Package 'auditor'

August 31, 2019

Title Model Audit - Verification, Validation, and Error Analysis

Version 1.0.0

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>), Przemyslaw Biecek [aut, ths] (<https://orcid.org/0000-0001-8423-1823>), Hubert Baniecki [aut], Tomasz Mikołajczyk [aut], Michal Burdukiewicz [ctb]

Maintainer Alicja Gosiewska <alicjagosiewska@gmail.com>

Repository CRAN

Date/Publication 2019-08-31 10:50:02 UTC

R topics documented:

`audit` *Create modelAudit object*

Description

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

```
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)
```

auditorData *auditorData*

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_object *Check object*

Description

Checks if the object is of desired class

Usage

```
check_object(object, type = "res")
```

Arguments

object	Object passed to the function
type	Type of check; default is res which stands for "model residuals". Other possible values: eva - model evaluation

check_residuals *Automated tests for model 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 particualar 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
 rf_exp <- explain(rf_model, data = dragons, y = dragons$life_length,
           predict_function = predict_function)
 check_residuals(rf_exp)

## End(Not run)
```

---

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

```
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

```
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

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

Arguments

object	An object of class 'explainer' created with function explain from the DALEX package.
B	number fo samplings

Value

standardized loess fit for 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_trend(lm_exp)
```

drwhy_geom_point *DrWhy's wrapper for geom_point function*

Description

Function which draws point layers in desired order

Usage

```
drwhy_geom_point(df, smooth = FALSE, alpha_val)
```

Arguments

df	Data frame prepared by (make_dataframe) function
smooth	Logical, if set to TRUE point are drawn with alpha (set in alpha_val argument). Default is FALSE
alpha_val	Numeric, level of alpha of points when smooth is drawn

drwhy_geom_smooth *DrWhy's wrapper for geom_smooth function*

Description

Function which draws smooth layers in desired order

Usage

```
drwhy_geom_smooth(df)
```

Arguments

df	Data frame prepared by (make_dataframe) function

make_dataframe *Make data frame*

Description

Makes data frame(s) from passed models

Usage

```
make_dataframe(object, ..., variable = NULL, nlabel = NULL,
  type = "res", quant = NULL, values = NULL, scale_error = TRUE,
  outliers = NA, residuals = TRUE, reverse_y = FALSE, score = NULL,
  new.score = NULL)
```

Arguments

object	Object passed to the function
...	Other model_audit objects to be plotted together
variable	Variable
nlabel	Number of labels in calculating 'model_cooksdistance'
type	Type of check; default is res which stands for "model residuals".
quant	if TRUE values on axis are on quantile scale in 'plotHalfNormal'
values	for 'plotModelCorrelation'
scale_error	A logical value indicating whether ECDF should be scaled by proportions of positive and negative proportions; 'plotECDF'
outliers	Number of outliers to be marked on 'plotECDF'
residuals	A logical value indicating whether residuals should be marked on 'plotECDF'

reverse_y	A logical value indicating whether values on y axis should be reversed on ‘plotECDF‘
score	Vector of standard scores for modelRankingPlot
new.score	Function for custom score for modelRankingPlot Other possible values: eva - model evaluation

---

model_cooksdistance *Cook's distances*

Description

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

Usage

```
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

Cook, R. Dennis (1977). "Detection of Influential Observations in Linear Regression". doi:10.2307/1268249.

Examples

```
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 Explaination*

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

```
model_evaluation(object)

modelEvaluation(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

```
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

```
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

Moral, R., Hinde, J., & Demétrio, C. (2017). Half-Normal Plots and Overdispersed Models in R: The hnp Package.doi:http://dx.doi.org/10.18637/jss.v081.i10

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)
```

model_performance *Create Model Performance Explaination*

Description

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

Usage

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

modelPerformance(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 plotted. Possible values are 'auc' 'cookdistance', 'dw', 'peak', 'halfnormal', 'mae', 'mse', 'rec', 'rmse', 'rroc', 'runs' (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.

Value

An object of the class 'auditor_model_performance'.

See Also

score_auc, score_cooksdistance, score_dw, score_peak, score_halfnormal, score_mae, score_mse, score_rec, sc

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)
```

`model_residual` *Create Model Residuals Explaination*

Description

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

Usage

```
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_pr

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_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 aduditor'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")
```

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).
...	other arguments dependent on the type of plot or additionl 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', 'resiual', 'roc', 'rroc', 'scalelocation', (for detailed description see corresponding functions in see also section).

See Also

plotD3_acf,plotD3_autocorrelation,plotD3_cooksdistance,plotD3_halfnormal,plotD3_residual,plotD3_lif

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)
```

---

plotD3_autocorrelation

Autocorrelation Plot in D3 with r2d3 package.

Description

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

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_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).
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 backgroud 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

```
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 backgroud 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

Cook, R. Dennis (1977). "Detection of Influential Observations in Linear Regression". doi:10.2307/1268249.

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

```
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 classificators.

Usage

```
plotD3_lift(object, ..., scale_plot = FALSE)

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.

Value

a ‘r2d3‘ object

See Also

plot_lift

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
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)
```

---

plotD3_prediction *Plot Prediction vs Target, Observed or Variable Values in D3 with r2d3 package.*

Description

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

Usage

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

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 defaul 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.

See Also

plot_prediction

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_prediction(mr_lm, abline = TRUE)
```

```
plotD3_prediction(mr_lm, variable = "height", smooth = 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)
plotD3_prediction(mr_lm, mr_rf, variable = "weight", smooth = TRUE)
```

`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

Bi J., Bennett K.P. (2003). Regression error characteristic curves, in: Twentieth International Conference on Machine Learning (ICML-2003), Washington, DC.

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 resudial values vs fitted, observed or variable values in the model.

Usage

```
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)

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 defaul 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 backgroud plots should be plotted. By default it's FALSE.

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 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

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` *Regression Receiver Operating Characteristic (RROC) in D3 with r2d3 package.*

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: $OVER = \sum(e_i|e_i > 0)$.

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 $fracn^2 2$.

Value

a 'r2d3' object

References

Hernández-Orallo, José. 2013. 'ROC Curves for Regression'. Pattern Recognition 46 (12): 3395–3411.

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)

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

# plot results
plotD3_rroc(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_rroc(mr_lm, mr_rf)
```

---

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

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

plotD3ScaleLocation(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 defaul 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 backgroud 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*

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

```
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).

...	other arguments dependent on the type of plot or additionl objects of classes 'auditor_model_residual', 'auditor_model_performance', 'auditor_model_evaluation', 'auditor_model_cooksdistance', 'auditor_model_halfnormal'.
type	the type of plot. Character or vector of characters. Possible values: 'acf', 'autocorrelation', 'cooksdistance', 'halfnormal', 'lift', 'pca', 'radar', 'correlation', 'prediction', 'rec', 'resiual', 'residual_boxplot','residual_density', 'roc', 'rroc', 'scalelocation', 'tsecdf' (for detailed description see corresponding functions in see also section).
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.

See Also

plot_acf,plot_autocorrelation,plot_cooksdistance,plot_halfnormal,plot_residual_boxplot,plot_lift,pl

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)
plot(mr_lm, type = "prediction")

hn_lm <- model_halfnormal(exp_lm)
plot(hn_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)
mp_lm <- model_performance(exp_lm)
plot(mp_lm, mp_rf)
```

`plot_autocorrelation` *Autocorrelation of Residuals Plot*

Description

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

Usage

```
plot_autocorrelation(object, ..., variable = "_y_hat_", smooth = FALSE)

plotAutocorrelation(object, ..., variable, smooth = 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).
`smooth`	Logical, if TRUE smooth line will be added.

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_autocorrelation(mr_lm)
plot(mr_lm, type = "autocorrelation")
plot_autocorrelation(mr_lm, smooth = TRUE)
plot(mr_lm, type = "autocorrelation", smooth = TRUE)
```

`plot_cooksdistance` *Influence of Observations Plot*

Description

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

Usage

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

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

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.

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 ggplot object.

References

Cook, R. Dennis (1977). "Detection of Influential Observations in Linear Regression". doi:10.2307/1268249.

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 (aternatively residuals), on the diagonal there are density plots of fitted values (alternatively residuals), in the right-top triangle there are correlations between fitte dvalues (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.
`...`	Other 'auditor_model_residual' objects to be plotted together.
`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

```
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

```
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 classificators.

Usage

```
plot_lift(object, ...)

plotLIFT(object, ...)
```

Arguments

object	An object of class 'auditor_model_evaluation' created with model_evaluation function.
...	Other 'auditor_model_evaluation' objects to be plotted together.

Value

A ggplot object.

See Also

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

```
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

```
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

```
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 = NULL which is a default option.

Value

A ggplot2 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_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

```
plot_radar(object, ..., score = c("mae", "mse", "rec", "rroc"),
  new_score = NULL, verbose = TRUE)

plotModelRanking(object, ..., score = c("MAE", "MSE", "REC", "RROC"),
  new_score = NULL)
```

Arguments

object	An object of class 'auditor_model_performance' created with model_performance function.
...	Other auditor_model_performance' objects to be plotted together.
score	Vector of score names to be plotted.
new_score	A named list of functions that take one argument: object of class ModelAudit and return a numeric value. The measure calculated by the function should have the property that lower score value indicates better model.
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

Bi J., Bennett K.P. (2003). Regression error characteristic curves, in: Twentieth International Conference on Machine Learning (ICML-2003), Washington, DC.

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` *Plot Residuals vs Observed, Fitted or Variable Values*

Description

A plot of residuals against fitted values, observed values or any variable.

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.
`...`	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.
`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 `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

```
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

```
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 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

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

plotROC(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

```
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)
```

---

plot_rroc *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

```
plot_rroc(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 $fracn^2 2$.

Value

A ggplot object.

References

Hernández-Orallo, José. 2013. 'ROC Curves for Regression'. Pattern Recognition 46 (12): 3395–3411.

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.

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)
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)
```

`prepare_object` *Prepare object for 'make_dataframe" function*

Description

Prepare object for 'make_dataframe" function

Usage

```
prepare_object(object, variable, nlabel, type, quant, values, scale_error,
  outliers, reverse_y, score, new.score)
```

Arguments

`object`	An audited model
`variable`	Variable
`nlabel`	Number of labels
`type`	Type of model passed
`quant`	Logical
`values`	Values
`scale_error`	Error scaled
`outliers`	Outliers
`reverse_y`	y reversed
`score`	Scores
`new.score`	New scores

print.auditor_model_cooksdistance

Prints Model Cook's Distances Summary

Description

Prints Model Cook's Distances Summary

Usage

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

Arguments

x	an object 'auditor_model_cooksdistance' created with model_cooksdistance function.
...	other parameters

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
model_cooksdistance(exp_lm)
```

print.auditor_model_evaluation

Prints Model Evaluation Summary

Description

Prints Model Evaluation Summary

Usage

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

Arguments

x	an object 'auditor_model_evaluation' created with model_evaluation 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,
                  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

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)
```

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, score = "mse", ...)
```

Arguments

object	An object of class 'explainer' created with function explain from the DALEX package.
score	The score to be calculated. Possible values: 'auc' 'cookdistance', 'dw', 'peak', 'halfnormal', 'mae', 'mse', 'rec', 'rmse', 'rroc', 'runs' (for detailed description see functions in see also section).
...	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_auc, score_cooksdistance, score_dw, score_peak, score_halfnormal, score_mae, score_mse, score_rec, sc

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, score = 'mae')
```

---

score_auc *Area Under ROC Curve (AUC)*

Description

Area Under Curve (AUC) for Receiver Operating Characteristic.

Usage

```
score_auc(object)

scoreROC(object)
```

Arguments

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

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.

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_cooksdistance(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)

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.

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_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 'score_audit'.

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)

scoreMAE(object)
```

Arguments

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

Value

an object of class 'score_audit'

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

```
score_mse(object)

scoreMSE(object)
```

Arguments

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

Value

an object of class score_audit

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_mse(exp_lm)
```

---

score_peak *Peak Score*

Description

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

Usage

```
score_peak(object, variable = NULL)

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.

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_peak(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)

scoreREC(object)
```

Arguments

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

Value

an object of class 'score_audit'.

References

J. Bi, and K. P. Bennet, "Regression error characteristic curves," in Proc. 20th Int. Conf. Machine Learning, Washington DC, 2003, pp. 43-50

See Also

plot_rec

Examples

```
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)
```

score_rmse *Root Mean Square Error*

Description

Root Mean Square Error.

Usage

```
score_rmse(object)

scoreRMSE(object)
```

Arguments

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

Value

An object of class 'score_audit'.

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_rmse(exp_lm)
```

score_rroc *Area Over the Curve for RROC Curves*

Description

The area over the Regression Receiver Operating Characteristic.

Usage

```
score_rroc(object)

scoreRROC(object)
```

Arguments

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

Value

an object of class 'score_audit'.

References

Hernández-Orallo, José. 2013. 'ROC Curves for Regression'. Pattern Recognition 46 (12): 3395–3411.

See Also

plot_rroc

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_rroc(exp_lm)
```

score_runs *Runs Score*

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

```
score_runs(object, variable = 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.

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)

# caluclate score
score_runs(exp_lm)
```

theme_drwhy *DrWhy Theme for ggplot objects*

Description

DrWhy Theme for ggplot objects

Usage

```
theme_drwhy()

theme_drwhy_colors(n = 2)
```

Arguments

n	number of colors to get

Value

theme for ggplot2 objects

Index