Package ‘DALEX’

January 16, 2023

Title moDel Agnostic Language for Exploration and eXplanation

Version 2.4.3

Description Any unverified black box model is the path to failure. Opaqueness leads to distrust. Distrust leads to ignorance. Ignorance leads to rejection. DALEX package xrays any model and helps to explore and explain its behaviour. 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 direct interpretability. DALEX package contains various methods that help to understand the link between input variables and model output. Implemented methods help to explore the model on the level of a single instance as well as a level of the whole dataset. All model explainers are model agnostic and can be compared across different models. DALEX package is the cornerstone for ‘DrWhy.AI’ universe of packages for visual model exploration. Find more details in (Biecek 2018) <arXiv:1806.08915>.

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.5)

Imports ggplot2, iBreakDown (>= 1.3.1), ingredients (>= 2.0)

Suggests gower, ranger, testthat, methods


BugReports https://github.com/ModelOriented/DALEX/issues

NeedsCompilation no

Author Przemyslaw Biecek [aut, cre] (<https://orcid.org/0000-0001-8423-1823>), Szymon Maksymiuk [aut] (<https://orcid.org/0000-0002-3120-1601>), Hubert Baniecki [aut] (<https://orcid.org/0000-0001-6661-5364>)

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

Repository CRAN

Date/Publication 2023-01-15 23:10:02 UTC
R topics documented:

- apartments ........................................... 3
- colors_discrete_drwhy .......................... 3
- covid .............................................. 4
- dragons .......................................... 5
- explain.default .................................. 5
- fifa ................................................ 9
- happiness ......................................... 11
- HR .................................................. 11
- install_dependencies ............................ 12
- loss_cross_entropy .............................. 12
- loss_yardstick .................................. 14
- model_diagnostics ............................... 15
- model_info ...................................... 16
- model_parts ..................................... 18
- model_performance .............................. 20
- model_profile ................................... 21
- plot.list ......................................... 23
- plot.model_diagnostics ......................... 24
- plot.model_parts ................................ 25
- plot.model_performance ....................... 26
- plot.model_profile ................................ 27
- plot.predict_diagnostics ...................... 29
- plot.predict_parts .............................. 30
- plot.predict_profile ............................ 31
- plot.shap_aggregated ......................... 32
- predict.explainer ............................... 33
- predict_diagnostics ......................... 34
- predict_parts .................................. 36
- predict_profile .................................. 38
- print.description ............................... 40
- print.explainer ................................ 41
- print.model_diagnostics ...................... 41
- print.model_info ................................ 42
- print.model_performance ..................... 42
- print.model_profile ........................... 43
- print.predict_diagnostics .................... 43
- set_theme_dalex .................................. 44
- shap_aggregated .................................. 45
- theme_drwhy ..................................... 46
- titanic ........................................... 46
- update_data ..................................... 48
- update_label .................................... 49
- variable_effect .................................. 49
- yhat ............................................... 50

Index 53
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

Description

DrWhy color palettes for ggplot objects

Usage

`colors_discrete_drwhy(n = 2)`
`colors_diverging_drwhy()`
`colors_breakdown_drwhy()`

Arguments

`n` number of colors for color palette
Value  
color palette as vector of characters

---

### covid  
*Data for early COVID mortality*

#### Description
Two datasets of characteristics of patients infected with COVID. It is important to note that these are not real patient data. This is simulated data, generated to have relationships consistent with real data (obtained from NIH), but the data itself is not real. Fortunately, they are sufficient for the purposes of our exercise.

#### Usage
```r
data(covid_summer)
data(covid_spring)
```

#### Format
a data frame with 10,000 rows each and 12 columns

#### Details
The data is divided into two sets covid_spring and covid_summer. The first is acquired in spring 2020 and will be used as training data while the second dataset is acquired in summer and will be used for validation. In machine learning, model validation is performed on a separate data set. This controls the risk of overfitting an elastic model to the data. If we do not have a separate set then it is generated using cross-validation, out of sample or out of time techniques.

It contains 20,000 rows related to COVID mortality. It contains 11 variables such as: Gender, Age, Cardiovascular.Diseases, Diabetes, Neurological.Diseases, Kidney.Diseases.

Source: [https://github.com/BetaAndBit/RML](https://github.com/BetaAndBit/RML)

#### Source
[https://github.com/BetaAndBit/RML](https://github.com/BetaAndBit/RML)
**dragons**

**Description**

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

**Usage**

```r
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**

**Description**

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

explain.default(
  model,
  data = NULL,
  y = NULL,
  predict_function = NULL,
  predict_function_target_column = NULL,
  residual_function = NULL,
  weights = NULL,
  ...
)

Arguments

model object - a model to be explained

data data.frame or matrix - data which will be used to calculate the explanations. If not provided, then it will be extracted from the model. Data should be passed without a target column (this shall be provided as the y argument). NOTE: If the 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 a numeric vector with predictions. By default it is yhat.
predict_function_target_column
Character or numeric containing either column name or column number in the
model prediction object of the class that should be considered as positive (i.e.
the class that is associated with probability 1). If NULL, the second column of
the output will be taken for binary classification. For a multiclass classification
setting, that parameter cause switch to binary classification mode with one vs
others probabilities.

residual_function
function that takes four arguments: model, data, target vector y and predict func-
tion (optionally). It should return a numeric vector with model residuals for
given data. If not provided, response residuals \((y - \hat{y})\) are calculated. By default
it is residual_function_default.

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 logical. If TRUE (default) then diagnostic messages will be printed
precalculate logical. If TRUE (default) then predicted_values and residual are calcu-
lated when explainer is created. This will happen also if verbose is TRUE. Set
both verbose and precalculate to FALSE to omit calculations.
colorize logical. If TRUE (default) then WARNINGS, ERRORS and NOTES are colorized.
Will work only in the R console. Now by default it is FALSE while knitting and
TRUE otherwise.
model_info a named list (package, version, type) containing information about model. If
NULL, DALEX will seek for information on it’s own.
type type of a model, either classification or regression. If not specified then
type will be extracted from model_info.

Details
Please NOTE that the model is the only required argument. But some explanations may expect that
other arguments will be provided too.

Value
An object of the class explainer.
It’s a list with the 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.

References

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)

# 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)

# user provided predict_function
aps_ranger <- ranger::ranger(m2.price~., data = apartments, num.trees = 50)
custom_predict <- function(X.model, newdata) {
  predict(X.model, newdata)$predictions
}
aps_ranger_exp <- explain(aps_ranger, data = apartments, y = apartments$m2.price,
predict_function = custom_predict)

# user provided residual_function
aps_ranger <- ranger::ranger(m2.price~., data = apartments, num.trees = 50)
custom_residual <- function(X.model, newdata, y, predict_function) {
  abs(y - predict_function(X.model, newdata))
}
aps_ranger_exp <- explain(aps_ranger, data = apartments,
y = apartments$m2.price,
residual_function = custom_residual)
# binary classification
```
titanic_ranger <- ranger::ranger(as.factor(survived)~., data = titanic_imputed, num.trees = 50, probability = TRUE)
```

# keep in mind that for binary classification y parameter has to be numeric with 0 and 1 values
```
titanic_ranger_exp <- explain(titanic_ranger, data = titanic_imputed, y = titanic_imputed$survived)
```

# multiclass task
```
hr_ranger <- ranger::ranger(status~., data = HR, num.trees = 50, probability = TRUE)
```

# keep in mind that for multiclass y parameter has to be a factor,
# with same levels as in training data
```
hr_ranger_exp <- explain(hr_ranger, data = HR, y = HR$status)
```

# set model_info
```
model_info <- list(package = "stats", ver = "3.6.2", type = "regression")
aps_lm_model4 <- lm(m2.price ~., data = apartments)
apls_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v", model_info = model_info)
```

# simple function
```
aps_fun <- function(x) 58*x$surface
aps_fun_explainer <- explain(aps_fun, data = apartments, y = apartments$m2.price, label="sfun")
model_performance(aps_fun_explainer)
```

# set model_info
```
model_info <- list(package = "stats", ver = "3.6.2", type = "regression")
apls_lm_model4 <- lm(m2.price ~., data = apartments)
apls_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v", model_info = model_info)
apls_lm_explainer4 <- explain(aps_lm_model4, data = apartments, label = "model_4v", weights = as.numeric(apartments$construction.year > 2000))
```

# more complex model
```
library("ranger")
apls_ranger_model4 <- ranger(m2.price ~., data = apartments, num.trees = 50)
apls_ranger_explainer4 <- explain(apls_ranger_model4, data = apartments, label = "model_ranger")
apls_ranger_explainer4
```

---

**FIFA 20 preprocessed data**

### Description

The `fifa` dataset is a preprocessed `players_20.csv` dataset which comes as a part of "FIFA 20 complete player dataset" at Kaggle.
Usage

data(fifa)

Format

a data frame with 5000 rows, 42 columns and rownames

Details

It contains 5000 ‘overall’ best players and 43 variables. These are:

- short_name (rownames)
- nationality of the player (not used in modeling)
- overall, potential, value_eur, wage_eur (4 potential target variables)
- age, height, weight, attacking skills, defending skills, goalkeeping skills (37 variables)

It is advised to leave only one target variable for modeling.

Source: [https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset](https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset)

All transformations:

1. take 43 columns: [3, 5, 7:9, 11:14, 45:78] (R indexing)
2. take rows with value_eur > 0
3. convert short_name to ASCII
4. remove rows with duplicated short_name (keep first)
5. sort rows on overall and take top 5000
6. set short_name column as rownames
7. transform nationality to factor
8. reorder columns

Source

The players_20.csv dataset was downloaded from the Kaggle site and went through few transformations. The complete dataset was obtained from [https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset#players_20.csv](https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-dataset#players_20.csv) on January 1, 2020.
Description

The happiness_train and happiness_test datasets are generated based on the "World Happiness Report" at Kaggle https://www.kaggle.com/datasets/unsdsn/world-happiness.

Usage

data(happiness_train)
data(happiness_test)

Format

two data frames with total 781 rows, 7 columns each and rownames

Details

It contains data for 781 countries and 7 variables. These are:

- score - Happiness score
- gdp_per_capita - GDP per capita
- social_support - Social support
- healthy_life_expectancy - Healthy life expectancy
- freedom_life_choices - Freedom to make life choices
- generosity - Generosity
- perceptions_of_corruption - Perceptions of corruption

Source


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("ingredients", "iBreakDown", "ggpubr"))

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)

loss_default(x)

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?

x either an explainer or type of the model. One of "regression", "classification", "multiclass".

Value

numeric - value of the loss function

Examples

library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
probability = TRUE)
loss_one_minus_auc(titanic_imputed$survived, yhat(titanic_ranger_model, titanic_imputed))

HR_ranger_model_multi <- ranger(status~., data = HR, num.trees = 50, probability = TRUE)
loss_cross_entropy(as.numeric(HR$status), yhat(HR_ranger_model_multi, HR))
**Description**

The `yardstick` package provides many auxiliary functions for calculating the predictive performance of the model. However, they have an interface that is consistent with the tidyverse philosophy. The `loss_yardstick` function adapts loss functions from the `yardstick` package to functions understood by DALEX. Type compatibility for y-values and for predictions must be guaranteed by the user.

**Usage**

```r
loss_yardstick(loss, reverse = FALSE, reference = 1)
```

**Arguments**

- `loss`: loss function from the `yardstick` package
- `reverse`: shall the metric be reversed? for loss metrics lower values are better. `reverse = TRUE` is useful for accuracy-like metrics
- `reference`: if the metric is reverse then it is calculated as `reference - loss`. The default value is 1.

**Value**

loss function that can be used in the `model_parts` function

**Examples**

```r
titanic_glm_model <- glm(survived~., data = titanic_imputed, family = "binomial")
explainer_glm <- DALEX::explain(titanic_glm_model,  
                             data = titanic_imputed[, -8],  
                             y = factor(titanic_imputed$ survived))
# See the 'How to use DALEX with the yardstick package' vignette
# which explains this model with measures implemented in the 'yardstick' package
```
model_diagnostics

Dataset Level Model Diagnostics

Description
This function performs model diagnostic of residuals. Residuals are calculated and plotted against
predictions, true y values or selected variables. Find information how to use this function here:

Usage
model_diagnostics(explainer, variables = NULL, ...)

Arguments
explainer a model to be explained, preprocessed by the explain function
variables character - name of variables to be explained. Default NULL stands for all vari-
ables
... other parameters

Value
An object of the class model_diagnostics. It’s a data frame with residuals and selected variables.

References
drwhy.ai/

Examples
library(DALEX)
apartments_lm_model <- lm(m2.price ~ ., data = apartments)
explainer_lm <- explain(apartments_lm_model,
data = apartments,
y = apartments$m2.price)
diag_lm <- model_diagnostics(explainer_lm)
diag_lm
plot(diag_lm)

library("ranger")
apartments_ranger_model <- ranger(m2.price ~ ., data = apartments)
explainer_ranger <- explain(apartments_ranger_model,
data = apartments,
y = apartments$m2.price)
diag_ranger <- model_diagnostics(explainer_ranger)
diag_ranger
plot(diag_ranger)
model_info

**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, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'lm'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'randomForest'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'svm'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'glm'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'lrm'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'glmnet'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'cv.glmnet'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'ranger'
model_info(model, is_multiclass = FALSE, ...)
```

```
# S3 method for class 'gbm'
model_info(model, is_multiclass = FALSE, ...)
```
## S3 method for class 'model_fit'
model_info(model, is_multiclass = FALSE, ...)

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

## S3 method for class 'rpart'
model_info(model, is_multiclass = FALSE, ...)

## Default S3 method:
model_info(model, is_multiclass = FALSE, ...)

### Arguments

- **model**
  - model object

- **is_multiclass**
  - if TRUE and task is classification, then multitask classification is set. Else is omitted. If `model_info` was executed within explain function. DALEX will recognize subtype on its own.

- **...**
  - another arguments

### Details

Currently supported packages are:

- class `cv.glmnet` and `glmnet` - models created with `glmnet` package
- class `glm` - generalized linear models
- class `lrm` - models created with `rms` package,
- 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

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

```r
library("ranger")
```
```r
model_regr_rf <- ranger::ranger(status~, data = HR, num.trees = 50, probability = TRUE)
model_info(model_regr_rf, is_multiclass = TRUE)
```

**Description**

From DALEX version 1.0 this function calls the `feature_importance` Find information how to use this function here: [https://ema.drwhy.ai/featureImportance.html](https://ema.drwhy.ai/featureImportance.html).

**Usage**

```r
model_parts(
  explainer,
  loss_function = loss_default(explainer$model_info$type),
  ..., 
  type = "variable_importance",
  N = n_sample,
  n_sample = 1000
)
```

**Arguments**

- `explainer`: a model to be explained, preprocessed by the `explain` function
- `loss_function`: a function that will be used to assess variable importance. By default it is 1-AUC for classification, cross entropy for multilabel classification and RMSE for regression. Custom, user-made loss function should accept two obligatory parameters (observed, predicted), where observed states for actual values of the target, while predicted for predicted values. If attribute "loss_accuracy" is associated with function object, then it will be plotted as name of the loss function.
- `...`: other parameters
- `type`: character, type of transformation that should be applied for dropout loss. `variable_importance` and `raw` results raw drop losses, `ratio` returns `drop_loss/drop_loss_full_model` while `difference` returns `drop_loss - drop_loss_full_model`
- `N`: number of observations that should be sampled for calculation of variable importance. If NULL then variable importance will be calculated on whole dataset (no sampling).
- `n_sample`: alias for N held for backwards compatibility. number of observations that should be sampled for calculation of variable importance.
**Value**

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

**References**


**Examples**

```r
# regression
library("ranger")
apartments_ranger_model <- ranger(m2.price~., data = apartments, num.trees = 50)
explainer_ranger <- explain(apartments_ranger_model, data = apartments[, -1],
                             y = apartments$m2.price, label = "Ranger Apartments")
model_parts_ranger_aps <- model_parts(explainer_ranger, type = "raw")
head(model_parts_ranger_aps, 8)
plot(model_parts_ranger_aps)

# binary classification
titanic_glm_model <- glm(survived~., data = titanic_imputed, family = "binomial")
explainer_glm_titanic <- explain(titanic_glm_model, data = titanic_imputed[, -8],
                                  y = titanic_imputed$survived)
logit <- function(x) exp(x)/(1+exp(x))
custom_loss <- function(observed, predicted){
  sum((observed - logit(predicted))^2)
}
attr(custom_loss, "loss_name") <- "Logit residuals"
model_parts_glm_titanic <- model_parts(explainer_glm_titanic, type = "raw",
                                        loss_function = custom_loss)
head(model_parts_glm_titanic, 8)
plot(model_parts_glm_titanic)

# multilabel classification
HR_ranger_model_HR <- ranger(status~., data = HR, num.trees = 50,
                              probability = TRUE)
explainer_ranger_HR <- explain(HR_ranger_model_HR, data = HR[, -6],
                               y = HR$status, label = "Ranger HR")
model_parts_ranger_HR <- model_parts(explainer_ranger_HR, type = "raw")
head(model_parts_ranger_HR, 8)
plot(model_parts_ranger_HR)
```
Description

Function `model_performance()` calculates various performance measures for classification and regression models. For classification models following measures are calculated: F1, accuracy, recall, precision and AUC. For regression models following measures are calculated: mean squared error, R squared, median absolute deviation.

Usage

`model_performance(explainer, ..., cutoff = 0.5)`

Arguments

- `explainer` a model to be explained, preprocessed by the `explain` function
- `...` other parameters
- `cutoff` a cutoff for classification models, needed for measures like recall, precision, ACC, F1. By default 0.5.

Value

An object of the class `model_performance`.

It’s a list with following fields:

- `residuals` - data frame that contains residuals for each observation
- `measures` - list with calculated measures that are dedicated for the task, whether it is regression, binary classification or multiclass classification.
- `type` - character that specifies type of the task.

References


Examples

```r
# regression
library("ranger")
apartments_ranger_model <- ranger(m2.price~., data = apartments, num.trees = 50)
explainer_ranger_apartments <- explain(apartments_ranger_model, data = apartments[, -1],
y = apartments$m2.price, label = "Ranger Apartments")
model_performance_ranger_aps <- model_performance(explainer_ranger_apartments)
model_performance_ranger_aps
```
model_profile

Dataset Level Variable Profile as Partial Dependence or Accumulated Local Dependence Explanations

Description

This function calculates explanations on a dataset level set that explore model response as a function of selected variables. The explanations can be calculated as Partial Dependence Profile or Accumulated Local Dependence Profile. Find information how to use this function here: https://ema.drwhy.ai/partialDependenceProfiles.html. The variable_profile function is a copy of model_profile.

Usage

model_profile(
  explainer,
  variables = NULL,
  N = 100,
  ...,
```r
model_profile

variables = NULL,
  k = NULL,
  center = TRUE,
  type = "partial"
)

variable_profile(
  explainer,
  variables = NULL,
  N = 100,
  ..., 
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "partial"
)

single_variable(explainer, variable, type = "pdp", ...)

Arguments

explainer a model to be explained, preprocessed by the explain function
variables character - names of variables to be explained
N number of observations used for calculation of aggregated profiles. By default 100. Use NULL to use all observations.
... other parameters that will be passed to ingredients::aggregate_profiles
groups a variable name that will be used for grouping. By default NULL which means that no groups shall be calculated
k number of clusters for the hclust function (for clustered profiles)
center shall profiles be centered before clustering
type the type of variable profile. Either partial, conditional or accumulated.
variable deprecated, use variables instead

Details

Underneath this function calls the partial_dependence or accumulated_dependence functions from the ingredients package.

Value

An object of the class model_profile. It’s a data frame with calculated average model responses.

References

Examples

titanic_glm_model <- glm(survived~., data = titanic_imputed, family = "binomial")
explainer_glm <- explain(titanic_glm_model, data = titanic_imputed)
model_profile_glm_fare <- model_profile(explainer_glm, "fare")
plot(model_profile_glm_fare)

library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
                               probability = TRUE)
explainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed)
model_profile_ranger <- model_profile(explainer_ranger)
plot(model_profile_ranger, geom = "profiles")

model_profile_ranger_1 <- model_profile(explainer_ranger, type = "partial",
                                        variables = c("age", "fare"))
plot(model_profile_ranger_1, variables = c("age", "fare"), geom = "points")

model_profile_ranger_2 <- model_profile(explainer_ranger, type = "partial", k = 3)
plot(model_profile_ranger_2, geom = "profiles")

model_profile_ranger_3 <- model_profile(explainer_ranger, type = "partial", groups = "gender")
plot(model_profile_ranger_3, geom = "profiles")

model_profile_ranger_4 <- model_profile(explainer_ranger, type = "accumulated")
plot(model_profile_ranger_4, geom = "profiles")

# Multiple profiles
model_profile_ranger_fare <- model_profile(explainer_ranger, "fare")
plot(model_profile_ranger_fare, model_profile_glm_fare)

---

plot.list  

Plot List of Explanations

Description

Plot List of Explanations

Usage

## S3 method for class 'list'
plot(x, ...)

Arguments

x           a list of explanations of the same class
...          other parameters
Value

An object of the class ggplot.

Examples

```r
library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
                               probability = TRUE)
explainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed[,-8],
                            y = titanic_imputed$survived)
mp_ranger <- model_performance(explainer_ranger)
titanic_ranger_model2 <- ranger(survived~gender + fare, data = titanic_imputed,
                                num.trees = 50, probability = TRUE)
explainer_ranger2 <- explain(titanic_ranger_model2, data = titanic_imputed[,-8],
                           y = titanic_imputed$survived,
                           label = "ranger2")
mp_ranger2 <- model_performance(explainer_ranger2)
plot(list(mp_ranger, mp_ranger2), geom = "prc")
plot(list(mp_ranger, mp_ranger2), geom = "roc")
tmp <- list(mp_ranger, mp_ranger2)
names(tmp) <- c("ranger", "ranger2")
plot(tmp)
```

plot.model_diagnostics

Plot Dataset Level Model Diagnostics

Description

Plot Dataset Level Model Diagnostics

Usage

```r
## S3 method for class 'model_diagnostics'
plot(x, ..., variable = "y_hat", yvariable = "residuals", smooth = TRUE)
```

Arguments

- **x**: a data.frame to be explained, preprocessed by the `model_diagnostics` function
- **...**: other object to be included to the plot
- **variable**: character - name of the variable on OX axis to be explained, by default `y_hat`
- **yvariable**: character - name of the variable on OY axis, by default `residuals`
- **smooth**: logical shall the smooth line be added
Value

an object of the class `model_diagnostics_explainer`.

Examples

```r
apartments_lm_model <- lm(m2.price ~ ., data = apartments)
explainer_lm <- explain(apartments_lm_model,
    data = apartments,
    y = apartments$m2.price)
diag_lm <- model_diagnostics(explainer_lm)
diag_lm
plot(diag_lm)

library("ranger")
apartments_ranger_model <- ranger(m2.price ~ ., data = apartments)
explainer_ranger <- explain(apartments_ranger_model,
    data = apartments,
    y = apartments$m2.price)
diag_ranger <- model_diagnostics(explainer_ranger)
diag_ranger
plot(diag_ranger)
plot(diag_ranger, diag_lm)
plot(diag_ranger, diag_lm, variable = "y")
plot(diag_ranger, diag_lm, variable = "construction.year")
plot(diag_ranger, variable = "y", yvariable = "y_hat")
```

---

**plot.model_parts**

Plot Variable Importance Explanations

**Description**

Plot Variable Importance Explanations

**Usage**

```r
## S3 method for class 'model_parts'
plot(x, ...)
```

**Arguments**

- `x` an object of the class `model_parts`
- `...` other parameters described below

**Value**

An object of the class `ggplot`. 
### Plot options

**variable_importance:**

- `max_vars` maximal number of features to be included in the plot. Default value is 10
- `show_boxplots` logical. If TRUE (default) boxplot will be plotted to show permutation data.
- `bar_width` width of bars. By default 10
- `desc_sorting` logical. Should the bars be sorted descending? By default TRUE
- `title` the plot’s title, by default ‘Feature Importance’
- `subtitle` a character. Plot subtitle. By default NULL - then subtitle is set to "created for the XXX, YYY model", where XXX, YYY are labels of given explainers.

---

**plot.model_performance**

*Plot Dataset Level Model Performance Explanations*

---

### Description

Plot Dataset Level Model Performance Explanations

### Usage

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

### Arguments

- `x` a model to be explained, preprocessed by the `explain` function
- `...` other parameters
- `geom` either "prc", "roc", "ecdf", "boxplot", "gain", "lift" or "histogram" 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` alias for `loss_function` held for backwards compatibility.
- `loss_function` function that calculates the loss for a model based on model residuals. By default it’s the root mean square. NOTE that this argument was called `lossFunction`. 
Value

An object of the class model_performance.

Examples

```r
library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
probability = TRUE)
explainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed[-8],
y = titanic_imputed$survived)
mp_ranger <- model_performance(explainer_ranger)
plot(mp_ranger)
plot(mp_ranger, geom = "boxplot", show_outliers = 1)
titanic_ranger_model2 <- ranger(survived~gender + fare, data = titanic_imputed,
num.trees = 50, probability = TRUE)
explainer_ranger2 <- explain(titanic_ranger_model2, data = titanic_imputed[-8],
y = titanic_imputed$survived,
label = "ranger2")
mp_ranger2 <- model_performance(explainer_ranger2)
plot(mp_ranger, mp_ranger2, geom = "prc")
plot(mp_ranger, mp_ranger2, geom = "roc")
plot(mp_ranger, mp_ranger2, geom = "lift")
plot(mp_ranger, mp_ranger2, geom = "gain")
plot(mp_ranger, mp_ranger2, geom = "boxplot")
plot(mp_ranger, mp_ranger2, geom = "histogram")
plot(mp_ranger, mp_ranger2, geom = "ecdf")
titanic_glm_model <- glm(survived~., data = titanic_imputed, family = "binomial")
explainer_glm <- explain(titanic_glm_model, data = titanic_imputed[-8],
y = titanic_imputed$survived, label = "glm",
predict_function = function(m,x) predict.glm(m,x,type = "response"))
mp_glm <- model_performance(explainer_glm)
plot(mp_glm)
titanic_lm_model <- lm(survived~., data = titanic_imputed)
explainer_lm <- explain(titanic_lm_model, data = titanic_imputed[-8],
y = titanic_imputed$survived, label = "lm")
mp_lm <- model_performance(explainer_lm)
plot(mp_lm)
plot(mp_ranger, mp_glm, mp_lm)
plot(mp_ranger, mp_glm, mp_lm, geom = "boxplot")
plot(mp_ranger, mp_glm, mp_lm, geom = "boxplot", show_outliers = 1)
```
Description

Plot Dataset Level Model Profile Explanations

Usage

```r
## S3 method for class 'model_profile'
plot(x, ..., geom = "aggregates")
```

Arguments

- `x`: a variable profile explanation, created with the `model_profile` function
- `...`: other parameters
- `geom`: either "aggregates", "profiles", "points" determines which will be plotted

Value

An object of the class `ggplot`.

`aggregates`:
- `color`: a character. Either name of a color, or hex code for a color, or `_label_` if models shall be colored, or `_ids_` if instances shall be colored
- `size`: a numeric. Size of lines to be plotted
- `alpha`: a numeric between 0 and 1. Opacity of lines
- `facet_ncol`: number of columns for the `facet_wrap`
- `variables`: if not NULL then only variables will be presented
- `title`: a character. Partial and accumulated dependence explainers have default value.
- `subtitle`: a character. If NULL value will be dependent on model usage.

Examples

```r
titanic_glm_model <- glm(survived~., data = titanic_imputed, family = "binomial")
extplainer_glm <- explain(titanic_glm_model, data = titanic_imputed)
expl_glm <- model_profile(explainer_glm, "fare")
plot(expl_glm)
```

```r
library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
                              probability = TRUE)
extplainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed)
expl_ranger <- model_profile(explainer_ranger)
plot(expl_ranger)
plot(expl_ranger, geom = "aggregates")
```

```r
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
                              probability = TRUE)
extplainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed)
expl_ranger <- model_profile(explainer_ranger, type = "partial", variables = c("age", "fare"))
plot(expl_ranger, geom = "points")
```

```r
vp_ra <- model_profile(explainer_ranger, type = "partial", k = 3)
plot(vp_ra)
```
### plot.predict_diagnostics

**Plot Instance Level Residual Diagnostics**

#### Description

Plot Instance Level Residual Diagnostics

#### Usage

```r
## S3 method for class 'predict_diagnostics'
plot(x, ...)
```

#### Arguments

- `x`  
  an object with instance level residual diagnostics created with `predict_diagnostics` function

- `...`  
  other parameters that will be passed to `plot.ceteris_paribus_explainer`.

#### Value

an ggplot2 object of the class gg.

#### Examples

```r
library("ranger")
titanic_glm_model <- ranger(survived ~ gender + age + class + fare + sibsp + parch,  
data = titanic_imputed)
explainer_glm <- explain(titanic_glm_model,  
data = titanic_imputed,  
y = titanic_imputed$survived)
johny_d <- titanic_imputed[24, c("gender", "age", "class", "fare", "sibsp", "parch")]
```
pl <- predict_diagnostics(explainer_glm, johny_d, variables = NULL)
plot(pl)

pl <- predict_diagnostics(explainer_glm, johny_d,
                           neighbors = 10,
                           variables = c("age", "fare"))
plot(pl)

pl <- predict_diagnostics(explainer_glm,
                           johny_d,
                           neighbors = 10,
                           variables = c("class", "gender"))
plot(pl)

plot.predict_parts  

Plot Variable Attribution Explanations

Description
Plot Variable Attribution Explanations

Usage
## S3 method for class 'predict_parts'
plot(x, ...)

Arguments
x  an object of the class predict_parts
...
other parameters described below

Value
An object of the class ggplot.

Plot options

break_down:
- max_features maximal number of features to be included in the plot. default value is 10
- min_max a range of OX axis. By default NA, therefore it will be extracted from the contributions of x. But it can be set to some constants, useful if these plots are to be used for comparisons.
- add_contributions if TRUE, variable contributions will be added to the plot.
- shift_contributions number describing how much labels should be shifted to the right, as a fraction of range. By default equal to 0.05.
plot.predict_profile

Plot Variable Profile Explanations

Description

Plot Variable Profile Explanations

Usage

```r
## S3 method for class 'predict_profile'
plot(x, ...)
```

Arguments

- `x` an object of the class `predict_profile`
- `...` other parameters

Value

An object of the class `ggplot`. 

- `vcolors` If `NA` (default), DrWhy colors are used.
- `vnames` a character vector, if specified then will be used as labels on OY axis. By default `NULL`.
- `digits` number of decimal places (`round`) or significant digits (`signif`) to be used.
- `rounding_function` a function to be used for rounding numbers.
- `plot_distributions` if `TRUE` then distributions of conditional proportions will be plotted. This requires `keep_distributions=TRUE` in the `break_down`, `local_attributions`, or `local_interactions`.
- `baseline` if numeric then vertical line starts in baseline.
- `title` a character. Plot title. By default "Break Down profile".
- `subtitle` a character. Plot subtitle. By default `NULL` - then subtitle is set to "created for the XXX, YYY model", where XXX, YYY are labels of given explainers.
- `max_vars` alias for the `max_features` parameter.

**shap:**

- `show_boxplots` logical if `TRUE` (default) boxplot will be plotted to show uncertainty of attributions.
- `vcolors` If `NA` (default), DrWhy colors are used.
- `max_features` maximal number of features to be included in the plot. default value is 10
- `max_vars` alias for the `max_features` parameter.

**oscillations:**

- `bar_width` width of bars. By default 10
Plot options

ceteris_paribus:

- color a character. Either name of a color or name of a variable that should be used for coloring
- size a numeric. Size of lines to be plotted
- alpha a numeric between 0 and 1. Opacity of lines
- facet_ncol number of columns for the facet_wrap
- variables if not NULL then only variables will be presented
- variable_type a character. If numerical then only numerical variables will be plotted. If categorical then only categorical variables will be plotted.
- title a character. Plot title. By default "Ceteris Paribus profile".
- subtitle a character. Plot subtitle. By default NULL - then subtitle is set to "created for the XXX, YYY model", where XXX, YYY are labels of given explainers.
- categorical_type a character. How categorical variables shall be plotted? Either "lines" (default) or "bars".

---

**plot.shap_aggregated**  
*Plot Generic for Break Down Objects*

**Description**

Displays a waterfall aggregated shap plot for objects of shap_aggregated class.

**Usage**

```r
## S3 method for class 'shap_aggregated'
plot(
  x,
  ...,
  shift_contributions = 0.05,
  add_contributions = TRUE,
  add_boxplots = TRUE,
  max_features = 10,
  title = "Aggregated SHAP"
)
```

**Arguments**

- **x** an explanation object created with function `explain`.
- **...** other parameters like `vcolors`, `vnames`, `min_max`, `digits`, `rounding_function`, `baseline`, `subtitle`, `baseline`, `max_vars`.
- **shift_contributions** number describing how much labels should be shifted to the right, as a fraction of range. By default equal to 0.05.
predict.explainer

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

Usage

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

model_prediction(explainer, new_data, ...)

Value

a ggplot2 object.

Examples

library("DALEX")
set.seed(1313)
model_titanic_glm <- glm(survived ~ gender + age + fare,
                         data = titanic_imputed, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
                                data = titanic_imputed,
                                y = titanic_imputed$survived,
                                label = "glm")

bd_glm <- shap_aggregated(explain_titanic_glm, titanic_imputed[1:10, ])
bd_glm
plot(bd_glm)
plot(bd_glm, max_features = 3)
plot(bd_glm, max_features = 3,
     vnames = c("average","+ male","+ young","+ cheap ticket","+ other factors","final"))
predict_diagnostics

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
explainer a model to be explained, object of the class explainer
new_data data.frame or matrix - observations for prediction

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,])

library("ranger")
HR_ranger_model <- ranger(status~., data = HR, num.trees = 50, probability = TRUE)
explainer_ranger <- explain(HR_ranger_model, data = HR)
predict(explainer_ranger, HR[1:3,])
model_prediction(explainer_ranger, HR[1:3,])

predict_diagnostics  Instance Level Residual Diagnostics

Description

This function performs local diagnostic of residuals. For a single instance its neighbors are identified in the validation data. Residuals are calculated for neighbors and plotted against residuals for all data. Find information how to use this function here: https://ema.drwhy.ai/localDiagnostics.html.

Usage

predict_diagnostics(
  explainer,
  new_observation,
  variables = NULL,
  ...,
  nbins = 20,
  neighbors = 50,
  distance = gower::gower_dist
)
predict_diagnostics

individual_diagnostics(
  explainer,
  new_observation,
  variables = NULL,
  ..., nbins = 20,
  neighbors = 50,
  distance = gower::gower_dist
)

Arguments

explainer a model to be explained, preprocessed by the 'explain' function
new_observation a new observation for which predictions need to be explained
variables character - name of variables to be explained
... other parameters
nbins number of bins for the histogram. By default 20
neighbors number of neighbors for histogram. By default 50.
distance the distance function, by default the gower_dist() function.

Value

An object of the class 'predict_diagnostics'. It's a data frame with calculated distribution of residuals.

References


Examples

library("ranger")
titanic_glm_model <- ranger(survived ~ gender + age + class + fare + sibsp + parch, data = titanic_imputed)
explainer_glm <- explain(titanic_glm_model, data = titanic_imputed, y = titanic_imputed$survived)
johny_d <- titanic_imputed[24, c("gender", "age", "class", "fare", "sibsp", "parch")]

id_johny <- predict_diagnostics(explainer_glm, johny_d, variables = NULL)
id_johny
plot(id_johny)

id_johny <- predict_diagnostics(explainer_glm, johny_d, neighbors = 10,
predict_parts

variables = c("age", "fare")

plot(id_johny)

id_johny <- predict_diagnostics(explainer_glm,
    johny_d,
    neighbors = 10,
    variables = c("class", "gender"))

plot(id_johny)

predict_parts **Instance Level Parts of the Model Predictions**

Description

Instance Level Variable Attributions as Break Down, SHAP, aggregated SHAP or Oscillations explanations. Model prediction is decomposed into parts that are attributed for particular variables. From DALEX version 1.0 this function calls the break_down or shap functions from the iBreakDown package or ceteris_paribus from the ingredients package. Find information how to use the break_down method here: [https://ema.drwhy.ai/breakDown.html](https://ema.drwhy.ai/breakDown.html). Find information how to use the shap method here: [https://ema.drwhy.ai/shapley.html](https://ema.drwhy.ai/shapley.html). Find information how to use the oscillations method here: [https://ema.drwhy.ai/ceterisParibusOscillations.html](https://ema.drwhy.ai/ceterisParibusOscillations.html). aSHAP method provides explanations for a set of observations based on SHAP.

Usage

```r
predict_parts(
    explainer,
    new_observation,
    ...,  
    N = if (substr(type, 1, 4) == "osci") 500 else NULL,
    type = "break_down"
)

predict_parts_oscillations(explainer, new_observation, ...)

predict_parts_oscillations_uni(
    explainer,
    new_observation,
    variable_splits_type = "uniform",
    ...
)

predict_parts_oscillations_emp(
    explainer,
    new_observation,
    variable_splits_type = "empirical",
    ...
)
```
predict_parts

new_observation,
variable_splits = NULL,
variables = colnames(explainer$data),
...
)
predict_parts_break_down(explainer, new_observation, ...)
predict_parts_break_down_interactions(explainer, new_observation, ...)
predict_parts_shap(explainer, new_observation, ...)
predict_parts_shap_aggregated(explainer, new_observation, ...)

variable_attribution(
  explainer,
  new_observation,
  ...,
  N = if (substr(type, 1, 4) == "osci") 500 else NULL,
  type = "break_down"
)

Arguments

explainer a model to be explained, preprocessed by the explain function
new_observation a new observation for which predictions need to be explained
... other parameters that will be passed to iBreakDown::break_down
N the maximum number of observations used for calculation of attributions. By default NULL (use all) or 500 (for oscillations).
type the type of variable attributions. Either shap, aggregated_shap, oscillations, oscillations_uni, oscillations_emp, break_down or break_down_interactions.
variable_splits_type how variable grids shall be calculated? Will be passed to ceteris_paribus.
variable_splits named list of splits for variables. It is used by oscillations based measures. Will be passed to ceteris_paribus.
variables names of variables for which splits shall be calculated. Will be passed to ceteris_paribus.

Value

Depending on the type there are different classes of the resulting object. It's a data frame with calculated average response.

References

**Examples**

```r
library(DALEX)

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

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

explainer_lm <- explain(model_lm,
    data = dragons,
    y = dragons$year_of_birth,
    label = "model_lm")

bd_lm <- predict_parts_break_down(explainer_lm, new_observation = new_dragon)
head(bd_lm)
plot(bd_lm)

library("ranger")
model_ranger <- ranger(life_length ~ year_of_birth + height +
    weight + scars + number_of_lost_teeth,
    data = dragons, num.trees = 50)

explainer_ranger <- explain(model_ranger,
    data = dragons,
    y = dragons$year_of_birth,
    label = "model_ranger")

bd_ranger <- predict_parts_break_down(explainer_ranger, new_observation = new_dragon)
head(bd_ranger)
plot(bd_ranger)
```

---

**predict_profile**

*Instance Level Profile as Ceteris Paribus*

**Description**

This function calculated individual profiles aka Ceteris Paribus Profiles. From DALEX version 1.0 this function calls the *ceteris_paribus* from the ingredients package. Find information how to use this function here: https://ema.drwhy.ai/ceterisParibus.html.
predict_profile

Usage

predict_profile(
  explainer,
  new_observation,
  variables = NULL,
  ...,  
  type = "ceteris_paribus",
  variable_splits_type = "uniform"
)

individual_profile(
  explainer,
  new_observation,
  variables = NULL,
  ...,  
  type = "ceteris_paribus",
  variable_splits_type = "uniform"
)

Arguments

explainer a model to be explained, preprocessed by the explain function
new_observation a new observation for which predictions need to be explained
variables character - names of variables to be explained
... other parameters
type character, currently only the ceteris_paribus is implemented
variable_splits_type how variable grids shall be calculated? Use "quantiles" (default) for percentiles or "uniform" to get uniform grid of points. Will be passed to ‘ingredients’.

Value

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

References


Examples

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 <- predict_profile(dragon_lm_explainer4, new_observation = new_dragon, variables = c("year_of_birth", "height", "scars"))
head(dragon_lm_predict4)
plot(dragon_lm_predict4, variables = c("year_of_birth", "height", "scars"))

library("ranger")
dragon_ranger_model4 <- ranger(life_length ~ year_of_birth + height + weight + scars + number_of_lost_teeth, data = dragons, num.trees = 50)
dragon_ranger_explainer4 <- explain(dragon_ranger_model4, data = dragons, y = dragons$year_of_birth, label = "model_ranger")
dragon_ranger_predict4 <- predict_profile(dragon_ranger_explainer4, new_observation = new_dragon, variables = c("year_of_birth", "height", "scars"))
head(dragon_ranger_predict4)
plot(dragon_ranger_predict4, variables = c("year_of_birth", "height", "scars"))

---

**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 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)
apss_lm_explainer4 <- explain(aps_lm_model4, data = apartments, y = apartments$m2.price,
    label = "model_4v")
apss_lm_explainer4

library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
    probability = TRUE)
explainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed[,-8],
    y = titanic_imputed$survived,
    label = "model_ranger")
explainer_ranger
```

Print Dataset Level Model Diagnostics

Description

Generic function

Usage

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


### print.model_performance

**Arguments**

- **x**
  - an object with dataset level residual diagnostics created with `model_diagnostics` function

- **...**
  - other parameters

**Description**

Print Dataset Level Model Performance Summary

**Usage**

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

**Arguments**

- **x**
  - a model to be explained, object of the class 'model_performance_explainer'

- **...**
  - other parameters
Examples

```r
library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 100,
                               probability = TRUE)
# It's a good practice to pass data without target variable
explainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed[, -8],
                            y = titanic_imputed$survived)
# resulting dataframe has predicted values and residuals
mp_ex_rn <- model_performance(explainer_ranger)
mp_ex_rn
plot(mp_ex_rn)
```

---

**print.model_profile**  
**Print Dataset Level Model Profile**

**Description**

Generic function

**Usage**

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

**Arguments**

- `x`: an object with dataset level profile created with `model_profile` function
- `...`: other parameters

---

**print.predict_diagnostics**  
**Print Instance Level Residual Diagnostics**

**Description**

Generic function

**Usage**

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

Arguments

x an object with instance level residual diagnostics created with predict_diagnostics function

... other parameters

---

set_theme_dalex Default Theme for DALEX plots

Description

Default Theme for DALEX plots

Usage

set_theme_dalex(
  default_theme = "drwhy",
  default_theme_vertical = default_theme
)

theme_default_dalex()

theme_vertical_default_dalex()

Arguments

default_theme object - string ("drwhy" or "ema") or an object of ggplot theme class. Will be applied by default by DALEX to all horizontal plots
default_theme_vertical object - string ("drwhy" or "ema") or an object of ggplot theme class. Will be applied by default by DALEX to all vertical plots

Value

list with current default themes

Examples

old <- set_theme_dalex("ema")

library("ranger")
apartments_ranger_model <- ranger(m2.price~., data = apartments, num.trees = 50)
explainer_ranger <- explain(apartments_ranger_model, data = apartments[, -1],
    y = apartments$m2.price, label = "Ranger Apartments")
model_parts_ranger_aps <- model_parts(explainer_ranger, type = "raw")
head(model_parts_ranger_aps, 8)
plot(model_parts_ranger_aps)
### shap_aggregated

**SHAP aggregated values**

**Description**

This function works in a similar way to shap function from iBreakDown but it calculates explanations for a set of observation and then aggregates them.

**Usage**

```r
shap_aggregated(explainer, new_observations, order = NULL, B = 25, ...)
```

**Arguments**

- `explainer` a model to be explained, preprocessed by the `explain` function
- `new_observations` a set of new observations with columns that correspond to variables used in the model.
- `order` if not NULL, then it will be a fixed order of variables. It can be a numeric vector or vector with names of variables.
- `B` number of random paths
- `...` other parameters like label, predict_function, data, x

**Value**

an object of the shap_aggregated class.

**References**

Examples

```r
library("DALEX")
set.seed(1313)
model_titanic_glm <- glm(survived ~ gender + age + fare,
data = titanic_imputed, family = "binomial")
explain_titanic_glm <- explain(model_titanic_glm,
data = titanic_imputed,
y = titanic_imputed$survived,
label = "glm")

bd_glm <- shap_aggregated(explain_titanic_glm, titanic_imputed[1:10, ])
bd_glm
plot(bd_glm, max_features = 3)
```

theme_drwhy

DrWhy Theme for ggplot objects

Description

DrWhy Theme for ggplot objects

Usage

```
theme_drwhy()
theme_ema()
theme_drwhy_vertical()
theme_ema_vertical()
```

Value

theme for ggplot2 objects

---

titanic

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**

```r
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](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 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 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 https://biostat.app.vumc.org/wiki/Main/DataSets.

References

https://www.encyclopedia-titanica.org and https://CRAN.R-project.org/package=stablelearner

---

**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 structure 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_effect

Dataset Level Variable Effect as Partial Dependency Profile or Accumulated Local Effects

Description

From DALEX version 1.0 this function calls the accumulated_dependence or partial_dependence from the ingredients package. Find information how to use this function here: https://ema.drwhy.ai/partialDependenceProfiles.html.

Usage

variable_effect(explainer, variables, ..., type = "partial_dependency")

variable_effect_partial_dependency(explainer, variables, ...)

variable_effect_accumulated_dependency(explainer, variables, ...)
Arguments

- explainer: a model to be explained, preprocessed by the 'explain' function
- variables: character - names of variables to be explained
- ...: other parameters
- type: character - type of the response to be calculated. Currently following options are implemented: 'partial_dependency' for Partial Dependency and 'accumulated_dependency' for Accumulated Local Effects

Value

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

References


Examples

titanic_glm_model <- glm(survived~., data = titanic_imputed, family = "binomial")
explainer_glm <- explain(titanic_glm_model, data = titanic_imputed)
expl_glm <- variable_effect(explainer_glm, "fare", "partial_dependency")
plot(expl_glm)

library("ranger")
titanic_ranger_model <- ranger(survived~., data = titanic_imputed, num.trees = 50,
probability = TRUE)
explainer_ranger <- explain(titanic_ranger_model, data = titanic_imputed)
expl_ranger <- variable_effect(explainer_ranger, variables = "fare",
type = "partial_dependency")
plot(expl_ranger)
plot(expl_ranger, expl_glm)

# Example for factor variable (with factorMerger)
expl_ranger_factor <- variable_effect(explainer_ranger, variables = "class")
plot(expl_ranger_factor)
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 it 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, ...)

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

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

## S3 method for class '\'function\''
yhat(X.model, newdata, ...)
## S3 method for class 'party'
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 created with `glm`,
- class `model_fit` - models created with `parsnip` package,
- class `lm` - linear models created with `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,
- class `lrm` - models created with `rms` package,
- class `rpart` - models created with `rpart` package.

### Value

An numeric matrix of predictions
Index

* HR
  HR, 11
* apartments
  apartments, 3
* covid_spring
  covid, 4
* covid_summer
  covid, 4
* dragons
  dragons, 5
* fifa
  fifa, 9
* happiness_test
  happiness, 11
* happiness_train
  happiness, 11
* titanic
  titanic, 46

accumulated_dependence, 22, 49
apartments, 3
apartments_test (apartments), 3
apartmentsTest (apartments), 3
break_down, 31, 36
ceteris_paribus, 36–38
colors_breakdown_drwhy
  (colors_discrete_drwhy), 3
colors_discrete_drwhy, 3
colors_diverging_drwhy
  (colors_discrete_drwhy), 3
covid, 4
covid_spring (covid), 4
covid_summer (covid), 4
dragons, 5
dragons_test (dragons), 5
explain, 20, 26, 32
explain (explain.default), 5
explain.default, 5
facet_wrap, 28, 32
feature_importance, 18
feature_importance (model_parts), 18
fifa, 9
glm, 52
happiness, 11
happiness_test (happiness), 11
happiness_train (happiness), 11
HR, 11
HR_test (HR), 11
HRTest (HR), 11
individual_diagnostics
  (predict_diagnostics), 34
individual_profile (predict_profile), 38
install_dependencies, 12
lm, 52
local_attributions, 31
local_interactions, 31
loss_accuracy (loss_cross_entropy), 12
loss_cross_entropy, 12
loss_default (loss_cross_entropy), 12
loss_one_minus_auc
  (loss_cross_entropy), 12
loss_root_mean_square
  (loss_cross_entropy), 12
loss_sum_of_squares
  (loss_cross_entropy), 12
loss_yardstick, 14
model_diagnostics, 15, 24, 42
model_info, 16, 42
model_parts, 18
model_performance, 20
model_prediction (predict.explainer), 33
model_profile, 21, 28, 43
partial_dependence, 22, 49
plot.list, 23
plot.model_diagnostics, 24
plot.model_parts, 25
plot.model_performance, 26
plot.model_profile, 27
plot.predict_diagnostics, 29
plot.predict_parts, 30
plot.predict_profile, 31
plot.shap_aggregated, 32
predict.explainer, 33
predict_diagnostics, 29, 34, 44
predict_parts, 36
predict_parts_break_down
  (predict_parts), 36
predict_parts_break_down_interactions
  (predict_parts), 36
predict_parts_iboakdown
  (predict_parts), 36
predict_parts_oscillations
  (predict_parts), 36
predict_parts_oscillations_emp
  (predict_parts), 36
predict_parts_oscillations_uni
  (predict_parts), 36
predict_parts_shap, 36
predict_parts_shap_aggregated
  (predict_parts), 36
predict_profile, 38
print.description, 40
print.explainer, 41
print.model_diagnostics, 41
print.model_info, 42
print.model_performance, 42
print.model_profile, 43
print.predict_diagnostics, 43

round, 31

set_theme_dalex, 44
shap, 36
shap_aggregated, 45
signif, 31
single_variable (model_profile), 21

theme_default_dalex (set_theme_dalex), 44
theme_drwhy, 46
titanic, 46
titanic_imputed (titanic), 46
update_data, 48
update_label, 49

variable_attribution (predict_parts), 36
variable_effect, 49
variable_effect_accumulated_dependency
  (variable_effect), 49
variable_effect_partial_dependency
  (variable_effect), 49
variable_importance (model_parts), 18
variable_profile (model_profile), 21

theme_ema (theme_drwhy), 46
theme_ema_vertical (theme_drwhy), 46
theme_vertical_default_dalex
  (set_theme_dalex), 44
theme_vertical_drwhy (theme_drwhy), 46
theme_vertical_drwhy_vertical (theme_drwhy), 46