

# Package ‘DALEX’

March 25, 2019

**Title** Descriptive mACHINE Learning EXplanations

**Version** 0.3.0

## 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.

It is a wrapper over packages 'pdp' (Greenwell 2017) <doi:10.32614/RJ-2017-016>, 'ALEPlot' (Apley 2018) <arXiv:1612.08468> and 'factorMerger' (Sitko and Biecek 2017) <arXiv:1709.04412>.

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 et 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 (>= 3.0)

**License** GPL

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Imports** ggplot2

**Suggests** gbm, randomForest, xgboost, ALEPlot, breakDown, pdp, factorMerger, ggpubr, testthat, dplyr

**URL** <https://pbiecek.github.io/DALEX/>

**BugReports** <https://github.com/pbiecek/DALEX/issues>

**NeedsCompilation** no

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

**Maintainer** Przemyslaw Biecek <[przemyslaw.biecek@gmail.com](mailto:przemyslaw.biecek@gmail.com)>

**Repository** CRAN

**Date/Publication** 2019-03-25 22:46:17 UTC

## R topics documented:

apartments . . . . .	2
dragons . . . . .	3
explain.default . . . . .	4
HR . . . . .	5
install_dependencies . . . . .	6
loss_cross_entropy . . . . .	6
model_feature_response . . . . .	7
model_performance . . . . .	9
plot.model_feature_response_explainer . . . . .	10
plot.model_performance_explainer . . . . .	11
plot.prediction_breakdown_explainer . . . . .	12
plot.variable_importance_explainer . . . . .	13
plot.variable_response_explainer . . . . .	15
predict.explainer . . . . .	16
prediction_breakdown . . . . .	17
print.explainer . . . . .	18
print.model_performance_explainer . . . . .	19
theme_drwhy . . . . .	20
theme_mi2 . . . . .	21
titanic . . . . .	21
variable_importance . . . . .	22
variable_response . . . . .	24
yhat . . . . .	25
<b>Index</b>	<b>27</b>

---

apartments	<i>Apartments Data</i>
------------	------------------------

---

### 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

---

dragons

*Dragon Data*

---

**Description**

Datasets dragons and dragons\_test are artificial, generated from 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

```
explain.default(model, data = NULL, y = NULL,
  predict_function = yhat, link = I, ..., label = tail(class(model),
  1))
```

```
explain(model, data = NULL, y = NULL, predict_function = yhat,
  link = I, ..., label = tail(class(model), 1))
```

### 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
y	numeric vector with outputs / scores. Currently used only by variable_dropout() explainer.
predict_function	function that takes two arguments: model and new data and returns numeric vector with predictions
link	function - a transformation/link function that shall be applied to raw model predictions
...	other parameters
label	character - the name of the model. By default it's extracted from the 'class' attribute of the model

### Details

Please NOTE, that the model is actually the only required argument. But some explainers may require that others 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

- `predict_function` function that may be used for model predictions, shall return a single numerical value for each observation.
- `class` class/classes of a model
- `label` label, by default it's the last value from the `class` vector, but may be set to any character.

### Examples

```
library("breakDown")

wine_lm_model4 <- lm(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)
wine_lm_explainer4 <- explain(wine_lm_model4, data = wine, label = "model_4v")
wine_lm_explainer4

## Not run:
library("randomForest")
wine_rf_model4 <- randomForest(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)
wine_rf_explainer4 <- explain(wine_rf_model4, data = wine, label = "model_rf")
wine_rf_explainer4

## End(Not run)
```

---

HR

*Human Resources Data*

---

### Description

Datasets HR and HR\_test are artificial, generated from 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("pdp", "ALEPlot", "breakDown",
  "ggpubr", "factorMerger"))
```

### Arguments

packages            which packages shall be installed?

---

loss\_cross\_entropy *Preimplemented Loss Functions*

---

### Description

Preimplemented Loss Functions

### Usage

```
loss_cross_entropy(observed, predicted, p_min = 1e-04, na.rm = TRUE)
```

### 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?

### Value

numeric - value of the loss function

**Examples**

```
## Not run:
library("randomForest")
HR_rf_model <- randomForest(status~., data = HR, ntree = 100)
loss_cross_entropy(HR$status, yhat(HR_rf_model))

## End(Not run)
```

---

model\_feature\_response

*Marginal Response 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**

```
model_feature_response(x, ...)

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

## Default S3 method:
model_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()'
...	other parameters
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

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 'model\_feature\_response\_explainer'. It's a data frame with calculated average response.

## References

Predictive Models: Visual Exploration, Explanation and Debugging [https://pbiecek.github.io/PM\\_VEE/](https://pbiecek.github.io/PM_VEE/)

## Examples

```
library("DALEX")

HR_glm_model <- glm(status == "fired" ~ ., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
expl_glm <- model_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 <- model_feature_response(explainer_rf, feature = "age", type = "pdp")
head(expl_rf)
plot(expl_rf)

expl_rf <- model_feature_response(explainer_rf, feature = "age", type = "pdp",
                                which_class = 2)
plot(expl_rf)

## End(Not run)
```



---

model_performance	<i>Model Performance Plots</i>
-------------------	--------------------------------

---

**Description**

Model Performance Plots

**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**

Predictive Models: Visual Exploration, Explanation and Debugging [https://pbiecek.github.io/PM\\_VEE/](https://pbiecek.github.io/PM_VEE/)

**Examples**

```
## Not run:
library("randomForest")
HR_rf_model <- randomForest(status == "fired"~., data = HR, ntree = 100)
explainer_rf <- explain(HR_rf_model, data = HR, y = HR$status == "fired")
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.model_feature_response_explainer
Plots 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 'model_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

```
library("DALEX")

HR_glm_model <- glm(status == "fired" ~., data = HR, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR)
expl_glm <- model_feature_response(explainer_glm, "hours", "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 <- model_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
  Model Performance Plots
```

---

## Description

Model Performance Plots

## Usage

```
## 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

```
## Not run:
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_rf <- model_performance(explainer_rf)
plot(mp_rf)
plot(mp_rf, geom = "boxplot", show_outliers = 1)

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"))
mp_glm <- model_performance(explainer_glm)
plot(mp_glm)

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

```

mp_lm <- model_performance(explainer_lm)
plot(mp_lm)

plot(mp_rf, mp_glm, mp_lm)
plot(mp_rf, mp_glm, mp_lm, geom = "boxplot")
plot(mp_rf, mp_glm, mp_lm, geom = "boxplot", show_outliers = 1)

## End(Not run)

```

---

```

plot.prediction_breakdown_explainer
Plots Local Explanations (Single Prediction)

```

---

## Description

Function 'plot.single\_prediction\_explainer' plots break down plots for a single prediction.

## Usage

```

## 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**

```

## Not run:
library("breakDown")
new.wine <- data.frame(citric.acid = 0.35,
  sulphates = 0.6,
  alcohol = 12.5,
  pH = 3.36,
  residual.sugar = 4.8)

wine_lm_model4 <- lm(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)
wine_lm_explainer4 <- explain(wine_lm_model4, data = wine, label = "model_4v")
wine_lm_predict4 <- prediction_breakdown(wine_lm_explainer4, observation = new.wine)
plot(wine_lm_predict4)

library("randomForest")
wine_rf_model4 <- randomForest(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)
wine_rf_explainer4 <- explain(wine_rf_model4, data = wine, label = "model_rf")
wine_rf_predict4 <- prediction_breakdown(wine_rf_explainer4, observation = new.wine)
plot(wine_rf_predict4)

# both models
plot(wine_rf_predict4, wine_lm_predict4)

library("gbm")
# create a gbm model
model <- gbm(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine,
  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 = wine, predict_function =
  function(model, x) predict(model, x, n.trees = 1000))
# create a new observation
exp_sgn <- prediction_breakdown(explainer_gbm, observation = new.wine)
head(exp_sgn)
plot(exp_sgn)
plot(wine_rf_predict4, wine_lm_predict4, exp_sgn)

## End(Not run)

```

**Description**

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 in different models.

**Usage**

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

**Arguments**

<code>x</code>	a variable dropout explainer produced with the 'variable_dropout' function
<code>...</code>	other explainers that shall be plotted together
<code>max_vars</code>	maximum number of variables that shall be presented for for each model
<code>bar_width</code>	width of bars. By default 10
<code>show_baseline</code>	logical. Should the baseline be included?

**Value**

a ggplot2 object

**Examples**

```
## 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")
vd_rf <- variable_importance(explainer_rf, type = "raw")
head(vd_rf)
plot(vd_rf)

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

head(vd_glm)
plot(vd_glm)

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

```

param <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
             objective = "binary:logistic", eval_metric = "auc")
HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)
explainer_xgb <- explain(HR_xgb_model, data = model_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)

# NOTE:
# if you like to have all importances hooked to 0, you can do this as well
vd_rf <- variable_importance(explainer_rf, type = "difference")
vd_glm <- variable_importance(explainer_glm, type = "difference",
                             loss_function = function(observed, predicted)
                             sum((observed - logit(predicted))^2))
vd_xgb <- variable_importance(explainer_xgb, type = "difference")
plot(vd_rf, vd_glm, vd_xgb, bar_width = 4)

## End(Not run)

```

---

plot.variable\_response\_explainer

*Plots 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 '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

## Value

a ggplot2 object

**Examples**

```

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      *Wrapper over the predict function*

---

**Description**

This function works for explain objects. It calls embedded predict function.

**Usage**

```

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

```

**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**

```
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 *Explanations for a Single Prediction*

---

**Description**

Explanations for a Single Prediction

**Usage**

```
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 <code>breakDown::broken.default()</code>

**Value**

An object of the class 'single\_prediction\_explainer'. It's a data frame with calculated average response.

**References**

Predictive Models: Visual Exploration, Explanation and Debugging [https://pbierek.github.io/PM\\_VEE/](https://pbierek.github.io/PM_VEE/)

**Examples**

```

library("breakDown")
new.wine <- data.frame(citric.acid = 0.35,
  sulphates = 0.6,
  alcohol = 12.5,
  pH = 3.36,
  residual.sugar = 4.8)

wine_lm_model4 <- lm(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)
wine_lm_explainer4 <- explain(wine_lm_model4, data = wine, label = "model_4v")
wine_lm_predict4 <- prediction_breakdown(wine_lm_explainer4, observation = new.wine)
head(wine_lm_predict4)
plot(wine_lm_predict4)

## Not run:
library("randomForest")
wine_rf_model4 <- randomForest(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)
wine_rf_explainer4 <- explain(wine_rf_model4, data = wine, label = "model_rf")
wine_rf_predict4 <- prediction_breakdown(wine_rf_explainer4, observation = new.wine)
head(wine_rf_predict4)
plot(wine_rf_predict4)

library("gbm")
# create a gbm model
model <- gbm(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine,
  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 = wine, predict_function =
  function(model, x) predict(model, x, n.trees = 1000))
# create a new observation
exp_sgn <- prediction_breakdown(explainer_gbm, observation = new.wine)
head(exp_sgn)
plot(exp_sgn)

exp_sgn <- prediction_breakdown(explainer_gbm, observation = new.wine, baseline = 0)
plot(exp_sgn)

## End(Not run)

```

---

print.explainer

*Prints Explainer Summary*


---

**Description**

Prints Explainer Summary

**Usage**

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

**Arguments**

x                    a model explainer created with the 'explain' function  
...                   other parameters

**Examples**

```
library("breakDown")  
  
wine_lm_model4 <- lm(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)  
wine_lm_explainer4 <- explain(wine_lm_model4, data = wine, label = "model_4v")  
wine_lm_explainer4  
  
## Not run:  
library("randomForest")  
wine_rf_model4 <- randomForest(quality ~ pH + residual.sugar + sulphates + alcohol, data = wine)  
wine_rf_explainer4 <- explain(wine_rf_model4, data = wine, label = "model_rf")  
wine_rf_explainer4  
  
## End(Not run)
```

---

```
print.model_performance_explainer  
                                  Model Performance Summary
```

---

**Description**

Model Performance Summary

**Usage**

```
## 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

```
## 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

## Usage

```
theme_drwhy()

theme_drwhy_vertical()

theme_drwhy_colors(n = 2)

theme_drwhy_colors_break_down()
```

## Arguments

n                    number of colors for color palette

## Value

theme for ggplot2 objects

---

`theme_mi2`*MI<sup>2</sup> Theme*

---

**Description**MI<sup>2</sup> Theme**Usage**`theme_mi2()`**Value**

theme object that can be added to ggplot2 plots

---

`titanic`*Passengers and Crew on the RMS Titanic*

---

**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)`**Format**

a data frame with 2207 rows and 11 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 aboard. 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 truly 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 Vanderbilt data set (see below).
- parch an ordered factor specifying the number of parents/children aboard; adopted from Vanderbilt data set (see below).
- survived a factor with two levels (no and yes) specifying whether the person has survived the sinking.

### 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

<https://www.encyclopedia-titanica.org>, <http://biostat.mc.vanderbilt.edu/DataSets> and <https://CRAN.R-project.org/package=stablelearner>

---

variable_importance	<i>Loss from Variable Dropout</i>
---------------------	-----------------------------------

---

### Description

Loss from Variable Dropout

### Usage

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

### Arguments

<code>explainer</code>	a model to be explained, preprocessed by the 'explain' function
<code>loss_function</code>	a function that will be used to assess variable importance
<code>...</code>	other parameters

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	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

Predictive Models: Visual Exploration, Explanation and Debugging [https://pbiecek.github.io/PM\\_VEE/](https://pbiecek.github.io/PM_VEE/)

## Examples

```
## 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")
vd_rf <- variable_importance(explainer_rf, type = "raw")
vd_rf

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))
vd_glm <- variable_importance(explainer_glm, type = "raw",
                             loss_function = function(observed, predicted)
                             sum((observed - logit(predicted))^2))
vd_glm

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

## End(Not run)
```

---

variable\_response      *Marginal Response 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 = explainer$link, ...)
```

### 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

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 'svariable\_response\_explainer'. It's a data frame with calculated average response.

### References

Predictive Models: Visual Exploration, Explanation and Debugging [https://pbiemek.github.io/PM\\_VEE/](https://pbiemek.github.io/PM_VEE/)



## Examples

```

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)

```

---

yhat

*Wrapper over the predict function*


---

## Description

This function is just a wrapper over the predict function. It sets different default parameters 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 'glm'  
yhat(X.model, newdata, ...)  
  
## S3 method for class 'ranger'  
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

**Value**

An numeric matrix of predictions

# Index

- \*Topic **HR**
  - HR, [5](#)
- \*Topic **apartments**
  - apartments, [2](#)
- \*Topic **dragons**
  - dragons, [3](#)
- \*Topic **titanic**
  - titanic, [21](#)
  
- apartments, [2](#)
- apartments\_test (apartments), [2](#)
- apartmentsTest (apartments), [2](#)
  
- dragons, [3](#)
- dragons\_test (dragons), [3](#)
  
- explain (explain.default), [4](#)
- explain.default, [4](#)
  
- HR, [5](#)
- HR\_test (HR), [5](#)
- HRTest (HR), [5](#)
  
- install\_dependencies, [6](#)
  
- loss\_accuracy (loss\_cross\_entropy), [6](#)
- loss\_cross\_entropy, [6](#)
- loss\_root\_mean\_square
  - (loss\_cross\_entropy), [6](#)
- loss\_sum\_of\_squares
  - (loss\_cross\_entropy), [6](#)
  
- model\_feature\_response, [7](#)
- model\_performance, [9](#)
  
- plot.model\_feature\_response\_explainer,
  - [10](#)
- plot.model\_performance\_explainer, [11](#)
- plot.prediction\_breakdown\_explainer,
  - [12](#)
- plot.variable\_importance\_explainer, [13](#)
  
- plot.variable\_response\_explainer, [15](#)
- predict.explainer, [16](#)
- prediction\_breakdown, [17](#)
- print.explainer, [18](#)
- print.model\_performance\_explainer, [19](#)
  
- single\_prediction
  - (prediction\_breakdown), [17](#)
- single\_variable (variable\_response), [24](#)
  
- theme\_drwhy, [20](#)
- theme\_drwhy\_colors (theme\_drwhy), [20](#)
- theme\_drwhy\_colors\_break\_down
  - (theme\_drwhy), [20](#)
- theme\_drwhy\_vertical (theme\_drwhy), [20](#)
- theme\_mi2, [21](#)
- titanic, [21](#)
  
- variable\_dropout (variable\_importance),
  - [22](#)
- variable\_importance, [22](#)
- variable\_response, [24](#)
  
- yhat, [25](#)