Package ‘auditor’

February 18, 2020

Title Model Audit - Verification, Validation, and Error Analysis

Version 1.2.0

Description Provides an easy to use unified interface for creating validation plots for any model. The ‘auditor’ helps to avoid repetitive work consisting of writing code needed to create residual plots. This visualizations allow to asses and compare the goodness of fit, performance, and similarity of models.

Depends R (>= 3.0.0)

License GPL

Encoding UTF-8

LazyData true

Imports DALEX, ggplot2, ggrepel, grid, gridExtra, hnp, scales

RoxygenNote 6.1.1

Suggests e1071, jsonlite, knitr, mgcv, r2d3, randomForest, rmarkdown, testthat

VignetteBuilder knitr

URL https://github.com/ModelOriented/auditor

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

NeedsCompilation no

Author Alicja Gosiewska [aut, cre] (https://orcid.org/0000-0001-6563-5742), Przemysław Biecek [aut, ths] (https://orcid.org/0000-0001-8423-1823), Hubert Baniecki [aut] (https://orcid.org/0000-0001-6661-5364), Tomasz Mikołajczyk [aut], Michal Burdakiewicz [ctb], Szymon Maksymiu [ctb]

Maintainer Alicja Gosiewska <alicjagosiewska@gmail.com>

Repository CRAN

Date/Publication 2020-02-18 10:40:02 UTC
R topics documented:

- audit .................................................. 3
- auditorData ............................................. 5
- check_residuals ........................................ 5
- check_residuals_autocorrelation ..................... 6
- check_residuals_outliers ............................. 7
- check_residuals_trend ............................... 7
- model_cooksdistance ................................ 8
- model_evaluation ....................................... 9
- model_halfnormal ..................................... 10
- model_performance ..................................... 11
- model_residual ......................................... 12
- plotD3 .................................................. 13
- plotD3_acf .............................................. 14
- plotD3_autocorrelation ................................ 15
- plotD3_cooksdistance .................................. 17
- plotD3_halfnormal .................................... 18
- plotD3_lift ............................................. 19
- plotD3_prediction ..................................... 21
- plotD3_rec .............................................. 22
- plotD3_residual ....................................... 24
- plotD3_roc ............................................... 25
- plotD3_rroc ............................................ 26
- plotD3_scalelocation .................................. 28
- plot_acf ................................................ 29
- plot_auditor ............................................ 30
- plot_autocorrelation .................................. 32
- plot_cooksdistance ................................... 33
- plot_correlation ....................................... 35
- plot_halfnormal ...................................... 36
- plot_lift ................................................ 37
- plot_pca ................................................ 38
- plot_prediction ....................................... 39
- plot_radar .............................................. 40
- plot_rec ................................................ 41
- plot_residual .......................................... 42
- plot_residual_boxplot ................................ 44
- plot_residual_density .................................. 45
- plot_roc ................................................ 46
- plot_rroc ............................................... 47
- plot_scalelocation ..................................... 48
- plot_tsecdf ............................................. 50
- print.auditor_model_cooksdistance .................. 51
- print.auditor_model_evaluation ....................... 52
- print.auditor_model_halfnormal ....................... 52
- print.auditor_model_performance ..................... 53
- print.auditor_model_residual ......................... 54
Function audit create modelAudit object for further validation of a model. Models may have very different structures. This function creates a unified representation of a model and calculates residuals, which can be further processed by various error analysis functions.

Function `audit()` is deprecated, please, use an object of class `explainer` created with function `explain` from the DALEX package.

Usage

```
audit(object, data = NULL, y = NULL, predict.function = NULL, residual.function = NULL, label = NULL)
```
Arguments

- **object**: An object containing a model or object of class explainer (see `explain`).
- **data**: Data.frame or matrix - data that will be used by further validation functions. If not provided, will be extracted from the model.
- **y**: Response vector that will be used by further validation functions. Some functions may require an integer vector containing binary labels with values 0,1. If not provided, will be extracted from the model.
- **predict.function**: Function that takes two arguments: model and data. It should return a numeric vector with predictions.
- **residual.function**: Function that takes three arguments: model, data and response vector. It should return a numeric vector with model residuals for given data. If not provided, response residuals ($y - \hat{y}$) are calculated.
- **label**: Character - the name of the model. By default it's extracted from the 'class' attribute of the model.

Value

An object of class ModelAudit, which contains:

- `model.class` class of the audited model,
- `label` the name of the model,
- `model` the audited model,
- `fitted.values` fitted values from model,
- `data` data used for fitting the model,
- `y` vector with values of predicted variable used for fitting the model,
- `predict.function` function that were used for model predictions,
- `residual.function` function that were used for calculating model residuals,
- `residuals`
- `std.residuals` standardized residuals - the residuals divided by theirs standard deviation.

Examples

```r
library(DALEX)

titanic <- na.omit(DALEX::titanic)
model_glm <- glm(survived ~ ., family = binomial, data = titanic)
audit_glm <- audit(model_glm)

p_fun <- function(model, data){predict(model, data, response = "link")}
audit_glm_newpred <- audit(model_glm, predict.function = p_fun)

library(randomForest)
model_rf <- randomForest(Species ~ ., data=iris)
audit_rf <- audit(model_rf)
```
Description

The auditor Data is an artificial data set. It consists of 2000 observations. First four of simulated variables are treated as continuous while the fifth one is categorical.

Examples

data("auditorData", package = "auditor")
head(auditorData)

check_residuals

Description

Currently three tests are performed - for outliers in residuals - for autocorrelation in target variable or in residuals - for trend in residuals as a function of target variable (detection of bias)

Usage

check_residuals(object, ...)

Arguments

object An object of class 'explainer' created with function explain from the DALEX package.

... other parameters that will be passed to further functions.

Value

list with statistics for particular checks

Examples

library(DALEX)
dragons <- DALEX::dragons[1:100, ]
lm_model <- lm(life_length ~ ., data = dragons)
lm_exp <- explain(lm_model, data = dragons, y = dragons$life_length)
library(auditor)
check_residuals(lm_exp)
## Not run:
library("ranger")
rf_model <- ranger(life_length ~ ., data = dragons)
predict_function <- function(m,x,...) predict(m, x, ...)$predictions
check_residuals_autocorrelation

Checks for autocorrelation in target variable or in residuals

Description

Checks for autocorrelation in target variable or in residuals

Usage

check_residuals_autocorrelation(object, method = "pearson")

Arguments

object An object of class 'explainer' created with function explain from the DALEX package.
method will be passed to the cor.test functions

Value

autocorrelation between target variable and between residuals

Examples

library(DALEX)
dragons <- DALEX::dragons[1:100, ]
lm_model <- lm(life_length ~ ., data = dragons)
lm_exp <- explain(lm_model, data = dragons, y = dragons$life_length)
library(auditor)
check_residuals_autocorrelation(lm_exp)
### check_residuals_outliers

**Checks for outliers**

**Description**

Checks for outliers

**Usage**

```r
check_residuals_outliers(object, n = 5)
```

**Arguments**

- **object**: An object of class 'explainer' created with function `explain` from the DALEX package.
- **n**: number of lowest and highest standardized residuals to be presented

**Value**

indexes of lowest and highest standardized residuals

**Examples**

```r
library(DALEX)
dragons <- DALEX::dragons[1:100, ]
lm_model <- lm(life_length ~ ., data = dragons)
library(auditor)
lm_exp <- explain(lm_model, data = dragons, y = dragons$life_length)
check_residuals_outliers(lm_exp)
```

---

### check_residuals_trend

**Checks for trend in residuals**

**Description**

Calculates loess fit for residuals and then extracts statistics that shows how far is this fit from one without trend

**Usage**

```r
check_residuals_trend(object, B = 20)
```
model_cooksdistance

**Arguments**

- **object**: An object of class 'explainer' created with function `explain` from the DALEX package.
- **B**: number of samplings

**Value**

standardized loess fit for residuals

**Examples**

```r
library(DALEX)
dragons <- DALEX::dragons[1:100, ]
lm_model <- lm(life_length ~ ., data = dragons)
lm_exp <- explain(lm_model, data = dragons, y = dragons$life_length)
library(auditor)
check_residuals_trend(lm_exp)
```

---

### model_cooksdistance

**Cook’s distances**

Calculates Cook’s distances for each observation. Please, note that it will work only for functions with specified ‘update’ method.

**Usage**

```r
model_cooksdistance(object)
observationInfluence(object)
```

**Arguments**

- **object**: An object of class 'explain' created with function `explain` from the DALEX package.

**Value**

An object of class 'auditor_model_cooksdistance'.

**References**

Examples

```r
# Load the required package
library(DALEX)

# Load the Titanic dataset
titanic <- na.omit(DALEX::titanic)

# Fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# Use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)

# Validate a model with auditor
library(auditor)
model_cooksdistance(exp_glm)
```

---

### model_evaluation

Create model evaluation explanation

**Description**

Creates explanation of classification model.

Returns, among others, true positive rate (tpr), false positive rate (fpr), rate of positive prediction (rpp), and true positives (tp).

Created object of class 'auditor_model_evaluation' can be used to plot Receiver Operating Characteristic (ROC) curve (plot `plot_roc`) and LIFT curve (plot `plot_lift`).

**Usage**

```r
model_evaluation(object)
```

**Arguments**

- `object`: An object of class 'explainer' created with function `explain` from the DALEX package.

**Value**

An object of class 'auditor_model_evaluation'.

**Examples**

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# Fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)
```
# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data= titanic, y = titanic$survived)

# validate a model with auditor
library(auditor)
model_evaluation(exp_glm)

model_halfnormal

## Create Halfnormal Explanation

### Description

Creates 'auditor_model_halfnormal' object that can be used for plotting halfnormal plot.

### Usage

```r
model_halfnormal(object, quant = FALSE, ...)
modelFit(object, quant = FALSE, ...)
```

### Arguments

- `object`: An object of class 'explainer' created with function `explain` from the DALEX package.
- `quant`: if TRUE values on axis are on quantile scale.
- `...`: other parameters passed do `hnp` function.

### Value

An object of the class 'auditor_model_halfnormal'.

### References


### Examples

```r
titanic <- na.omit(DALEX::titanic[1:100,])

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm)
```
# validate a model with auditor
library(auditor)
model_halfnormal(exp_glm)

---

**model_performance**

*Create Model Performance Explanation*

**Description**

Creates auditor_model_performance object that can be used to plot radar with ranking of models.

**Usage**

```r
model_performance(object, score = c("mae", "mse", "rec", "rroc"),
                  new_score = NULL, data = NULL, ...)
model_performance(object, score = c("mae", "mse", "rec", "rroc"),
                  new_score = NULL)
```

**Arguments**

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **score**: Vector of score names to be calculated. Possible values: acc, auc, cooksdistance, dw, f1, gini, halfnormal, mae, mse, peak, precision, r2, rec, recall, rmse, rroc, runs, specificity, one_minus_acc, one_minus_auc, one_minus_f1, one_minus_gini, one_minus_precision, one_minus_recall, one_minus_specificity (for detailed description see functions in see also section). Pass NULL if you want to use only custom scores by `new_score` parameter.
- **new_score**: A named list of functions that take one argument: object of class 'explainer' and return a numeric value. The measure calculated by the function should have the property that lower score value indicates better model.
- **data**: New data that will be used to calculate scores. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the score list.

**Value**

An object of the class auditor_model_performance.

**See Also**

- `score_acc`, `score_auc`, `score_cooksdistance`, `score_dw`, `score_f1`, `score_gini`, `score_halfnormal`,
- `score_mae`, `score_mse`, `score_peak`, `score_precision`, `score_r2`, `score_rec`, `score_recall`,
- `score_rmse`, `score_rroc`, `score_runs`, `score_specificity`, `score_one_minus_acc`, `score_one_minus_auc`,
- `score_one_minus_f1`, `score_one_minus_precision`, `score_one_minus_gini`, `score_one_minus_recall`,
- `score_one_minus_specificity`
Examples

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)

# validate a model with auditor
library(auditor)
model_performance(exp_glm)
```

---

**model_residual**  
Create Model Residuals Explanation

**Description**

Creates an 'auditor_model_residual' object that contains sorted residuals. An object can be further used to generate plots. For the list of possible plots see the see also section.

**Usage**

```r
model_residual(object)
modelResiduals(object)
```

**Arguments**

- `object` An object of class 'explainer' created with function `explain` from the DALEX package.

**See Also**

plot_acf, plot_autocorrelation, plot_residual, plot_residual_boxplot, plot_pca, plot_correlation, plot_prediction, plot_rec, plot_residual_density, plot_residual, plot_rroc, plot_scalelocation, plot_tsecdf

**Examples**

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)
```
# validate a model with auditor
library(auditor)
model_residual(exp_glm)

plotD3

Model Diagnostic Plots in D3 with r2d3 package.

Description
This function provides several diagnostic plots for regression and classification models. Provide object created with one of auditor's computational functions, `model_residual`, `model_cooksdistance`, `model_evaluation`, `model_performance`, `model_evaluation`.

Usage

plotD3(x, ...)

plotD3_auditor(x, ..., type = "residual")

## S3 method for class 'auditor_model_residual'
plotD3(x, ..., type = "residual")

## S3 method for class 'auditor_model_halfnormal'
plotD3(x, ..., type = "residual")

## S3 method for class 'auditor_model_evaluation'
plotD3(x, ..., type = "residual")

## S3 method for class 'auditor_model_cooksdistance'
plotD3(x, ..., type = "residual")

Args

- **x** object of class 'auditor_model_residual' (created with `model_residual` function), 'auditor_model_performance' (created with `model_performance` function), 'auditor_model_evaluation' (created with `model_evaluation` function), 'auditor_model_cooksdistance' (created with `model_cooksdistance` function), or 'auditor_model_halfnormal' (created with `model_halfnormal` function).
- **...** other arguments dependent on the type of plot or additional objects of classes 'auditor_model_residual', 'auditor_model_performance', 'auditor_model_evaluation', 'auditor_model_cooksdistance', 'auditor_model_halfnormal'.
- **type** the type of plot. Single character. Possible values: 'acf', 'autocorrelation', 'cooksdistance', 'halfnormal', 'lift', 'prediction', 'rec', 'residual', 'roc', 'rroc', 'scalelocation', (for detailed description see corresponding functions in see also section).
plotD3_acf

See Also

plotD3_acf, plotD3_autocorrelation, plotD3_cooksdistance, plotD3_halfnormal, plotD3_residual, plotD3_lift, plotD3_prediction, plotD3_rec, plotD3_roc, plotD3_rroc, plotD3_scalelocation

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plotD3(mr_lm)
plotD3(mr_lm, type = "prediction")

hn_lm <- model_halfnormal(exp_lm)
plotD3(hn_lm)

plotD3_acf

Plot Autocorrelation Function in D3 with r2d3 package.

Description

Plot Autocorrelation Function of models’ residuals.

Usage

plotD3_acf(object, ..., variable = NULL, alpha = 0.95,
   scale_plot = FALSE)

plotD3ACF(object, ..., variable = NULL, alpha = 0.95,
   scale_plot = FALSE)

Arguments

object An object of class ‘auditor_model_residual’ created with model_residual function.

... Other ‘auditor_model_residual’ objects to be plotted together.
variable  Name of variable to order residuals on a plot. If variable="_y_", the data is ordered by a vector of actual response (y parameter passed to the explain function). If variable = "_y_hat_" the data on the plot will be ordered by predicted response. If variable = NULL, unordered observations are presented.

alpha  Confidence level of the interval.

scale_plot Logical, indicates whenever the plot should scale with height. By default it’s FALSE.

Value

a ‘r2d3‘ object.

Examples

dragons <- DALEX::dragons[1:100, ]
# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plotD3_acf(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plotD3_acf(mr_lm, mr_rf)
Usage

plotD3_autocorrelation(object, ..., variable = NULL, points = TRUE,
    smooth = FALSE, point_count = NULL, single_plot = TRUE,
    scale_plot = FALSE, background = FALSE)

plotD3Autocorrelation(object, ..., variable = NULL, points = TRUE,
    smooth = FALSE, point_count = NULL, single_plot = TRUE,
    scale_plot = FALSE, background = FALSE)

Arguments

object An object of class 'auditor_model_residual' created with model_residu-
    al function.
...
other 'auditor_model_residual' objects to be plotted together.
variable Name of variable to order residuals on a plot. If variable="_y_", the data
    is ordered by a vector of actual response (y parameter passed to the explain
    function).
points Logical, indicates whenever observations should be added as points. By defaul-
    it's TRUE.
smooth Logical, indicates whenever smoothed lines should be added. By default it's
    FALSE.
point_count Number of points to be plotted per model. Points will be chosen randomly. By
    default plot all of them.
single_plot Logical, indicates whenever single or facets should be plotted. By default it's
    TRUE.
scale_plot Logical, indicates whenever the plot should scale with height. By default it's
    FALSE.
background Logical, available only if single_plot = FALSE. Indicates whenever backgrou-
    nd plots should be plotted. By default it's FALSE.

Value

a ‘r2d3’ object.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plotD3_autocorrelation(mr_lm)
plotD3_autocorrelation(mr_lm, smooth = TRUE)

plotD3_cooksdistance  

*Influence of observations Plot in D3 with r2d3 package.*

### Description

Plot of Cook’s distances used for estimate the influence of an single observation.

### Usage

```r
plotD3_cooksdistance(object, ..., nlabel = 3, single_plot = FALSE, scale_plot = FALSE, background = FALSE)
plotD3CooksDistance(object, ..., nlabel = 3, single_plot = FALSE, scale_plot = FALSE, background = FALSE)
```

### Arguments

- **object**  
  An object of class `auditor_model_cooksdistance` created with `model_cooksdistance` function.
- **...**  
  Other objects of class `auditor_model_cooksdistance`.
- **nlabel**  
  Number of observations with the biggest Cook’s distances to be labeled.
- **single_plot**  
  Logical, indicates whenever single or facets should be plotted. By default it’s FALSE.
- **scale_plot**  
  Logical, indicates whenever the plot should scale with height. By default it’s FALSE.
- **background**  
  Logical, available only if single_plot = FALSE. Indicates whenever background plots should be plotted. By default it’s FALSE.

### Details

Cook’s distance is a tool for identifying observations that may negatively affect the model. They may be also used for indicating regions of the design space where it would be good to obtain more observations. Data points indicated by Cook’s distances are worth checking for validity.

Cook’s Distances are calculated by removing the i-th observation from the data and recalculating the model. It shows how much all the values in the model change when the i-th observation is removed.

For model classes other than lm and glm the distances are computed directly from the definition.
Value

a ‘r2d3’ object.

References


See Also

plot_cooksdistance

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
cd_lm <- model_cooksdistance(exp_lm)

# plot results
plotD3_cooksdistance(cd_lm, nlabel = 5)

plotD3_halfnormal

Plot Half-Normal in D3 with r2d3 package.

Description

The half-normal plot is one of the tools designed to evaluate the goodness of fit of a statistical models. It is a graphical method for comparing two probability distributions by plotting their quantiles against each other. Points on the plot correspond to ordered absolute values of model diagnostic (i.e. standardized residuals) plotted against theoretical order statistics from a half-normal distribution.

Usage

plotD3_halfnormal(object, ..., quantiles = FALSE, sim = 99, scale_plot = FALSE)

plotD3HalfNormal(object, ..., quantiles = FALSE, sim = 99, scale_plot = FALSE)
**Arguments**

- **object**: An object of class 'auditor_model_halfnormal' created with `model_halfnormal` function.
- **...**: Other 'auditor_model_halfnormal' objects.
- **quantiles**: If TRUE values on axis are on quantile scale.
- **sim**: Number of residuals to simulate.
- **scale_plot**: Logical, indicates whenever the plot should scale with height. By default it's FALSE.

**Value**

A 'r2d3' object.

**See Also**

- `model_halfnormal`
- `score_halfnormal`, `plot_halfnormal`

**Examples**

```r
dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
hn_lm <- model_halfnormal(exp_lm)

# plot results
plotD3_halfnormal(hn_lm)
```

---

**plotD3_lift**

*Plot LIFT in D3 with r2d3 package.*

**Description**

LIFT is a plot of the rate of positive prediction against true positive rate for the different thresholds. It is useful for measuring and comparing the accuracy of the classifiers.
plotD3_lift(object, ..., scale_plot = FALSE, zeros = TRUE)
plotD3LIFT(object, ..., scale_plot = FALSE)

Arguments:
- **object**: An object of class 'auditor_model_evaluation' created with `model_evaluation` function.
- **...**: Other 'auditor_model_evaluation' objects to be plotted together.
- **scale_plot**: Logical, indicates whenever the plot should scale with height. By default it's FALSE.
- **zeros**: Logical. It makes the lines start from the \((0, 0)\) point. By default it's TRUE.

Value:
a 'r2d3' object

See Also:
- `plot_lift`

Examples:
```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)

# validate a model with auditor
library(auditor)
eva_glm <- model_evaluation(exp_glm)

# plot results
plotD3_lift(eva_glm)

model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic)
exp_glm_2 <- DALEX::explain(model_glm_2, data = titanic, y = titanic$survived, label = "glm2")
eva_glm_2 <- model_evaluation(exp_glm_2)

plotD3_lift(eva_glm, eva_glm_2)
```
**Description**

Function plotD3_prediction plots predicted values observed or variable values in the model.

**Usage**

```r
plotD3_prediction(object, ..., variable = "_y_", points = TRUE,  
smooth = FALSE, abline = FALSE, point_count = NULL,  
single_plot = TRUE, scale_plot = FALSE, background = FALSE)
```

```r
plotD3Prediction(object, ..., variable = NULL, points = TRUE,  
smooth = FALSE, abline = FALSE, point_count = NULL,  
single_plot = TRUE, scale_plot = FALSE, background = FALSE)
```

**Arguments**

- `object` An object of class 'auditor_model_residual.
- `...` Other modelAudit or modelResiduals objects to be plotted together.
- `variable` Name of variable to order residuals on a plot. If `variable="_y_"`, the data is ordered by a vector of actual response (y parameter passed to the explain function). If `variable="_y_hat_"` the data on the plot will be ordered by predicted response. If `variable=NULL`, unordered observations are presented.
- `points` Logical, indicates whenever observations should be added as points. By default it’s TRUE.
- `smooth` Logical, indicates whenever smoothed lines should be added. By default it’s FALSE.
- `abline` Logical, indicates whenever function y = x should be added. Works only with variable = NULL which is a default option.
- `point_count` Number of points to be plotted per model. Points will be chosen randomly. By default plot all of them.
- `single_plot` Logical, indicates whenever single or facets should be plotted. By default it’s TRUE.
- `scale_plot` Logical, indicates whenever the plot should scale with height. By default it’s FALSE.
- `background` Logical, available only if single_plot = FALSE. Indicates whenever backgroud plots should be plotted. By default it’s FALSE.

**Value**

a ‘r2d3‘ object.
plotD3_rec

Regression Error Characteristic Curves (REC) in D3 with r2d3 package.

Description

Error Characteristic curves are a generalization of ROC curves. On the x axis of the plot there is an error tolerance and on the y axis there is a percentage of observations predicted within the given tolerance.

Usage

plotD3_rec(object, ..., scale_plot = FALSE)
plotD3REC(object, ..., scale_plot = FALSE)
Arguments

object  
An object of class ‘auditor_model_residual’ created with `model_residual` function.

...  
Other ‘auditor_model_residual’ objects to be plotted together.

scale_plot  
Logical, indicates whenever the plot should scale with height. By default it’s FALSE.

Details

REC curve estimates the Cumulative Distribution Function (CDF) of the error
Area Over the REC Curve (REC) is a biased estimate of the expected error

Value

a ‘r2d3’ object.

References


See Also

`plot_rec`

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)
plotD3_rec(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length ~ ., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plotD3_rec(mr_lm, mr_rf)
plotD3_residual  

*Plot Residuals vs Observed, Fitted or Variable Values in D3 with r2d3 package.*

**Description**

Function `plotD3_residual` plots residual values vs fitted, observed or variable values in the model.

**Usage**

```r
plotD3_residual(object, ..., variable = "_y_", points = TRUE, 
smooth = FALSE, std_residuals = FALSE, nlabel = 0, 
point_count = NULL, single_plot = TRUE, scale_plot = FALSE, 
background = FALSE)
```

```r
plotD3Residual(object, ..., variable = NULL, points = TRUE, 
smooth = FALSE, std_residuals = FALSE, point_count = NULL, 
single_plot = TRUE, scale_plot = FALSE, background = FALSE)
```

**Arguments**

- **object**: An object of class `auditor_model_residual` created with `model_residual` function.
- **...**: Other `auditor_model_residual` objects to be plotted together.
- **variable**: Name of variable to order residuals on a plot. If `variable = "_y_"`, the data is ordered by a vector of actual response (y parameter passed to the `explain` function). If `variable = "_y_hat_"` the data on the plot will be ordered by predicted response. If `variable = NULL`, unordered observations are presented.
- **points**: Logical, indicates whenever observations should be added as points. By default it’s TRUE.
- **smooth**: Logical, indicates whenever smoothed lines should be added. By default it’s FALSE.
- **std_residuals**: Logical, indicates whenever standardized residuals should be used. By default it’s FALSE.
- **nlabel**: Number of observations with the biggest residuals to be labeled.
- **point_count**: Number of points to be plotted per model. Points will be chosen randomly. By default plot all of them.
- **single_plot**: Logical, indicates whenever single or facets should be plotted. By default it’s TRUE.
- **scale_plot**: Logical, indicates whenever the plot should scale with height. By default it’s FALSE.
- **background**: Logical, available only if `single_plot = FALSE`. Indicates whenever background plots should be plotted. By default it’s FALSE.
plotD3_roc

Value

a ‘r2d3’ object.

See Also

plot_residual

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plotD3_residual(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plotD3_residual(mr_lm, mr_rf)

plotD3_roc

Receiver Operating Characteristic (ROC) in D3 with r2d3 package.

Description

Receiver Operating Characterstic Curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for the different thresholds. It is useful for measuring and comparing the accuracy of the classificators.

Usage

plotD3_roc(object, ..., nlabel = NULL, scale_plot = FALSE)
Arguments

object  
An object of class 'auditor_model_evaluation' created with \texttt{model_evaluation} function.

...  
Other 'auditor_model_evaluation' objects to be plotted together.

nlabel  
Number of cutoff points to show on the plot. Default is 'NULL'.

scale_plot  
Logical, indicates whenever the plot should scale with height. By default it's FALSE.

Value

a 'r2d3' object

See Also

\texttt{plot_roc}

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- as.numeric(titanic$survived == "yes")

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# validate a model with auditor
library(auditor)
eva_glm <- model_evaluation(exp_glm)

# plot results
plotD3_roc(eva_glm)

# add second model
model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic)
exp_glm_2 <- DALEX::explain(model_glm_2, data = titanic, y = titanic$survived, label = "glm2")
eva_glm_2 <- model_evaluation(exp_glm_2)

plotD3_roc(eva_glm, eva_glm_2)
plotD3_rroc

Description
The basic idea of the ROC curves for regression is to show model asymmetry. The RROC is a plot where on the x-axis we depict total over-estimation and on the y-axis total under-estimation.

Usage
plotD3_rroc(object, ..., scale_plot = FALSE)

Arguments
object
An object of class 'auditor_model_residual' created with model_residual function.

... Other 'auditor_model_residual' objects to be plotted together.

scale_plot
Logical, indicates whenever the plot should scale with height. By default it’s FALSE.

Details
For RROC curves we use a shift, which is an equivalent to the threshold for ROC curves. For each observation we calculate new prediction: \( \hat{y}' = \hat{y} + s \) where \( s \) is the shift. Therefore, there are different error values for each shift: \( e_i = \hat{y}'_i - y_i \). Over-estimation is calculated as: \( \text{OVER} = \sum (e_i | e_i > 0) \). Under-estimation is calculated as: \( \text{UNDER} = \sum (e_i | e_i < 0) \). The shift equals 0 is represented by a dot.

The Area Over the RROC Curve (AOC) equals to the variance of the errors multiplied by \( \frac{\text{var}(e)}{2} \).

Value
a ‘r2d3‘ object

References

See Also
plotD3_rroc

Examples
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)
## plotD3_scalelocation

Scale Location Plot in D3 with r2d3 package.

### Description

Function `plotD3_scalelocation` plots square root of the absolute value of the residuals vs target, observed or variable values in the model. A vertical line corresponds to median.

### Usage

```r
plotD3_scalelocation(object, ..., variable = NULL, smooth = FALSE,
                      peaks = FALSE, point_count = NULL, single_plot = TRUE,
                      scale_plot = FALSE, background = FALSE)
```

### Arguments

- **object**: An object of class ‘auditor_model_residual’ created with `model_residual` function.
- **...**: Other ‘auditor_model_residual’ objects to be plotted together.
- **variable**: Name of variable to order residuals on a plot. If `variable="_y_"`, the data is ordered by a vector of actual response (y parameter passed to the `explain` function). If `variable="_y_hat_"` the data on the plot will be ordered by predicted response. If `variable=NULL`, unordered observations are presented.
- **smooth**: Logical, indicates whenever smoothed lines should be added. By default it’s `FALSE`.
- **peaks**: Logical, indicates whenever peak observations should be highlighted. By default it’s `FALSE`.
- **point_count**: Number of points to be plotted per model. Points will be chosen randomly. By default plot all of them.
plot_acf

single_plot Logical, indicates whenever single or facets should be plotted. By default it’s TRUE.
scale_plot Logical, indicates whenever the plot should scale with height. By default it’s FALSE.
background Logical, available only if single_plot = FALSE. Indicates whenever background plots should be plotted. By default it’s FALSE.

Value

a ‘r2d3’ object.

See Also

plot_scalelocation

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plotD3_scalelocation(mr_lm, peaks = TRUE)

---

plot_acf  

**Autocorrelation Function Plot**

**Description**

Plot Autocorrelation Function of models’ residuals.

**Usage**

plot_acf(object, ..., variable = NULL, alpha = 0.95)

plotACF(object, ..., variable = NULL, alpha = 0.95)
Arguments

object

An object of class 'auditor_model_residual' created with `model_residual` function.

...  

Other 'auditor_model_residual' objects to be plotted together.

variable

Name of variable to order residuals on a plot. If `variable = "_y_"`, the data is ordered by a vector of actual response (y parameter passed to the `explain` function). If `variable = "_y_hat_"` the data on the plot will be ordered by predicted response. If `variable = NULL`, unordered observations are presented.

alpha

Confidence level of the interval.

Value

A ggplot object.

Examples

dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot(mr_lm, type = "acf")
plot_acf(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length ~ ., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)

mr_rf <- model_residual(exp_rf)
plot_acf(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type = "acf")

---

plot_auditor  

Model Diagnostic Plots
plot_auditor

Description

This function provides several diagnostic plots for regression and classification models. Provide object created with one of aduditor’s computational functions, `model_residual`, `model_cooksdistance`, `model_evaluation`, `model_performance`, `model_evaluation`.

Usage

```r
plot_auditor(x, ..., type = "residual", ask = TRUE, grid = TRUE)
## S3 method for class 'auditor_model_residual'
plot(x, ..., type = "residual",
     ask = TRUE, grid = TRUE)
## S3 method for class 'auditor_model_performance'
plot(x, ..., type = "residual",
     ask = TRUE, grid = TRUE)
## S3 method for class 'auditor_model_halfnormal'
plot(x, ..., type = "residual",
     ask = TRUE, grid = TRUE)
## S3 method for class 'auditor_model_evaluation'
plot(x, ..., type = "residual",
     ask = TRUE, grid = TRUE)
## S3 method for class 'auditor_model_cooksdistance'
plot(x, ..., type = "residual",
     ask = TRUE, grid = TRUE)
```

Arguments

- `x`: object of class ‘auditor_model_residual’ (created with `model_residual` function), ‘auditor_model_performance’ (created with `model_performance` function), ‘auditor_model_evaluation’ (created with `model_evaluation` function), ‘auditor_model_cooksdistance’ (created with `model_cooksdistance` function), or ‘auditor_model_halfnormal’ (created with `model_halfnormal` function).
- `ask`: logical; if TRUE, the user is asked before each plot, see `par(ask=)`.
- `grid`: logical; if TRUE plots will be plotted on the grid.
**plot_autocorrelation**

**Autocorrelation of Residuals Plot**

**Description**

Plot of i-th residual vs i+1-th residual.

**Usage**

```r
plot_autocorrelation(object, ..., variable = "_y_hat_", smooth = FALSE)
plotAutocorrelation(object, ..., variable, smooth = FALSE)
```
**plot_cooksdistance**

**Influence of Observations Plot**

**Description**

Plot of Cook’s distances used for estimate the influence of an single observation.

**Usage**

```r
plot_cooksdistance(object, ..., nlabel = 3)
```

```r
plotCooksDistance(object, ..., nlabel = 3)
```
plot_cooksdistance

Arguments

object  An object of class ‘auditor_model_cooksdistance’ created with \texttt{model_cooksdistance} function.

...  Other objects of class ‘auditor_model_cooksdistance’.

nlabel  Number of observations with the biggest Cook’s distances to be labeled.

Details

Cook’s distance is a tool for identifying observations that may negatively affect the model. They may be also used for indicating regions of the design space where it would be good to obtain more observations. Data points indicated by Cook’s distances are worth checking for validity.

Cook’s Distances are calculated by removing the i-th observation from the data and recalculting the model. It shows how much all the values in the model change when the i-th observation is removed.

For model classes other than lm and glm the distances are computed directly from the definition.

Value

A ggplot object.

References


Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
cd_lm <- model_cooksdistance(exp_lm)

# plot results
plot_cooksdistance(cd_lm)
plot(cd_lm, type = "cooksdistance")
plot_correlation

Correlation of Model's Residuals Plot

Description
Matrix of plots. Left-down triangle consists of plots of fitted values (alternatively residuals), on the diagonal there are density plots of fitted values (alternatively residuals), in the right-top triangle there are correlations between fitted values (alternatively residuals).

Usage

plot_correlation(object, ..., values = "fit")

plotModelCorrelation(object, ..., values = "fit")

Arguments

object An object of class 'auditor_model_residual' created with model_residual function.
...
values "fit" for model fitted values or "res" for residual values.

Value

Invisibly returns a gtable object.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

library(randomForest)
model_rf <- randomForest(life_length ~ ., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)

# plot results
plot_correlation(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type = "correlation")
plot_halfnormal  

**Half-Normal plot**

**Description**

The half-normal plot is one of the tools designed to evaluate the goodness of fit of a statistical models. It is a graphical method for comparing two probability distributions by plotting their quantiles against each other. Points on the plot correspond to ordered absolute values of model diagnostic (i.e. standardized residuals) plotted against theoretical order statistics from a half-normal distribution.

**Usage**

```r
plot_halfnormal(object, ..., quantiles = FALSE, sim = 99)

plotHalfNormal(object, ..., quantiles = FALSE, sim = 99)
```

**Arguments**

- `object` An object of class 'auditor_model_halfnormal' created with `model_halfnormal` function.
- `...` Other 'auditor_model_halfnormal' objects.
- `quantiles` If TRUE values on axis are on quantile scale.
- `sim` Number of residuals to simulate.

**Value**

A ggplot object.

**See Also**

- `model_halfnormal`
- `score_halfnormal`

**Examples**

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
```
library(auditor)

hn_lm <- model_halfnormal(exp_lm)

# plot results
plot_halfnormal(hn_lm)
plot(hn_lm)

plot_lift  

---

LIFT Chart

Description

LIFT is a plot of the rate of positive prediction against true positive rate for the different thresholds. It is useful for measuring and comparing the accuracy of the classifiers.

Usage

plot_lift(object, ..., zeros = TRUE)

plotLIFT(object, ...)

Arguments

object  
An object of class 'auditor_model_evaluation' created with \texttt{model_evaluation} function.

...  
Other 'auditor_model_evaluation' objects to be plotted together.

zeros  
Logical. It makes the lines start from the \((0,0)\) point. By default it's \texttt{TRUE}.

Value

A ggplot object.

See Also

\texttt{model_evaluation}

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)

# validate a model with auditor
library(auditor)
eva_glm <- model_evaluation(exp_glm)

# plot results
plot_lift(eva_glm)
plot(eva_glm, type = "lift")

model_glm_2 <- glm(survived ~ . - age, family = binomial, data = titanic)
exp_glm_2 <- DALEX::explain(model_glm_2, data = titanic, y = titanic$survived, label = "glm2")
eva_glm_2 <- model_evaluation(exp_glm_2)

plot_lift(eva_glm, eva_glm_2)
plot(eva_glm, eva_glm_2, type = "lift")

---

**plot_pca**

### Principal Component Analysis of models

**Description**

Principal Component Analysis of models residuals. PCA can be used to assess the similarity of the models.

**Usage**

```r
plot_pca(object, ..., scale = TRUE)
plotModelPCA(object, ..., scale = TRUE)
```

**Arguments**

- `object` An object of class 'auditor_model_residual' created with `model_residual` function.
- `...` Other 'auditor_model_residual' objects to be plotted together.
- `scale` A logical value indicating whether the models residuals should be scaled before the analysis.

**Value**

A ggplot object.

**Examples**

```r
dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)
```

---

**plot_pca**

### Principal Component Analysis of models

**Description**

Principal Component Analysis of models residuals. PCA can be used to assess the similarity of the models.

**Usage**

```r
plot_pca(object, ..., scale = TRUE)
plotModelPCA(object, ..., scale = TRUE)
```

**Arguments**

- `object` An object of class 'auditor_model_residual' created with `model_residual` function.
- `...` Other 'auditor_model_residual' objects to be plotted together.
- `scale` A logical value indicating whether the models residuals should be scaled before the analysis.

**Value**

A ggplot object.

**Examples**

```r
dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)
```
# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)

# plot results
plot_pca(mr_lm, mr_rf)

---

**plot_prediction**

**Predicted response vs Observed or Variable Values**

**Description**

Plot of predicted response vs observed or variable Values.

**Usage**

```r
plot_prediction(object, ..., variable = "_y_", smooth = FALSE,
abline = FALSE)
plotPrediction(object, ..., variable = NULL, smooth = FALSE,
abline = FALSE)
```

**Arguments**

- `object`: An object of class `auditor_model_residual`.
- `...`: Other modelAudit or modelResiduals objects to be plotted together.
- `variable`: Name of variable to order residuals on a plot. If `variable="_y_"`, the data is ordered by a vector of actual response (y parameter passed to the `explain` function). If `variable="_y_hat_"` the data on the plot will be ordered by predicted response. If `variable=NULL`, unordered observations are presented.
- `smooth`: Logical, indicates whenever smooth line should be added.
- `abline`: Logical, indicates whenever function ‘y = x’ should be added. Works only with `variable = "_y_"` (which is a default option) or when ‘variable‘ equals actual response variable.

**Value**

A ggplot2 object.
Examples

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot_prediction(mr_lm, abline = TRUE)
plot_prediction(mr_lm, variable = "height", smooth = TRUE)
plot(mr_lm, type = "prediction", abline = TRUE)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plot_prediction(mr_lm, mr_rf, variable = "height", smooth = TRUE)
```

---

### plot_radar

**Model Ranking Plot**

#### Description

Radar plot with model score. Score are scaled to [0,1], each score is inversed and divided by maximum score value.

#### Usage

```r
plot_radar(object, ..., verbose = TRUE)
plotModelRanking(object, ..., verbose = TRUE)
```

#### Arguments

- `object` An object of class 'auditor_model_performance' created with `model_performance` function.
- `...` Other `auditor_model_performance` objects to be plotted together.
- `verbose` Logical, indicates whether values of scores should be printed.
**Value**

ggplot object

**Examples**

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mp_lm <- model_performance(exp_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mp_rf <- model_performance(exp_rf)

# plot results
plot_radar(mp_lm, mp_rf)

---

**plot_rec**

*Regression Error Characteristic Curves (REC)*

**Description**

Error Characteristic curves are a generalization of ROC curves. On the x axis of the plot there is an error tolerance and on the y axis there is a percentage of observations predicted within the given tolerance.

**Usage**

plot_rec(object, ...)

plotREC(object, ...)

**Arguments**

object An object of class auditor_model_residual created with `model_residual` function.

... Other auditor_model_residual objects to be plotted together.
Details

REC curve estimates the Cumulative Distribution Function (CDF) of the error
Area Over the REC Curve (REC) is a biased estimate of the expected error

Value

A ggplot object.

References


See Also

plot_roc, plot_rroc

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)
plot_rec(mr_lm)
plot(mr_lm, type = "rec")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plot_rec(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type = "rec")
plot_residual

Usage

plot_residual(object, ..., variable = "_y_", smooth = FALSE,
              std_residuals = FALSE, nlabel = 0)

plotResidual(object, ..., variable = NULL, smooth = FALSE,
              std_residuals = FALSE, nlabel = 0)

Arguments

object An object of class 'auditor_model_residual' created with model_residual function.
...
variable Name of variable to order residuals on a plot. If variable="_y_", the data is ordered by a vector of actual response (y parameter passed to the explain function). If variable="_y_hat_" the data on the plot will be ordered by predicted response. If variable = NULL, unordered observations are presented.
smooth Logical, indicates whenever smoothed lines should be added. By default it’s FALSE.
std_residuals Logical, indicates whenever standardized residuals should be used.
nlabel Number of observations with the biggest absolute values of residuals to be labeled.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot_residual(mr_lm)
plot(mr_lm, type = "residual")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plot_residual(mr_lm, mr_rf)
plot(mr_rf, mr_rf, type = "residual")
plot_residual_boxplot  

Plot Boxplots of Residuals

Description
A boxplot of residuals.

Usage
plot_residual_boxplot(object, ...)

plotResidualBoxplot(object, ...)

Arguments
object  
An object of class 'auditor_model_residual' created with \texttt{model_residual} function.

...  
Other 'auditor_model_residual' objects to be plotted together.

See Also
plot_residual

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot_residual_boxplot(mr_lm)
plot(mr_lm, type = "residual_boxplot")

library(randomForest)
model_rf <- randomForest(life_length ~ ., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)

mr_rf <- model_residual(exp_rf)

plot_residual_boxplot(mr_lm, mr_rf)
plot(mr_lm, mr_rf)
**plot_residual_density**  
*Residual Density Plot*

---

**Description**
Density of model residuals.

**Usage**

```r
plot_residual_density(object, ..., variable = "")
plotResidualDensity(object, ..., variable = NULL)
```

**Arguments**
- `object`: An object of class ’auditor_model_residual’ created with `model_residual` function.
- `...`: Other ’auditor_model_residual’ objects to be plotted together.
- `variable`: Split plot by variable’s factor level or median. If `variable="_y_"`, the plot will be splitted by actual response (y parameter passed to the `explain` function). If `variable="_y_hat_"` the plot will be splitted by predicted response. If `variable = NULL`, the plot will be splitted by observation index. If `variable = ""` plot is not splitted (default option).

**Value**
ggplot object

**See Also**
`plot_residual`

**Examples**

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot_residual_density(mr_lm)
```
plot(mr_lm, type = "residual_density")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plot_residual_density(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type = "residual_density")

---

**plot_roc**

**Receiver Operating Characteristic (ROC)**

**Description**

Receiver Operating Characteristic Curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for the different thresholds. It is useful for measuring and comparing the accuracy of the classifiers.

**Usage**

```r
plot_roc(object, ..., nlabel = NULL)
```

**Arguments**

- `object`: An object of class 'auditor_model_evaluation' created with `model_evaluation` function.
- `...`: Other 'auditor_model_evaluation' objects to be plotted together.
- `nlabel`: Number of cutoff points to show on the plot. Default is 'NULL'.

**Value**

A ggplot object.

**See Also**

`plot_rroc, plot_rec`

**Examples**

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- as.numeric(titanic$survived == "yes")

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)
```
# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# validate a model with auditor
library(auditor)
eva_glm <- model_evaluation(exp_glm)

# plot results
plot_roc(eva_glm)
plot(eva_glm)

# add second model
model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic)
exp_glm_2 <- DALEX::explain(model_glm_2, data = titanic, y = titanic$survived, label = "glm2")
eva_glm_2 <- model_evaluation(exp_glm_2)

plot_roc(eva_glm, eva_glm_2)
plot(eva_glm, eva_glm_2)

---

**Regression Receiver Operating Characteristic (RROC)**

**Description**

The basic idea of the ROC curves for regression is to show model asymmetry. The RROC is a plot where on the x-axis we depict total over-estimation and on the y-axis total under-estimation.

**Usage**

```r
plot_roc(object, ...)  
plotRROC(object, ...)
```

**Arguments**

- `object`: An object of class `auditor_model_residual` created with `model_residual` function.
- `...`: Other `auditor_model_residual` objects to be plotted together.

**Details**

For RROC curves we use a shift, which is an equivalent to the threshold for ROC curves. For each observation we calculate new prediction: \( \hat{y}' = \hat{y} + s \) where \( s \) is the shift. Therefore, there are different error values for each shift: \( e_i = \hat{y}_i' - y_i \).

Over-estimation is calculated as: \( OVER = \sum(e_i|e_i > 0) \).

Under-estimation is calculated as: \( UNDER = \sum(e_i|e_i < 0) \).

The shift equals 0 is represented by a dot.

The Area Over the RROC Curve (AOC) equals to the variance of the errors multiplied by \( frac{n^2}{2} \).
Value

A ggplot object.

References


See Also

plot_roc, plot_rec

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot_rroc(mr_lm)
plot(mr_lm, type = "rroc")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plot_rroc(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type="rroc")

plot_scalelocation

Scale location plot

Description

Variable values vs square root of the absolute value of the residuals. A vertical line corresponds to median.
plot_scalelocation

Usage

plot_scalelocation(object, ..., variable = "_y_", smooth = FALSE, peaks = FALSE)

plotScaleLocation(object, ..., variable = NULL, smooth = FALSE, peaks = FALSE)

Arguments

object An object of class 'auditor_model_residual' created with model_residual function.

... Other 'auditor_model_residual' objects to be plotted together.

variable Name of variable to order residuals on a plot. If variable="_y_", the data is ordered by a vector of actual response (y parameter passed to the explain function). If variable="_y_hat_" the data on the plot will be ordered by predicted response. If variable = NULL, unordered observations are presented.

smooth Logical, indicates whenever smoothed lines should be added. By default it's FALSE.

peaks A logical value. If TRUE peaks are marked on plot by black dots.

Value

A ggplot object.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)
mr_lm <- model_residual(exp_lm)

# plot results
plot_scalelocation(mr_lm)
plot(mr_lm, type = "scalelocation")
plot_tsecdf

Two-sided Cumulative Distribution Function

Description

Cumulative Distribution Function for positive and negative residuals.

Usage

plot_tsecdf(object, ..., scale_error = TRUE, outliers = NA, residuals = TRUE, reverse_y = FALSE)

plotTwoSidedECDF(object, ..., scale_error = TRUE, outliers = NA, residuals = TRUE, reverse_y = FALSE)

Arguments

object An object of class ‘auditor_model_residual’ created with model_residual function.

... Other modelAudit objects to be plotted together.

scale_error A logical value indicating whether ECDF should be scaled by proportions of positive and negative proportions.

outliers Number of outliers to be marked.

residuals A logical value indicating whether residuals should be marked.

reverse_y A logical value indicating whether values on y axis should be reversed.

Value

A ggplot object.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
library(auditor)

mr_lm <- model_residual(exp_lm)
plot_tsecdf(mr_lm)
plot(mr_lm, type="tsecdf")

library(randomForest)
```r
model_rf <- randomForest(life_length~., data = dragons)
exp_rf <- DALEX::explain(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(exp_rf)
plot_tsecdf(mr_lm, mr_rf, reverse_y = TRUE)
```

### Description

Prints Model Cook’s Distances Summary

### Usage

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

### Arguments

- `x`: an object `auditor_model_cooksdistance` created with `model_cooksdistance` function.
- `...`: other parameters

### Examples

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
model_cooksdistance(exp_lm)
```
print.auditor_model_evaluation

Prints Model Evaluation Summary

Description

Prints Model Evaluation Summary

Usage

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

Arguments

- `x` an object `auditor_model_evaluation` created with `model_evaluation` function.
- `...` other parameters

Examples

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived,
                          predict_function = function(m, d) predict(m, newdata = d, type = "response"))

# validate a model with auditor
library(auditor)
model_evaluation(exp_glm)
```

print.auditor_model_halfnormal

Prints Model Halfnormal Summary

Description

Prints Model Halfnormal Summary
print.auditor_model_performance

Usage

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

Arguments

x an object auditor_model_halfnormal created with model_halfnormal function.
...
other parameters

Examples

titanic <- na.omit(DALEX::titanic[1:100,])

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm)

# validate a model with auditor
library(auditor)
model_halfnormal(exp_glm)

print.auditor_model_performance

Prints Model Performance Summary

Description

Prints Model Performance Summary

Usage

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

Arguments

x an object auditor_model_performance created with model_performance function.
...
other parameters
Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)

# validate a model with auditor
library(auditor)
model_performance(exp_glm)

print.auditor_model_residual

Prints Model Residual Summary

Description
Prints Model Residual Summary

Usage

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

Arguments

x               an object auditor_model_residual created with model_residual function.
...
other parameters

Examples

titanic <- na.omit(DALEX::titanic)

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)
titanic$survived <- titanic$survived == "yes"

# use DALEX package to wrap up a model into explainer
exp_glm <- DALEX::explain(model_glm, data = titanic, y = titanic$survived)

# validate a model with auditor
library(auditor)
model_residual(exp_glm)
print.auditor_score

print.auditor_score  Prints of Models Scores

Description

Prints of Models Scores

Usage

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

Arguments

x an object auditor_score created with score function.
...
other parameters

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"
# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)
# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)
# calculate score
score(exp_glm, type = "auc")

score

Model Scores computations

Description

This function provides several scores for model validation and performance assessment. Scores can be also used to compare models.

Usage

score(object, type = "mse", data = NULL, ...)
Arguments

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **type**: The score to be calculated. Possible values: `acc`, `auc`, `cooksdistance`, `dw`, `f1`, `gini`, `halfnormal`, `mae`, `mse`, `peak`, `precision`, `r2`, `rec`, `recall`, `rmse`, `rroc`, `runs`, `specificity`, `one_minus_acc`, `one_minus_auc`, `one_minus_f1`, `one_minus_gini`, `one_minusPrecision`, `one_minus_recall`, `one_minus_specificity` (for detailed description see functions in see also section).
- **data**: New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.
- **...**: Other arguments dependent on the type of score.

Value

An object of class `auditor_score`, except Cooks distance, where numeric vector is returned.

See Also

-score_acc, score_auc, score_cooksdistance, score_dw, score_f1, score_gini, score_halfnormal, score_mae, score_mse, score_peak, score_precision, score_r2, score_rec, score_recall, score_rmse, score_rroc, score_runs, score_specificity, score_one_minus_acc, score_one_minus_auc, score_one_minus_f1, score_one_minus_gini, score_one_minus_precision, score_one_minus_recall, score_one_minus_specificity

Examples

dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score(exp_lm, type = 'mae')

<table>
<thead>
<tr>
<th>score_acc</th>
<th>Accuracy</th>
</tr>
</thead>
</table>

Description

Accuracy

Usage

score_acc(object, cutoff = 0.5, data = NULL, ...)
**score_auc**

**Arguments**

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **cutoff**: Threshold value, which divides model predicted values (`y_hat`) to calculate confusion matrix. By default it's 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.

**Value**

An object of class `auditor_score`.

**Examples**

```r

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_acc(exp_glm)
```

---

**score_auc**

*Area Under ROC Curve (AUC)*

**Description**

Area Under Curve (AUC) for Receiver Operating Characteristic.

**Usage**

```r

test_auc <- score_auc(object, data = NULL, ...)
```

**Arguments**

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.
Value

An object of class auditor_score.

See Also

plot_roc

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_auc(exp_glm)

score_cooksdistance

Score based on Cooks Distance

Description

Cook's distance are used for estimate of the influence of an single observation.

Usage

score_cooksdistance(object, verbose = TRUE, ...)

scoreCooksDistance(object, verbose = TRUE)

Arguments

object An object of class explainer created with function explain from the DALEX package.

verbose If TRUE progress is printed.

... Other arguments dependent on the type of score.
Details

Cook’s distance is a tool for identifying observations that may negatively affect the model. They may be also used for indicating regions of the design space where it would be good to obtain more observations. Data points indicated by Cook’s distances are worth checking for validity.

Cook’s Distances are calculated by removing the i-th observation from the data and recalculating the model. It shows how much all the values in the model change when the i-th observation is removed.

Models of classes other than lm and glm the distances are computed directly from the definition, so this may take a while.

Value

A vector of Cook’s distances for each observation.

numeric vector

See Also

score

Examples

dragons <- DALEX::dragons[1:100, ]
# fit a model
model_lm <- lm(life_length ~ ., data = dragons)
# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)
# calculate score
score_cooks_distance(exp_lm)

---

score_dw

Durbin-Watson Score

Description

Score based on Durbin-Watson test statistic. The score value is helpful in comparing models. It is worth pointing out that results of tests like p-value makes sense only when the test assumptions are satisfied. Otherwise test statistic may be considered as a score.

Usage

score_dw(object, variable = NULL, data = NULL, ...)
scoreDW(object, variable = NULL)
Arguments

object An object of class explainer created with function `explain` from the DALEX package.

variable Name of model variable to order residuals.

data New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.

... Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_dw(exp_lm)

__________________________________________

score_fi  \hspace{1cm} F1 Score

Description

F1 Score

Usage

score_f1(object, cutoff = 0.5, data = NULL, ...)

Arguments

object An object of class explainer created with function `explain` from the DALEX package.

cutoff Treshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it's 0.5.

data New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.

... Other arguments dependent on the type of score.
Value
An object of class auditor_score.

Examples
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_f1(exp_glm)

Gini Coefficient

The Gini coefficient measures the inequality among values of a frequency distribution. A Gini coefficient equals 0 means perfect equality, where all values are the same. A Gini coefficient equals 100

Usage
score_gini(object, data = NULL, ...)

Arguments

object An object of class explainer created with function explain from the DALEX package.
data New data that will be used to calculate the score. Pass NULL if you want to use data from object.
...
Other arguments dependent on the type of score.

Value
An object of class auditor_score.

See Also
plot_roc
Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_gini(exp_glm)

---

score_halfnormal   Half-Normal Score

Description

Score is approximately: \[ \sum #\{res_i \leq simres_{i,j}\} - n \] with the distinction that each element of sum is also scaled to take values from [0,1].

\(res_i\) is a residual for i-th observation, \(simres_{i,j}\) is the residual of j-th simulation for i-th observation, and \(n\) is the number of simulations for each observation. Scores are calculated on the basis of simulated data, so they may differ between function calls.

Usage

score_halfnormal(object, ...)
scoreHalfNormal(object, ...)

Arguments

object     An object of class explainer created with function explain from the DALEX package.
...
Extra arguments passed to hnp.

Value

An object of class auditor_score.

Examples

dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)
# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_halfnormal(exp_lm)

---

### score_mae

**Mean Absolute Error**

**Description**

Mean Absolute Error.

**Usage**

score_mae(object, data = NULL, ...)

scoreMAE(object)

**Arguments**

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.

**Value**

An object of class auditor_score.

**See Also**

score

**Examples**

dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_mae(exp_lm)
**score_mse**  

*Mean Square Error*

**Description**

Mean Square Error.

**Usage**

```r
score_mse(object, data = NULL, ...)
```

```r
scoreMSE(object)
```

**Arguments**

- `object`: An object of class `explainer` created with function `explain` from the DALEX package.
- `data`: New data that will be used to calculate the score. Pass `NULL` if you want to use `data` from `object`.
- `...`: Other arguments dependent on the type of score.

**Value**

An object of class `auditor_score`.

**See Also**

`score`

**Examples**

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_mse(exp_lm)
```
score_one_minus_acc

One minus accuracy

Description

One minus accuracy

Usage

score_one_minus_acc(object, cutoff = 0.5, data = NULL, ...)

Arguments

object An object of class explainer created with function explain from the DALEX package.
cutoff Treshold value, which divides model predicted values to calculate confusion matrix. By default it's 0.5.
data New data that will be used to calculate the score. Pass NULL if you want to use data from object.
... Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_one_minus_acc(exp_glm)
score_one_minus_auc 

One minus Area Under ROC Curve (AUC)

Description
One minus Area Under Curve (AUC) for Receiver Operating Characteristic.

Usage
score_one_minus_auc(object, data = NULL, ...)

Arguments
- object: An object of class explainer created with function explain from the DALEX package.
- data: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- ...: Other arguments dependent on the type of score.

Value
An object of class auditor_score.

Examples
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_one_minus_auc(exp_glm)
score_one_minus_f1

One Minus F1 Score

Description

One Minus F1 Score

Usage

score_one_minus_f1(object, cutoff = 0.5, data = NULL, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object of class explainer created with function explain from the DALEX package.</td>
</tr>
<tr>
<td>cutoff</td>
<td>Threshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it's 0.5.</td>
</tr>
<tr>
<td>data</td>
<td>New data that will be used to calculate the score. Pass NULL if you want to use data from object.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments dependent on the type of score.</td>
</tr>
</tbody>
</table>

Value

An object of class auditor_score.

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_one_minus_f1(exp_glm)
score_one_minus_gini  

One minus Gini Coefficient

Description

One minus Gini COefficient 100 0 expresses maximal inequality of values.

Usage

score_one_minus_gini(object, data = NULL, ...)

Arguments

object  
An object of class explainer created with function explain from the DALEX package.

data  
New data that will be used to calculate the score. Pass NULL if you want to use data from object.

...  
Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_one_minus_gini(exp_glm)
Description

One Minus Precision

Usage

```r
score_one_minus_precision(object, cutoff = 0.5, data = NULL, ...)
```

Arguments

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **cutoff**: Threshold value, which divides model predicted values (`y_hat`) to calculate confusion matrix. By default it’s `0.5`.
- **data**: New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.
- **...**: Other arguments dependent on the type of score.

Value

An object of class `auditor_score`.

Examples

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_one_minus_precision(exp_glm)
```
score_one_minus_recall

One minus recall

Description

One minus recall

Usage

score_one_minus_recall(object, cutoff = 0.5, data = NULL, ...)

Arguments

object       An object of class explainer created with function explain from the DALEX package.
cutoff       Treshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it's 0.5.
data         New data that will be used to calculate the score. Pass NULL if you want to use data from object.
...          Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

#create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_one_minus_recall(exp_glm)
score_one_minus_specificity

One minus specificity

Description

One minus specificity

Usage

score_one_minus_specificity(object, cutoff = 0.5, data = NULL, ...)

Arguments

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **cutoff**: Threshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it’s 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```r
  titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

  # fit a model
  model_glm <- glm(survived ~ ., family = binomial, data = titanic)

  # create an explainer
  exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

  # calculate score
  score_one_minus_specificity(exp_glm)
```
**Description**

This score is calculated on the basis of Peak test, which is used for checking for homoscedasticity of residuals in regression analyses.

**Usage**

```r
score_peak(object, variable = NULL, data = NULL, ...)
```

```r
scorePeak(object)
```

**Arguments**

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **variable**: Name of model variable to order residuals.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.

**Value**

An object of class auditor_score.

**Examples**

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_peak(exp_lm)
```
### Description

Precision

### Usage

```r
score_precision(object, cutoff = 0.5, data = NULL, ...)
```

### Arguments

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **cutoff**: Threshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it's 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.

### Value

An object of class auditor_score.

### Examples

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_precision(exp_glm)
```
score_r2

Description

The R2 is the coefficient of determination. An R2 coefficient equals 0 means that model explains none of the variability of the response. An R2 coefficient equals 1 means that model explains all the variability of the response.

Usage

score_r2(object, data = NULL, ...)

Arguments

object An object of class explainer created with function explain from the DALEX package.
data New data that will be used to calculate the score. Pass NULL if you want to use data from object.
... Other arguments dependent on the type of score.

Value

An object of class auditor_score.

See Also

score

Examples

dragons <- DALEX::dragons[1:100,]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# use DALEX package to wrap up a model into explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score with auditor
library(auditor)
score_r2(exp_lm)
### score_rec

**Area Over the Curve for REC Curves**

**Description**

The area over the Regression Error Characteristic curve is a measure of the expected error for the regression model.

**Usage**

```
    score_rec(object, data = NULL, ...)
    scoreREC(object)
```

**Arguments**

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **data**: New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.
- **...**: Other arguments dependent on the type of score.

**Value**

An object of class `auditor_score`.

**References**


**See Also**

`plot_rec`

**Examples**

```r
    dragons <- DALEX::dragons[1:100, ]

    # fit a model
    lm_model <- lm(life_length ~ ., data = dragons)

    # create an explainer
    lm_exp <- DALEX::explain(lm_model, data = dragons, y = dragons$life_length)

    # calculate score
    score_rec(lm_exp)
```
Description
Recall

Usage

score_recall(object, cutoff = 0.5, data = NULL, ...)

Arguments

- **object**: An object of class explainer created with function `explain` from the DALEX package.
- **cutoff**: Treshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it's 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **...**: Other arguments dependent on the type of score.

Value
An object of class auditor_score.

Examples

```r
titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_recall(exp_glm)
```
**Description**

Root Mean Square Error.

**Usage**

\[
\text{score\_rmse}(\text{object}, \text{data} = \text{NULL}, ...)  
\text{score\_RMSE}(\text{object})
\]

**Arguments**

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **data**: New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.
- **...**: Other arguments dependent on the type of score.

**Value**

An object of class `auditor\_score`.

**See Also**

`score`

**Examples**

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$s\_life\_length)

# calculate score
score\_rmse(exp_lm)
```
**score_rroc**

*Area Over the Curve for RROC Curves*

**Description**

The area over the Regression Receiver Operating Characteristic.

**Usage**

```r
score_rroc(object, data = NULL, ...)
```

```r
scoreRROC(object)
```

**Arguments**

- `object` An object of class `explainer` created with function `explain` from the DALEX package.
- `data` New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.
- `...` Other arguments dependent on the type of score.

**Value**

An object of class `auditor_score`.

**References**


**See Also**

- `plot_rroc`

**Examples**

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_rroc(exp_lm)
```
### Description

Score based on Runs test statistic. Note that this test is not very strong. It utilizes only signs of the residuals. The score value is helpful in comparing models. It is worth pointing out that results of tests like p-value makes sense only when the test assumptions are satisfied. Otherwise test statistic may be considered as a score.

### Usage

```r
score_runs(object, variable = NULL, data = NULL, ...)
scoreRuns(object, variable = NULL)
```

### Arguments

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **variable**: name of model variable to order residuals.
- **data**: New data that will be used to calculate the score. Pass `NULL` if you want to use data from `object`.
- **...**: Other arguments dependent on the type of score.

### Value

An object of class `auditor_score`.

### Examples

```r
dragons <- DALEX::dragons[1:100, ]

# fit a model
model_lm <- lm(life_length ~ ., data = dragons)

# create an explainer
exp_lm <- DALEX::explain(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_runs(exp_lm)
```
Description

Specificity

Usage

score_specificity(object, cutoff = 0.5, data = NULL, ...)

Arguments

object
An object of class explainer created with function explain from the DALEX package.
cutoff
Treshold value, which divides model predicted values (y_hat) to calculate confusion matrix. By default it's 0.5.
data
New data that will be used to calculate the score. Pass NULL if you want to use data from object.
...
Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

titanic <- na.omit(DALEX::titanic)
titanic$survived <- titanic$survived == "yes"

# fit a model
model_glm <- glm(survived ~ ., family = binomial, data = titanic)

# create an explainer
exp_glm <- DALEX::explain(model_glm, y = titanic$survived)

# calculate score
score_specificity(exp_glm)
Index

audit, 3
auditorData, 5
check_residuals, 5
check_residuals_autocorrelation, 6
check_residuals_outliers, 7
check_residuals_trend, 7
explain, 3–12, 15, 16, 21, 24, 28, 30, 33, 39, 43, 45, 49, 56–58, 60–80
gtable, 35
hnp, 10, 62
model_cooksdistance, 8, 13, 17, 31, 34, 51
model_evaluation, 9, 13, 20, 26, 31, 37, 46, 52
model_halfnormal, 10, 13, 19, 31, 36, 53
model_performance, 11, 13, 31, 40, 53
model_residual, 12, 13, 14, 16, 23, 24, 27, 28, 30, 31, 33, 35, 38, 41, 43–45, 47, 49, 50, 54
modelEvaluation (model_evaluation), 9
modelFit (model_halfnormal), 10
modelPerformance (model_performance), 11
modelResiduals (model_residual), 12
observationInfluence
  (model_cooksdistance), 8
par, 31
plot.auditor_model_cooksdistance
  (plot_auditor), 30
plot.auditor_model_evaluation
  (plot_auditor), 30
plot.auditor_model_halfnormal
  (plot_auditor), 30
plot.auditor_model_performance
  (plot_auditor), 30
plot.acf, 12, 29, 32
plot_auditor, 30
plot_autocorrelation, 12, 32, 33
plot_cooksdistance, 18, 32, 33
plot_correlation, 12, 32, 35
plot_halfnormal, 19, 32, 36
plot_lift, 9, 20, 32, 37
plot_pca, 12, 32, 38
plot_prediction, 12, 22, 32, 39
plot_radar, 32, 40
plot_rec, 12, 23, 32, 41, 46, 48, 75
plot_residual, 12, 25, 32, 42, 44, 45
plot_residual_boxplot, 12, 32, 44
plot_residual_density, 12, 32, 45
plot_roc, 9, 26, 32, 42, 46, 48, 58, 61
plot_rroc, 12, 32, 42, 46, 47, 78
plot_scalelocation, 12, 29, 32, 48
plot_tsecdf, 12, 32, 50
plotACF (plot_acf), 29
plotAutocorrelation
  (plot_autocorrelation), 32
plotCooksDistance (plot_cooksdistance), 33
plotD3Acf, 14, 14
plotD3_auditor (plotD3), 13
plotD3_autocorrelation, 14, 15
plotD3_cooksdistance, 14, 17
plotD3_halfnormal, 14, 18
plotD3_lift, 14, 19
plotD3_prediction, 14, 21
plotD3_rec, 14, 22
plotD3_residual, 14, 24
plotD3_rroc, 14, 25
plotD3_scalelocation, 14, 28
plotD3ACF (plotD3_acf), 14
plotD3Autocorrelation
  (plotD3_autocorrelation), 15
plotD3CooksDistance
  (plotD3_cooksdistance), 17
plotD3HalfNormal (plotD3_halfnormal), 18
plotD3LIFT (plotD3_lift), 19
plotD3Prediction (plotD3_prediction), 21
plotD3Residual (plotD3_residual), 24
plotD3ScaleLocation
  (plotD3_scalelocation), 28
plotHalfNormal (plot_halfnormal), 36
plotLIFT (plot_lift), 37
plotModelCorrelation
  (plot_correlation), 35
plotModelPCA (plot_pca), 38
plotModelRanking (plot_radar), 40
plotPrediction (plot_prediction), 39
plotResidual (plot_residual), 42
plotResidualBoxplot
  (plot_residual_boxplot), 44
plotResidualDensity
  (plot_residual_density), 45
plotROC (plot_roc), 46
plotRROC (plot_rroc), 47
plotScaleLocation (plot_scalelocation), 48
plotTwoSidedECDF (plot_tsecdf), 50
print.auditor_model_cooksdistance, 51
print.auditor_model_evaluation, 52
print.auditor_model_halfnormal, 52
print.auditor_model_performance, 53
print.auditor_model_residual, 54
print.auditor_score, 55
score, 55, 55, 59, 63, 64, 74, 77
score_acc, 11, 56, 56
score_auc, 11, 56, 57
score_cooksdistance, 11, 56, 58
score_dw, 11, 56, 59
score_f1, 11, 56, 60
score_gini, 11, 56, 61
score_halfnormal, 11, 19, 36, 56, 62
score_mae, 11, 56, 63
score_mse, 11, 56, 64
score_one_minus_acc, 11, 56, 65
score_one_minus_auc, 11, 56, 66
score_one_minus_f1, 11, 56, 67
score_one_minus_gini, 11, 56, 68
score_one_minus_precision, 11, 56, 69
score_one_minus_recall, 11, 56, 70
score_one_minus_specificity, 11, 56, 71
score_peak, 11, 56, 72
score_precision, 11, 56, 73
score_r2, 11, 56, 74
score_rec, 11, 56, 75
score_recall, 11, 56, 76
score_rmse, 11, 56, 77
score_rroc, 11, 56, 78
score_runs, 11, 56, 79
score_specificity, 11, 56, 80
scoreCooksDistance
  (score_cooksdistance), 58
scoreDW (score_dw), 59
scoreHalfNormal (score_halfnormal), 62
scoreMAE (score_mae), 63
scoreMSE (score_mse), 64
scorePeak (score_peak), 72
scoreREC (score_rec), 75
scoreRMSE (score_rmse), 77
scoreROC (score_auc), 57
scoreRROC (score_rroc), 78
scoreRuns (score_runs), 79