

Package ‘auditor’

May 7, 2026

Title Model Audit - Verification, Validation, and Error Analysis

Version 1.3.5

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 (>= 3.5.0)

License GPL

Encoding UTF-8

LazyData true

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

RoxygenNote 7.2.3

Suggests jsonlite, knitr, markdown, mgcv, r2d3, randomForest, rmarkdown, spelling, testthat, covr

VignetteBuilder knitr

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

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

Language en-US

NeedsCompilation no

Author Alicja Gosiewska [aut, cre] (ORCID: <https://orcid.org/0000-0001-6563-5742>),
Przemyslaw Biecek [aut, ths] (ORCID: <https://orcid.org/0000-0001-8423-1823>),
Hubert Baniecki [aut] (ORCID: <https://orcid.org/0000-0001-6661-5364>),
Tomasz Mikołajczyk [aut],
Michal Burdukiewicz [ctb],
Szymon Maksymiuk [ctb]

Maintainer Alicja Gosiewska <alicjagosiewska@gmail.com>

Repository CRAN

Date/Publication 2023-10-30 15:40:07 UTC

Contents

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	15
plotD3_autocorrelation	16
plotD3_cooksdistance	17
plotD3_halfnormal	19
plotD3_lift	20
plotD3_prediction	21
plotD3_rec	23
plotD3_residual	24
plotD3_roc	26
plotD3_rroc	28
plotD3_scalelocation	29
plot_acf	31
plot_auditor	32
plot_autocorrelation	34
plot_cooksdistance	35
plot_correlation	36
plot_halfnormal	37
plot_lift	38
plot_pca	39
plot_prc	40
plot_prediction	42
plot_radar	43
plot_rec	44
plot_residual	45
plot_residual_boxplot	47
plot_residual_density	48
plot_rroc	49
plot_scalelocation	50
plot_tsecdf	52
print.auditor_model_cooksdistance	53
print.auditor_model_evaluation	54
print.auditor_model_halfnormal	55
print.auditor_model_performance	55
print.auditor_model_residual	56

print.auditor_score	57
score	57
score_acc	58
score_auc	59
score_auprc	60
score_cooksdistance	61
score_dw	62
score_f1	63
score_gini	64
score_halfnormal	65
score_mae	66
score_mse	67
score_one_minus_acc	68
score_one_minus_auc	69
score_one_minus_auprc	70
score_one_minus_f1	71
score_one_minus_gini	72
score_one_minus_precision	73
score_one_minus_recall	74
score_one_minus_specificity	75
score_peak	76
score_precision	77
score_r2	78
score_rec	79
score_recall	80
score_rmse	81
score_roc	82
score_runs	83
score_specificity	84

Index	85
--------------	-----------

audit	<i>Deprecated</i>
-------	-------------------

Description

The `audit()` function is deprecated, use [explain](#) from the DALEX package instead.

Usage

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

```

    predict_function = NULL,
    residual_function = NULL
  )

```

Arguments

<code>object</code>	An object containing a model or object of class explainer (see explain).
<code>data</code>	Data.frame or matrix - data that will be used by further validation functions. If not provided, will be extracted from the model.
<code>y</code>	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.
<code>predict.function</code>	Function that takes two arguments: model and data. It should return a numeric vector with predictions.
<code>residual.function</code>	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.
<code>label</code>	Character - the name of the model. By default it's extracted from the 'class' attribute of the model.
<code>predict_function</code>	Function that takes two arguments: model and data. It should return a numeric vector with predictions.
<code>residual_function</code>	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.

Value

An object of class explainer.

Examples

```

data(titanic_imputed, package = "DALEX")

model_glm <- glm(survived ~ ., family = binomial, data = titanic_imputed)
audit_glm <- audit(model_glm,
  data = titanic_imputed,
  y = titanic_imputed$survived)

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

```

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

auditorData	<i>Artificial auditorData</i>
-------------	-------------------------------

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.

Usage

```
data(auditorData)
```

Format

a data frame with 2000 rows and 5 columns

Examples

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

check_residuals	<i>Automated tests for model residuals</i>
-----------------	--------------------------------------------

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

```
dragons <- DALEX::dragons[1:100, ]
lm_model <- lm(life_length ~ ., data = dragons)
lm_audit <- audit(lm_model, data = dragons, y = dragons$life_length)
check_residuals(lm_audit)
## Not run:
library("randomForest")
rf_model <- randomForest(life_length ~ ., data = dragons)
rf_audit <- audit(rf_model, data = dragons, y = dragons$life_length)
check_residuals(rf_audit)

## 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

```
dragons <- DALEX::dragons[1:100, ]
lm_model <- lm(life_length ~ ., data = dragons)
lm_audit <- audit(lm_model, data = dragons, y = dragons$life_length)
check_residuals_autocorrelation(lm_audit)
```

check_residuals_outliers
Checks for outliers

Description

Outlier checks

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

```
dragons <- DALEX::dragons[1:100, ]  
lm_model <- lm(life_length ~ ., data = dragons)  
lm_audit <- audit(lm_model, data = dragons, y = dragons$life_length)  
check_residuals_outliers(lm_audit)
```

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 of 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)
```

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 explainer created with function [explain](#) from the DALEX package.

Value

An object of the class auditor_model_cooksdistance.

References

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

Examples

```
data(titanic_imputed, package = "DALEX")

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

# use DALEX package to wrap up a model into explainer
glm_audit <- audit(model_glm,
                  data = titanic_imputed,
```

```
      y = titanic_imputed$survived)

# validate a model with auditor
mc <- model_cooksdistance(glm_audit)
mc

plot(mc)
```

model_evaluation	<i>Create model evaluation explanation</i>
------------------	--------------------------------------------

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 the class `auditor_model_evaluation`.

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data= titanic_imputed,
                  y = titanic_imputed$survived)

# validate a model with auditor
me <- model_evaluation(glm_audit)
```

```
me  
plot(me)
```

model_halfnormal	<i>Create Halfnormal Explanation</i>
------------------	--------------------------------------

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

<code>object</code>	An object of class <code>explainer</code> created with function <code>explain</code> from the DALEX package.
<code>quant</code>	if TRUE values on axis are on quantile scale.
<code>...</code>	other parameters passed do <code>hnp</code> 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

```
data(titanic_imputed, package = "DALEX")  
  
# fit a model  
model_glm <- glm(survived ~ ., family = binomial, data = titanic_imputed)  
  
glm_audit <- audit(model_glm,  
                  data = titanic_imputed,  
                  y = titanic_imputed$survived)  
  
# validate a model with auditor  
mh <- model_halfnormal(glm_audit)  
mh
```

```
plot(mh)
```

```
model_performance      Create Model Performance Explanation
```

Description

Creates `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,
  data = NULL,
  ...
)
```

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

Arguments

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

```
data(titanic_imputed, package = "DALEX")

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

# use DALEX package to wrap up a model into explainer
glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# validate a model with auditor
library(auditor)
mp <- model_performance(glm_audit)
mp

plot(mp)
```

model_residual

Create Model Residuals Explanation

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

<code>object</code>	An object of class <code>explainer</code> created with function explain from the DALEX package.
<code>...</code>	other parameters

Value

An object of the class `auditor_model_residual`.

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

```
library(DALEX)

# fit a model
model_glm <- glm(m2.price ~ ., data = apartments)

glm_audit <- explain(model_glm,
                    data = apartments,
                    y = apartments$m2.price)

# validate a model with auditor
mr <- model_residual(glm_audit)
mr

plot(mr)
```

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")
```

Arguments

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

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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

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

hn_lm <- model_halfnormal(lm_audit)
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 <code>model_residual</code> function.
...	Other 'auditor_model_residual' objects to be plotted together.
variable	Name of variable to order residuals on a plot. If <code>variable = "_y_"</code> , the data is ordered by a vector of actual response (y parameter passed to the <code>explain</code> function). If <code>variable = "_y_hat_"</code> the data on the plot will be ordered by predicted response. If <code>variable = NULL</code> , 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# plot results
plotD3_acf(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length ~ ., data = dragons)
```

```
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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 default 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 background 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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 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

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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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.

Usage

```
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**

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

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

# plot results
plot_roc(eva_glm)
plot(eva_glm)

#add second model
model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic_imputed)
glm_audit_2 <- audit(model_glm_2,
                    data = titanic_imputed,
                    y = titanic_imputed$survived,
                    label = "glm2")
eva_glm_2 <- model_evaluation(glm_audit_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 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 background 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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

```
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 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.

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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# plot results
plotD3_residual(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
plotD3_residual(mr_lm, mr_rf)
```

plotD3_roc

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

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

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

Arguments

object	An object of class <code>auditor_model_evaluation</code> created with <code>model_evaluation</code> function.
...	Other <code>auditor_model_evaluation</code> objects to be plotted together.
nlabel	Number of cutoff points to show on the plot. Default is <code>NULL</code> .
scale_plot	Logical, indicates whenever the plot should scale with height. By default it's <code>FALSE</code> .

Value

a `r2d3` object

See Also

[plot_roc](#)

Examples

```
data(titanic_imputed, package = "DALEX")

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

# use DALEX package to wrap up a model into explainer
glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

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

# plot results
plot_roc(eva_glm)
plot(eva_glm)

#add second model
model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic_imputed)
glm_audit_2 <- audit(model_glm_2,
                   data = titanic_imputed,
                   y = titanic_imputed$survived,
                   label = "glm2")
eva_glm_2 <- model_evaluation(glm_audit_2)

plotD3_roc(eva_glm, eva_glm_2)
```

plotD3_rroc	<i>Regression Receiver Operating Characteristic (RROC) in D3 with r2d3 package.</i>
-------------	-------------------------------------------------------------------------------------

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 $frac{n^2$.

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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# plot results
plotD3_rroc(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.
variable	Name of variable to order residuals on a plot. If <code>variable = "_y_"</code> , the data is ordered by a vector of actual response (y parameter passed to the <code>explain</code> function). If <code>variable = "_y_hat_"</code> the data on the plot will be ordered by predicted response. If <code>variable = NULL</code> , 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.
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 <code>single_plot = FALSE</code> . 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor

```

```

mr_lm <- model_residual(lm_audit)

# 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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.
variable	Name of variable to order residuals on a plot. If <code>variable = "_y_"</code> , the data is ordered by a vector of actual response (<code>y</code> parameter passed to the <code>explain</code> function). If <code>variable = "_y_hat_"</code> the data on the plot will be ordered by predicted response. If <code>variable = NULL</code> , 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

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

```

```
plot_acf(mr_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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 auditor'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 additional 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', 'residual', 'residual_boxplot', 'residual_density', '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 <code>par(ask=)</code> .
grid	logical; if TRUE plots will be plotted on the grid.

Value

A ggplot object.

See Also

[plot_acf](#), [plot_autocorrelation](#), [plot_cooksdistance](#), [plot_halfnormal](#), [plot_residual_boxplot](#), [plot_lift](#), [plot_pca](#), [plot_radar](#), [plot_correlation](#), [plot_prediction](#), [plot_rec](#), [plot_residual_density](#), [plot_residual](#), [plot_roc](#), [plot_rroc](#), [plot_scalelocation](#), [plot_tsecdf](#)

Examples

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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

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

hn_lm <- model_halfnormal(lm_audit)
plot(hn_lm)

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)

mp_rf <- model_performance(rf_audit)
mp_lm <- model_performance(lm_audit)
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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.
variable	Name of variable to order residuals on a plot. If <code>variable="_y_"</code> , the data is ordered by a vector of actual response (y parameter passed to the <code>explain</code> 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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 <code>auditor_model_cooksdistance</code> created with <code>model_cooksdistance</code> function.
...	Other objects of class <code>auditor_model_cooksdistance</code> .
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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)
```

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

# 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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> 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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

library(randomForest)
```

```
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)

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

plot_halfnormal	<i>Half-Normal plot</i>
-----------------	-------------------------

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 <code>auditor_model_halfnormal</code> created with model_halfnormal function.
...	Other <code>auditor_model_halfnormal</code> 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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 <code>auditor_model_evaluation</code> created with <code>model_evaluation</code> function.
...	Other <code>auditor_model_evaluation</code> objects to be plotted together.
zeros	Logical. It makes the lines start from the $(0, 0)$ point. By default it's TRUE.

Value

A ggplot object.

See Also

[model_evaluation](#)

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

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

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

model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic_imputed)
glm_audit_2 <- audit(model_glm_2,
                   data = titanic_imputed,
                   y = titanic_imputed$survived,
                   label = "glm2")
eva_glm_2 <- model_evaluation(glm_audit_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, arrow_size = 2)
```

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

Arguments

object	An object of class <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.

scale	A logical value indicating whether the models residuals should be scaled before the analysis.
arrow_size	Width of the arrows.

Value

A ggplot object.

Examples

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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)

# plot results
plot_pca(mr_lm, mr_rf)
```

plot_prc

Precision-Recall Curve (PRC)

Description

Precision-Recall Curve summarize the trade-off between the true positive rate and the positive predictive value for a model. It is useful for measuring performance and comparing classifiers.

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

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

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

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

Arguments

object	An object of class <code>auditor_model_evaluation</code> created with <code>model_evaluation</code> function.
...	Other <code>auditor_model_evaluation</code> objects to be plotted together.
nlabel	Number of cutoff points to show on the plot. Default is <code>NULL</code> .

Value

A `ggplot` object.

A `ggplot` object.

See Also

[plot_rroc](#), [plot_rec](#)

Examples

```
library(DALEX)

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

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

# plot results
plot_prc(eva_glm)
plot(eva_glm)

#add second model
model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic_imputed)
glm_audit_2 <- audit(model_glm_2,
                   data = titanic_imputed,
                   y = titanic_imputed$survived,
                   label = "glm2")
eva_glm_2 <- model_evaluation(glm_audit_2)

plot_prc(eva_glm, eva_glm_2)
plot(eva_glm, eva_glm_2)

data(titanic_imputed, package = "DALEX")

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

# use DALEX package to wrap up a model into explainer
```

```

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

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

# plot results
plot_roc(eva_glm)
plot(eva_glm)

#add second model
model_glm_2 <- glm(survived ~ .-age, family = binomial, data = titanic_imputed)
glm_audit_2 <- audit(model_glm_2,
                   data = titanic_imputed,
                   y = titanic_imputed$survived,
                   label = "glm2")
eva_glm_2 <- model_evaluation(glm_audit_2)

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

```

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 <code>auditor_model_residual</code> .
...	Other <code>auditor_model_residual</code> objects to be plotted together.
variable	Name of variable to order residuals on a plot. If <code>variable="_y_"</code> , the data is ordered by a vector of actual response (y parameter passed to the explain function). If <code>variable = "_y_hat_"</code> the data on the plot will be ordered by predicted response. If <code>variable = NULL</code> , 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 <code>variable = "_y_"</code> (which is a default option) or when <code>variable</code> equals actual response variable.

Value

A ggplot2 object.

Examples

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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# 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)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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, ..., verbose = TRUE)
```

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

Arguments

object	An object of class <code>auditor_model_performance</code> created with <code>model_performance</code> function.
...	Other <code>auditor_model_performance</code> objects to be plotted together.
verbose	Logical, indicates whether values of scores should be printed.

Value

A ggplot object.

Examples

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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mp_rf <- model_performance(rf_audit)

# 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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> 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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# plot results
plot_residual(mr_lm)
```

```
plot(mr_lm, type = "residual")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.

Value

A ggplot object.

See Also

[plot_residual](#)

Examples

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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
mr_lm <- model_residual(lm_audit)
```

```

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

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
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 = "", show_rugs = TRUE)
```

```
plotResidualDensity(object, ..., variable = NULL)
```

Arguments

object	An object of class <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.
variable	Split plot by variable's factor level or median. If <code>variable="_y_"</code> , the plot will be split by actual response (y parameter passed to the <code>explain</code> function). If <code>variable = "_y_hat_"</code> the plot will be split by predicted response. If <code>variable = NULL</code> , the plot will be split by observation index. If <code>variable = ""</code> plot is not split (default option).
show_rugs	Adds rugs layer to the plot. By default it's TRUE

Value

A ggplot object.

See Also

[plot_residual](#)

Examples

```

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

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

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

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

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

```

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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> 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

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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

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

# plot results
plot_rrroc(mr_lm)
plot(mr_lm, type = "rrroc")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
plot_rrroc(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type="rrroc")
```

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 <code>auditor_model_residual</code> created with <code>model_residual</code> function.
...	Other <code>auditor_model_residual</code> objects to be plotted together.
variable	Name of variable to order residuals on a plot. If <code>variable = "_y_"</code> , the data is ordered by a vector of actual response (y parameter passed to the <code>explain</code> function). If <code>variable = "_y_hat_"</code> the data on the plot will be ordered by predicted response. If <code>variable = NULL</code> , 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)  
  
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)  
  
# validate a model with auditor  
mr_lm <- model_residual(lm_audit)  
  
# 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

<code>object</code>	An object of class 'auditor_model_residual' created with <code>model_residual</code> function.
<code>...</code>	Other modelAudit objects to be plotted together.
<code>scale_error</code>	A logical value indicating whether ECDF should be scaled by proportions of positive and negative proportions.
<code>outliers</code>	Number of outliers to be marked.
<code>residuals</code>	A logical value indicating whether residuals should be marked.
<code>reverse_y</code>	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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# validate a model with auditor
mr_lm <- model_residual(lm_audit)
plot_tsecdf(mr_lm)
plot(mr_lm, type="tsecdf")

library(randomForest)
model_rf <- randomForest(life_length~., data = dragons)
rf_audit <- audit(model_rf, data = dragons, y = dragons$life_length)
mr_rf <- model_residual(rf_audit)
plot_tsecdf(mr_lm, mr_rf, reverse_y = TRUE)

```

```
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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

```

```
# calculate score
model_cooksdistance(lm_audit)
```

```
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 <code>model_evaluation</code> function.
...	other parameters

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data= titanic_imputed,
                  y = titanic_imputed$survived)

# validate a model with auditor
model_evaluation(glm_audit)
```

```
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

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# validate a model with auditor
model_halfnormal(glm_audit)
```

```
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

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                   data = titanic_imputed,
                   y = titanic_imputed$survived)

# validate a model with auditor
model_performance(glm_audit)
```

```
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

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                   data = titanic_imputed,
                   y = titanic_imputed$survived)
```

```
# validate a model with auditor
model_residual(glm_audit)
```

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

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score(glm_audit, 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 <code>explain</code> from the DALEX package.
type	The score to be calculated. Possible values: <code>acc</code> , <code>auc</code> , <code>cookdistance</code> , <code>dw</code> , <code>f1</code> , <code>gini</code> , <code>halfnormal</code> , <code>mae</code> , <code>mse</code> , <code>peak</code> , <code>precision</code> , <code>r2</code> , <code>rec</code> , <code>recall</code> , <code>rmse</code> , <code>rroc</code> , <code>runs</code> , <code>specificity</code> , <code>one_minus_acc</code> , <code>one_minus_auc</code> , <code>one_minus_f1</code> , <code>one_minus_gini</code> , <code>one_minus_precision</code> , <code>one_minus_recall</code> , <code>one_minus_specificity</code> (for detailed description see functions in <code>see also</code> section).
data	New data that will be used to calculate the score. Pass <code>NULL</code> 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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score(lm_audit, type = 'mae')
```

score_acc

Accuracy

Description

Accuracy

Usage

```
score_acc(object, cutoff = 0.5, data = NULL, y = NULL, ...)
```

Arguments

object	An object of class explainer created with function <code>explain</code> from the DALEX package.
cutoff	Threshold value, which divides model predicted values (<code>y_hat</code>) 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class `auditor_score`.

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_acc(glm_audit)
```

score_auc	<i>Area Under ROC Curve (AUC)</i>
-----------	-----------------------------------

Description

Area Under Curve (AUC) for Receiver Operating Characteristic.

Usage

```
score_auc(object, data = NULL, y = NULL, ...)

scoreROC(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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

See Also

[plot_roc](#)

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_auc(glm_audit)
```

score_auprc

Area under precision-recall curve

Description

Area under precision-recall (AUPRC) curve.

Usage

```
score_auprc(object, data = NULL, y = 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

# create an explainer
glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_auprc(glm_audit)
```

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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_cooksdistance(lm_audit)
```

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, y = 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.
y	New y parameter will be used to calculate score.
...	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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_dw(lm_audit)
```

score_f1	<i>F1 Score</i>
----------	-----------------

Description

F1 Score

Usage

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

Arguments

object	An object of class explainer created with function explain from the DALEX package.
cutoff	Threshold value, which divides model predicted values (\hat{y}) 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class `auditor_score`.

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_f1(glm_audit)
```

score_gini	<i>Gini Coefficient</i>
------------	-------------------------

Description

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, y = NULL, ...)
```

Arguments

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

Value

An object of class `auditor_score`.

See Also

[plot_roc](#)

Examples

```

library(DALEX)

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

# create an explainer
exp_glm <- explain(model_glm,
                   data = titanic_imputed,
                   y = titanic_imputed$survived)

# calculate score
score_gini(exp_glm)

```

score_halfnormal	<i>Half-Normal Score</i>
------------------	--------------------------

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.
...	...

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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_halfnormal(lm_audit)
```

score_mae	<i>Mean Absolute Error</i>
-----------	----------------------------

Description

Mean Absolute Error.

Usage

```
score_mae(object, data = NULL, y = 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.
y	New y parameter will be used to calculate score.
...	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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_mae(lm_audit)

```

score_mse	<i>Mean Square Error</i>
-----------	--------------------------

Description

Mean Square Error.

Usage

```

score_mse(object, data = NULL, y = NULL, ...)

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.
y	New y parameter will be used to calculate score.
...	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)

lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_mse(lm_audit)
```

score_one_minus_acc *One minus accuracy*

Description

One minus accuracy

Usage

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

Arguments

object	An object of class explainer created with function explain from the DALEX package.
cutoff	Threshold 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

# create an explainer
glm_audit <- audit(model_glm,
```

```

      data = titanic_imputed,
      y = titanic_imputed$survived)

# calculate score
score_one_minus_acc(glm_audit)

```

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, y = 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```

data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_one_minus_auc(glm_audit)

```

score_one_minus_auprc *One Minus area under precision-recall curve*

Description

One Minus Area under precision-recall (AUPRC) curve.

Usage

```
score_one_minus_auprc(object, data = NULL, y = 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

# create an explainer
glm_audit <- audit(model_glm,
                   data = titanic_imputed,
                   y = titanic_imputed$survived)

# calculate score
score_one_minus_auprc(glm_audit)
```

score_one_minus_f1	<i>One Minus F1 Score</i>
--------------------	---------------------------

Description

One Minus F1 Score

Usage

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

Arguments

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

Value

An object of class `auditor_score`.

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_one_minus_f1(glm_audit)
```

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, y = 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_one_minus_gini(glm_audit)
```

`score_one_minus_precision`*One Minus Precision*

Description

One Minus Precision

Usage

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

Arguments

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

Value

An object of class `auditor_score`.

Examples

```
library(DALEX)

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

# create an explainer
exp_glm <- explain(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$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, y = NULL, ...)
```

Arguments

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

Value

An object of class `auditor_score`.

Examples

```
library(DALEX)

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

# create an explainer
exp_glm <- explain(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$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, y = NULL, ...)
```

Arguments

object	An object of class explainer created with function explain from the DALEX package.
cutoff	Threshold value, which divides model predicted values (\hat{y}) 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

# create an explainer
glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_one_minus_specificity(glm_audit)
```

score_peak	<i>Peak Score</i>
------------	-------------------

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, data = NULL, y = 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.
data	New data that will be used to calculate the score. Pass NULL if you want to use data from object.
y	New y parameter will be used to calculate score.
...	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  
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)  
  
# calculate score  
score_peak(lm_audit)
```

score_precision	<i>Precision</i>
-----------------	------------------

Description

Precision

Usage

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

Arguments

object	An object of class explainer created with function explain from the DALEX package.
cutoff	Threshold value, which divides model predicted values (\hat{y}) 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_precision(glm_audit)
```

`score_r2`*R-squared*

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, y = NULL, ...)
```

Arguments

<code>object</code>	An object of class explainer created with function explain from the DALEX package.
<code>data</code>	New data that will be used to calculate the score. Pass NULL if you want to use data from object.
<code>y</code>	New y parameter will be used to calculate score.
<code>...</code>	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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score with auditor
score_r2(lm_audit)
```

score_rec	<i>Area Over the Curve for REC Curves</i>
-----------	-------------------------------------------

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, y = 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

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_audit <- audit(lm_model, data = dragons, y = dragons$life_length)

# calculate score
score_rec(lm_audit)
```

score_recall	<i>Recall</i>
--------------	---------------

Description

Recall

Usage

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

Arguments

object	An object of class explainer created with function explain from the DALEX package.
cutoff	Threshold value, which divides model predicted values (\hat{y}) 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class `auditor_score`.

Examples

```
data(titanic_imputed, package = "DALEX")

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

# create an explainer
glm_audit <- audit(model_glm,
                  data = titanic_imputed,
                  y = titanic_imputed$survived)

# calculate score
score_recall(glm_audit)
```

score_rmse	<i>Root Mean Square Error</i>
------------	-------------------------------

Description

Root Mean Square Error.

Usage

```
score_rmse(object, data = NULL, y = NULL, ...)  
  
scoreRMSE(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.
y	New y parameter will be used to calculate score.
...	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  
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)  
  
# calculate score  
score_rmse(lm_audit)
```

score_rroc	<i>Area Over the Curve for RROC Curves</i>
------------	--------------------------------------------

Description

The area over the Regression Receiver Operating Characteristic.

Usage

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

```
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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_rroc(lm_audit)
```

 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, data = NULL, y = NULL, ...)
```

```
scoreRuns(object, variable = NULL)
```

Arguments

object	An object of class explainer created with function <code>explain</code> 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.
y	New y parameter will be used to calculate score.
...	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
lm_audit <- audit(model_lm, data = dragons, y = dragons$life_length)

# calculate score
score_runs(lm_audit)
```

score_specificity *Specificity*

Description

Specificity

Usage

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

Arguments

object	An object of class explainer created with function explain from the DALEX package.
cutoff	Threshold value, which divides model predicted values (\hat{y}) 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.
y	New y parameter will be used to calculate score.
...	Other arguments dependent on the type of score.

Value

An object of class auditor_score.

Examples

```
data(titanic_imputed, package = "DALEX")

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

exp_glm <- audit(model_glm,
                 data = titanic_imputed,
                 y = titanic_imputed$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, 22, 25, 30, 31, 34, 42, 46, 48, 51, 58–61, 63–84

- gtable, 36

- hnp, 10

- model_cooksdistance, 8, 13, 14, 18, 32, 35, 53
- model_evaluation, 9, 13, 14, 20, 27, 32, 38, 41, 54
- model_halfnormal, 10, 14, 19, 20, 32, 37, 55
- model_performance, 11, 13, 14, 32, 43, 56
- model_residual, 12, 13–16, 23, 25, 28, 30–32, 34, 36, 39, 44, 46–49, 51, 52, 56
- modelEvaluation (model_evaluation), 9
- modelFit (model_halfnormal), 10
- modelPerformance (model_performance), 11
- modelResiduals (model_residual), 12

- observationInfluence (model_cooksdistance), 8

- par, 33
- plot.auditor_model_cooksdistance (plot_auditor), 32
- plot.auditor_model_evaluation (plot_auditor), 32
- plot.auditor_model_halfnormal (plot_auditor), 32
- plot.auditor_model_performance (plot_auditor), 32

- plot.auditor_model_residual (plot_auditor), 32
- plot_acf, 13, 31, 33
- plot_auditor, 32
- plot_autocorrelation, 13, 33, 34
- plot_cooksdistance, 18, 33, 35
- plot_correlation, 13, 33, 36
- plot_halfnormal, 20, 33, 37
- plot_lift, 9, 21, 33, 38
- plot_pca, 13, 33, 39
- plot_prc, 40
- plot_prediction, 13, 23, 33, 42
- plot_radar, 33, 43
- plot_rec, 13, 24, 33, 41, 44, 50, 79
- plot_residual, 13, 26, 33, 45, 47, 48
- plot_residual_boxplot, 13, 33, 47
- plot_residual_density, 13, 33, 48
- plot_roc, 9, 27, 33, 45, 50, 60, 64
- plot_roc (plot_prc), 40
- plot_rrroc, 13, 33, 41, 45, 49, 82
- plot_scalelocation, 13, 30, 33, 50
- plot_tsecdf, 13, 33, 52
- plotACF (plot_acf), 31
- plotAutocorrelation (plot_autocorrelation), 34
- plotCooksDistance (plot_cooksdistance), 35
- plotD3, 13
- plotD3_acf, 14, 15
- plotD3_auditor (plotD3), 13
- plotD3_autocorrelation, 14, 16
- plotD3_cooksdistance, 14, 17
- plotD3_halfnormal, 14, 19
- plotD3_lift, 14, 20
- plotD3_prediction, 14, 21
- plotD3_rec, 14, 23
- plotD3_residual, 14, 24
- plotD3_roc, 14, 26
- plotD3_rrroc, 14, 28, 28

plotD3_scalelocation, [14](#), [29](#)
 plotD3ACF (plotD3_acf), [15](#)
 plotD3Autocorrelation
 (plotD3_autocorrelation), [16](#)
 plotD3CooksDistance
 (plotD3_cooksdistance), [17](#)
 plotD3HalfNormal (plotD3_halfnormal), [19](#)
 plotD3LIFT (plotD3_lift), [20](#)
 plotD3Prediction (plotD3_prediction), [21](#)
 plotD3REC (plotD3_rec), [23](#)
 plotD3Residual (plotD3_residual), [24](#)
 plotD3ScaleLocation
 (plotD3_scalelocation), [29](#)
 plotHalfNormal (plot_halfnormal), [37](#)
 plotLIFT (plot_lift), [38](#)
 plotModelCorrelation
 (plot_correlation), [36](#)
 plotModelPCA (plot_pca), [39](#)
 plotModelRanking (plot_radar), [43](#)
 plotPrediction (plot_prediction), [42](#)
 plotREC (plot_rec), [44](#)
 plotResidual (plot_residual), [45](#)
 plotResidualBoxplot
 (plot_residual_boxplot), [47](#)
 plotResidualDensity
 (plot_residual_density), [48](#)
 plotROC (plot_prc), [40](#)
 plotRROC (plot_rroc), [49](#)
 plotScaleLocation (plot_scalelocation),
 [50](#)
 plotTwoSidedECDF (plot_tsecdf), [52](#)
 print.auditor_model_cooksdistance, [53](#)
 print.auditor_model_evaluation, [54](#)
 print.auditor_model_halfnormal, [55](#)
 print.auditor_model_performance, [55](#)
 print.auditor_model_residual, [56](#)
 print.auditor_score, [57](#)

 score, [57](#), [57](#), [62](#), [66](#), [67](#), [78](#), [81](#)
 score_acc, [12](#), [58](#), [58](#)
 score_auc, [12](#), [58](#), [59](#)
 score_auprc, [60](#)
 score_cooksdistance, [12](#), [58](#), [61](#)
 score_dw, [12](#), [58](#), [62](#)
 score_f1, [12](#), [58](#), [63](#)
 score_gini, [12](#), [58](#), [64](#)
 score_halfnormal, [12](#), [20](#), [37](#), [58](#), [65](#)
 score_mae, [12](#), [58](#), [66](#)
 score_mse, [12](#), [58](#), [67](#)

 score_one_minus_acc, [12](#), [58](#), [68](#)
 score_one_minus_auc, [12](#), [58](#), [69](#)
 score_one_minus_auprc, [70](#)
 score_one_minus_f1, [12](#), [58](#), [71](#)
 score_one_minus_gini, [12](#), [58](#), [72](#)
 score_one_minus_precision, [12](#), [58](#), [73](#)
 score_one_minus_recall, [12](#), [58](#), [74](#)
 score_one_minus_specificity, [12](#), [58](#), [75](#)
 score_peak, [12](#), [58](#), [76](#)
 score_precision, [12](#), [58](#), [77](#)
 score_r2, [12](#), [58](#), [78](#)
 score_rec, [12](#), [58](#), [79](#)
 score_recall, [12](#), [58](#), [80](#)
 score_rmse, [12](#), [58](#), [81](#)
 score_rroc, [12](#), [58](#), [82](#)
 score_runs, [12](#), [58](#), [83](#)
 score_specificity, [12](#), [58](#), [84](#)
 scoreCooksDistance
 (score_cooksdistance), [61](#)
 scoreDW (score_dw), [62](#)
 scoreHalfNormal (score_halfnormal), [65](#)
 scoreMAE (score_mae), [66](#)
 scoreMSE (score_mse), [67](#)
 scorePeak (score_peak), [76](#)
 scoreREC (score_rec), [79](#)
 scoreRMSE (score_rmse), [81](#)
 scoreROC (score_auc), [59](#)
 scoreRROC (score_rroc), [82](#)
 scoreRuns (score_runs), [83](#)