Package 'modeldb'

November 1, 2023

Title Fits Models Inside the Database

Version 0.3.0

Description Uses 'dplyr' and 'tidyeval' to fit statistical models inside the database. It currently supports KMeans and linear regression models.

License MIT + file LICENSE

URL https://modeldb.tidymodels.org, https://github.com/tidymodels/modeldb

BugReports https://github.com/tidymodels/modeldb/issues

Depends R (>= 3.6)

Imports cli, dplyr (>= 0.7), ggplot2, progress, purrr, rlang (>= 1.1.1), tibble, tidypredict

Suggests covr, DBI, dbplyr, knitr, methods, nycflights13, rmarkdown, RSQLite, testthat (>= 3.2.0)

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Edgar Ruiz [aut], Max Kuhn [aut, cre]

Maintainer Max Kuhn <max@posit.co>

Repository CRAN

Date/Publication 2023-11-01 14:30:02 UTC

R topics documented:

`add_dummy_variables` *Creates dummy variables*

Description

It uses 'tidyeval' and 'dplyr' to create dummy variables based for categorical variables.

Usage

```
add_dummy_variables(
  df,
  x,
  values = c(),
  auto_values = FALSE,
  remove_original = TRUE
)
```

Arguments

`df`	A Local or remote data frame
`x`	Categorical variable
`values`	Possible known values of the categorical variable. If not passed then the function will take an additional step to figure the unique values of the variable.
`auto_values`	Safeguard argument to prevent the function from figuring the unique values if the values argument is empty. If it is ok for this function to obtain the unique values, set to TRUE. Defaults to FALSE.
`remove_original`	It removes the original variable from the returned table. Defaults to TRUE.

Examples

```
library(dplyr)

mtcars %>%
  add_dummy_variables(cyl, values = c(4, 6, 8))

mtcars %>%
  add_dummy_variables(cyl, auto_values = TRUE)
```

as_parsed_model.modeldb_lm *Prepares parsed model object*

Description

Prepares parsed model object

Usage

```
## S3 method for class 'modeldb_lm'
as_parsed_model(x)
```

Arguments

x	A parsed model object

linear_regression_db *Fits a Linear Regression model*

Description

It uses 'tidyeval' and 'dplyr' to create a linear regression model.

Usage

```
linear_regression_db(df, y_var = NULL, sample_size = NULL, auto_count = FALSE)
```

Arguments

df	A Local or remote data frame
y_var	Dependent variable
sample_size	Prevents a table count. It is only used for models with three or more independent variables
auto_count	Serves as a safeguard in case sample_size is not passed inadvertently. Defaults to FALSE. If it is ok for the function to count how many records are in the sample, then set to TRUE. It is only used for models with three or more independent variables

Details

The linear_regression_db() function only calls one of three unexported functions. The function used is determined by the number of independent variables. This is so any model of one or two variables can use a simpler formula, which in turn will have less SQL overhead.

Examples

```
library(dplyr)

mtcars %>%
  select(mpg, wt, qsec) %>%
  linear_regression_db(mpg)
```

`plot_kmeans` *Visualize a KMeans Cluster with lots of data*

Description

It uses 'ggplot2' to display the results of a KMeans routine. Instead of a scatterplot, it uses a square grid that displays the concentration of intersections per square. The number of squares in the grid can be customized for more or less fine grain.

Usage

```
plot_kmeans(df, x, y, resolution = 50, group = center)

db_calculate_squares(df, x, y, group, resolution = 50)
```

Arguments

`df`	A Local or remote data frame with results of KMeans clustering
`x`	A numeric variable for the x axis
`y`	A numeric variable for the y axis
`resolution`	The number of squares in the grid. Defaults to 50. Meaning a 50 x 50 grid.
`group`	A discrete variable containing the grouping for the KMeans. It defaults to 'center'

Details

For large result-sets in remote sources, downloading every intersection will be a long running, costly operation. The approach of this function is to devide the x and y plane in a grid and have the remote source figure the total number of intersections, returned as a single number. This reduces the granularity of the visualization, but it speeds up the results.

Examples

```
plot_kmeans(mtcars, mpg, wt, group = am)
```

`simple_kmeans_db` *Simple kmeans routine that works in-database*

Description

It uses 'tidyeval' and 'dplyr' to run multiple cycles of kmean calculations, expressed in dplyr formulas until an the optimal centers are found.

Usage

```
simple_kmeans_db(
  df,
  ...,
  centers = 3,
  max_repeats = 100,
  initial_kmeans = NULL,
  safeguard_file = "kmeans.csv",
  verbose = TRUE
)
```

Arguments

`df`	A Local or remote data frame
`...`	A list of variables to be used in the kmeans algorithm
`centers`	The number of centers. Defaults to 3.
`max_repeats`	The maximum number of cycles to run. Defaults to 100.
`initial_kmeans`	A local dataframe with initial centroid values. Defaults to NULL.
`safeguard_file`	Each cycle will update a file specified in this argument with the current centers. Defaults to 'kmeans.csv'. Pass NULL if no file is desired.
`verbose`	Indicates if the progress bar will be displayed during the model's fitting.

Details

Because each cycle is an independent 'dplyr' operation, or SQL operation if using a remote source, the latest centroid data frame is saved to the parent environment in case the process needs to be canceled and then restarted at a later point. Passing the `current_kmeans` as the `initial_kmeans` will allow the operation to pick up where it left off.

Examples

```
library(dplyr)

mtcars %>%
  simple_kmeans_db(mpg, qsec, wt) %>%
  glimpse()
```

Index