Package: aai (via r-universe)

September 13, 2024

Type Package

Title Functions, apps, exercises and other R related stuff used in ``AI - Aalborg Intelligence"

Version 0.2.0

Author Ege Rubak, Torben Tvedebrink, Mikkel Meyer Andersen, Lisbeth Fajstrup

Maintainer Ege Rubak <rubak@math.aau.dk>

Description Functions, apps, exercises and other R related stuff used in ``AI - Aalborg Intelligence" The project (2020 - 2026) is supported by the Novo Nordisk Foundation to develop teaching material to be used in the Danish highschools to strengthen the understanding of AI while explaining how basic maths is used in the some popular AI methods.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0

Imports tidyr, dplyr, magrittr, ggplot2, tibble, purrr, spaa, proxy, class, DT, forcats, knitr, patchwork, readr, rio, rmarkdown, shiny, shinyWidgets, shinycssloaders, shinyjs

Depends R (>= 4.0)

Repository https://aalborg-intelligence.r-universe.dev

RemoteUrl https://github.com/aalborg-intelligence/aai

RemoteRef HEAD

RemoteSha 6fcba07a15e3bdfee847539c17ed388bc21699e9

Contents

`aai` *aai: Functions, apps, exercises and other R related stuff used in "AI - Aalborg Intelligence"*

Description

Functions, apps, exercises and other R related stuff used in "AI - Aalborg Intelligence" The project (2020 - 2026) is supported by the Novo Nordisk Foundation to develop teaching material to be used in the Danish high schools to strengthen the understanding of AI while explaining how basic maths is used in the some popular AI methods.

`classification_test_data` *Fictive data set used to demonstrate some concepts on classification*

Description

A data frame containing the length add weight of SOMETHING. In ‘classification_train_data‘ the class is also given.

Usage

```
classification_test_data
```

Format

A data frame with 10 rows and 2 variables:

Længde Length

Vægt Weight

`classification_train_data` *Fictive data set used to demonstrate some concepts on classification*

Description

A data frame containing the length add weight of SOMETHING. In 'classification_train_data' the class is also given.

Usage

```
classification_train_data
```

Format

A data frame with 150 rows and 3 variables:

Længde Length

Vægt Weight

Type Class

`dt_simple` *Simple function for DT output*

Description

Simple function for DT output

Usage

```
dt_simple(tab, ...)
```

Arguments

`tab`	The table to format
`...`	Arguments passed to 'DT::datatable'

dt_table *Short function for DT output*

Description

Short function for DT output

Usage

```
dt_table(tab, ...)
```

Arguments

tab	The table to format
...	Arguments passed to ‘DT::datatable‘

kable_ *Simple function for kable output*

Description

Simple function for kable output

Usage

```
kable_(tab, ...)
```

Arguments

tab	The table to format
...	Arguments passed to ‘knitr::kable‘

kMD_plot *Data for plotting a grid based the mean of the K nearest neighbors*

Description

Data for plotting a grid based the mean of the K nearest neighbors

Usage

```
kMD_plot(K = 3, .train, response = "Type", grid = 100)
```

Arguments

K	Number of neighbors
.train	Training data
response	Name of the class variable
grid	Resolution of the grid - higher values gives finer grid

kNN *Wrapper around 'class::knn'*

Description

Wrapper around 'class::knn'

Usage

```
kNN(K = 3, .test, .train, response = "Type")
```

Arguments

K	Number of nearest neighbors to use
.test	Data that should be classified based on the training data
.train	Annotated training data that should be classified the test data
response	Name of the response variable

kNN.cv *Actual cross-validation function for kNN.*

Description

Actual cross-validation function for kNN.

Usage

```
kNN.cv(K = 3, .train, response = "Type", fold = 10)
```

Arguments

K	Vector of nearest neighbor values (the k in kNN)
.train	The data to use kNN on
response	The variable name of the response
fold	The number of folds to use in cross validation

Examples

```
data(classification_train_data)
K_LOO <- tibble(K = 1:15,
LOO = kNN.loo(K, .train = classification_train_data)
) %>%
rowwise() %>%
mutate(CV = list(kNN.cv(K, .train = classification_train_data)))

K_LOO %>% ggplot(aes(x = factor(K))) +
geom_boxplot(data = unnest(K_LOO, CV), aes(y = CV)) +
geom_point(aes(y = LOO), colour = "#999999") +
labs(x = "K", y = "Accuracy")
```

kNN.loo *Wrapper around 'class::knn.cv' which does Leave one Out (LoO)*

Description

Wrapper around 'class::knn.cv' which does Leave one Out (LoO)

Usage

```
kNN.loo(K = 3, .train, response = "Type")
```

Arguments

K	Number of nearest neighbors to use (can be a vector)
.train	Annotated training data that should be classified the test data
response	Name of the response variable

---

kNN1 *Wrapper around 'class::knn1'*

Description

Wrapper around 'class::knn1'

Usage

```
kNN1(.test, .train, response = "Type")
```

Arguments

.test	Data that should be classified based on the training data
.train	Annotated training data that should be classified the test data
response	Name of the response variable

---

kNN_plot *Visualise a kNN trainer*

Description

Visualise a kNN trainer

Usage

```
kNN_plot(K = 3, .train, response = "Type", grid = 100)
```

Arguments

K	Number of neighbors to use
.train	The training data
response	The name of the response/class variable
grid	The resolution of the grid. Larger numbers gives higher resolution (and slower performance).

Examples

```
k <- 3
kNN_plot(.train = classification_train_data, K = k) %>%
ggplot() + labs(title = paste("K =", k)) +
geom_rect(aes(xmin = Længde_0, xmax = Længde_1, ymin = Vægt_0, ymax = Vægt_1, fill = Type), alpha = 0.3) +
geom_point(data = train, aes(x = Længde, y = Vægt, colour = Type))
```

`meandist_to_k_nearest` *Mean distance to k nearest*

Description

Mean distance to k nearest

Usage

```
meandist_to_k_nearest(
  K = 3,
  .test,
  .train,
  response = "Type",
  dist = FALSE,
  info = TRUE
)
```

Arguments

`K`	Number of nearest neighbors
`.train`	The training data
`return_all`	Logical. Should the distance to the nearest K be returned or just the mean distance of them?

Value

If 'return_all = FALSE' a dataframe of the mean distance to each class of 'response' is returned. If 'return_all = TRUE' a list is returned - 'top_K' is as above, 'all' contains the closest neighbors from each class.

Examples

```
data(classification_train_data)
meandist_to_k_nearest_(K = 3, .train = classification_train_data) %>%
  mutate(same_Type = ifelse(obs_Type == Type, "Y", "N")) %>%
  ggplot(aes(x = obs_Type, y = Distance, fill = Type, colour = same_Type)) +
  labs(x = "Type of the observation", fill = "Type of the nearest points") +
  theme(legend.position = "top") +
 guides(colour = FALSE) + scale_colour_manual(values = c("Y" = "#666666", "N" = "#000000")) +
  geom_boxplot() + coord_flip()
```

meandist_to_k_nearest_ *Mean distance to k nearest*

Description

Mean distance to k nearest

Usage

```
meandist_to_k_nearest_(K = 5, .train, response = "Type", return_all = FALSE)
```

Arguments

K	Number of nearest neighbors
.train	The training data
return_all	Logical. Should the distance to the nearest K be returned or just the mean distance of them?

Value

If 'return_all = FALSE' a dataframe of the mean distance to each class of 'response' is returned. If 'return_all = TRUE' a list is returned - 'top_K' is as above, 'all' contains the closest neighbors from each class.

perceptron31 *Fictive data set used to demonstrate some concepts in perceptron document*

Description

A data frame containing the responses to two fictive questions on the scale -2,-1,0,1,2 together with a classification color.

Usage

```
perceptron31
```

Format

A data frame with 31 rows and 3 variables:

x1 Answer to first question.

x2 Answer to second question.

col Class

predict_grid *Create grid for new data*

Description

Create grid for new data

Usage

```
predict_grid(pred_grd, newdata)
```

Arguments

pred_grd	Returned from ?
newdata	New data to be used in prediction

pred_grid *Helper function for making preditive grid*

Description

Helper function for making preditive grid

Usage

```
pred_grid(
  data,
  step = 10,
  response = "Type",
  pred_var = "Prediction",
  center = 0
)
```

Arguments

data	Dataset
step	Step size in each data variable
response	The name of the response variable
center	If not through zero, then through ‘center‘

pred_max *Method for predicting the majority vote or "?" if ties*

Description

Method for predicting the majority vote or "?" if ties

Usage

```
pred_max(n, x)
```

Arguments

n	Counts
x	Data vector

pred_plot_grid *Helper function for making preditive grid*

Description

Helper function for making preditive grid

Usage

```
pred_plot_grid(pred_grd, pred_var = "Prediction", remove = TRUE)
```

Arguments

pred_grd	Output from YY function
remove	Is parsed to the ‘remove‘ argument of ‘tidyr::separate()‘

Print *Makes print return all rows in a tibble*

Description

Makes print return all rows in a tibble

Usage

```
Print(...)
```

Arguments

...	Arguments passed to ‘knitr::kable‘

seq_cut *Create discretised version with some pretty labels*

Description

Create discretised version with some pretty labels

Usage

```
seq_cut(x, step, center, breaks = FALSE)
```

Arguments

x	Data variable
step	Step size
center	If not through zero, then through ‘center‘
breaks	Logical. Should break labels we returned?

Examples

```
seq_cut(rnorm(100), step = 2, center = 0, breaks = TRUE)
```

seq_zero *Create breaks for ‘seq_cut‘*

Description

Create breaks for ‘seq_cut‘

Usage

```
seq_zero(x, step, center)
```

Arguments

x	Data variable
step	Step size
center	If not through zero, then through ‘center‘

Examples

```
seq_zero(rnorm(100), step = 2, center = 0)
```

xy_plot *Plot of data for exercise by Jan B Sørensen on classification*

Description

Plot of data for exercise by Jan B Sørensen on classification

Usage

```
xy_plot(train, x, y, colour, test = NULL, selected = NULL)
```

Arguments

train	Training data set
x, y, colour	parameters controlling the x and y axis and point colours
test	Test data set
selected	points to highlight

Examples

```
data(classification_train_data)
data(classification_test_data)
type_cols <- c("1" = "#E41A1C", "2" = "#377EB8", "3" = "#4DAF4A", "?" = "#444444")
xy_plot(train = classification_train_data, x = Længde, y = Vægt, colour = Type) +
  scale_colour_manual(values = type_cols)
```

Index