Package 'healthyR.ai'

September 3, 2021

Title The Machine Learning and AI Modeling Companion to 'healthyR'

Version 0.0.2

Description
Hospital machine learning and ai data analysis workflow tools, modeling, and automations. This library provides many useful tools to review common administrative hospital data. Some of these include predicting length of stay, and readmits. The aim is to provide a simple and consistent verb framework that takes the guesswork out of everything.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

URL https://github.com/spsanderson/healthyR.ai

BugReports https://github.com/spsanderson/healthyR.ai/issues

Imports magrittr, rlang (>= 0.1.2), utils, broom, ggrepel, tibble, dplyr, ggplot2, tidyr, forcats, timetk, recipes, parsnip, rsample, purrr, h2o

Suggests rmarkdown, knitr, roxygen2, stats, tidyquant, cli, crayon, rstudioapi, healthyR.data, scales

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Steven Sanderson [aut, cre, cph]

Maintainer Steven Sanderson <spsanderson@gmail.com>

Repository CRAN

Date/Publication 2021-09-03 00:10:19 UTC

R topics documented:

`get_juiced_data` *Get the Juiced Data*

Description

This is a simple function that will get the juiced data from a recipe.

Usage

```
get_juiced_data(.recipe_object)
```

Arguments

`.recipe_object`	The recipe object you want to pass.

Details

Instead of typing out something like: `recipe_object %>% prep() %>% juice() %>% glimpse()`

Value

A tibble of the prepped and juiced data from the given recipe

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Data Wrangling: `pca_your_recipe()`

Examples

```
suppressPackageStartupMessages(library(timetk))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(healthyR.data))
suppressPackageStartupMessages(library(rsample))
suppressPackageStartupMessages(library(recipes))

data_tbl <- healthyR_data %>%
    select(visit_end_date_time) %>%
    summarise_by_time(
        .date_var = visit_end_date_time,
        .by       = "month",
        value     = n()
    ) %>%
    set_names("date_col","value") %>%
    filter_by_time(
        .date_var = date_col,
        .start_date = "2013",
        .end_date = "2020"
    )

splits <- initial_split(data = data_tbl, prop = 0.8)

rec_obj <- recipe(value ~., training(splits))

get_juiced_data(rec_obj)
```

`hai_control_chart` *Create a control chart*

Description

Create a control chart, aka Shewhart chart: https://en.wikipedia.org/wiki/Control_chart.

Usage

```
hai_control_chart(
  .data,
  .value_col,
  .x_col,
  .center_line = mean,
  .std_dev = 3,
  .plt_title = NULL,
  .plt_catpion = NULL,
  .plt_font_size = 11,
  .print_plot = TRUE
)
```

Arguments

.data	data frame or a path to a csv file that will be read in
.value_col	variable of interest mapped to y-axis (quoted, ie as a string)
.x_col	variable to go on the x-axis, often a time variable. If unspecified row indices will be used (quoted)
.center_line	Function used to calculate central tendency. Defaults to mean
.std_dev	Number of standard deviations above and below the central tendency to call a point influenced by "special cause variation." Defaults to 3
.plt_title	Plot title
.plt_catpion	Plot caption
.plt_font_size	Font size; text elements will be scaled to this
.print_plot	Print the plot? Default = TRUE. Set to FALSE if you want to assign the plot to a variable for further modification, as in the last example.

Details

Control charts, also known as Shewhart charts (after Walter A. Shewhart) or process-behavior charts, are a statistical process control tool used to determine if a manufacturing or business process is in a state of control. It is more appropriate to say that the control charts are the graphical device for Statistical Process Monitoring (SPM). Traditional control charts are mostly designed to monitor process parameters when underlying form of the process distributions are known. However, more advanced techniques are available in the 21st century where incoming data streaming can-be monitored even without any knowledge of the underlying process distributions. Distribution-free control charts are becoming increasingly popular.

Value

Generally called for the side effect of printing the control chart. Invisibly, returns a ggplot object for further customization.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
data_tbl <- tibble::tibble(
    day = sample(c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"),
                 100, TRUE),
    person = sample(c("Tom", "Jane", "Alex"), 100, TRUE),
    count = rbinom(100, 20, ifelse(day == "Friday", .5, .2)),
    date = Sys.Date() - sample.int(100))

hai_control_chart(.data = data_tbl, .value_col = count, .x_col = date)

# In addition to printing or writing the plot to file, hai_control_chart
# returns the plot as a ggplot2 object, which you can then further customize
```

```
library(ggplot2)
my_chart <- hai_control_chart(data_tbl, count, date)
my_chart +
  ylab("Number of Adverse Events") +
  scale_x_date(name = "Week of ... ", date_breaks = "week") +
  theme(axis.text.x = element_text(angle = -90, vjust = 0.5, hjust=1))
```

`hai_kmeans_automl` *Automatic K-Means H2O*

Description

This is a wrapper around the `h2o::h2o.kmeans()` function that will return a list object with a lot of useful and easy to use tidy style information.

Usage

```
hai_kmeans_automl(
  .data,
  .split_ratio = 0.8,
  .seed = 1234,
  .centers = 10,
  .standardize = TRUE,
  .print_model_summary = TRUE,
  .predictors,
  .categorical_encoding = "auto",
  .initialization_mode = "Furthest",
  .max_iterations = 100
)
```

Arguments

`.data`	The data that is to be passed for clustering.
`.split_ratio`	The ratio for training and testing splits.
`.seed`	The default is 1234, but can be set to any integer.
`.centers`	The default is 1. Specify the number of clusters (groups of data) in a data set.
`.standardize`	The default is set to TRUE. When TRUE all numeric columns will be set to zero mean and unit variance.
`.print_model_summary`	This is a boolean and controls if the model summary is printed to the console. The default is TRUE.
`.predictors`	This must be in the form of c("column_1", "column_2", ... "column_n")
`.categorical_encoding`	Can be one of the following:

- "auto"
- "enum"
- "one_hot_explicit"
- "binary"
- "eigen"
- "label_encoder"
- "sort_by_response"
- "enum_limited"

`.initialization_mode`
: This can be one of the following:

- "Random"
- "Furthest (default)
- "PlusPlus"

`.max_iterations`
: The default is 100. This specifies the number of training iterations

Value

A list object

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Kmeans: `hai_kmeans_automl_predict()`, `hai_kmeans_mapped_tbl()`, `hai_kmeans_obj()`, `hai_kmeans_scree_data_tbl()`, `hai_kmeans_scree_plt()`, `hai_kmeans_tidy_tbl()`, `hai_kmeans_user_item_tbl()`

Examples

```
## Not run:
h2o.init()
output <- hai_kmeans_automl(
    .data = iris,
    .predictors = c("Sepal.Width","Sepal.Length","Petal.Width","Petal.Length"),
    .standardize = FALSE
)
h2o.shutdown()

## End(Not run)
```

`hai_kmeans_automl_predict` *Automatic K-Means H2O*

Description

This is a wrapper around the `h2o::h2o.predict()` function that will return a list object with a lot of useful and easy to use tidy style information.

Usage

```
hai_kmeans_automl_predict(.input)
```

Arguments

`.input`	This is the output of the `hai_kmeans_automl()` function.

Details

This function will internally take in the output assigned from the `hai_kmeans_automl()` function only and return a list of useful information. The items that are returned are as follows:

1. prediction - The h2o dataframe of predictions
2. prediction_tbl - The h2o predictions in tibble format
3. valid_tbl - The validation data in tibble format
4. pred_full_tbl - The entire validation set with the predictions attached using `base::cbind()`. The predictions are in a column called `predicted_cluster` and are in the formate of a factor using `forcats::as_factor()`

Value

A list object

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Kmeans: `hai_kmeans_automl()`, `hai_kmeans_mapped_tbl()`, `hai_kmeans_obj()`, `hai_kmeans_scree_data_tbl()`, `hai_kmeans_scree_plt()`, `hai_kmeans_tidy_tbl()`, `hai_kmeans_user_item_tbl()`

Examples

```
## Not run:
h2o.init()

output <- hai_kmeans_automl(
    .data = iris,
    .predictors = c("Sepal.Width","Sepal.Length","Petal.Width","Petal.Length"),
    .standardize = FALSE
)

pred <- hai_kmeans_automl_predict(output)

h2o.shutdown()

## End(Not run)
```

---

hai_kmeans_mapped_tbl *K-Means Mapping Function*

Description

Create a tibble that maps the `hai_kmeans_obj()` using `purrr::map()` to create a nested data.frame/tibble that holds n centers. This tibble will be used to help create a scree plot.

Usage

```
hai_kmeans_mapped_tbl(.data, .centers = 15)
```

Arguments

`.data`	You must have a tibble in the working environment from the `hai_kmeans_user_item_tbl()`
`.centers`	How many different centers do you want to try

Details

Takes in a single parameter of .centers. This is used to create the tibble and map the `hai_kmeans_obj()` function down the list creating a nested tibble.

Value

A nested tibble

Author(s)

Steven P. Sanderson II, MPH

See Also

https://en.wikipedia.org/wiki/Scree_plot

Other Kmeans: hai_kmeans_automl_predict(), hai_kmeans_automl(), hai_kmeans_obj(), hai_kmeans_scree_data_tbl(), hai_kmeans_scree_plt(), hai_kmeans_tidy_tbl(), hai_kmeans_user_item_tbl()

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

ui_tbl <-  hai_kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

hai_kmeans_mapped_tbl(ui_tbl)
```

---

hai_kmeans_obj *K-Means Object*

Description

Takes the output of the hai_kmeans_user_item_tbl() function and applies the k-means algorithm to it using stats::kmeans()

Usage

```
hai_kmeans_obj(.data, .centers = 5)
```

Arguments

.data	The data that gets passed from hai_kmeans_user_item_tbl()
.centers	How many initial centers to start with

Details

Uses the stats::kmeans() function and creates a wrapper around it.

Value

A stats k-means object

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Kmeans: hai_kmeans_automl_predict(), hai_kmeans_automl(), hai_kmeans_mapped_tbl(), hai_kmeans_scree_data_tbl(), hai_kmeans_scree_plt(), hai_kmeans_tidy_tbl(), hai_kmeans_user_item_tbl()

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

hai_kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 ) %>%
 hai_kmeans_obj()
```

---

hai_kmeans_scree_data_tbl

K-Means Scree Plot Data Table

Description

Take data from the hai_kmeans_mapped_tbl() and unnest it into a tibble for inspection and for use in the hai_kmeans_scree_plt() function.

Usage

```
hai_kmeans_scree_data_tbl(.data)
```

Arguments

.data	You must have a tibble in the working environment from the hai_kmeans_mapped_tbl()

Details

Takes in a single parameter of .data from `hai_kmeans_mapped_tbl()` and transforms it into a tibble that is used for `hai_kmeans_scree_plt()`. It will show the values (tot.withinss) at each center.

Value

A nested tibble

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Kmeans: `hai_kmeans_automl_predict()`, `hai_kmeans_automl()`, `hai_kmeans_mapped_tbl()`, `hai_kmeans_obj()`, `hai_kmeans_scree_plt()`, `hai_kmeans_tidy_tbl()`, `hai_kmeans_user_item_tbl()`

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

ui_tbl <-  hai_kmeans_user_item_tbl(
   .data          = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmm_tbl <- hai_kmeans_mapped_tbl(ui_tbl)

hai_kmeans_scree_data_tbl(kmm_tbl)
```

---

hai_kmeans_scree_plt *K-Means Scree Plot*

Description

Create a scree-plot from the `hai_kmeans_mapped_tbl()` function.

Usage

```
hai_kmeans_scree_plt(.data)
```

Arguments

.data	The data from the hai_kmeans_mapped_tbl() function

Details

Outputs a scree-plot

Value

A ggplot2 plot

Author(s)

Steven P. Sanderson II, MPH

See Also

https://en.wikipedia.org/wiki/Scree_plot

Other Kmeans: hai_kmeans_automl_predict(), hai_kmeans_automl(), hai_kmeans_mapped_tbl(), hai_kmeans_obj(), hai_kmeans_scree_data_tbl(), hai_kmeans_tidy_tbl(), hai_kmeans_user_item_tbl()

Examples

```
library(healthyR.data)
library(dplyr)
library(tidyquant)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

ui_tbl <-  hai_kmeans_user_item_tbl(
   .data          = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
 )

kmm_tbl <- hai_kmeans_mapped_tbl(ui_tbl)

hai_kmeans_scree_plt(.data = kmm_tbl)
```

hai_kmeans_tidy_tbl *K-Means Object Tidy Functions*

Description

K-Means tidy functions

Usage

```
hai_kmeans_tidy_tbl(.kmeans_obj, .data, .tidy_type = "tidy")
```

Arguments

.kmeans_obj	A stats::kmeans() object
.data	The user item tibble created from hai_kmeans_user_item_tbl()
.tidy_type	"tidy","glance", or "augment"

Details

Takes in a k-means object and its associated user item tibble and then returns one of the items asked for. Either: broom::tidy(), broom::glance() or broom::augment(). The function defaults to broom::tidy().

Value

A tibble

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Kmeans: hai_kmeans_automl_predict(), hai_kmeans_automl(), hai_kmeans_mapped_tbl(), hai_kmeans_obj(), hai_kmeans_scree_data_tbl(), hai_kmeans_scree_plt(), hai_kmeans_user_item_tbl()

Examples

```
library(healthyR.data)
library(dplyr)
library(broom)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
```

```
   as_tibble()

uit_tbl <- hai_kmeans_user_item_tbl(
   .data           = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
)

km_obj  <- hai_kmeans_obj(uit_tbl)

hai_kmeans_tidy_tbl(
   .kmeans_obj  = km_obj
   , .data      = uit_tbl
   , .tidy_type = "augment"
 )

hai_kmeans_tidy_tbl(
   .kmeans_obj  = km_obj
   , .data      = uit_tbl
   , .tidy_type = "glance"
 )

hai_kmeans_tidy_tbl(
   .kmeans_obj  = km_obj
   , .data      = uit_tbl
   , .tidy_type = "tidy"
 ) %>%
   glimpse()
```

`hai_kmeans_user_item_tbl` *K-Means User Item Tibble*

Description

Takes in a data.frame/tibble and transforms it into an aggregated/normalized user-item tibble of proportions. The user will need to input the parameters for the rows/user and the columns/items.

Usage

```
hai_kmeans_user_item_tbl(.data, .row_input, .col_input, .record_input)
```

Arguments

`.data`	The data that you want to transform
`.row_input`	The column that is going to be the row (user)
`.col_input`	The column that is going to be the column (item)

`.record_input` The column that is going to be summed up for the aggregation and normalization process.

Details

This function should be used before using a k-mean model. This is commonly referred to as a user-item matrix because "users" tend to be on the rows and "items" (e.g. orders) on the columns. You must supply a column that can be summed for the aggregation and normalization process to occur.

Value

A aggregated/normalized user item tibble

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Kmeans: `hai_kmeans_automl_predict()`, `hai_kmeans_automl()`, `hai_kmeans_mapped_tbl()`, `hai_kmeans_obj()`, `hai_kmeans_scree_data_tbl()`, `hai_kmeans_scree_plt()`, `hai_kmeans_tidy_tbl()`

Examples

```
library(healthyR.data)
library(dplyr)

data_tbl <- healthyR_data%>%
   filter(ip_op_flag == "I") %>%
   filter(payer_grouping != "Medicare B") %>%
   filter(payer_grouping != "?") %>%
   select(service_line, payer_grouping) %>%
   mutate(record = 1) %>%
   as_tibble()

hai_kmeans_user_item_tbl(
   .data          = data_tbl
   , .row_input    = service_line
   , .col_input    =  payer_grouping
   , .record_input = record
)
```

---

pca_your_recipe *Perform PCA*

Description

This is a simple function that will perform PCA analysis on a passed recipe.

Usage

```
pca_your_recipe(.recipe_object, .data, .threshold = 0.75)
```

Arguments

`.recipe_object`	The recipe object you want to pass.
`.data`	The full data set that is used in the original recipe object passed into `.recipe_object` in order to obtain the baked data of the transform.
`.threshold`	A number between 0 and 1. A fraction of the total variance that should be covered by the components.

Details

This is a simple wrapper around some recipes functions to perform a PCA on a given recipe. This function will output a list and return it invisible. All of the components of the analysis will be returned in a list as their own object that can be selected individually. A scree plot is also included. The items that get returned are:

1. pca_transform - This is the pca recipe.
2. variable_loadings
3. variable_variance
4. pca_estimates
5. pca_juiced_estimates
6. pca_baked_data
7. pca_variance_df
8. pca_variance_scree_plt
9. pca_rotation_df

Value

A list object with several components.

Author(s)

Steven P. Sanderson II, MPH

See Also

https://recipes.tidymodels.org/reference/step_pca.html

Other Data Wrangling: `get_juiced_data()`

Examples

```
suppressPackageStartupMessages(library(timetk))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(purrr))
suppressPackageStartupMessages(library(healthyR.data))
suppressPackageStartupMessages(library(rsample))
suppressPackageStartupMessages(library(recipes))
suppressPackageStartupMessages(library(ggplot2))

data_tbl <- healthyR_data %>%
    select(visit_end_date_time) %>%
    summarise_by_time(
        .date_var = visit_end_date_time,
        .by       = "month",
        value     = n()
    ) %>%
    set_names("date_col","value") %>%
    filter_by_time(
        .date_var = date_col,
        .start_date = "2013",
        .end_date = "2020"
    )

splits <- initial_split(data = data_tbl, prop = 0.8)

rec_obj <- recipe(value ~ ., training(splits)) %>%
    step_timeseries_signature(date_col) %>%
    step_rm(matches("(iso$)|(xts$)|(hour)|(min)|(sec)|(am.pm)"))

output_list <- pca_your_recipe(rec_obj, .data = data_tbl)
output_list$pca_variance_scree_plt
```

Index