

Package ‘applicable’

July 22, 2025

Title A Compilation of Applicability Domain Methods

Version 0.1.1

Description A modeling package compiling applicability domain methods in R. It combines different methods to measure the amount of extrapolation new samples can have from the training set. See Gadaleta et al (2016) <[doi:10.4018/IJQSPR.2016010102](https://doi.org/10.4018/IJQSPR.2016010102)> for an overview of applicability domains.

License MIT + file LICENSE

URL <https://github.com/tidymodels/applicable>,
<https://applicable.tidymodels.org>

BugReports <https://github.com/tidymodels/applicable/issues>

Depends ggplot2, R (>= 3.6)

Imports dplyr, glue, hardhat (>= 1.3.1), Matrix, proxyC, purrr, rlang,
stats, tibble, tidyr, tidyselect, utils

Suggests covr, isotree (>= 0.6.1-1), knitr, modeldata, recipes (>= 1.0.10), rmarkdown, spelling, testthat (>= 3.0.0), xml2

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Author Marly Gotti [aut, cre],
Max Kuhn [aut],
RStudio [cph]

Maintainer Marly Gotti <marlygotti@gmail.com>

Repository CRAN

Date/Publication 2024-04-25 00:00:04 UTC

Contents

ames_new	2
apd_hat_values	3
apd_isolation	4
apd_pca	6
apd_similarity	7
autoplot.apd_pca	9
autoplot.apd_similarity	10
binary	11
okc_binary	11
print.apd_hat_values	12
print.apd_pca	12
print.apd_similarity	13
score	14
score.apd_hat_values	14
score.apd_isolation	15
score.apd_pca	16
score.apd_similarity	17
Index	19

ames_new	<i>Recent Ames Iowa Houses</i>
----------	--------------------------------

Description

More data related to the set described by De Cock (2011) where data were recorded for 2,930 properties in Ames IA.

Details

This data sets includes three more properties added since the original reference. There are less fields in this data set; only those that could be transcribed from the assessor’s office were included.

Value

ames_new a tibble

Source

De Cock, D. (2011). "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project," *Journal of Statistics Education*, Volume 19, Number 3.
<https://www.cityofames.org> (see Assessor’s department site)
<http://jse.amstat.org/v19n3/decock/DataDocumentation.txt>
<http://jse.amstat.org/v19n3/decock.pdf>

apd_hat_values	<i>Fit a apd_hat_values</i>
----------------	-----------------------------

Description

apd_hat_values() fits a model.

Usage

```
apd_hat_values(x, ...)  
  
## Default S3 method:  
apd_hat_values(x, ...)  
  
## S3 method for class 'data.frame'  
apd_hat_values(x, ...)  
  
## S3 method for class 'matrix'  
apd_hat_values(x, ...)  
  
## S3 method for class 'formula'  
apd_hat_values(formula, data, ...)  
  
## S3 method for class 'recipe'  
apd_hat_values(x, data, ...)
```

Arguments

x	Depending on the context: • A data frame of predictors. • A matrix of predictors. • A recipe specifying a set of preprocessing steps created from <code>recipes::recipe()</code>.
...	Not currently used, but required for extensibility.
formula	A formula specifying the predictor terms on the right-hand side. No outcome should be specified.
data	When a recipe or formula is used, data is specified as: • A data frame containing the predictors.

Value

A apd_hat_values object.

Examples

```

predictors <- mtcars[, -1]

# Data frame interface
mod <- apd_hat_values(predictors)

# Formula interface
mod2 <- apd_hat_values(mpg ~ ., mtcars)

# Recipes interface
library(recipes)
rec <- recipe(mpg ~ ., mtcars)
rec <- step_log(rec, disp)
mod3 <- apd_hat_values(rec, mtcars)

```

apd_isolation

Fit an isolation forest to estimate an applicability domain.

Description

apd_isolation() fits an isolation forest model.

Usage

```

apd_isolation(x, ...)

## Default S3 method:
apd_isolation(x, ...)

## S3 method for class 'data.frame'
apd_isolation(x, ...)

## S3 method for class 'matrix'
apd_isolation(x, ...)

## S3 method for class 'formula'
apd_isolation(formula, data, ...)

## S3 method for class 'recipe'
apd_isolation(x, data, ...)

```

Arguments

- x
- Depending on the context:
- A **data frame** of predictors.
 - A **matrix** of predictors (see the `categ_cols` argument of `isotree::isolation.forest()`).
 - A **recipe** specifying a set of preprocessing steps created from `recipes::recipe()`.

...	Options to pass to <code>isotree::isolation.forest()</code> . Options should not include data.
formula	A formula specifying the predictor terms on the right-hand side. No outcome should be specified.
data	When a recipe or formula is used, data is specified as: • A data frame containing the predictors.

Details

In an isolation forest, splits are designed to isolate individual data points. The tree construction process takes random split locations on randomly selected predictors. As splits are made in the tree, the algorithm tracks when data points are isolated as more splits are made. The first points that are isolated are thought to be outliers or anomalous. From these results, an anomaly score can be constructed.

This function creates an isolation forest on the training set and measures the reference distribution of the scores when re-predicting the training set. When scoring new data, the raw anomaly score is produced along with the sample's corresponding percentile of the reference distribution.

Value

A `apd_isolation` object.

References

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." 2008 *Eighth IEEE International Conference on Data Mining. IEEE*, 2008. Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-based anomaly detection." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012): 3.

Examples

```
if (rlang::is_installed(c("isotree", "modeldata"))) {
  library(dplyr)

  data(cells, package = "modeldata")

  cells_tr <- cells %>% filter(case == "Train") %>% select(-case, -class)
  cells_te <- cells %>% filter(case != "Train") %>% select(-case, -class)

  if_mod <- apd_isolation(cells_tr, ntrees = 10, nthreads = 1)
  if_mod
}
```

apd_pca	<i>Fit a apd_pca</i>
---------	----------------------

Description

apd_pca() fits a model.

Usage

```
apd_pca(x, ...)

## Default S3 method:
apd_pca(x, ...)

## S3 method for class 'data.frame'
apd_pca(x, threshold = 0.95, ...)

## S3 method for class 'matrix'
apd_pca(x, threshold = 0.95, ...)

## S3 method for class 'formula'
apd_pca(formula, data, threshold = 0.95, ...)

## S3 method for class 'recipe'
apd_pca(x, data, threshold = 0.95, ...)
```

Arguments

x	Depending on the context: • A data frame of predictors. • A matrix of predictors. • A recipe specifying a set of preprocessing steps created from <code>recipes::recipe()</code>.
...	Not currently used, but required for extensibility.
threshold	A number indicating the percentage of variance desired from the principal components. It must be a number greater than 0 and less or equal than 1.
formula	A formula specifying the predictor terms on the right-hand side. No outcome should be specified.
data	When a recipe or formula is used, data is specified as: • A data frame containing the predictors.

Details

The function computes the principal components that account for up to either 95% or the provided threshold of variability. It also computes the percentiles of the absolute value of the principal components. Additionally, it calculates the mean of each principal component.

Value

A apd_pca object.

Examples

```
predictors <- mtcars[, -1]

# Data frame interface
mod <- apd_pca(predictors)

# Formula interface
mod2 <- apd_pca(mpg ~ ., mtcars)

# Recipes interface
library(recipes)
rec <- recipe(mpg ~ ., mtcars)
rec <- step_log(rec, disp)
mod3 <- apd_pca(rec, mtcars)
```

apd_similarity

Applicability domain methods using binary similarity analysis

Description

apd_similarity() is used to analyze samples in terms of similarity scores for binary data. All features in the data should be binary (i.e. zero or one).

Usage

```
apd_similarity(x, ...)
```

Default S3 method:

```
apd_similarity(x, quantile = NA_real_, ...)
```

S3 method for class 'data.frame'

```
apd_similarity(x, quantile = NA_real_, ...)
```

S3 method for class 'matrix'

```
apd_similarity(x, quantile = NA_real_, ...)
```

S3 method for class 'formula'

```
apd_similarity(formula, data, quantile = NA_real_, ...)
```

S3 method for class 'recipe'

```
apd_similarity(x, data, quantile = NA_real_, ...)
```

Arguments

x	Depending on the context: • A data frame of binary predictors. • A matrix of binary predictors. • A recipe specifying a set of preprocessing steps created from <code>recipes::recipe()</code>.
...	Options to pass to <code>proxyC::simil()</code> , such as <code>method</code> . If no options are specified, <code>method = "jaccard"</code> is used.
quantile	A real number between 0 and 1 or NA for how the similarity values for each sample versus the training set should be summarized. A value of NA specifies that the mean similarity is computed. Otherwise, the appropriate quantile is computed.
formula	A formula specifying the predictor terms on the right-hand side. No outcome should be specified.
data	When a recipe or formula is used, data is specified as: • A data frame containing the binary predictors. Any predictors with no 1's will be removed (with a warning).

Details

The function computes measures of similarity for different samples points. For example, suppose samples A and B both contain p binary variables. First, a 2x2 table is constructed between A and B *across their elements*. The table will contain p entries across the four cells (see the example below). From this, different measures of likeness are computed.

For a training set of n samples, a new sample is compared to each, resulting in n similarity scores. These can be summarized into a single value; the median similarity is used by default by the scoring function.

For this method, the computational methods are fairly taxing for large data sets. The training set must be stored (albeit in a sparse matrix format) so object sizes may become large.

By default, the computations are run in parallel using *all possible cores*. To change this, call the `setThreadOptions` function in the `RcppParallel` package.

Value

A `apd_similarity` object.

References

Leach, A. and Gillet V. (2007). *An Introduction to Chemoinformatics*. Springer, New York

Examples

```
data(qsar_binary)

jacc_sim <- apd_similarity(binary_tr)
jacc_sim
```

```

# plot the empirical cumulative distribution function (ECDF) for the training set:
library(ggplot2)
autoplot(jacc_sim)

# Example calculations for two samples:
A <- as.matrix(binary_tr[1, ])
B <- as.matrix(binary_tr[2, ])
xtab <- table(A, B)
xtab

# Jaccard statistic
xtab[2, 2] / (xtab[1, 2] + xtab[2, 1] + xtab[2, 2])

# Hamman statistic
((xtab[1, 1] + xtab[2, 2]) - (xtab[1, 2] + xtab[2, 1])) / sum(xtab)

# Faith statistic
(xtab[1, 1] + xtab[2, 2] / 2) / sum(xtab)

# Summarize across all training set similarities
mean_sim <- score(jacc_sim, new_data = binary_unk)
mean_sim

```

autoplot.apd_pca

Plot the distribution function for principal components

Description

Plot the distribution function for principal components

Usage

```

## S3 method for class 'apd_pca'
autoplot(object, ...)

```

Arguments

object	An object produced by apd_pca.
...	An optional set of dplyr selectors, such as <code>dplyr::matches()</code> or <code>dplyr::starts_with()</code> for selecting which variables should be shown in the plot.

Value

A ggplot object that shows the distribution function for each principal component.

Examples

```
library(ggplot2)
library(dplyr)
library(modeldata)
data(biomass)

biomass_ad <- apd_pca(biomass[, 3:8])

autoplot(biomass_ad)
# Using selectors in `...`
autoplot(biomass_ad, distance) + scale_x_log10()
autoplot(biomass_ad, matches("PC[1-2]"))
```

```
autoplot.apd_similarity
```

Plot the cumulative distribution function for similarity metrics

Description

Plot the cumulative distribution function for similarity metrics

Usage

```
## S3 method for class 'apd_similarity'
autoplot(object, ...)
```

Arguments

object	An object produced by apd_similarity.
...	Not currently used.

Value

A ggplot object that shows the cumulative probability versus the unique similarity values in the training set. Not that for large samples, this is an approximation based on a random sample of 5,000 training set points.

Examples

```
set.seed(535)
tr_x <- matrix(
  sample(0:1, size = 20 * 50, prob = rep(.5, 2), replace = TRUE),
  ncol = 20
)
model <- apd_similarity(tr_x)
```

`binary`*Binary QSAR Data*

Description

Binary QSAR Data

Details

These data are from two different sources on quantitative structure-activity relationship (QSAR) modeling and contain 67 predictors that are either 0 or 1. The training set contains 4,330 samples and there are five unknown samples (both from the Mutagen data in the QSARdata package).

Value

`binary_tr`, `binary_ukn`
data frame frames with 67 columns

Examples

```
data(qsar_binary)
str(binary_tr)
```

`okc_binary`*OkCupid Binary Predictors*

Description

OkCupid Binary Predictors

Details

Data originally from Kim (2015) includes a training and test set consistent with Kuhn and Johnson (2020). Predictors include ethnicity indicators and a set of keywords derived from text essay data.

Value

`okc_binary_train`, `okc_binary_test`
data frame frames with 61 columns

Source

Kim (2015), "OkCupid Data for Introductory Statistics and Data Science Courses", *Journal of Statistics Education*, Volume 23, Number 2.

Kuhn and Johnson (2020), *Feature Engineering and Selection*, Chapman and Hall/CRC . <https://bookdown.org/max/FES/> and <https://github.com/topepo/FES>

Examples

```
data(okc_binary)
str(okc_binary_train)
```

```
print.apd_hat_values
```

Print number of predictors and principal components used.

Description

Print number of predictors and principal components used.

Usage

```
## S3 method for class 'apd_hat_values'
print(x, ...)
```

Arguments

x	A apd_hat_values object.
...	Not currently used, but required for extensibility.

Value

None

Examples

```
model <- apd_hat_values(~ Sepal.Length + Sepal.Width, iris)
print(model)
```

```
print.apd_pca
```

Print number of predictors and principal components used.

Description

Print number of predictors and principal components used.

Usage

```
## S3 method for class 'apd_pca'
print(x, ...)
```

Arguments

x	A apd_pca object.
...	Not currently used, but required for extensibility.

Value

None

Examples

```
model <- apd_pca(~ Sepal.Length + Sepal.Width, iris)
print(model)
```

print.apd_similarity *Print number of predictors and principal components used.*

Description

Print number of predictors and principal components used.

Usage

```
## S3 method for class 'apd_similarity'
print(x, ...)
```

Arguments

x	A apd_similarity object.
...	Not currently used, but required for extensibility.

Value

None

Examples

```
set.seed(535)
tr_x <- matrix(
  sample(0:1, size = 20 * 50, prob = rep(.5, 2), replace = TRUE),
  ncol = 20
)
model <- apd_similarity(tr_x)
print(model)
```

score	<i>A scoring function</i>
-------	---------------------------

Description

A scoring function

Usage

```
score(object, ...)
```

```
## Default S3 method:
score(object, ...)
```

Arguments

object	Depending on the context: • A data frame of predictors. • A matrix of predictors. • A recipe specifying a set of preprocessing steps created from <code>recipes::recipe()</code>.
...	Not currently used, but required for extensibility.

Value

A tibble of predictions.

score.apd_hat_values	<i>Score new samples using hat values</i>
----------------------	---

Description

Score new samples using hat values

Usage

```
## S3 method for class 'apd_hat_values'
score(object, new_data, type = "numeric", ...)
```

Arguments

object	A apd_hat_values object.
new_data	A data frame or matrix of new predictors.
type	A single character. The type of predictions to generate. Valid options are: • "numeric" for a numeric value that summarizes the hat values for each sample across the training set.
...	Not used, but required for extensibility.

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in new_data. For type = "numeric", the tibble contains two columns hat_values and hat_values_pctl5. The column hat_values_pctl5 is in percent units so that a value of 11.5 indicates that, in the training set, 11.5 percent of the training set samples had smaller values than the sample being scored.

Examples

```
train_data <- mtcars[1:20, ]
test_data <- mtcars[21:32, ]

hat_values_model <- apd_hat_values(train_data)

hat_values_scoring <- score(hat_values_model, new_data = test_data)
hat_values_scoring
```

score.apd_isolation	<i>Predict from a apd_isolation</i>
---------------------	-------------------------------------

Description

Predict from a apd_isolation

Usage

```
## S3 method for class 'apd_isolation'
score(object, new_data, type = "numeric", ...)
```

Arguments

object	A apd_isolation object.
new_data	A data frame or matrix of new samples.
type	A single character. The type of predictions to generate. Valid options are: "numeric" for numeric predictions.
...	Not used, but required for extensibility.

Details

About the score

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in new_data. The score column is the raw prediction from `isotree::predict.isolation_forest()` while score_pctl compares this value to the reference distribution of the score created by predicting the training set. A value of X means that X percent of the training data have scores less than the predicted value.

See Also[apd_isolation\(\)](#)**Examples**

```

if (rlang::is_installed(c("isotree", "modeldata"))) {
  library(dplyr)

  data(cells, package = "modeldata")

  cells_tr <- cells %>% filter(case == "Train") %>% select(-case, -class)
  cells_te <- cells %>% filter(case != "Train") %>% select(-case, -class)

  if_mod <- apd_isolation(cells_tr, ntrees = 10, nthreads = 1)
  score(if_mod, cells_te)
}

```

score.apd_pca

*Predict from a apd_pca***Description**

Predict from a apd_pca

Usage

```

## S3 method for class 'apd_pca'
score(object, new_data, type = "numeric", ...)

```

Arguments

object	A apd_pca object.
new_data	A data frame or matrix of new samples.
type	A single character. The type of predictions to generate. Valid options are: • "numeric" for numeric predictions.
...	Not used, but required for extensibility.

Details

The function computes the principal components of the new data and their percentiles as compared to the training data. The number of principal components computed depends on the threshold given at fit time. It also computes the multivariate distance between each principal component and its mean.

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in `new_data`.

Examples

```
train <- mtcars[1:20, ]
test <- mtcars[21:32, -1]

# Fit
mod <- apd_pca(mpg ~ cyl + log(drat), train)

# Predict, with preprocessing
score(mod, test)
```

`score.apd_similarity` *Score new samples using similarity methods*

Description

Score new samples using similarity methods

Usage

```
## S3 method for class 'apd_similarity'
score(object, new_data, type = "numeric", add_percentile = TRUE, ...)
```

Arguments

<code>object</code>	A <code>apd_similarity</code> object.
<code>new_data</code>	A data frame or matrix of new predictors.
<code>type</code>	A single character. The type of predictions to generate. Valid options are: "numeric" for a numeric value that summarizes the similarity values for each sample across the training set.
<code>add_percentile</code>	A single logical; should the percentile of the similarity score <i>relative to the training set values</i> be computed?
<code>...</code>	Not used, but required for extensibility.

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in `new_data`. For `type = "numeric"`, the tibble contains a column called "similarity". If `add_percentile = TRUE`, an additional column called `similarity_pctl` will be added. These values are in percent units so that a value of 11.5 indicates that, in the training set, 11.5 percent of the training set samples had smaller values than the sample being scored.

Examples

```
data(qsar_binary)

jacc_sim <- apd_similarity(binary_tr)

mean_sim <- score(jacc_sim, new_data = binary_unk)
mean_sim
```

Index

* datasets

- ames_new, [2](#)
- binary, [11](#)
- okc_binary, [11](#)

- ames_new, [2](#)
- apd_hat_values, [3](#)
- apd_isolation, [4](#)
- apd_isolation(), [16](#)
- apd_pca, [6](#)
- apd_similarity, [7](#)
- autoplot.apd_pca, [9](#)
- autoplot.apd_similarity, [10](#)

- binary, [11](#)
- binary_tr (binary), [11](#)
- binary_unk (binary), [11](#)

- isotree::isolation.forest(), [4](#), [5](#)
- isotree::predict.isolation_forest(),
[15](#)

- okc_binary, [11](#)
- okc_binary_test (okc_binary), [11](#)
- okc_binary_train (okc_binary), [11](#)

- print.apd_hat_values, [12](#)
- print.apd_pca, [12](#)
- print.apd_similarity, [13](#)

- qsar_binary (binary), [11](#)

- recipes::recipe(), [3](#), [4](#), [6](#), [8](#), [14](#)

- score, [14](#)
- score.apd_hat_values, [14](#)
- score.apd_isolation, [15](#)
- score.apd_pca, [16](#)
- score.apd_similarity, [17](#)