Package: KMEANS.KNN (via r-universe)

September 15, 2024

Title KMeans and KNN Clustering Package

Version 0.1.0

Description Implementation of Kmeans clustering algorithm and a supervised KNN (K Nearest Neighbors) learning method. It allows users to perform unsupervised clustering and supervised classification on their datasets. Additional features include data normalization, imputation of missing values, and the choice of distance metric. The package also provides functions to determine the optimal number of clusters for Kmeans and the best k-value for KNN: knn_Function(), find_Knn_best_k(), KMEANS_FUNCTION(), and find_Kmeans_best_k().

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports factoextra, cluster, ggplot2, stats, assertthat, class, caret, grDevices

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://GitHubID.github.io/KMEANS.KNN

Repository https://lallogo45.r-universe.dev

RemoteUrl https://github.com/lallogo45/kmeans.knn

RemoteRef HEAD

RemoteSha 2c1177c505abab99df5843278736d55b3c6d7b16

Contents

`find_Kmeans_best_k` *find_Kmeans_best_k*

Description

find_Kmeans_best_k

Usage

```
find_Kmeans_best_k(data, max_k = 10, Method = "coude", verbose = FALSE)
```

Arguments

`data`	The dataset for which K-means clustering will be performed.
`max_k`	The maximum number of clusters to consider. It defaults to 10.
`Method`	The method used to determine the optimal number of clusters. Acceptable values are "coude" (elbow method), "silhouette" (silhouette method), or "gap" (gap statistics).
`verbose`	Logical. If TRUE, additional output is provided.

Value

This function does not return a value but prints the optimal number of clusters based on the chosen method and plots the corresponding graph.

Examples

```
data(iris)
find_Kmeans_best_k(iris[,-5],9,Method = "coude")
```

`find_Knn_best_k` *find_Knn_best_k*

Description

This function finds the best k-value for KNN based on the provided data.

Usage

```
find_Knn_best_k(data, target_column, k_values, Prop_train = 0.8)
```

Arguments

data	A dataframe containing the dataset to be used.
target_column	A string specifying the name of the target column in the dataset.
k_values	A numeric vector containing the different k-values to be tested.
Prop_train	A numeric value between 0 and 1 indicating the proportion of the dataset to be used for training.

Value

A list containing a dataframe with k-values and their corresponding accuracies, and the best k-value with its accuracy.

Examples

```
data(iris)
find_Knn_best_k(iris,"Species",1:10,Prop_train=0.8)
```

KMEANS_FUNCTION *KMEANS_FUNCTION*

Description

This function implements the K-Means algorithm for data clustering. It provides options for data preprocessing, such as normalization and imputation of missing values.

Usage

```
KMEANS_FUNCTION(
  data,
  k,
  max_iter = 100,
  nstart = 25,
  distance_metric = "euclidean",
  scale_data = FALSE,
  impute_data = "mean"
)
```

Arguments

data	A dataframe containing the numerical data to be clustered.
k	The number of clusters to form.
max_iter	The maximum number of iterations for the K-Means algorithm.
nstart	The number of times to randomly initialize the centroids.
distance_metric	The distance metric to use ('euclidean' or 'manhattan').
scale_data	A boolean indicating whether the data should be normalized.
impute_data	The imputation method for missing values ('mean', 'median', 'mode').

Value

A list containing the following elements: - clusters: A vector indicating the cluster of each point. - centers: The coordinates of the centroids of each cluster. - additional_info: Additional information such as total distance and number of iterations.

Examples

```
data(iris)
data_iris <- iris[, -5] # Exclude the species column
results <- KMEANS_FUNCTION(data_iris, k = 3)
print(results$clusters)
```

---

knn_Function *knn_Function*

Description

This function implements a custom K-Nearest Neighbors (KNN) algorithm with data preprocessing options. It predicts the class of a new point based on the k closest neighbors in the feature space.

Usage

```
knn_Function(
  new_points,
  dataset,
  k = 5,
  distance_metric = "gower",
  target_variable,
  scale_data = TRUE,
  impute_data = "mean",
  weight_votes = TRUE
)
```

Arguments

new_points	A dataframe of new points to be classified.
dataset	A dataframe of training data.
k	The number of nearest neighbors to consider.
distance_metric	The distance metric for calculating neighbors ('gower', 'euclidean', 'manhattan').
target_variable	The name of the target variable in 'dataset'.
scale_data	A boolean to indicate whether the data should be normalized.
impute_data	The imputation method for missing values ('mean', 'median', 'mode').
weight_votes	A boolean to indicate whether votes should be weighted by the inverse of the distance.

Value

A list containing 'Predictions' with the predicted class for each new point, 'Data' with the 'new_points' dataframe and an additional column for predictions, 'Distances' with the distances of the k nearest neighbors, and 'Imputed_Values' with the imputed values for missing variables.

Examples

```
# Loading training data (e.g., iris)
data(iris)

# Preparing new points for prediction (e.g., two new observations)
new_points <- data.frame(Sepal.Length = c(5.1, 7.7, 1.3, 0.2, 5.1),
Sepal.Width = c(3.5, 2.6, 5, 3.7, 3.5),
Petal.Length = c(1.4 , 6.9, 4.5, 6, 3.4),
Petal.Width = c(10.1, 7.6, 5.6, 8.4, 5.2))

# Calling the custom KNN function
results <- knn_Function(new_points, dataset = iris, k = 3, target_variable = "Species")

# Displaying predictions
print(results$Predictions)
```

Index