Package 'DALEX'

January 7, 2019

Title Descriptive mAchine Learning EXplanations

Version 0.2.6

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' and 'ALEPlot'.
The single_prediction() explainer attributes parts of a model prediction to particular variables used in the model.
It is a wrapper over 'breakDown' package.
The variable_dropout() explainer calculates variable importance scores based on variable shuffling.
All these explainers can be plotted with generic plot() function and compared across different models.

Depends R (>= 3.0)

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports pdp, ggplot2, ALEPlot, breakDown, factorMerger, ggpubr

Suggests gbm, randomForest, xgboost, 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>

Repository CRAN

Date/Publication 2019-01-07 19:50:12 UTC

R topics documented:

`apartments` *Apartments Data*

Description

Datasets `apartments` and `apartmentsTest` are artificial, generated form 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

`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 HRTest 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 - gender 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‘.

`model_performance` *Model Performance Plots*

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'.

Examples

```
 ## Not run:
library("breakDown")
library("randomForest")
HR_rf_model <- randomForest(left~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data, y = HR_data$left)
model_performance(explainer_rf)

HR_glm_model <- glm(left~., data = breakDown::HR_data, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR_data, y = HR_data$left,
                    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(left~., data = breakDown::HR_data)
explainer_lm <- explain(HR_lm_model, data = HR_data, y = HR_data$left)
model_performance(explainer_lm)

## 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("breakDown")
library("randomForest")
HR_rf_model <- randomForest(left~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data, y = HR_data$left)
mp_rf <- model_performance(explainer_rf)
plot(mp_rf)
plot(mp_rf, geom = "boxplot", show_outliers = 1)

HR_glm_model <- glm(left~., data = breakDown::HR_data, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR_data, y = HR_data$left, 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(left~., data = breakDown::HR_data)
```

```
explainer_lm <- explain(HR_lm_model, data = HR_data, y = HR_data$left)
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` = "#d8b365", `0` =
  "#f5f5f5", `1` = "#5ab4ac", X = "darkgrey"), digits = 3,
  rounding_function = round)
```

Arguments

`x`	a single prediction exlainer 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)
 exp_sgn
 plot(exp_sgn)
 plot(wine_rf_predict4, wine_lm_predict4, exp_sgn)

## End(Not run)
```

`plot.variable_importance_explainer` *Plots Global Model Explanations (Variable Importance)*

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 mdoels.

Usage

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

Arguments

`x`	a variable dropout exlainer 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

Value

a ggplot2 object

Examples

```
 ## Not run:
library("breakDown")
library("randomForest")
HR_rf_model <- randomForest(left~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data, y = HR_data$left)
vd_rf <- variable_importance(explainer_rf, type = "raw")
vd_rf
plot(vd_rf)

HR_glm_model <- glm(left~., data = breakDown::HR_data, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR_data, y = HR_data$left)
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
plot(vd_glm)

library("xgboost")
model_martix_train <- model.matrix(left~.-1, breakDown::HR_data)
data_train <- xgb.DMatrix(model_martix_train, label = breakDown::HR_data$left)
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_data$left, label = "xgboost")
vd_xgb <- variable_importance(explainer_xgb, type = "raw")
vd_xgb
plot(vd_xgb)

plot(vd_rf, vd_glm, vd_xgb)

# 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)

## 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, ...)
```

Arguments

x	a single variable exlainer produced with the 'single_variable' function
...	other explainers that shall be plotted together

Value

a ggplot2 object

Examples

```
library("breakDown")
logit <- function(x) exp(x)/(1+exp(x))

HR_glm_model <- glm(left~., data = breakDown::HR_data, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR_data, trans = logit)
expl_glm <- variable_response(explainer_glm, "satisfaction_level", "pdp", trans=logit)
```

```
plot(expl_glm)

 ## Not run:
library("randomForest")
HR_rf_model <- randomForest(factor(left)~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data,
                      predict_function = function(model, x)
                            predict(model, x, type = "prob")[,2])
expl_rf  <- variable_response(explainer_rf, variable = "satisfaction_level",
                      type = "pdp", which.class = 2, prob = TRUE)
plot(expl_rf)

plot(expl_rf, expl_glm)

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

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

# both models
plot(expl_rf, expl_glm)

## 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 observarvation 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.

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)
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)
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)
 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 expaliner 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(left~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data, y = HR_data$left)
mp_ex_rf <- model_performance(explainer_rf)
mp_ex_rf
plot(mp_ex_rf)

## End(Not run)
```

---

theme_mi2 *MI^2 Theme*

Description

MI^2 Theme

Usage

```
theme_mi2()
```

Value

theme object that can be added to ggplot2 plots

---

variable_importance *Loss from Variable Dropout*

Description

Loss from Variable Dropout

Usage

```
variable_importance(explainer, loss_function = function(observed,
  predicted) sum((observed - predicted)^2), ..., type = "raw",
  n_sample = 1000)
```

Arguments

explainer	a model to be explained, preprocessed by the 'explain' function
loss_function	a function thet 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.

Examples

```
 ## Not run:
library("breakDown")
library("randomForest")
HR_rf_model <- randomForest(left~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data, y = HR_data$left)
vd_rf <- variable_importance(explainer_rf, type = "raw")
vd_rf

HR_glm_model <- glm(left~., data = breakDown::HR_data, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR_data, y = HR_data$left)
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(left~.-1, breakDown::HR_data)
data_train <- xgb.DMatrix(model_martix_train, label = breakDown::HR_data$left)
param <- list(max_depth = 2, eta = 1, silent = 1, nthread = 2,
              objective = "binary:logistic", eval_metric = "auc")
HR_xgb_model <- xgb.train(param, data_train, nrounds = 50)
explainer_xgb <- explain(HR_xgb_model, data = model_martix_train,
                     y = HR_data$left, 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.

Examples

```
library("breakDown")
logit <- function(x) exp(x)/(1+exp(x))

HR_glm_model <- glm(left~., data = breakDown::HR_data, family = "binomial")
explainer_glm <- explain(HR_glm_model, data = HR_data, trans=logit)
expl_glm <- variable_response(explainer_glm, "satisfaction_level", "pdp")
expl_glm
```

```
 ## Not run:
library("randomForest")
HR_rf_model <- randomForest(factor(left)~., data = breakDown::HR_data, ntree = 100)
explainer_rf  <- explain(HR_rf_model, data = HR_data,
                      predict_function = function(model, x)
                             predict(model, x, type = "prob")[,2])
expl_rf  <- variable_response(explainer_rf, variable = "satisfaction_level", type = "pdp",
                      which.class = 2, prob = TRUE)
expl_rf
plot(expl_rf)

## 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 '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