Package ‘manymodelr’

October 13, 2022

Title Build and Tune Several Models
Version 0.3.7
Description Frequently one needs a convenient way to build and tune several models in one go. The goal is to provide a number of machine learning convenience functions. It provides the ability to build, tune and obtain predictions of several models in one function. The models are built using functions from 'caret' with easier to read syntax. 
License GPL-2
Encoding UTF-8
RoxygenNote 7.1.2
Imports dplyr (>= 1.0.0), ggplot2 (>= 3.3.5), lme4 (>= 1.1.27.1)
Depends R (>= 3.6.0), caret (>= 6.0.88), Metrics (>= 0.1.4), e1071 (>= 1.7.8)
URL https://github.com/Nelson-Gon/manymodelr
BugReports https://github.com/Nelson-Gon/manymodelr/issues
Suggests knitr, rmarkdown, testthat, covr, markdown
VignetteBuilder knitr
LazyData True
NeedsCompilation no
Author Nelson Gonzabato [aut, cre]
Maintainer Nelson Gonzabato <gonzabato@hotmail.com>
Repository CRAN
Date/Publication 2021-11-15 09:20:09 UTC

R topics documented:

add_model_predictions .......................................................... 2
add_model_residuals ............................................................. 3
**add_model_predictions**

Add predictions to the data set. A dplyr compatible way to add predictions to a data set.

**Description**

Add predictions to the data set. A dplyr compatible way to add predictions to a data set.

**Usage**

```
add_model_predictions(model = NULL, old_data = NULL, new_data = NULL)
```

**Arguments**

- `model` A model object from `fit_model`
- `old_data` The data set to which predicted values will be added.
- `new_data` The data set to use for predicting.

**Value**

A data.frame object with a new column for predicted values

**See Also**

`fit_model` `extract_model_info`
**add_model_residuals**

**Examples**

```r
data("yields", package="manymodelr")
yields1 <- yields[1:50,]
yields2 <- yields[51:100,]
lm_model <- fit_model(yields1, "weight", "height", "lm")
head(add_model_predictions(lm_model, yields1, yields2))
```

**Description**

A dplyr compatible convenience function to add residuals to a data set.

**Usage**

```r
add_model_residuals(model = NULL, old_data = NULL)
```

**Arguments**

- `model`: A model object from `fit_model`.
- `old_data`: The data set to which predicted values will be added.

**Value**

A data.frame object with residuals added.

**Examples**

```r
data("yields", package="manymodelr")
yields1 <- yields[1:50,]
yields2 <- yields[51:100,]
lm_model <- fit_model(yields1, "weight", "height", "lm")
head(add_model_residuals(lm_model, yields2))
```

---

**agg_by_group**

**A convenient way to perform grouped operations**

**Description**

This function performs operations by grouping the data.

**Usage**

```r
agg_by_group(df = NULL, my_formula = NULL, func = NULL, ...)
```
**Arguments**

- **df**: The data set for which correlations are required
- **my_formula**: A formula such as `A~B` where B is the grouping variable (normally a factor). See examples below
- **func**: The kind of operation e.g `sum`, `mean`, `min`, `max`, `many` or `modelr::get_mode`
- **...**: Other arguments to `aggregate` see `?aggregate` for details

**Value**

A grouped data.frame object with results of the chosen operation.

**Examples**

```r
head(agg_by_group(airquality,.~Month,sum))
```

---

**drop_non_numeric**

*Drops non numeric columns from a data.frame object*

**Description**

Drops non numeric columns from a data.frame object

**Usage**

```r
drop_non_numeric(df)
```

**Arguments**

- **df**: A data.frame object for which non-numeric columns will be dropped

**Examples**

```r
drop_non_numeric(data.frame(A=1:2, B=c("A", "B")))
```
extract_model_info  
Extract important model attributes

Description

Provides a convenient way to extract any kind of model information from common model objects

Usage

extract_model_info(model_object = NULL, what = NULL, ...)

Arguments

model_object A model object for example a linear model object, generalized linear model object, analysis of variance object.
what character. The attribute you would like to obtain for instance p_value
... Arguments to other functions e.g. AIC, BIC, deviance etc

Details

This provides a convenient way to extract model information for any kind of model. For linear models, one can extract such attributes as coefficients, p value("p_value"), standard error("std_err"), estimate, t value("t_value"), residuals, aic and other known attributes. For analysis of variance (aov), other attributes like sum squared(ssq), mean squared error(msq), degrees of freedom(df),p_value.

Examples

# perform analysis of variance
data("yields", package="manymodelr")
aov_mod <- fit_model(yields, "weight","height + normal","aov")
exttract_model_info(aov_mod, "ssq")
extrack_model_info(aov_mod, c("ssq","predictors"))
# linear regression
lm_model <- fit_model(yields, "weight","height","lm")
extrack_model_info(lm_model, c("aic","bic"))
## glm
glm_model <- fit_model(yields, "weight","height","glm")
extrack_model_info(glm_model,"aic")
fit_models

**Description**

Fit several models with different response variables.

### Usage

```r
fit_models(
  df = NULL,
  yname = NULL,
  xname = NULL,
  modeltype = NULL,
  drop_non_numeric = FALSE,
  ...
)
```

### Arguments

- `df` A data.frame object
- `yname` The outcome variable
- `xname` The predictor variable(s)
- `modeltype` A character specifying the model type e.g. lm for linear model
- `drop_non_numeric` Should non numeric columns be dropped? Defaults to FALSE
- `...` Other arguments to specific model types.

### Examples

```r
data("yields", package="manymodelr")
fit_model(yields, "height", "weight", "lm")
fit_model(yields, "weight", "height + I(yield)**2", "lm")
```
get_data_Stats

Usage

```r
fit_models(
  df = NULL,
  yname = NULL,
  xname = NULL,
  modeltype = NULL,
  drop_non_numeric = FALSE,
  ...
)
```

Arguments

- `df`: A data.frame object
- `yname`: The outcome variable
- `xname`: The predictor variable(s)
- `modeltype`: A character specifying the model type e.g `lm` for linear model
- `drop_non_numeric`: Should non numeric columns be dropped? Defaults to FALSE
- `...`: Other arguments to specific model types.

Value

A list of model objects that can be used later.

Examples

```r
data("yields", package="manymodelr")
fit_models(df=yields,yname=c("height","yield"),xname="weight",modeltype="lm") #many model types
fit_models(df=yields,yname=c("height","yield"),xname="weight",modeltype=c("lm", "glm"))
```

get_data_Stats

A pipe friendly way to get summary stats for exploratory data analysis

Description

A pipe friendly way to get summary stats for exploratory data analysis

Usage

```r
get_data_Stats(
  x = NULL,
  func = NULL,
  exclude = NULL,
  na.rm = FALSE,
  ...
)
```
get_stats(
  x = NULL,
  func = NULL,
  exclude = NULL,
  na.rm = FALSE,
  na_action = NULL,
  ...
)

Arguments

x The data for which stats are required
func The nature of function to apply
exclude What kind of data should be excluded? Use for example c("character","factor") to drop character and factor columns
na.rm Logical. Should NAs be removed. Defaults to FALSE.
na_action If na.rm is set to TRUE, this uses na_replace to replace missing values.
... Other arguments to na_replace See ?na_replace for details.

Details

A convenient wrapper especially useful for get_mode

Value

A data.frame object showing the requested stats

Examples

head(get_data_Stats(airquality,mean,na.rm = TRUE,na_action = "get_mode"))
get_stats(airquality,mean,"non_numeric",na.rm = TRUE,na_action = "get_mode")

get_exponent

Get the exponent of any number or numbers

Description

Get the exponent of any number or numbers

Usage

get_exponent(y = NULL, x = NULL)
get_mode

Arguments

y The number or numeric columns for which an exponent is required
x The power to which y is raised

Details

Depends on the expo and expo1 functions in expo

Value

A data.frame object showing the value, power and result

Examples

df<-data.frame(A=c(1123, 25657, 3987))
get_exponent(df, 3)
get_exponent(1:5, 2)

get_mode

A convenience function that returns the mode

Description

A convenience function that returns the mode

Usage

get_mode(x, na.rm = TRUE)

Arguments

x The dataframe or vector for which the mode is required.
na.rm Logical. Should ‘NA’ s be dropped? Defaults to ‘TRUE’

Details

Useful when used together with get_stats in a pipe fashion. These functions are for exploratory data analysis The smallest number is returned if there is a tie in values The function is currently slow for greater than 300,000 rows. It may take up to a minute. may work with inaccuracies. By default, NAs are discarded.

Value

A data.frame or vector showing the mode of the variable(s)
Examples

test<-c(1,2,3,3,3,3,4,5)
test2<-c(455,7878,908981,NA,456,455,7878,7878,NA)
get_mode(test)
get_mode(test2)
## Not run:
mtcars %>%
get_data_Stats(get_mode)
get_data_Stats(mtcars,get_mode)
## End(Not run)

get_this

Helper function to easily access elements

Description

Helper function to easily access elements

Usage

get_this(where = NULL, what = NULL)

Arguments

where  Where do you want to get it from? Currently only supports ‘list’s and ‘data.frame’ objects.
what   What do you want to extract from the ‘data.frame’ or ‘list’? No quotes. See examples below.

Details

This is a helper function useful if you would like to extract data from the output of ‘multi_model_1’.

Examples

my_list<-list(A=520),list(B=456,C=567))
get_this(where="A",my_list)
get_this(my_list,"C")
# use values
get_this(my_list, "B")
get_var_corr

Get correlations between variables

Description
This function returns the correlations between different variables.

Usage
get_var_corr(
  df,
  comparison_var = NULL,
  other_vars = NULL,
  method = "pearson",
  drop_columns = c("factor", "character"),
  ...
)

Arguments

df                The data set for which correlations are required
comparison_var    The variable to compare to
other_vars        variables for which correlation with comparison_var is required. If not supplied, all variables will be used.
method             The method used to perform the correlation test as defined in 'cor.test'. Defaults to pearson.
drop_columns      A character vector specifying column classes to drop. Defaults to c("factor","character")
...                Other arguments to 'cor.test' see ?cor.test for details

Value
A data.frame object containing correlations between comparison_var and each of other_vars

Examples

# Get correlations between all variables
get_var_corr(mtcars,"mpg")
# Use only a few variables
get_var_corr(mtcars,"mpg", other_vars = c("disp","drat"), method = "kendall",exact=FALSE)
**get_var_corr_**

Get correlations for combinations

Description

Get correlations for combinations

Usage

```r
get_var_corr_(
  df,
  subset_cols = NULL,
  drop_columns = c("character", "factor"),
  ...
)
```

Arguments

- **df**: A ‘data.frame’ object for which correlations are required in combinations.
- **subset_cols**: A ‘list’ of length 2. The values in the list correspond to the comparison and other_Var arguments in ‘get_var_corr’. See examples below.
- **drop_columns**: A character vector specifying column classes to drop. Defaults to c("factor","character")
- **...**: Other arguments to ‘get_var_corr’

Details

This function extends get_var_corr by providing an opportunity to get correlations for combinations of variables. It is currently slow and may take up to a minute depending on system specifications.

Value

A data.frame object with combinations.

Examples

```r
get_var_corr_(mtcars,method="pearson")
#use only a subset of the data.
get_var_corr_(mtcars,
  subset_cols = list(c("mpg","vs"), c("disp","wt")),
  method="spearman",exact=FALSE)
```
multi_model_1

Simultaneously train and predict on new data.

Description

This function provides a convenient way to train several model types. It allows a user to predict on new data and depending on the metrics, the user is able to decide which model predictions to finally use. The models are built based on Max Kuhn’s models in the caret package.

Usage

multi_model_1(
  old_data,
  yname,
  xname,
  method = NULL,
  metric = NULL,
  control = NULL,
  new_data = NULL,
  ...
)

Arguments

old_data The data holding the training dataset
yname The outcome variable
xname The predictor variable(s)
method A vector containing methods to be used as defined in the caret package
metric One of several metrics. Accuracy, RMSE, MAE, etc
control See caret ?trainControl for details.
new_data A data set to validate the model or for which predictions are required
... Other arguments to caret’s train function

Details

Most of the details of the parameters can be found in the caret package documentation. This function is meant to help in exploratory analysis to make an informed choice of the best models

Value

A list containing two objects. A tibble containing a summary of the metrics per model, a tibble containing predicted values and information concerning the model
References
Kuhn (2008), "Building Predictive Models in R Using the caret" (http://www.jstatsoft.org/article/view/v028i05/v28i05.pold_data)

Examples

data("yields", package="manymodelr")
train_set<-createDataPartition(yields$normal,p=0.8,list=FALSE)
valid_set<-yields[-train_set,]
train_set<-yields[train_set,]
ctrl<-trainControl(method="cv",number=5)
set.seed(233)
m<-multi_model_1(train_set,"normal",".",c("knn","rpart"),
"Accuracy",ctrl,new_data =valid_set)
m$Predictions
m$Metrics
m$modelInfo

multi_model_2

Fit and predict in one function

Description
Fit and predict in one function

Usage

multi_model_2(old_data, new_data, yname, xname, modeltype, ...)

Arguments

old_data The data set to which predicted values will be added.
new_data The data set to use for predicting.
yname The outcome variable
xname The predictor variable(s)
modeltype A character specifying the model type e.g lm for linear model
... Other arguments to specific model types.

Examples

# fit a linear model and get predictions
multi_model_2(iris[1:50,],iris[50:99,],"Sepal.Length","Petal.Length","lm")
# multilinear
multi_model_2(iris[1:50,],iris[50:99,],"Sepal.Length",
"Petal.Length + Sepal.Width","lm")
# glm
multi_model_2(iris[1:50,],iris[50:99,],"Sepal.Length","Petal.Length","glm")
na_replace

Replace missing values

Description

Replace missing values

Usage

na_replace(df, how = NULL, value = NULL)

Arguments

- **df**: The data set (data.frame or vector) for which replacements are required
- **how**: How should missing values be replaced? One of ffill, samples, value or any other known method e.g mean, median, max, min. The default is NULL meaning no imputation is done. For character vectors, the use of ‘get_mode’ is also supported. No implementation for class factor (yet).
- **value**: If how is set to value, this allows the user to provide a specific fill value for the NAs.

Details

This function currently does not support grouping although this may be achieved with some inaccuracies using grouping functions from other packages.

Value

A data.frame object with missing values replaced.

Examples

head(na_replace(airquality, how="value", value="Missing"))

na_replace_grouped

Replace NAs by group

Description

A convenient way to replace NAs by group.

Usage

na_replace_grouped(df, group_by_cols = NULL, ...)


Arguments

- **df**: A data.frame object for which grouped NA replacement is desired.
- **group_by_cols**: The column(s) used to use for the grouping.
- **...**: Other arguments to `na_replace`

Value

A 'data.frame' object with 'NA's replaced.

Examples

```r
test2 <- data.frame(A=c("A","A","A","B","B","B"),
B=c(NA,5,2,2,NA,2))
head(na_replace_grouped(test2,"A",how="value","Replaced")==)
```

---

**plot_corr**

Plot a correlations matrix

**Description**

This function plots the results produced by `get_var_corr_`.

**Usage**

```r
plot_corr(
  df,
  x = "comparison_var",
  y = "other_var",
  xlabel = "comparison_variable",
  ylabel = "other_variable",
  title = "Correlations Plot",
  plot_style = "circles",
  title_just = 0.5,
  round_which = NULL,
  colour_by = NULL,
  decimals = 2,
  show_which = "corr",
  size = 12.6,
  value_angle = 360,
  shape = 16,
  value_size = 3.5,
  value_col = "black",
  width = 1.1,
  custom_cols = c("indianred2", "green2", "gray34"),
  legend_labels = waiver(),
  legend_title = NULL,
)```
plot_corr

    signif_cutoff = 0.05,
    signif_size = 7,
    signif_col = "gray13",
... )

Arguments

df          The data to be plotted. A ‘data.frame’ object produced by ‘get_var_corr_’
x           Value for the x axis. Defaults to "comparison_var"
y           Values for the y axis. Defaults to "other_var."
xlabel       label for the x axis
ylabel       label for the y axis
title        plot title.
plot_style   One of squares and circles(currently).
title_just   Justification of the title. Defaults to 0.5, title is centered.
round_which  Character. The column name to be rounded off.
colour_by    The column to use for coloring. Defaults to "correlation". Colour strength thus
             indicates the strength of correlations.
decimals     Numeric. To how many decimal places should the rounding be done? Defaults
             to 2.
show_which   Character. One of either corr or signif to control whether to show the correlation
             values or significance stars of the correlations. This is case sensitive and defaults
             to corr i.e. correlation values are shown.
size         Size of the circles for plot_style set to circles
value_angle  What angle should the text be?
shape        Values for the shape if plot_style is circles
value_size   Size of the text.
value_col    What colour should the text in the squares/circles be?
width        width value for plot_style set to squares.
custom_cols  A vector(length 2) of colors to use for the plot. The first colour specifies the
             lower end of the correlations. The second specifies the higher end.
legend_labels Text to use for the legend labels. Defaults to the default labels produced by the
               plot method.
legend_title Title to use for the legend.
signif_cutoff Numeric. If show_signif is TRUE, this defines the cutoff point for significance.
               Defaults to 0.05.
signif_size   Numeric. Defines size of the significance stars.
signif_col    Character. Defines the col for the significance stars.
...           Other arguments to get_var_corr_
Details

This function uses ‘ggplot2’ backend. ‘ggplot2’ is thus required for the plots to work. Since the correlations are obtained by ‘get_var_corr_’, the default is to omit correlation between a variable and itself. Therefore blanks in the plot would indicate a correlation of 1.

Value

A ‘ggplot2’ object showing the correlations plot.

Examples

```r
plot_corr(mtcars,show_which = "corr",
          round_values = TRUE,
          round_which = "correlation",decimals = 2,x="other_var",
          y="comparison_var",plot_style = "circles",width = 1.1,
          custom_cols = c("green","blue","red"),colour_by = "correlation")
```

rowdiff

Get row differences between values

Description

This function returns the differences between rows depending on the user’s choice.

Usage

```r
rowdiff(
  df,
  direction = "forward",
  exclude = NULL,
  na.rm = FALSE,
  na_action = NULL,
  ...
)
```

Arguments

- **df** The data set for which differences are required
- **direction** One of forward and reverse. The default is forward meaning the differences are calculated in such a way that the difference between the current value and the next is returned
- **exclude** A character vector specifying what classes should be removed. See examples below
- **na.rm** Logical. Should missing values be removed? The missing values referred to are those introduced during the calculation ie when subtracting a row with itself. Defaults to FALSE.
\textbf{select\_col}  

\begin{verbatim}
na_action If \texttt{na.rm} is \texttt{TRUE}, how should missing values be replaced? Depending on the
type as set out in \texttt{na_replace'}, the value can be replaced as per the user's re-
quirement.

... Other arguments to \texttt{na_replace'}.  
\end{verbatim}

\textbf{Value}  

A data.frame object of row differences

\textbf{See Also}  

\texttt{na\_replace}

\textbf{Examples}  

\begin{verbatim}
# Remove factor columns
data("yields", package="manymodelr")
rowdiff(yields, exclude = "factor", direction = "reverse")
rowdiff(yields[1:5,], exclude="factor", na.rm = TRUE,
na_action = "get_mode",direction = "reverse")
\end{verbatim}

\begin{verbatim}
select\_col A convenient selector gadget
\end{verbatim}

\textbf{Description}  

A convenient selector gadget

\textbf{Usage}  

\texttt{select\_col(df, \ldots)}

\textbf{Arguments}  

\begin{verbatim}
df The data set from which to select a column
\ldots columns to select, no quotes
\end{verbatim}

\textbf{Details}  

A friendly way to select a column or several columns. Mainly for non-pipe usage It is recommended
to use known select functions to do pipe manipulations. Otherwise convert to tibble

\textbf{Value}  

Returns a dataframe with selected columns
select_percentile

Get the row corresponding to a given percentile

Description

Get the row corresponding to a given percentile

Usage

select_percentile(df = NULL, percentile = NULL, descend = FALSE)

Arguments

df A `data.frame` object for which a percentile is required. Other data structures are not yet supported.
percentile The percentile required eg 10 percentile
descend Logical. Should the data be arranged in descending order? Defaults to FALSE.

Details

Returns the value corresponding to a percentile. Returns mean values if the position of the percentile is whole number. Values are sorted in ascending order. You can change this by setting `descend` to TRUE.

Value

A dataframe showing the row corresponding to the required percentile.

Examples

data("yields", package="manymodelr")
select_percentile(yields, 5)
### Description

A simulated data set of plant yields, height, weight, and a binary class

### Usage

```r
yields
```

### Author(s)

Nelson Gonzabato
Index

add_model_predictions, 2
add_model_residuals, 3
agg_by_group, 3

drop_non_numeric, 4

extract_model_info, 2, 5

fit_model, 2, 6
fit_models, 6

get_data_Stats, 7
get_exponent, 8
get_mode, 9
get_stats(get_data_Stats), 7
get_this, 10
get_var_corr, 11
get_var_corr_, 12

multi_model_1, 13
multi_model_2, 14

na_replace, 15, 19
na_replace_grouped, 15

plot_corr, 16

rowdiff, 18

select_col, 19
select_percentile, 20

yields, 21