Package ‘auditor’

October 12, 2022

Title  Model Audit - Verification, Validation, and Error Analysis

Version  1.3.3

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  7.1.1

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] (<https://orcid.org/0000-0001-6563-5742>), Przemyslaw Biecek [aut, ths] (<https://orcid.org/0000-0001-8423-1823>), Hubert Baniecki [aut] (<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  2021-07-26 18:40:04 UTC
R topics documented:

audit ................................................. 3
auditorData ......................................... 5
check_residuals ..................................... 5
check_residuals_autocorrelation .................. 6
check_residuals_outliers ......................... 7
check_residuals_trend ............................ 7
model_cooksdistance ............................. 8
model_evaluation ................................... 9
model_halfnormal .................................. 10
model_performance ............................... 11
model_residual .................................... 12
plotD3 .............................................. 13
plotD3_acf ........................................ 15
plotD3_autocorrelation ............................ 16
plotD3_cooksdistance ............................. 18
plotD3_halfnormal ................................ 19
plotD3_lift ........................................ 20
plotD3_prediction ................................. 22
plotD3_rec ........................................ 24
plotD3_residual ................................... 25
plotD3_roc ......................................... 27
plotD3_rroc ....................................... 28
plotD3_scalelocation .............................. 30
plot_acf ........................................... 31
plot_auditor ....................................... 32
plot_autocorrelation ............................. 34
plot_cooksdistance ............................... 35
plot_correlation .................................. 36
plot_halfnormal .................................. 37
plot_lift .......................................... 38
plot_pca .......................................... 40
plot_prc .......................................... 41
plot_prediction ................................... 42
plot_radar ........................................ 44
plot_rec .......................................... 45
plot_residual ..................................... 46
plot_residual_boxplot ............................ 47
plot_residual_density ........................... 48
plot_rroc ......................................... 50
plot_scalelocation ............................... 51
plot_tsecdf ...................................... 52
print.auditor_model_cooksdistance ............ 54
print.auditor_model_evaluation ................. 54
print.auditor_model_halfnormal ................. 55
print.auditor_model_performance ............... 56
print.auditor_model_residual .................... 57
The `audit()` function is deprecated, use `explain` from the DALEX package instead.
audit

predict_function = NULL,
residual_function = NULL
)

Arguments

object      An object containing a model or object of class explainer (see explain).
data        Data.frame or matrix - data that will be used by further validation functions. If not provided, will be extracted from the model.
y          Response vector that will be used by further validation functions. Some functions may require an integer vector containing binary labels with values 0,1. If not provided, will be extracted from the model.

predict.function  Function that takes two arguments: model and data. It should return a numeric vector with predictions.
residual.function  Function that takes three arguments: model, data and response vector. It should return a numeric vector with model residuals for given data. If not provided, response residuals \((y - \hat{y})\) are calculated.

label      Character - the name of the model. By default it’s extracted from the ‘class’ attribute of the model.

predict.function  Function that takes two arguments: model and data. It should return a numeric vector with predictions.
residual.function  Function that takes three arguments: model, data and response vector. It should return a numeric vector with model residuals for given data. If not provided, response residuals \((y - \hat{y})\) are calculated.

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**

*Artificial auditorData*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(auditorData)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>a data frame with 2000 rows and 5 columns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
</table>
| data("auditorData", package = "auditor")
head(auditorData) |

---

**check_residuals**

*Automated tests for model residuals*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>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)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>check_residuals(object, ...)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>list with statistics for particular checks</td>
</tr>
</tbody>
</table>
check_residuals_autocorrelation

Checks for autocorrelation in target variable or in residuals

**Description**

Checks for autocorrelation in target variable or in residuals

**Usage**

```r
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**

```r
dragons <- DALEX::dragons[1:100,]
lm_model <- lm(life_length ~ ., data = dragons)
lm_audit <- audit(lm_model, data = dragons, y = dragons$l life_length)
check_residuals(lm_audit)
## Not run:
library("randomForest")
rfe_model <- randomForest(life_length ~ ., data = dragons)
rfe_audit <- audit(rfe_model, data = dragons, y = dragons$l life_length)
check_residuals(rfe_audit)
## End(Not run)
```
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 Calculates loess fit for residuals and then extracts statistics that shows how far is this fit from one without trend

Description
Checks for trend in residuals
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)
model_cooksdistance

**Arguments**

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

- **B**: number of samplings

**Value**

standardized loess fit for residuals

**Examples**

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

An object of the class `auditor_model_cooksdistance`.

**References**

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

Create model evaluation explanation

Description

Creates explanation of classification model.

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

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

Usage

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

```r
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`  
*Create Halfnormal Explanation*

**Description**

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

**Usage**

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

**Arguments**

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

- **quant**  
  if TRUE values on axis are on quantile scale.

- **...**  
  other parameters passed do `hnp` function.

**Value**

An object of the class auditor_model_halfnormal.

**References**

Examples

```r
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)

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**

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

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

**Arguments**

- **object**: An object of class `explainer` created with function `explain` from the DALEX package.
- **score**: Vector of score names to be calculated. Possible values: `acc, auc, cookdistance, dw, f1, gini, halfnormal, mae, mse, peak, precision, r2, rec, recall, rmse, rroc, runs, specificity, one_minus_acc, one_minus_auc, one_minus_f1`. 

---
model_residual

one_minus_gini, one_minus_precision, one_minus_recall, one_minus_specificity
(for detailed description see functions in see also section). Pass NULL if you want
to use only custom scores by new_score parameter.

new_score A named list of functions that take one argument: object of class `explainer’ and
return a numeric value. The measure calculated by the function should have the
property that lower score value indicates better model.

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

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)
Usage

model_residual(object, ...)  
modelResiduals(object, ...)

Arguments

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

```r
plotD3(x, ...)
```

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

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

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

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

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

Arguments

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

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

```r
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(lmAudit)
plotD3(hn_lm)

---

plotD3.acf  
Plot Autocorrelation Function in D3 with r2d3 package.

Description
Plot Autocorrelation Function of models’ residuals.

Usage
plotD3.acf(object, ..., variable = NULL, alpha = 0.95, scale_plot = FALSE)
plotD3ACF(object, ..., variable = NULL, alpha = 0.95, scale_plot = FALSE)

Arguments
- **object**: An object of class ‘auditor_model_residual’ created with `model_residual` function.
- **...**: Other ‘auditor_model_residual’ objects to be plotted together.
- **variable**: Name of variable to order residuals on a plot. If `variable = "_y_"`, the data is ordered by a vector of actual response (y parameter passed to the `explain` function). If `variable = "_y_hat_"` the data on the plot will be ordered by predicted response. If `variable = NULL`, unordered observations are presented.
- **alpha**: Confidence level of the interval.
- **scale_plot**: Logical, indicates whenever the plot should scale with height. By default it’s FALSE.

Value
a ‘r2d3’ object.

Examples

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

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

## Description

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

## Usage

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

```r
plotD3Autocorrelation(
  object,
  ..., 
  variable = NULL,
  points = TRUE,
  smooth = FALSE,
  point_count = NULL,
  single_plot = TRUE,
  scale_plot = FALSE,
  background = FALSE
)
```
plotD3_autocorrelation

Arguments

object  
An object of class ‘auditor_model_residual’ created with model_residual

...  
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)
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


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 classificators.

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 is a function that plots predicted values observed or variable values in the model.

**Description**

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

**Usage**

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

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

**Arguments**

- `object`: An object of class `auditor_model_residual`.
- `...`: Other modelAudit or modelResiduals objects to be plotted together.
- `variable`: Name of variable to order residuals on a plot. If `variable="_y_"`, the data is ordered by a vector of actual response (y parameter passed to the `explain` function). If `variable="_y_hat_"` the data on the plot will be ordered by predicted response. If `variable=NULL`, unordered observations are presented.
- `points`: Logical, indicates whenever observations should be added as points. By default it's `TRUE`. 
smooth  Logical, indicates whenever smoothed lines should be added. By default it’s FALSE.

abline  Logical, indicates whenever function y = x should be added. Works only with variable = NULL which is a default option.

point_count  Number of points to be plotted per model. Points will be chosen randomly. By default plot all of them.

single_plot  Logical, indicates whenever single or facets should be plotted. By default it’s TRUE.

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

background  Logical, available only if single_plot = FALSE. Indicates whenever 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)

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


See Also

plot_rec
plotD3_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)
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 auditor_model_evaluation created with `model_evaluation` function.
- **...**: Other auditor_model_evaluation objects to be plotted together.
- **nlabel**: Number of cutoff points to show on the plot. Default is NULL.
- **scale_plot**: Logical, indicates whenever the plot should scale with height. By default it’s FALSE.

### Value

a r2d3 object

### See Also

plot_roc
Examples

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_roc**

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

---

**Description**

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

**Usage**

plotD3_roc(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


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
)

Arguments

object  An object of class auditor_model_residual created with model_residual function.

...  Other auditor_model_residual objects to be plotted together.

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

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

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 single_plot = FALSE. Indicates whenever background plots should be plotted. By default it’s FALSE.

Value

a r2d3 object

See Also

plot_scalelocation

Examples

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

Arguments

object

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

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

variable

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

alpha

Confidence level of the interval.

Value

A ggplot object.

Examples

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

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

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**

**Usage**

```r
plot_auditor(x, ..., type = "residual", ask = TRUE, grid = TRUE)
```

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

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

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

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

```r
## 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 `par(ask=)`.

- **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 auditor_model_residual created with model_residual function.

- **...**
  Other auditor_model_residual objects to be plotted together.

- **variable**
  Name of variable to order residuals on a plot. If variable="_y_", the data is ordered by a vector of actual response (y parameter passed to the explain function).

- **smooth**
  Logical, if TRUE smooth line will be added.
**plot_cooksdistance**

Influence of Observations Plot

**Description**

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

**Usage**

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

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

**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.

**Examples**

```r
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)
```
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


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")

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")
plot_halfnormal

Arguments

object  An object of class auditor_model_residual created with model_residual function.

... Other auditor_model_residual objects to be plotted together.

values "fit" for model fitted values or "res" for residual values.

Value

Invisibly returns a gtable object.

Examples

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

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

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

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

plot_halfnormal  Half-Normal plot

Description

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

Usage

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

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

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

Value

A ggplot object.

See Also

- `model_halfnormal`
- `score_halfnormal`

Examples

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

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

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

```r
plot_lift(object, ..., zeros = TRUE)
plotLIFT(object, ...)
```
Arguments

object  An object of class auditor_model_evaluation created with `model_evaluation` function.

...  Other auditor_model_evaluation objects to be plotted together.

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")
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 auditor_model_residual created with model_residual function.

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

scale: A logical value indicating whether the models residuals should be scaled before the analysis.

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$lifetime_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$lifetime_length)

mr_rf <- model_residual(rf_audit)

# plot results
plot_pca(mr_lm, mr_rf)
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 classificators.

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 classificators.

Usage

plot_prc(object, ..., nlabel = NULL)
plot_roc(object, ..., nlabel = NULL)
plotROC(object, ..., nlabel = NULL)

Arguments

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

Value

A ggplot object.
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
plot_prediction

<table>
<thead>
<tr>
<th>Predicted response vs Observed or Variable Values</th>
</tr>
</thead>
</table>

**Description**

Plot of predicted response vs observed or variable Values.
plot_prediction

Usage

plot_prediction(object, ..., variable = "_y_", smooth = FALSE, abline = FALSE)

plotPrediction(object, ..., variable = NULL, smooth = FALSE, abline = FALSE)

Arguments

object An object of class auditor_model_residual.

... Other auditor_model_residual objects to be plotted together.

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

smooth Logical, indicates whenever smooth line should be added.

abline Logical, indicates whenever function y = x should be added. Works only with variable="_y_" (which is a default option) or when variable equals actual response variable.

Value

A ggplot2 object.

Examples

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)
### Description

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

### Usage

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

### Arguments

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

### Value

A ggplot object.

### Examples

```r
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)
```
Regression Error Characteristic Curves (REC)

Description

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

Usage

plot_rec(object, ...)
plotREC(object, ...)

Arguments

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

Details

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

Value

A ggplot object.

References


See Also

plot_roc, plot_rroc

Examples

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

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

# use DALEX package to wrap up a model into explainer
lm_audit <- audit(model_lm, data = dragons, y = dragons$lifetime_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**

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

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

**Arguments**

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

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

std_residuals  Logical, indicates whenever standardized residuals should be used.

nlabel  Number of observations with the biggest absolute values of residuals to be labeled.

Examples

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

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

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")
Arguments

object An object of class auditor_model_residual created with model_residual function.

... Other auditor_model_residual 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 auditor_model_residual created with model_residual function.

... Other auditor_model_residual objects to be plotted together.

variable Split plot by variable’s factor level or median. If variable="_y_", the plot will be split by actual response (y parameter passed to the explain function). If variable = "_y_hat_" the plot will be split by predicted response. If variable = NULL, the plot will be split by observation index If variable = "" 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")
Regression Receiver Operating Characteristic (RROC)

Description

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

Usage

```r
plot_rroc(object, 
```
```
plotRROC(object, 
```

Arguments

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

Details

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

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

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

The shift equals 0 is represented by a dot.

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

Value

A ggplot object.

References


See Also

- `plot_roc`, `plot_rec`
Examples

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

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

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_rroc(mr_lm)
plot(mr_lm, type = "rroc")

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_rroc(mr_lm, mr_rf)
plot(mr_lm, mr_rf, type="rroc"")

plot_scalelocation

Scale location plot

Description

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

Usage

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

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

Arguments

object An object of class auditor_model_residual created with model_residual function.

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

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

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

Value

A ggplot object.

Examples

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

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

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

object An object of class 'auditor_model_residual' created with model_residual function.
... Other modelAudit objects to be plotted together.
scale_error A logical value indicating whether ECDF should be scaled by proportions of positive and negative proportions.
outliers Number of outliers to be marked.
residuals A logical value indicating whether residuals should be marked.
reverse_y A logical value indicating whether values on y axis should be reversed.

Value

A ggplot object.

Examples

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

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

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 \texttt{model_cooksdistance} function.

...
other parameters

Examples

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

# fit a model
model_lm <- \texttt{lm(life\_length \sim ., data = dragons)}

# create an explainer
lm_audit <- \texttt{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, ...)


print.auditor_model_halfnormal

Arguments

x  an object auditor_model_evaluation created with model_evaluation 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)

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_evaluation(glm_audit)
```r
    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

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

### Arguments

- `x`:
  - An object `auditor_model_performance` created with `model_performance` function.
- `...`:
  - Other parameters

### Examples

```r
  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,
                     data = titanic_imputed,
                     y = titanic_imputed$survived)

  # validate a model with auditor
  model_performance(glm_audit)
```
print.auditor_model_residual

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

Description
Prints of Models Scores

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

score

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

Value

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

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')

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

Description

Accuracy

Usage

score_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 (y_hat) to calculate confusion matrix. By default it's 0.5.
data New data that will be used to calculate the score. Pass NULL if you want to use data from object.
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)


<table>
<thead>
<tr>
<th>score_auc</th>
<th>Area Under ROC Curve (AUC)</th>
</tr>
</thead>
</table>

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$survived)
y = titanic_imputed$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

score_cooksdistance

score_auprc
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          F1 Score

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 (y_hat) to calculate confusion matrix. By default it's 0.5.
data           New data that will be used to calculate the score. Pass NULL if you want to use data from object.
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,
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 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

```r
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
cscore_gini(exp_glm)

---

**Score Half-Normal**

**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**

```r
score_halfnormal(object, ...)
```

```r
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**

```r
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)
```
**Description**

Mean Absolute Error.

**Usage**

```r
score_mae(object, data = NULL, y = NULL, ...)
```

```r
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**

```r
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

| score_mse | Mean Square Error |

**Description**

Mean Square Error.

**Usage**

```r
cscore_mse(object, data = NULL, y = NULL, ...)
cscoreMSE(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**

```r
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 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**

```r
# 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

```r
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

```r
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**

```r
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)
```
One Minus F1 Score

Usage

\[ \text{score\_one\_minus\_f1}(\text{object}, \text{cutoff} = 0.5, \text{data} = \text{NULL}, \text{y} = \text{NULL}, \ldots) \]

Arguments

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

```r
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

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

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

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

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

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

```r
score_one_minus_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 (y_hat) to calculate confusion matrix. By default it's 0.5.

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

- **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

```r
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**

```r
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 (y_hat) to calculate confusion matrix. By default it’s 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **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**

```r
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)
```
**Description**

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

**Usage**

```r
score_peak(object, variable = NULL, data = NULL, 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**

```r
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)
```
**Description**

Precision

**Usage**

```r
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 (y_hat) to calculate confusion matrix. By default it's 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **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**

```r
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)
```
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

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)

# 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)
Area Over the Curve for REC Curves

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

Usage
score_rec(object, data = NULL, 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

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)
Description
Recall

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

Arguments

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

Value
An object of class auditor_score.

Examples

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

Root Mean Square Error

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**

*Area Over the Curve for RROC Curves*

**Description**

The area over the Regression Receiver Operating Characteristic.

**Usage**

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

```r
scoreRROC(object)
```

**Arguments**

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


**See Also**

- `plot_rroc`

**Examples**

```r
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$lifetime_length)

# calculate score
score_rroc(lm_audit)
```
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

\[
\text{score\_runs}(\text{object}, \text{variable} = \text{NULL}, \text{data} = \text{NULL}, y = \text{NULL}, \ldots)
\]

\[
\text{scoreRuns}(\text{object}, \text{variable} = \text{NULL})
\]

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

```r

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

---

**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 (y_hat) to calculate confusion matrix. By default it’s 0.5.
- **data**: New data that will be used to calculate the score. Pass NULL if you want to use data from object.
- **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**
```r
data(titanic_imputed, package = "DALEX")

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

exp_glm <- audit(model_glm, 
                 family = binomial, data = titanic_imputed)

# 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, 3–11, 13, 15, 17, 22, 26, 30, 32, 34, 43, 47, 49, 52, 58–85
gtable, 37
hnp, 10
model_cooksdistance, 8, 13, 14, 18, 32, 33, 35, 54
model_evaluation, 9, 13, 14, 21, 27, 32, 33, 39, 41, 55
model_halfnormal, 10, 14, 20, 33, 38, 55
model_performance, 11, 13, 14, 32, 33, 44, 56
model_residual, 12, 13–15, 17, 24, 26, 28, 30, 32–34, 37, 40, 45, 46, 48–51, 53, 57
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, 19, 33, 35
plot_correlation, 13, 33, 36
plot_halfnormal, 20, 33, 37
plot_lift, 9, 21, 33, 38
plot_pca, 13, 33, 40
plot_prc, 41
plot_prediction, 13, 23, 33, 42
plot_radar, 33, 44
plot_rec, 13, 24, 33, 41, 45, 50, 80
plot_residual, 13, 26, 33, 46, 48, 49
plot_residual_boxplot, 13, 33, 47
plot_residual_density, 13, 33, 48
plot_roc, 9, 27, 33, 45, 50, 60, 65
plot_roc (plot_prc), 41
plot_rroc, 13, 33, 41, 45, 50, 83
plot_scalelocation, 13, 31, 33, 51
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, 18
plotD3_halfnormal, 14, 19
plotD3_lift, 14, 20
plotD3_prediction, 14, 22
plotD3_rec, 14, 24
plotD3_residual, 14, 25
plotD3_roc, 14, 27
plotD3_rroc, 14, 28, 29
plotD3_scalelocation, 14, 30
plotD3_ACF (plotD3_acf), 15
plotD3_Autocorrelation
   (plotD3_autocorrelation), 16
plotD3_CooksDistance
   (plotD3_cooksdistance), 18
plotD3_HalfNormal (plotD3_halfnormal), 19
plotD3_LIFT (plotD3_lift), 20
plotD3_Prediction (plotD3_prediction), 22
plotD3_REC (plotD3_rec), 24
plotD3_Residual (plotD3_residual), 25
plotD3_ScaleLocation
   (plotD3_scalelocation), 30
plot_HalfNormal (plot_halfnormal), 37
plot_LIFT (plot_lift), 38
plot_ModelCorrelation
   (plot_correlation), 36
plot_ModelPCA (plot_pca), 40
plot_ModelRanking (plot_radar), 44
plot_Prediction (plot_prediction), 42
plot_REC (plot_rec), 45
plot_Residual (plot_residual), 46
plot_Residual_Boxplot
   (plot_residual_boxplot), 47
plot_Residual_Density
   (plot_residual_density), 48
plot_ROC (plot_prc), 41
plot_RROC (plot_rroc), 50
plot_ScaleLocation (plot_scalelocation), 51
plot_TwoSided_ECDF (plot_tsecdf), 52
print.auditor_model_cooksdistance, 54
print.auditor_model_evaluation, 54
print.auditor_model_halfnormal, 55
print.auditor_model_performance, 56
print.auditor_model_residual, 57
print.auditor_score, 57
score, 58, 58, 62, 67, 68, 79, 82
score_acc, 12, 59, 59
score_auc, 12, 59, 60
score_auprc, 61
score_cooksdistance, 12, 59, 62
score_dw, 12, 59, 63
score_f1, 12, 59, 64
score_gini, 12, 59, 65
score_halfnormal, 12, 20, 38, 59, 66
score_mae, 12, 59, 67
score_mse, 12, 59, 68