Package ‘viraldomain’

January 21, 2024

Title Applicability Domain Methods of Viral Load and CD4 Lymphocytes

Version 0.0.3

Description Provides methods for assessing the applicability domain of models that predict viral load and CD4 (Cluster of Differentiation 4) lymphocyte counts. These methods help determine the extent of extrapolation when making predictions.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 2.10)

LazyData true

Imports applicable, dplyr, earth, ggplot2, kknn, nnet, parsnip, recipes, stats, vdiffr, workflows

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Juan Pablo Acuña González [aut, cre] (<`https://orcid.org/0009-0003-6029-6560`>)

Maintainer Juan Pablo Acuña González <`22253567@uagro.mx`>

Repository CRAN

Date/Publication 2024-01-21 13:00:08 UTC

R topics documented:

`knn_domain_score` *Calculate the K-Nearest Neighbor model domain applicability score*

Description

This function fits a K-Nearest Neighbor (KNN) model to the provided data and computes a domain applicability score based on PCA distances.

Usage

```
knn_domain_score(
  featured,
  train_data,
  knn_hyperparameters,
  test_data,
  threshold_value
)
```

Arguments

`featured`	The name of the response variable to predict.
`train_data`	The training dataset containing predictor variables and the response variable.
`knn_hyperparameters`	A list of hyperparameters for the KNN model, including: • `neighbors`: The number of neighbors to consider. • `weight_func`: The weight function to use. • `dist_power`: The distance power parameter.
`test_data`	The test dataset for making predictions.
`threshold_value`	The threshold value used for computing domain scores.

Value

A data frame containing the computed domain scores for each observation in the test dataset.

Examples

```
set.seed(123)
library(dplyr)
featured <- "cd_2022"
# Adding jitter to original features
train_data = viral |>
transmute(cd_2022 = jitter(cd_2022), vl_2022 = jitter(vl_2022))
test_data = sero |>
transmute(cd_2022 = jitter(cd_2022), vl_2022 = jitter(vl_2022))
```

```
knn_hyperparameters <- list(neighbors = 5, weight_func = "optimal", dist_power = 0.3304783)
threshold_value <- 0.99
# Call the function
knn_domain_score(featured, train_data, knn_hyperparameters, test_data, threshold_value)
```

`mars_domain_score` *Calculate the MARS model domain applicability score*

Description

This function fits a MARS (Multivariate Adaptive Regression Splines) model to the provided data and computes a domain applicability score based on PCA distances.

Usage

```
mars_domain_score(
  featured_col,
  train_data,
  mars_hyperparameters,
  test_data,
  threshold_value
)
```

Arguments

`featured_col`	The name of the featured column.
`train_data`	A data frame containing the training data.
`mars_hyperparameters`	A list of hyperparameters for the MARS model, including: • `num_terms`: The number of terms to include in the MARS model. • `prod_degree`: The degree of interaction terms to include. • `prune_method`: The method used for pruning the MARS model.
`test_data`	A data frame containing the test data.
`threshold_value`	The threshold value for the domain score.

Value

A tibble with the domain applicability scores.

Examples

```
library(viraldomain)
library(dplyr)

# Set the seed for reproducibility
set.seed(1234)

# Create a tibble with the training data
data(viral)

# Number of imputations needed
num_imputations <- sum(viral$vl_2022 <= 40)  # Count values below 40 cpm

# Impute unique values
imputed_values <- unique(rexp(num_imputations, rate = 1/13))

# Create a new tibble with mutated/imputed viral load
imputed_viral <- viral |>
  mutate(imputed_vl_2022 = ifelse(vl_2022 <= 40, imputed_values, vl_2022),
         log10_imputed_vl_2022 = log10(ifelse(vl_2022 <= 40, imputed_values, vl_2022)),
         jittered_log10_imputed_vl_2022 = jitter(log10_imputed_vl_2022))

# Create a new tibble with mutated/imputed cd4 counts
imputed_viral <- imputed_viral |>
  mutate(
    jittered_cd_2022 = ifelse(
    duplicated(cd_2022),
    cd_2022 + sample(1:100, length(cd_2022), replace = TRUE),
    cd_2022
    )
  )

# New data frame with mutated/imputed columns
imp_viral <- imputed_viral |>
select(jittered_cd_2022, jittered_log10_imputed_vl_2022) |>
scale() |>
as.data.frame()

# Set the seed for reproducibility
set.seed(1234)

# Create a tibble with the testing data
data(sero)

# Number of imputations needed
num_imputations <- sum(sero$vl_2022 <= 40)  # Count values below 40 cpm

# Impute unique values
imputed_values <- unique(rexp(num_imputations, rate = 1/13))

# Create a new tibble with mutated/imputed viral load
imputed_sero <- sero |>
```

```
  mutate(imputed_vl_2022 = ifelse(vl_2022 <= 40, imputed_values, vl_2022),
         log10_imputed_vl_2022 = log10(ifelse(vl_2022 <= 40, imputed_values, vl_2022)),
         jittered_log10_imputed_vl_2022 = jitter(log10_imputed_vl_2022))

# Create a new tibble with mutated/imputed cd
imputed_sero <- imputed_sero |>
  mutate(
    jittered_cd_2022 = ifelse(
    duplicated(cd_2022),
    cd_2022 + sample(1:100, length(cd_2022), replace = TRUE),
    cd_2022
    )
  )

# New data frame with mutated/imputed columns
imp_sero <- imputed_sero |>
select(jittered_cd_2022, jittered_log10_imputed_vl_2022) |>
scale() |>
as.data.frame()

# Specify your function parameters
featured_col <- "jittered_cd_2022"
train_data <- imp_viral
mars_hyperparameters <- list(num_terms = 3, prod_degree = 1, prune_method = "none")
test_data <- imp_sero
threshold_value <- 0.99

# Call the function
mars_domain_score(featured_col, train_data, mars_hyperparameters, test_data, threshold_value)
```

---

nn_domain_score *Calculate the Neural Network model domain applicability score*

Description

This function fits a Neural Network model to the provided data and computes a domain applicability score based on PCA distances.

Usage

```
nn_domain_score(
  featured_col,
  train_data,
  nn_hyperparameters,
  test_data,
  threshold_value
)
```

Arguments

`featured_col`	The name of the featured column in the training data.
`train_data`	The training data used to fit the Neural Network model.
`nn_hyperparameters`	A list of Neural Network hyperparameters, including hidden_units, penalty, and epochs.
`test_data`	The testing domain data used to calculate the domain applicability score.
`threshold_value`	The threshold value for domain applicability scoring.

Value

A tibble with the domain applicability scores.

Examples

```
library(viraldomain)
library(dplyr)

# Set the seed for reproducibility
set.seed(1234)

# Create a tibble with the training data
data(viral)

# Number of imputations needed
num_imputations <- sum(viral$vl_2022 <= 40)  # Count values below 40 cpm

# Impute unique values
imputed_values <- unique(rexp(num_imputations, rate = 1/13))

# Create a new tibble with mutated/imputed viral load
imputed_viral <- viral |>
  mutate(imputed_vl_2022 = ifelse(vl_2022 <= 40, imputed_values, vl_2022),
         log10_imputed_vl_2022 = log10(ifelse(vl_2022 <= 40, imputed_values, vl_2022)),
         jittered_log10_imputed_vl_2022 = jitter(log10_imputed_vl_2022))

# Create a new tibble with mutated/imputed cd4 counts
imputed_viral <- imputed_viral |>
  mutate(
    jittered_cd_2022 = ifelse(
    duplicated(cd_2022),
    cd_2022 + sample(1:100, length(cd_2022), replace = TRUE),
    cd_2022
    )
  )

# New data frame with mutated/imputed columns
imp_viral <- imputed_viral |>
select(jittered_cd_2022, jittered_log10_imputed_vl_2022) |>
```

```
scale() |>
as.data.frame()

# Set the seed for reproducibility
set.seed(1234)

# Create a tibble with the testing data
data(sero)

# Number of imputations needed
num_imputations <- sum(sero$vl_2022 <= 40)  # Count values below 40 cpm

# Impute unique values
imputed_values <- unique(rexp(num_imputations, rate = 1/13))

# Create a new tibble with mutated/imputed viral load
imputed_sero <- sero |>
  mutate(imputed_vl_2022 = ifelse(vl_2022 <= 40, imputed_values, vl_2022),
         log10_imputed_vl_2022 = log10(ifelse(vl_2022 <= 40, imputed_values, vl_2022)),
         jittered_log10_imputed_vl_2022 = jitter(log10_imputed_vl_2022))

# Create a new tibble with mutated/imputed cd
imputed_sero <- imputed_sero |>
  mutate(
    jittered_cd_2022 = ifelse(
    duplicated(cd_2022),
    cd_2022 + sample(1:100, length(cd_2022), replace = TRUE),
    cd_2022
    )
  )

# New data frame with mutated/imputed columns
imp_sero <- imputed_sero |>
select(jittered_cd_2022, jittered_log10_imputed_vl_2022) |>
scale() |>
as.data.frame()

# Specify your function parameters
featured_col <- "jittered_cd_2022"
train_data <- imp_viral
nn_hyperparameters <- list(hidden_units = 1, penalty = 0.3746312,  epochs =  480)
test_data <- imp_sero
threshold_value <- 0.99

# Call the function
nn_domain_score(featured_col, train_data, nn_hyperparameters, test_data, threshold_value)
```

normalized_domain_plot

Create a Normalized Domain Plot

Description

This function generates a domain plot for a normalized model based on PCA distances of the provided data.

Usage

```
normalized_domain_plot(
  features,
  train_data,
  test_data,
  treshold_value,
  impute_hyperparameters
)
```

Arguments

`features` A list containing the following elements:

- `featured_col`: The name of the featured column.
- `features_vl`: A character vector of feature names related to viral load.
- `features_cd`: A character vector of feature names related to cluster of differentiation.

`train_data` A data frame containing the training data.

`test_data` A data frame containing the test data.

`treshold_value` The threshold value for the domain plot.

`impute_hyperparameters`
A list of hyperparameters for imputation, including:

- `indetect`: The undetectable viral load level.
- `tasa_exp`: The exponential distribution rate of undetectable values.
- `semi`: The seed for random number generation (for reproducibility).

Value

A domain plot visualizing the distances of imputed values.

Examples

```
data(viral)
data(sero)
 # Adding "jitter_" prefix to original variable
features <- list(
  featured_col = "jittered_cd_2022",
  features_vl = "vl_2022",
  features_cd = "cd_2022"
  )
train_data = viral |>
dplyr::select("cd_2022", "vl_2022")
test_data = sero
```

```
treshold_value = 0.99
impute_hyperparameters = list(indetect = 40, tasa_exp = 1/13, semi = 123)
normalized_domain_plot(features, train_data, test_data, treshold_value, impute_hyperparameters)
```

`sero` *Seropositive Data for Applicability Domain Testing*

Description

This dataset is designed for testing the applicability domain of methods related to HIV research. It provides a tibble with 53 rows and 2 columns containing numeric measurements of CD4 lymphocyte counts (cd_2022) and viral load (vl_2022) for seropositive individuals in 2022. These measurements are vital indicators of HIV disease status. This dataset is ideal for evaluating the performance and suitability of various HIV-predictive models and as an aid in developing diagnostic tools within a seropositive context.

Usage

```
data(sero)
```

Format

A tibble (data frame) with 53 rows and 2 columns.

Note

To explore more rows of this dataset, you can use the `print(n = ...)` function.

Author(s)

Juan Pablo Acuña González 22253567@uagro.mx

Examples

```
data(sero)
sero
```

simple_domain_plot *Create a Simple Domain Plot*

Description

This function generates a domain plot for a simple model based on PCA distances of the provided data.

Usage

```
simple_domain_plot(
  features,
  train_data,
  test_data,
  treshold_value,
  impute_hyperparameters
)
```

Arguments

`features`	A list of features according to their modeling roles. It should contain the following elements: • 'featured_col': Name of the featured column in the training data. When specifying the featured column, use "jitter_*" as a prefix to the featured variable of interest. • 'features_vl': Names of the columns containing viral load data (numeric values). • 'features_cd': Names of the columns containing CD4 data (numeric values).
`train_data`	The training data used to fit the MARS model.
`test_data`	The testing domain data used to calculate PCA distances.
`treshold_value`	The threshold for domain applicability scoring.
`impute_hyperparameters`	A list of parameters for imputation including 'indetect' (undetectable viral load level), 'tasa_exp' (exponential distribution rate of undetectable values), and 'semi' (set a seed for reproducibility).

Value

A domain plot showing PCA distances.

Examples

```
data(viral)
data(sero)
 # Adding "jitter_" prefix to original variable
features <- list(
  featured_col = "jittered_cd_2022",
  features_vl = "vl_2022",
  features_cd = "cd_2022"
  )
train_data = viral |>
dplyr::select("cd_2022", "vl_2022")
test_data = sero
treshold_value = 0.99
impute_hyperparameters = list(indetect = 40, tasa_exp = 1/13, semi = 123)
simple_domain_plot(features, train_data, test_data, treshold_value, impute_hyperparameters)
```

`train` *Training Data for Applicability Domain Analysis*

Description

This dataset contains training data for viral load models applicability domain analysis. It includes CD4 and viral load measurements for different years.

Usage

```
data(train)
```

Format

A tibble (data frame) with 26 rows and 6 columns.

Note

To explore more rows of this dataset, you can use the `print(n = ...)` function.

Author(s)

Juan Pablo Acuña González 22253567@uagro.mx

Examples

```
data(train)
train
```

`viral` *Predictive Modeling Data for Viral Load and CD4 Lymphocyte Counts*

Description

This dataset serves as input for predictive modeling tasks related to HIV research. It contains numeric measurements of CD4 lymphocyte counts (cd) and viral load (vl) at three different time points: 2019, 2021, and 2022. These measurements are crucial indicators of HIV disease progression.

Usage

```
data(viral)
```

Format

A tibble (data frame) with 35 rows and 6 columns.

Note

To explore more rows of this dataset, you can use the `print(n = ...)` function.

Author(s)

Juan Pablo Acuña González 22253567@uagro.mx

Examples

```
data(viral)
viral
```

Index