Package ‘autostats’

August 19, 2022

Type Package
Title Auto Stats
Version 0.3.1
Maintainer Harrison Tietze <harrison4192@gmail.com>
Description Automatically do statistical exploration. Create formulas using 'tidyselect' syntax, and then determine cross-validated model accuracy and variable contributions using 'glm' and 'xgboost'. Contains additional helper functions to create and modify formulas. Has a flagship function to quickly determine relationships between categorical and continuous variables in the data set.
Encoding UTF-8
Imports dplyr, stringr, tidyselect, purrr, janitor, tibble, rlang, stats, rlist, broom, magrittr, ggeasy, ggplot2, jtools, gtools, ggthemes, patchwork, tidyr, xgboost, flextable, parsnip, recipes, rsample, tune, workflows, forcats, framecleaner, presenter, yardstick, dials, readr, lubridate, party, data.table, FOCI, XICOR, agtboost, nnet, recosystem, doParallel
RoxygenNote 7.2.1
BugReports https://github.com/Harrison4192/autostats/issues
Suggests knitr, rmarkdown, parallel, igraph, moreparty, broom.mixed, hardhat, glmnet, Ckmeans.1d.dp, ggstance, Matrix, BBmisc
VignetteBuilder knitr
License MIT + file LICENSE
NeedsCompilation no
Author Harrison Tietze [aut, cre]
Depends R (>= 3.5.0)
Repository CRAN
Date/Publication 2022-08-19 13:10:06 UTC
Description

A wrapper around `lm` and `anova` to run a regression of a continuous variable against categorical variables. Used for determining the whether the mean of a continuous variable is statistically significant amongst different levels of a categorical variable.

Usage

```r
auto_anova(
  data,
  
  ..., 
  baseline = c("mean", "median", "first_level", "user_supplied"),
  user_supplied_baseline = NULL,
  sparse = FALSE,
  pval_thresh = 0.1
)
```
Arguments

- **data**: a data frame
- **...**: tidyselect specification or cols
- **baseline**: choose from "mean", "median", "first_level", "user_supplied". what is the baseline to compare each category to? can use the mean and median of the target variable as a global baseline
- **user_supplied_baseline**: if intercept is "user_supplied", can enter a numeric value
- **sparse**: default FALSE; if true returns a truncated output with only significant results
- **pval_thresh**: control significance level for sparse output filtering

Details

Columns can be inputted as unquoted names or tidyselect. Continuous and categorical variables are automatically determined. If no character or factor column is present, the column with the lowest amount of unique values will be considered the categorical variable.

Description of columns in the output

- **target**: continuous variables
- **predictor**: categorical variables
- **level**: levels in the categorical variables
- **estimate**: difference between level target mean and baseline
- **target_mean**: target mean per level
- **n**: rows in predictor level
- **std.error**: standard error of target in predictor level
- **level_p.value**: p.value for t.test of whether target mean differs significantly between level and baseline
- **level_significance**: level p.value represented by stars
- **predictor_p.value**: p.value for significance of entire predictor given by F test
- **predictor_significance**: predictor p.value represented by stars
- **conclusion**: text interpretation of tests

Value

data frame

Examples

```r
iris %>%
auto_anova(tidyselect::everything()) -> iris_anova

iris_anova %>%
print(width = Inf)
```
auto_boxplot

Description

Wraps `geom_boxplot` to simplify creating boxplots.

Usage

```r
auto_boxplot(
  .data,
  continuous_outcome,
  categorical_variable,
  categorical_facets = NULL,
  alpha = 0.3,
  width = 0.15,
  color_dots = "black",
  color_box = "red"
)
```

Arguments

- `.data` data
- `continuous_outcome` continuous y variable. unquoted column name
- `categorical_variable` categorical x variable. unquoted column name
- `categorical_facets` categorical facet variable. unquoted column name
- `alpha` alpha points
- `width` width of jitter
- `color_dots` dot color
- `color_box` box color

Value

`ggplot`

Examples

```r
iris %>%
auto_boxplot(continuous_outcome = Petal.Width, categorical_variable = Species)
```
Description

Finds the correlation between numeric variables in a data frame, chosen using tidyselect. Additional parameters for the correlation test can be specified as in `cor.test`.

Usage

```r
auto_cor(
  .data, 
  ..., 
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything", "na.or.complete"), 
  method = c("pearson", "kendall", "spearman", "xicor"), 
  include_nominals = TRUE, 
  max_levels = 5L, 
  sparse = TRUE, 
  pval_thresh = 0.1
)
```

Arguments

- `.data` data frame
- `...` tidyselect cols
- `use` method to deal with na. Default is to remove rows with NA
- `method` correlation method. default is pearson, but also supports xicor.
- `include_nominals` logicals, default TRUE. Dummyfify nominal variables?
- `max_levels` maximum numbers of dummies to be created from nominal variables
- `sparse` logical, default TRUE. Filters and arranges cor table
- `pval_thresh` threshold to filter out weak correlations

Details

includes the asymmetric correlation coefficient xi from `xicor`

Value

data frame of correlations
Examples

```r
iris %>%
  auto_cor()
# don't use sparse if you're interested in only one target variable
iris %>%
  auto_cor(sparse = FALSE) %>%
  dplyr::filter(x == "Petal.Length")
```

Description

Runs a cross validated xgboost and regularized linear regression, and reports accuracy metrics. Automatically determines whether the provided formula is a regression or classification.

Usage

```r
auto_model_accuracy(
  data,
  formula,
  ..., 
  n_folds = 4,
  as_flextable = TRUE,
  include_linear = FALSE,
  theme = "tron",
  seed = 1,
  mtry = 1,
  trees = 15L,
  min_n = 1L,
  tree_depth = 6L,
  learn_rate = 0.3,
  loss_reduction = 0,
  sample_size = 1,
  stop_iter = 10L,
  counts = FALSE,
  penalty = 0.015,
  mixture = 0.35
)
```

Arguments

- `data` : data frame
- `formula` : formula
- `...` : any other params for xgboost
**auto_tune_xgboost**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_folds</td>
<td>number of cross validation folds</td>
</tr>
<tr>
<td>as_flextable</td>
<td>if FALSE, returns a tibble</td>
</tr>
<tr>
<td>include_linear</td>
<td>if TRUE includes a regularized linear model</td>
</tr>
<tr>
<td>theme</td>
<td>make_flextable theme</td>
</tr>
<tr>
<td>seed</td>
<td>seed</td>
</tr>
<tr>
<td>mtry</td>
<td># Randomly Selected Predictors (xgboost: colsample_bynode) (type: numeric, range 0 - 1) (or type: integer if count = TRUE)</td>
</tr>
<tr>
<td>trees</td>
<td># Trees (xgboost: nrounds) (type: integer, default: 15L)</td>
</tr>
<tr>
<td>min_n</td>
<td>Minimal Node Size (xgboost: min_child_weight) (type: integer, default: 1L); [typical range: 2-10] Keep small value for highly imbalanced class data where leaf nodes can have smaller size groups. Otherwise increase size to prevent overfitting outliers.</td>
</tr>
<tr>
<td>tree_depth</td>
<td>Tree Depth (xgboost: max_depth) (type: integer, default: 6L); Typical values: 3-10</td>
</tr>
<tr>
<td>learn_rate</td>
<td>Learning Rate (xgboost: eta) (type: double, default: 0.3); Typical values: 0.01-0.3</td>
</tr>
<tr>
<td>loss_reduction</td>
<td>Minimum Loss Reduction (xgboost: gamma) (type: double, default: 0.0); range: 0 to Inf; typical value: 0 - 20 assuming low-mid tree depth</td>
</tr>
<tr>
<td>sample_size</td>
<td>Proportion Observations Sampled (xgboost: subsample) (type: double, default: 1.0); Typical values: 0.5 - 1</td>
</tr>
<tr>
<td>stop_iter</td>
<td># Iterations Before Stopping (xgboost: early_stop) (type: integer, default: 15L) only enabled if validation set is provided</td>
</tr>
<tr>
<td>counts</td>
<td>if TRUE specify mtry as an integer number of cols. Default FALSE to specify mtry as fraction of cols from 0 to 1</td>
</tr>
<tr>
<td>penalty</td>
<td>linear regularization parameter</td>
</tr>
<tr>
<td>mixture</td>
<td>linear model parameter, combines l1 and l2 regularization</td>
</tr>
</tbody>
</table>

**Value**

- a table
auto_tune_xgboost

Usage

auto_tune_xgboost(
  .data,  
  formula,  
  tune_method = c("grid", "bayes"),  
  event_level = c("first", "second"),  
  n_fold = 5L,  
  seed = 1,  
  n_iter = 100L,  
  save_output = FALSE,  
  parallel = TRUE,  
  trees = tune::tune(),  
  min_n = tune::tune(),  
  mtry = tune::tune(),  
  tree_depth = tune::tune(),  
  learn_rate = tune::tune(),  
  loss_reduction = tune::tune(),  
  sample_size = tune::tune(),  
  stop_iter = tune::tune(),  
  counts = FALSE,  
  tree_method = c("auto", "exact", "approx", "hist", "gpu_hist"),  
  monotone_constraints = 0L,  
  num_parallel_tree = 1L,  
  lambda = 1,  
  alpha = 0,  
  scale_pos_weight = 1,  
  verbosity = 0L
)

Arguments

.data dataframe

formula formula

tune_method method of tuning. defaults to grid

event_level for binary classification, which factor level is the positive class. specify "second" for second level

n_fold integer. n folds in resamples

seed seed

n_iter n iterations for tuning (bayes); paramter grid size (grid)

save_output FALSE. If set to TRUE will write the output as an rds file

parallel default TRUE; If set to TRUE, will enable parallel processing on resamples for grid tuning

trees # Trees (xgboost: nrounds) (type: integer, default: 15L)

min_n Minimal Node Size (xgboost: min_child_weight) (type: integer, default: 1L); [typical range: 2-10] Keep small value for highly imbalanced class data where
leaves. Otherwise increase size to prevent overfitting outliers.

mtry
# Randomly Selected Predictors (xgboost: colsample_bytree) (type: numeric, range 0 - 1) (or type: integer if count = TRUE)

tree_depth
Tree Depth (xgboost: max_depth) (type: integer, default: 6L); Typical values: 3-10

learn_rate
Learning Rate (xgboost: eta) (type: double, default: 0.3); Typical values: 0.01-0.3

loss_reduction
Minimum Loss Reduction (xgboost: gamma) (type: double, default: 0.0); range: 0 to Inf; typical value: 0 - 20 assuming low-mid tree depth

sample_size
Proportion Observations Sampled (xgboost: subsample) (type: double, default: 1.0); Typical values: 0.5 - 1

stop_iter
# Iterations Before Stopping (xgboost: early_stop) (type: integer, default: 15L) only enabled if validation set is provided

counts
if TRUE specify mtry as an integer number of cols. Default FALSE to specify mtry as fraction of cols from 0 to 1

tree_method
xgboost tree_method. default is auto. reference: tree method docs

monotone_constraints
an integer vector with length of the predictor cols, of -1, 1, 0 corresponding to decreasing, increasing, and no constraint respectively for the index of the predictor col. reference: monotonicity docs.

num_parallel_tree
should be set to the size of the forest being trained. default 1L

lambda
[default=1] L2 regularization term on weights. Increasing this value will make model more conservative.

alpha
[default=0] L1 regularization term on weights. Increasing this value will make model more conservative.

scale_pos_weight
[default=1] Control the balance of positive and negative weights, useful for unbalanced classes. if set to TRUE, calculates sum(negative instances) / sum(positive instances)

verbosity
[default=1] Verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug).

Details
Default is to tune all 7 xgboost parameters. Individual parameter values can be optionally fixed to reduce tuning complexity.

Value
workflow object
Examples

```r
if(FALSE){

iris %>%
  framecleaner::create_dummies() -> iris1

iris1 %>%
  tidy_formula(target = Petal.Length) -> petal_form

iris1 %>%
  rsample::initial_split() -> iris_split

iris_split %>%
  rsample::analysis() -> iris_train

iris_split %>%
  rsample::assessment() -> iris_val

iris_train %>%
  auto_tune_xgboost(formula = petal_form, n_iter = 10,
  parallel = TRUE, method = "bayes") -> xgb_tuned

xgb_tuned %>%
  fit(iris_train) %>%
  parsnip::extract_fit_engine() -> xgb_tuned_fit

xgb_tuned_fit %>%
  tidy_predict(newdata = iris_val, form = petal_form) -> iris_val1

}
```

Description

Performs a t.test on 2 populations for numeric variables.

Usage

```r
auto_t_test(data, col, ..., var_equal = FALSE, abbrv = TRUE)
```

Arguments

data	dataframe
col	a column with 2 categories representing the 2 populations
auto_variable_contributions

... numeric variables to perform t.test on. Default is to select all numeric variables
var_equal default FALSE; t.test parameter
abbrev default TRUE; remove some extra columns from output

Value
dataframe

Examples

iris %>%
dplyr::filter(Species != "setosa") %>%
auto_t_test(col = Species)

auto_variable_contributions

Plot Variable Contributions

Description

Return a variable importance plot and coefficient plot from a linear model. Used to easily visualize
the contributions of explanatory variables in a supervised model

Usage

auto_variable_contributions(data, formula, scale = TRUE)

Arguments

data dataframe
formula formula
scale logical. If FALSE puts coefficients on original scale

Value

a ggplot object

Examples

iris %>%
framecleaner::create_dummies() %>%
auto_variable_contributions(
  tidy_formula(., target = Petal.Width)
)
iris %>%
  auto_variable_contributions(
    tidy_formula(., target = Species)
  )

---

**cap_outliers**

**Description**

Caps the outliers of a numeric vector by percentiles. Also outputs a plot of the capped distribution

**Usage**

```r
cap_outliers(x, q = 0.05, type = c("both", "upper", "lower"))
```

**Arguments**

- `x`: numeric vector
- `q`: decimal input to the quantile function to set cap. default .05 caps at the 95 and 5th percentile
- `type`: chr vector. where to cap: both, upper, or lower

**Value**

numeric vector

**Examples**

```r
cap_outliers(iris$Petal.Width)
```

---

**create_monotone_constraints**

**Description**

helper function to create the integer vector to pass to the monotone_constraints argument in xgboost
eval_preds

Usage

create_monotone_constraints(
   .data,
   formula,
   decreasing = NULL,
   increasing = NULL
)

Arguments

.data   dataframe, training data for tidy_xgboost
formula formula used for tidy_xgboost
decreasing character vector or tidyselect regular expression to designate decreasing cols
increasing character vector or tidyselect regular expression to designate increasing cols

Value

a named integer vector with entries of 0, 1, -1

Examples

iris %>%
framecleaner::create_dummies(Species) -> iris_dummy

iris_dummy %>%
tidy_formula(target= Petal.Length) -> petal_form

iris_dummy %>%
create_monotone_constraints(petal_form,
   decreasing = tidyselect::matches("Petal|Species"),
   increasing = "Sepal.Width")

eval_preds  eval_preds

description

Automatically evaluates predictions created by tidy_predict. No need to supply column names.

Usage

eval_preds(.data, ..., softprob_model = NULL)
Arguments

.data   dataframe as a result of tidy_predict
...     additional metrics from yarstick to be calculated
softprob_model  character name of the model used to create multiclass probabilities

Value

tibble of summarized metrics

f_charvec_to_formula  charvec to formula

Description

takes the lhs and rhs of a formula as character vectors and outputs a formula

Usage

f_charvec_to_formula(lhs, rhs)

Arguments

lhs     lhs atomic chr vec
rhs     rhs chr vec

Value

formula

Examples

lhs <- "Species"
rhs <- c("Petal.Width", "Custom_Var")

f_charvec_to_formula(lhs, rhs)
**f_formula_to_charvec**  
*Formula rhs to chr vec*

**Description**

Accepts a formula and returns the rhs as a character vector.

**Usage**

```r
f_formula_to_charvec(f, include_lhs = FALSE, .data = NULL)
```

**Arguments**

- **f**: formula
- **include_lhs**: FALSE. If TRUE, appends lhs to beginning of vector
- **.data**: dataframe for names if necessary

**Value**

chr vector

**Examples**

```r
iris %>%
tidy_formula(target = Species, tidyselect::everything()) -> f

f

f %>%
f_formula_to_charvec()
```

**f_modify_formula**  
*Modify Formula*

**Description**

Modify components of a formula by adding / removing vars from the rhs or replacing the lhs.

**Usage**

```r
f_modify_formula(
    f,
    rhs_remove = NULL,
    rhs_add = NULL,
    lhs_replace = NULL,
    negate = TRUE
)
```
Arguments

- **f**: formula
- **rhs_remove**: regex or character vector for dropping variables from the rhs
- **rhs_add**: character vector for adding variables to rhs
- **lhs_replace**: string to replace formula lhs if supplied
- **negate**: should `rhs_remove` keep or remove the specified vars. Set to `FALSE` to keep

Value

- formula

Examples

```r
iris %>%
tidy_formula(target = Species, tidyselect::everything()) -> f

f

f %>%
f_modify_formula(
rhs_remove = c("Petal.Width", "Sepal.Length"),
rhs_add = "Custom_Variable"
)

f %>%
f_modify_formula(
rhs_remove = "Petal",
lhs_replace = "Petal.Length"
)
```

Description

s3 method to extract params of a model with names consistent for use in the ‘autostats‘ package

Usage

```r
get_params(model, ...)
```
Arguments

model a model
... additional arguments

Value

list of params

Examples

```r
iris %>%
  framecleaner::create_dummies() -> iris_dummies

iris_dummies %>%
  tidy_formula(target = Petal.Length) -> p_form

iris_dummies %>%
  tidy_xgboost(p_form, mtry = .5, trees = 5L, loss_reduction = 2, sample_size = .7) -> xgb

## reuse these parameters to find the cross validated error
rlang::exec(auto_model_accuracy, data = iris_dummies, formula = p_form, !!!get_params(xgb))
```

Description

Imputes missing values of a numeric matrix using stochastic gradient descent. `recosystem`

Usage

```r
impute_recosystem(
  .data,
  lrate = c(0.05, 0.1),
  costp_l1 = c(0, 0.05),
  costq_l1 = c(0, 0.05),
  costp_l2 = c(0, 0.05),
  costq_l2 = c(0, 0.05),
  nthread = 8,
  loss = "l2",
  niter = 15,
  verbose = FALSE,
  nfold = 4,
  seed = 1
)
```
Arguments

.data        long format data frame
lrate        learning rate
costp_l1     l1 cost p
costq_l1     l1 cost q
costp_l2     l2 cost p
costq_l2     l2 cost q
nthread      nthreads
loss         loss function. also can use “l1”
niter        training iterations for tune
verbose      show training loss?
nfold        folds for tune validation
seed         seed for randomness

Details

input is a long data frame with 3 columns: ID col, Item col (the column names from pivoting longer), and the ratings (values from pivoting longer)

pre-processing generally requires pivoting a wide user x item matrix to long format. The missing values from the matrix must be retained as NA values in the rating column. The values will be predicted and filled in by the algorithm. Output is a long data frame with the same number of rows as input, but no missing values.

This function automatically tunes the recosystem learner before applying. Parameter values can be supplied for tuning. To avoid tuning, use single values for the parameters.

Value

long format data frame

Description

Boosted tree regression using the agtboost package. Variable importance plot is printed along with returning the model. Noise features are eliminated from the plot.

Usage

tidy_agtboost(.data, formula, ...)

---

tidy_agtboost     tidy agtboost
Arguments

.data dataframe
formula formula
... additional parameters to pass to gbt.train

Details

agtboost: Adaptive and Automatic Gradient Tree Boosting Computations

Value

agtboost model of class Rcpp_ENSEMBLE

Examples

iris %>%
tidy_formula(target = Petal.Length) -> f1

iris %>%
tidy_agtboost(f1)

Description

Runs a conditional inference forest.

Usage

tidy_cforest(data, formula, seed = 1)

Arguments

data dataframe
formula formula
seed seed integer

Value

a cforest model
Examples

```r
iris %>%
tidy_cforest(
  tidy_formula(., Petal.Width)
) -> iris_cfor

iris_cfor

iris_cfor %>%
visualize_model()
```

---

description

tidy conditional inference tree. Creates easily interpretable decision tree models that be shown with the `visualize_model` function. Statistical significance required for a split, and minimum necessary samples in a terminal leaf can be controlled to create the desired tree visual.

Usage

tidy_ctree(.data, formula, minbucket = 7L, mincriterion = 0.95, ...)

Arguments

- `.data` dataframe
- `formula` formula
- `minbucket` minimum amount of samples in terminal leaves, default is 7
- `mincriterion` \((1 - \alpha)\) value between 0 - 1, default is .95. lowering this value creates more splits, but less significant
- `...` optional parameters to `ctree_control`

Value

a ctree object

Examples

```r
iris %>%
tidy_formula(., Sepal.Length) -> sepal_form

iris %>%
tidy_ctree(sepal_form) %>%
visualize_model()
```
tidy_foci

Description
variable selection with FOCI

Usage

```r
tidy_foci(.data, formula, ...)
```

Arguments

- `.data` data
- `formula` formula
- `...` other arguments to FOCI

Value
data frame

Examples

```r
iris %>%
tidy_focite(spal_form, minbucket = 30) %>%
visualize_model(plot_type = "box")

iris %>%
tidy_foci(Species ~ .) %>%
d1 %>%
tibble::as_tibble()
```
tidy_formula  
**tidy formula construction**

**Description**

Takes a dataframe and allows for use of tidyselect to construct a formula.

**Usage**

```r
 tidy_formula(data, target, ...) 
```

**Arguments**

- `data`  
  dataframe
- `target`  
  lhs
- `...`  
  tidyselect. rhs

**Value**

a formula

**Examples**

```r
 iris %>%
 tidy_formula(Species, tidyselect::everything())
```

tidy_glm  
**tidy glm**

**Description**

Runs either a linear regression, logistic regression, or multinomial classification. The model is automatically determined based off the nature of the target variable.

**Usage**

```r
 tidy_glm(data, formula) 
```

**Arguments**

- `data`  
  dataframe
- `formula`  
  formula
tidy_predict

Value

glm model

Examples

# linear regression
iris %>%
tidy_glm(
tidy_formula(., target = Petal.Width)) -> glm1

glm1

glm1 %>%
visualize_model()

# multinomial classification


tidy_formula(iris, target = Species) -> species_form

iris %>%
tidy_glm(species_form) -> glm2

glm2 %>%
visualize_model()

# logistic regression

iris %>%
dplyr::filter(Species != "setosa") %>%
tidy_glm(species_form) -> glm3

suppressWarnings({
  glm3 %>%
  visualize_model()})

Description

tidy predict

Usage

tidy_predict(
  model,
  newdata,
  form = NULL,
tidy_shap

```r

olddata = NULL,
bind_preds = FALSE,
...
)

## S3 method for class 'Rcpp_ENSEMBLE'
tidy_predict(model, newdata, form = NULL, ...)

## S3 method for class 'glm'
tidy_predict(model, newdata, form = NULL, ...)

## Default S3 method:
tidy_predict(model, newdata, form = NULL, ...)

## S3 method for class 'BinaryTree'
tidy_predict(model, newdata, form = NULL, ...)

## S3 method for class 'xgb.Booster'
tidy_predict(
  model,
  newdata,
  form = NULL,
  olddata = NULL,
  bind_preds = FALSE,
...)

Arguments

model          model
newdata        dataframe
form           the formula used for the model
olddata        training data set
bind_preds      set to TRUE if newdata is a dataset without any labels, to bind the new and old
data with the predictions under the original target name
...
other parameters to pass to predict

Value

dataframe

```

tidy_shap    tidy shap
tidy_xgboost

Description

plot and summarize shapley values from an xgboost model

Usage

tidy_shap(model, newdata, form = NULL, ..., top_n = 12, aggregate = NULL)

Arguments

model: xgboost model
newdata: dataframe similar to model input
form: formula used for model
...: additional parameters for shapley value
top_n: top n features
aggregate: a character vector. Predictors containing the string will be aggregated, and renamed to that string.

Details

returns a list with the following entries

shap_tbl: table of shaply values
shap_summary: table summarizing shapley values. Includes correlation between shaps and feature values.
swarmplot: one plot showing the relation between shaps and features
scatterplots: returns the top 9 most important features as determined by sum of absolute shapley values, as a faceted scatterplot of feature vs shap

Value

list

Description

Accepts a formula to run an xgboost model. Automatically determines whether the formula is for classification or regression. Returns the xgboost model.
Usage

tidy_xgboost(
  .data,
  formula,
  ...,  
mtry = 1,
  trees = 15L,
  min_n = 1L,
  tree_depth = 6L,
  learn_rate = 0.3,
  loss_reduction = 0,
  sample_size = 1,
  stop_iter = 10L,
  counts = FALSE,
  tree_method = c("auto", "exact", "approx", "hist", "gpu_hist"),
  monotone_constraints = 0L,
  num_parallel_tree = 1L,
  lambda = 1,
  alpha = 0,
  scale_pos_weight = 1,
  verbosity = 0L,
  validate = TRUE
)

Arguments

.data dataframe
formula formula
... additional parameters to be passed to set_engine
mtry # Randomly Selected Predictors (xgboost: colsample_bynode) (type: numeric, range 0 - 1) (or type: integer if count = TRUE)
trees # Trees (xgboost: nrounds) (type: integer, default: 15L)
min_n Minimal Node Size (xgboost: min_child_weight) (type: integer, default: 1L); [typical range: 2-10] Keep small value for highly imbalanced class data where leaf nodes can have smaller size groups. Otherwise increase size to prevent overfitting outliers.
tree_depth Tree Depth (xgboost: max_depth) (type: integer, default: 6L); Typical values: 3-10
learn_rate Learning Rate (xgboost: eta) (type: double, default: 0.3); Typical values: 0.01-0.3
loss_reduction Minimum Loss Reduction (xgboost: gamma) (type: double, default: 0.0); range: 0 to Inf; typical value: 0 - 20 assuming low-mid tree depth
sample_size Proportion Observations Sampled (xgboost: subsample) (type: double, default: 1.0); Typical values: 0.5 - 1
stop_iter # Iterations Before Stopping (xgboost: early_stop) (type: integer, default: 15L) only enabled if validation set is provided
counts if TRUE specify mtry as an integer number of cols. Default FALSE to specify mtry as fraction of cols from 0 to 1

```
tree_method  xgboost tree_method. default is auto. reference: tree method docs
monotone_constraints
```

an integer vector with length of the predictor cols, of -1, 1, 0 corresponding to decreasing, increasing, and no constraint respectively for the index of the predictor col. reference: monotonicity docs.

```
tnum_parallel_tree
```

should be set to the size of the forest being trained. default 1L

```
lambda [default=1] L2 regularization term on weights. Increasing this value will make model more conservative.
alpha [default=0] L1 regularization term on weights. Increasing this value will make model more conservative.
```

```
scscale_pos_weight
```

[default=1] Control the balance of positive and negative weights, useful for unbalanced classes. if set to TRUE, calculates sum(negative instances) / sum(positive instances)

```
verbosverbosity
```

[default=1] Verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug).

```
validate
```

default TRUE. report accuracy metrics on a validation set.

Details

reference for parameters: xgboost docs

Value

```
xgb.Booster model
```

Examples

```
ooptions(rlang_trace_top_env = rlang::current_env())

# regression on numeric variable

iris %>%
  framecleaner::create_dummies(Species) -> iris_dummy

iris_dummy %>%
  tidy_formula(target= Petal.Length) -> petal_form

iris_dummy %>%
  tidy_xgboost(
    petals_form,
    trees = 500,
    mtry = .5
  ) -> xg1
```
tidy_xgboost

```r
xg1 %>%
  visualize_model(top_n = 2)

xg1 %>%
  tidy_predict(newdata = iris_dummy, form = petal_form) -> iris_preds

iris_preds %>%
  eval_prets()

# binary classification
# returns probability and labels

iris %>%
  tidy_formula(Species) -> species_form

iris %>%
  dplyr::filter(Species != "versicolor") %>%
  dplyr::mutate(Species = forcats::fct_drop(Species)) -> iris_binary

iris_binary %>%
  tidy_xgboost(formula = species_form, trees = 50L, mtry = 0.2) -> xgb_bin

xgb_bin %>%
  tidy_predict(newdata = iris_binary, form = species_form) -> iris_binary1

iris_binary1 %>%
  eval_prets()

# multiclass classification that returns labels

iris %>%
  tidy_xgboost(species_form,
               objective = "multi:softmax",
               trees = 100,
               tree_depth = 3L,
               loss_reduction = 0.5) -> xgb2

xgb2 %>%
  tidy_predict(newdata = iris, form = species_form) -> iris_preds

# additional yardstick metrics can be supplied to the dots in eval_prets

iris_preds %>%
  eval_prets(yardstick::j_index)
```
# multiclass classification that returns probabilities

```r
iris %>%
tidy_xgboost(species_form,
  objective = "multi:softprob",
  trees = 50L,
  sample_size = .2,
  mtry = .5,
  tree_depth = 2L,
  loss_reduction = 3) -> xgb2_prob
```

# predict on the data that already has the class labels, so the resulting data frame # has class and prob predictions

```r
xgb2_prob %>%
tidy_predict(newdata = iris_preds, form = species_form) -> iris_preds1
```

# also requires the labels in the dataframe to evaluate preds # the model name must be supplied as well. Then roc metrics can be calculated

```r
# iris_preds1 %>%
# eval_preds( yardstick::average_precision, softprob_model = "xgb2_prob"
# )
```

---

**Description**

`s3` method to automatically visualize the output of a model object. Additional arguments can be supplied for the original function. Check the corresponding plot function documentation for any custom arguments.

**Usage**

```r
visualize_model(model, ..., method = NULL)
```

```r
## S3 method for class 'RandomForest'
visualize_model(model, ..., method)
```

```r
## S3 method for class 'BinaryTree'
visualize_model(model, ..., method)
```

```r
## S3 method for class 'glm'
visualize_model(model, ..., method)
```
visualize_model

## S3 method for class 'multinom'
visualize_model(model, ..., method)

## S3 method for class 'xgb.Booster'
visualize_model(model, ..., method)

## Default S3 method:
visualize_model(model, ..., method)

### Arguments

- **model**: a model
- **...**: additional arguments
- **method**: choose amongst different visualization methods

### Value

- a plot
Index

auto_anova, 2
auto_boxplot, 4
auto_cor, 5
auto_model_accuracy, 6
auto_t_test, 10
auto_tune_xgboost, 7
auto_variable_contributions, 11

cap_outliers, 12
cor.test, 5
create_monotone_constraints, 12
ctree_control, 20

eval_preds, 13
f_charvec_to_formula, 14
f_formula_to_charvec, 15
f_modify_formula, 15

gbt.train, 19
gem_boxplot, 4
get_params, 16

impute_recosystem, 17

set_engine, 26

tidy_agtboost, 18
tidy_cforest, 19
tidy_ctree, 20
tidy_foci, 21
tidy_formula, 22
tidy_glm, 22
tidy_predict, 13, 14, 23
tidy_shap, 24
tidy_xgboost, 25

visualize_model, 20, 29

xicor, 5