randomForest(status == "fired" ~., data = HR)
explainer_rf <- explain(HR_rf_model, data = HR)
expl_rf <- variable_response(explainer_rf, variable = "age",
                           type = "pdp")
plot(expl_rf)
plot(expl_rf, expl_glm)

# Example for factor variable (with factorMerger)
expl_rf <- variable_response(explainer_rf, variable = "evaluation",
                           type = "factor")
plot(expl_rf)

expl_glm <- variable_response(explainer_glm, variable = "evaluation",
                           type = "factor")
plot(expl_glm)

# both models
plot(expl_rf, expl_glm)
## End(Not run)
```

---

**yhat**

Wrap Various Predict Functions

**Description**

This function is a wrapper over various predict functions for different models and different model structures. The wrapper returns a single numeric score for each new observation. To do this it uses different extraction techniques for models from different classes, like for classification random forest is forces the output to be probabilities not classes itself.

**Usage**

`yhat(X.model, newdata, ...)`
## S3 method for class 'lm'
yhat(X.model, newdata, ...)

## S3 method for class 'randomForest'
yhat(X.model, newdata, ...)

## S3 method for class 'svm'
yhat(X.model, newdata, ...)

## S3 method for class 'gbm'
yhat(X.model, newdata, ...)

## S3 method for class 'glm'
yhat(X.model, newdata, ...)

## S3 method for class 'cv.glmnet'
yhat(X.model, newdata, ...)

## S3 method for class 'glmnet'
yhat(X.model, newdata, ...)

## S3 method for class 'ranger'
yhat(X.model, newdata, ...)

## S3 method for class 'model_fit'
yhat(X.model, newdata, ...)

## S3 method for class 'train'
yhat(X.model, newdata, ...)

## Default S3 method:
yhat(X.model, newdata, ...)

### Arguments

- **X.model** object - a model to be explained
- **newdata** data.frame or matrix - observations for prediction
- **...** other parameters that will be passed to the predict function

### Details

Currently supported packages are:

- class 'cv.glmnet' and 'glmnet' - models created with 'glmnet' package
- class 'glm' - generalized linear models
- class 'model_fit' - models created with 'parsnip' package
- class 'lm' - linear models created with 'stats::lm'
- class 'ranger' - models created with 'ranger' package
• class 'randomForest' - random forest models created with 'randomForest' package
• class 'svm' - support vector machines models created with the 'e1071' package
• class 'train' - models created with 'caret' package
• class 'gbm' - models created with 'gbm' package

**Value**

An numeric matrix of predictions
Index

*Topic **HR**
  HR, 9
*Topic **apartments**
  apartments, 3
*Topic **dragons**
  dragons, 4
*Topic **titanic**
  titanic, 26

accumulated_dependency, 8, 30
apartments, 3
apartments_test (apartments), 3
apartmentsTest (apartments), 3
break_down, 21

colors_breakdown_drwhy
  (colors_discrete_drwhy), 3
colors_discrete_drwhy, 3
colors_diverging_drwhy
  (colors_discrete_drwhy), 3
dragons, 4
dragons_test (dragons), 4
explain, 13, 15
explain (explain.default), 5
explain.default, 5
feature_importance, 28
feature_response, 7
HR, 9
HR_test (HR), 9
HRTest (HR), 9
install_dependencies, 10

loss_accuracy (loss_cross_entropy), 10
loss_cross_entropy, 10
loss_one_minus_auc
  (loss_cross_entropy), 10

loss_root_mean_square
  (loss_cross_entropy), 10
loss_sum_of_squares
  (loss_cross_entropy), 10
model_info, 11, 24
model_performance, 13

partial_dependency, 8, 30
plot.feature_response_explainer, 14
plot.model_performance_explainer, 15
plot.prediction_breakdown_explainer,
  16
plot.variable_importance_breakdown_explainer, 18
plot.variable_response_explainer, 14, 19, 19
predict.explainer, 20
prediction_breakdown, 21
print.description, 23
print.explainer, 23
print.model_info, 24
print.model_performance_explainer, 24

single_prediction, 16
single_prediction
  (prediction_breakdown), 21
single_variable, 19
single_variable (variable_response), 30
theme_drwhy, 25
theme_drwhy_colors
  (colors_discrete_drwhy), 3
theme_drwhy_colors_breakdown
  (colors_discrete_drwhy), 3
theme_drwhy_vertical (theme_drwhy), 25
theme_mi2 (theme_drwhy), 25
titanic, 26
titanic_imputed (titanic), 26

update_data, 27
update_label, 28
INDEX

variable_dropout, 18
variable_dropout (variable_importance), 28
variable_importance, 28
variable_response, 30
yhat, 31