Package 'CASMI'

January 20, 2025

Type Package

Title 'CASMI'-Based Functions

Version 1.2.2

Description Contains Coverage Adjusted Standardized Mutual Information ('CASMI')-based functions. 'CASMI' is a fundamental concept of a series of methods. For more information about 'CASMI' and 'CASMI'-related methods, please refer to the corresponding publications (e.g., a feature selection method, Shi, J., Zhang, J., & Ge, Y. (2019) <doi:10.3390/e21121179>, and a dataset quality measurement method, Shi, J., Zhang, J., & Ge, Y. (2019) <doi:10.1109/ICHI.2019.8904553>) or contact the package author for the latest updates.

Imports EntropyEstimation, entropy, stats

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

Author Jingyi (Catherine) Shi [aut, cre, cph, ctb],
Jialin Zhang [ctb]

Maintainer Jingyi (Catherine) Shi <jshi@math.msstate.edu>

Repository CRAN

Date/Publication 2024-02-16 23:10:16 UTC

Contents

AQI *AQI Index*

Description

A quantitative measure of dataset quality. The AQI Index score indicates the degree that how features are associated with the outcome in a dataset. (synonyms of "feature": "variable" "factor" "attribute")
For more information, please refer to the corresponding publication: Shi, J., Zhang, J. and Ge, Y. (2019), "An Association-Based Intrinsic Quality Index for Healthcare Dataset Ranking" <doi:10.1109/ICHI.2019.8904553>

Usage

```
AQI(data, alpha.filter = 0.2)
```

Arguments

data	data frame (features as columns and observations as rows). The outcome variable (Y) MUST be the last column. It requires at least one features and only one outcome. Both the features (Xs) and the outcome (Y) MUST be discrete (if not naturally discrete, you may try the 'autoBin.binary' function in the same package).
alpha.filter	level of significance for the mutual information test of independence in step 2 (<doi:10.1109/ICHI.2019.8904553>). By default, 'alpha.filter = 0.2'.

Value

The AQI Index score.

Examples

```
## Generate a toy dataset: "data"
n <- 10000
set.seed(1)
x1 <- rbinom(n, 3, 0.5) + 0.2
set.seed(2)
x2 <- rbinom(n, 2, 0.8) + 0.5
set.seed(3)
x3 <- rbinom(n, 5, 0.3)
set.seed(4)
error <- round(runif(n, min=-1, max=1))
y <- x1 + x3 + error
data <- data.frame(cbind(x1, x2, x3, y))
colnames(data) <- c("feature1", "feature2", "feature3", "Y")

## Calculate the AQI score of "data"
AQI(data)
```

autoBin.binary *Auto Binning for Quantitative Variables - Binary*

Description

Automatically suggest an optimal cutting point for categorizing a quantitative variable before using the **CASMI**-based functions. This function does binary cutting, that is, to convert the quantitative variable into a categorical variable with two levels/categories.

Usage

```
autoBin.binary(data, index)
```

Arguments

data	data frame (features as columns and observations as rows). An outcome variable is required. The outcome variable (Y) MUST be the last column.
index	index or a vector of indices of the quantitative variables that need to be automatically categorized.

Value

'autoBin.binary()' returns the entire data frame after automatic binary categorization for the selected quantitative variable(s).

Examples

```
## Use the "iris" dataset embedded in R
data("iris")
newData <- autoBin.binary(iris, c(1,2,3,4))
newData
```

CASMI.selectFeatures **CASMI**-*Based Feature Selection*

Description

Selects the most relevant features toward an outcome. It automatically learns the number of features to be selected, and the selected features are ranked. The method automatically handles the feature redundancy issue. (synonyms of "feature": "variable" "factor" "attribute")
For more information, please refer to the corresponding publication: Shi, J., Zhang, J. and Ge, Y. (2019), "**CASMI**—An Entropic Feature Selection Method in Turing's Perspective" <doi:10.3390/e21121179>

Usage

```
CASMI.selectFeatures(
  data,
  feature.na.handle = "stepwise",
  alpha.filter = 0.1,
  alpha = 0.05,
  intermediate.steps = TRUE,
  kappa.star.cap = 1,
  feature.num.cap = ncol(data)
)
```

Arguments

`data`	data frame (features as columns and observations as rows). The outcome variable (Y) MUST be the last column. It requires at least one features and only one outcome. Both the features (Xs) and the outcome (Y) MUST be discrete (if not naturally discrete, you may try the 'autoBin.binary' function in the same package).
`feature.na.handle`	options for handling NA values in the data. There are three options: '"stepwise", "na.omit", "NA as a category"'. 'feature.na.handle = "stepwise"' excludes NA rows only when a particular variable is being calculated. For example, suppose we have data(Feature1: A, NA, B; Feature2: C, D, E; Feature3: F, G, H; Outcome: O, P, Q); the second observation will be excluded only when a particular step includes Feature1, but will not be excluded when a step calculates among Feature2, Feature3, and the Outcome. This option is designed to take advantage of a maximum number of data points. 'feature.na.handle = "na.omit"' excludes observations with any NA values at the beginning of the analysis. 'feature.na.handle = "NA as a category"' regards the NA value as a new category. This is designed to be used when NA values in the data have a consistent meaning instead of being missing values. For example, in survey data asking for comments, each NA value might consistently mean "no opinion." By default, 'feature.na.handle = "stepwise"'.
`alpha.filter`	level of significance for the mutual information test of independence in step 1 of the features selection (initial screening). The smaller the alpha.filter, the fewer the features sent to step 2 (<doi:10.3390/e21121179>). By default, 'alpha.filter = 0.1'.
`alpha`	level of significance for the confidence intervals in final results. By default, 'alpha = 0.05'.
`intermediate.steps`	output the intermediate process. By default, 'intermediate.steps = TRUE'. Set to 'FALSE' for showing only summary results.
`kappa.star.cap`	a threshold of 'kappa*' for pausing the feature selection process. The program will automatically pause at the first feature of which the 'kappa*' value exceeds the kappa.star.cap threshold. By default, 'kappa.star.cap = 1.0', which is the maximum possible value. A lower value may result in fewer final features but less computing time.

`feature.num.cap`
: the maximum number of features to be selected. A lower value may result in fewer final features but less computing time.

Value

'CASMI.selectFeatures()' returns selected features and relevant information, including the estimated Kappa* for all selected features ('$KappaStar') and the corresponding confidence interval ('$KappaStarCI'). The selected features are ranked. The Standardized Mutual Information using the z estimator ('SMIz') and the corresponding confidence interval ('SMIz.Low' for lower bound, 'SMIz.Upr' for upper bound) are given for each selected feature ('Var.Idx' for column index, 'Var.Name' for column name). The p-value from the mutual information test of independence using the z estimator ('p.MIz') is given for each selected feature.

Examples

```
## Generate a toy dataset: "data"
## Features 1 and 3 are associated with Y, while feature 2 is irrelevant.
## The outcome variable Y must be discrete and be the LAST column. Features must be discrete.
n <- 10000
set.seed(1)
x1 <- rbinom(n, 3, 0.5) + 0.2
set.seed(2)
x2 <- rbinom(n, 2, 0.8) + 0.5
set.seed(3)
x3 <- rbinom(n, 5, 0.3)
set.seed(4)
error <- round(runif(n, min=-1, max=1))
y <- x1 + x3 + error
data <- data.frame(cbind(x1, x2, x3, y))
colnames(data) <- c("feature1", "feature2", "feature3", "Y")

## Select features and provide relevant results for the toy dataset "data"
CASMI.selectFeatures(data)

## For showing only the summary results
CASMI.selectFeatures(data, intermediate.steps = FALSE)

## Adjust 'feature.num.cap' for including fewer features.
## A lower 'feature.num.cap' value may result in fewer final features but less computing time.
## For example, if needing only the top one feature based on the toy dataset:
CASMI.selectFeatures(data, feature.num.cap = 1)
```

Index