Package ‘UAHDataScienceO’

February 20, 2025

Type Package

Title Educational Outlier Detection Algorithms with Step-by-Step Tutorials

Version 1.0.0

Maintainer Andriy Protsak Protsak <andriy.protsak@edu.uah.es>

Description Provides implementations of some of the most important outlier detection algorithms. Includes a tutorial mode option that shows a description of each algorithm and provides a step-by-step execution explanation of how it identifies outliers from the given data with the specified input parameters. References include the works of Azzedine Boukerche, Lining Zheng, and Omar Alfandi (2020) <doi:10.1145/3381028>, Abir Smiti (2020) <doi:10.1016/j.cosrev.2020.100306>, and Xiaogang Su, Chih-Ling Tsai (2011) <doi:10.1002/widm.19>.

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation no

Author Andres Missiego Manjon [aut],
Juan Jose Cuadrado Gallego [aut]
(<https://orcid.org/0000-0001-8178-5556>),
Andriy Protsak Protsak [aut, cre],
Universidad de Alcala de Henares [cph]

Repository CRAN

Date/Publication 2025-02-20 17:20:08 UTC

Contents

`boxandwhiskers` *Box And Whiskers*

Description

This function implements the box & whiskers algorithm to detect outliers

Usage

```
boxandwhiskers(data, d, learn)
```

Arguments

`data`	Input data.
`d`	Degree of outlier or distance at which an event is considered an outlier
`learn`	if TRUE the tutorial mode is activated (the algorithm will include an explanation detailing the theory behind the outlier detection algorithm and a step by step explanation of how is the data processed to obtain the outliers following the theory mentioned earlier)

Value

Numeric vector containing the indices of detected outliers.

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))))
inputData = data.frame(inputData)
boxandwhiskers(inputData,2,FALSE) # Can be set to TRUE
```

`compare_multivariate_methods` *Compare Multivariate Outlier Detection Methods*

Description

Compares multiple multivariate outlier detection methods on the same dataset

Usage

```
compare_multivariate_methods(data, methods, params)
```

Arguments

`data`	Input dataset (must be a data.frame)
`methods`	Vector of method names to compare. Available methods are: "lof", "dbscan", "knn", "mahalanobis"
`params`	List of parameters for each method. Must contain named lists: • lof: list(K=numeric, threshold=numeric) • dbscan: list(max_distance_threshold=numeric, min_pts=numeric) • knn: list(d=numeric, K=numeric) • mahalanobis: list(alpha=numeric)

Value

None, produces a visualization matrix comparing the outliers detected by each method.

Author(s)

Andriy Protsak

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))))
inputData = data.frame(inputData)
methods = c("lof", "dbscan", "knn", "mahalanobis")
params = list(
  lof = list(K=3, threshold=0.5),
  dbscan = list(max_distance_threshold=4, min_pts=3),
  knn = list(d=3, K=2),
  mahalanobis = list(alpha=0.7)
)
compare_multivariate_methods(inputData, methods, params)
```

`compare_univariate_methods` *Compare Univariate Outlier Detection Methods*

Description

Compares univariate outlier detection methods on the flattened dataset

Usage

```
compare_univariate_methods(data, methods, params)
```

Arguments

`data`	Input dataset (must be a data.frame)
`methods`	Vector of method names to compare. Available methods are: "z_score", "boxandwhiskers"
`params`	List of parameters for each method. Must contain named lists: • z_score: list(d=numeric) • boxandwhiskers: list(d=numeric)

Value

None, produces a visualization matrix comparing the outliers detected by each method.

Author(s)

Andriy Protsak

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))))
inputData = data.frame(inputData)
methods = c("z_score", "boxandwhiskers")
params = list(
  z_score = list(d=2),
  boxandwhiskers = list(d=2)
)
compare_univariate_methods(inputData, methods, params)
```

DBSCAN_method *DBSCAN_method*

Description

Outlier detection method using DBSCAN

Usage

```
DBSCAN_method(inputData, max_distance_threshold, min_pts, learn)
```

Arguments

`inputData` Input Data (must be a data.frame)

`max_distance_threshold` This is used to calculate the distance between all the points and check if the euclidean distance is less than the max_distance_threshold parameter to decide if add it to the neighbors or not

`min_pts` the minimum number of points to form a dense region

`learn` if TRUE the tutorial mode is activated (the algorithm will include an explanation detailing the theory behind the outlier detection algorithm and a step by step explanation of how is the data processed to obtain the outliers following the theory mentioned earlier)

Value

Numeric vector containing the indices of detected outliers.

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))));
inputData = data.frame(inputData);
eps = 4;
min_pts = 3;
DBSCAN_method(inputData, eps, min_pts, FALSE); #Can be set to TRUE
```

euclidean_distance	*euclidean_distance*

Description

This function calculates the euclidean distance between 2 points. They must have the same number of dimensions

Usage

```
euclidean_distance(p1, p2)
```

Arguments

p1	One of the points that will be used by the algorithm with N dimensions
p2	The other point that will be used by the algorithm with N dimensions

Value

Euclidean Distance calculated between the two N-dimensional points

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))));
inputData = data.frame(inputData);
point1 = inputData[1,];
point2 = inputData[4,];
distance = euclidean_distance(point1, point2);
```

knn	*knn*

Description

This function implements the knn algorithm for outlier detection

Usage

```
knn(data, d, K, learn)
```

Arguments

- `data` Input Data (must be a data.frame)
- `d` Degree of outlier or distance at which an event is considered an outlier
- `K` Nearest neighbor for which an event must have a degree of outlier to be considered an outlier
- `learn` if TRUE the tutorial mode is activated (the algorithm will include an explanation detailing the theory behind the outlier detection algorithm and a step by step explanation of how is the data processed to obtain the outliers following the theory mentioned earlier)

Value

Numeric vector containing the indices of detected outliers.

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))))
inputData = data.frame(inputData)
knn(inputData,3,2,FALSE) #Can be changed to TRUE
```

`lof` *lof*

Description

Local Outlier Factor algorithm to detect outliers

Usage

```
lof(inputData, K, threshold, learn)
```

Arguments

- `inputData` Input Data (must be a data.frame)
- `K` This number represents the nearest neighbor to use to calculate the density of each point. This value is chosen arbitrarily and is responsibility of the data scientist/user to select a number adequate to the dataset.
- `threshold` Value that is used to classify the points comparing it to the calculated ARDs of the points in the dataset. If the ARD is smaller, the point is classified as an outliers. If not, the point is classified as a normal point (inlier)

learn — if TRUE the tutorial mode is activated (the algorithm will include an explanation detailing the theory behind the outlier detection algorithm and a step by step explanation of how is the data processed to obtain the outliers following the theory mentioned earlier)

Value

Numeric vector containing the indices of detected outliers.

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))));
inputData = data.frame(inputData);
lof(inputData,3,0.5,FALSE) #Can be changed to TRUE
```

---

mahalanobis_distance *mahalanobis_distance*

Description

Calculates the mahalanobis_distance given the input data

Usage

```
mahalanobis_distance(value, sample_mean, sample_covariance_matrix)
```

Arguments

value — Point to calculate the mahalanobis_distance

sample_mean — Sample mean

sample_covariance_matrix — Sample Covariance Matrix

Value

Mahalanobis distance associated to the point

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))));
inputData = data.frame(inputData);
inputData = as.matrix(inputData);
sampleMeans = c();
for(i in 1:ncol(inputData)){
  column = inputData[,i];
  calculatedMean = sum(column)/length(column);
  print(sprintf("Calculated mean for column %d: %f", i, calculatedMean))
  sampleMeans = c(sampleMeans, calculatedMean);
}
covariance_matrix = cov(inputData);
distance = mahalanobis_distance(inputData[3,], sampleMeans, covariance_matrix);
```

---

mahalanobis_method *mahalanobis_method*

Description

Detect outliers using the Mahalanobis Distance method

Usage

```
mahalanobis_method(inputData, alpha, learn)
```

Arguments

inputData	Input Data dataset that will be processed (with or not the step by step explanation) to obtain the underlying outliers. It must be a data.frame type.
alpha	Significance level alpha. This value indicates the proportion that it is expected to be outliers out of the dataset. It has to be in the range from 0 to 1
learn	if TRUE the tutorial mode is activated (the algorithm will include an explanation detailing the theory behind the outlier detection algorithm and a step by step explanation of how is the data processed to obtain the outliers following the theory mentioned earlier)

Value

Numeric vector containing the indices of detected outliers.

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))));
inputData = data.frame(inputData);
mahalanobis_method(inputData, 0.7, FALSE); #Can be set to TRUE
```

---

manhattan_dist *manhattan_dist*

Description

Calculates the manhattan distance between two 2D points

Usage

```
manhattan_dist(A, B)
```

Arguments

A	One of the 2D points
B	The other 2D point

Value

Manhattan distance calculated between point A and B

Author(s)

Andres Missiego Manjon

Examples

```
distance = manhattan_dist(c(1,2), c(3,4));
```

`mean_outliersLearn` *mean_outliersLearn*

Description

Calculates the mean of the given data vector

Usage

```
mean_outliersLearn(data)
```

Arguments

`data`	Input Data that will be processed to calculate the mean. It must be a vector

Value

Mean of the input data

Author(s)

Andres Missiego Manjon

Examples

```
mean = mean_outliersLearn(c(2,3,2.3,7.8));
```

`quantile_outliersLearn` *quantile_outliersLearn*

Description

Function that obtains the 'v' quantile

Usage

```
quantile_outliersLearn(data, v)
```

Arguments

`data`	Input Data
`v`	Goes from 0 to 1 (e.g. 0.25). Indicates the quantile that wants to be obtained

Value

Quantile v calculated

Author(s)

Andres Missiego Manjon

Examples

```
q = quantile_outliersLearn(c(12,2,3,4,1,13), 0.60)
```

---

sd_outliersLearn *sd_outliersLearn*

Description

Calculates the standard deviation of the input data given the mean.

Usage

```
sd_outliersLearn(data, mean)
```

Arguments

`data`	Input Data that will be used to calculate the standard deviation. Must be a vector
`mean`	Mean of the input data vector of the function.

Value

Standard Deviation of the input data

Author(s)

Andres Missiego Manjon

Examples

```
inputData = c(1,2,3,4,5,6,1);
mean = sum(inputData)/length(inputData);
sd = sd_outliersLearn(inputData, mean);
```

transform_to_vector *transform_to_vector*

Description

Transform any type of data to a vector

Usage

```
transform_to_vector(data)
```

Arguments

data	Input data that will be transformed into a vector

Value

Data formatted as a vector

Author(s)

Andres Missiego Manjon

Examples

```
numeric_data = c(1, 2, 3)
character_data = c("a", "b", "c")
logical_data = c(TRUE, FALSE, TRUE)
factor_data = factor(c("A", "B", "A"))
integer_data = as.integer(c(1, 2, 3))
complex_data = complex(real = c(1, 2, 3), imaginary = c(4, 5, 6))
list_data = list(1, "apple", TRUE)
data_frame_data = data.frame(x = c(1, 2, 3), y = c("a", "b", "c"))

transformed_numeric = transform_to_vector(numeric_data)
transformed_character = transform_to_vector(character_data)
transformed_logical = transform_to_vector(logical_data)
transformed_factor = transform_to_vector(factor_data)
transformed_integer = transform_to_vector(integer_data)
transformed_complex = transform_to_vector(complex_data)
transformed_list = transform_to_vector(list_data)
transformed_data_frame = transform_to_vector(data_frame_data)
```

z_score_method *z_score_method*

Description

This function implements the outlier detection algorithm using standard deviation and mean

Usage

```
z_score_method(data, d, learn)
```

Arguments

`data`	Input Data that will be processed with or without the tutorial mode activated
`d`	Degree of outlier or distance at which an event is considered an outlier
`learn`	if TRUE the tutorial mode is activated (the algorithm will include an explanation detailing the theory behind the outlier detection algorithm and a step by step explanation of how is the data processed to obtain the outliers following the theory mentioned earlier)

Value

Numeric vector containing the indices of detected outliers.

Author(s)

Andres Missiego Manjon

Examples

```
inputData = t(matrix(c(3,2,3.5,12,4.7,4.1,5.2,
4.9,7.1,6.1,6.2,5.2,14,5.3),2,7,dimnames=list(c("r","d"))))
inputData = data.frame(inputData)
z_score_method(inputData,2,FALSE) #Can be changed to TRUE
```

Index