Package 'knnp'

May 8, 2026

Version 2.0.0

Date 2020-01-10

Title Time Series Prediction using K-Nearest Neighbors Algorithm (Parallel)

Depends R (>= 3.6.0)

Imports parallelDist, forecast, stats, utils, doParallel, foreach, plyr

Suggests tseries, tsibble, datasets

Description Two main functionalities are provided. One of them is predicting values with k-nearest neighbors algorithm and the other is optimizing the parameters k and d of the algorithm. These are carried out in parallel using multiple threads.

License AGPL-3

URL https://github.com/Grasia/knnp

BugReports https://github.com/Grasia/knnp/issues

RoxygenNote 7.0.2

NeedsCompilation no

Author Daniel Bastarrica Lacalle [aut, cre], Javier Berdecio Trigueros [aut], Javier Arroyo Gallardo [aut], Albert Meco Alias [aut]

Maintainer Daniel Bastarrica Lacalle <danibast@ucm.es>

Repository CRAN

Date/Publication 2020-01-10 23:30:02 UTC

Contents

knn *Generic function to make a prediction for a time series. If a knn model is provided as the first argument, knn_forecast will be directly called. If single values are provided as k and d as no parameter search can be perfomed, knn_forecast will be called automatically. If no values are provided for k and/or d, values 1 to 50 will be used by default.*

Description

Generic function to make a prediction for a time series. If a knn model is provided as the first argument, knn_forecast will be directly called. If single values are provided as k and d as no parameter search can be perfomed, knn_forecast will be called automatically. If no values are provided for k and/or d, values 1 to 50 will be used by default.

Usage

```
knn(
  y,
  k = 1:50,
  d = 1:50,
  initial = NULL,
  distance = "euclidean",
  error_measure = "MAE",
  weight = "proportional",
  v = 1,
  threads = 1
)
```

Arguments

y	A time series or a trained kNN model generated by the knn_param_search function. In case that a model is provided the knn_forecast function will be automatically called.
k	Values of k's to be analyzed or chosen k for knn forecasting. Default value is 1 to 50.
d	Values of d's to be analyzed or chosen d for knn forecasting. Default value is 1 to 50.
initial	Variable that determines the limit of the known past for the first instant predicted.
distance	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, camberra and others. For more information about the supported metrics check the values that 'method' argument of function parDist (from parallelDist package) can take as this is the function used to calculate the distances. Link to package info: https://cran.r-project.org/web/packages/parallelDist Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "camberra", "chord".

error_measure	Type of metric to evaluate the prediction error. Five metrics supported: **ME** Mean Error **RMSE** Root Mean Squared Error **MAE** Mean Absolute Error **MPE** Mean Percentage Error **MAPE** Mean Absolute Percentage Error
weight	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proportional, average, linear. **proportional** the weight assigned to each neighbor is inversely proportional to its distance **average** all neighbors are assigned with the same weight **linear** nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.
v	Variable to be predicted if given multivariate time series.
threads	Number of threads to be used when parallelizing, default is 1

Value

A matrix of errors, optimal k and d. All tested ks and ks and all the used metrics.

Examples

```
knn(AirPassengers, 1:5, 1:3)
knn(LakeHuron, 1:10, 1:6)
```

`knn_elements` *Creates a matrix to be used for calculating distances. The most recent 'element' is put in the first row of the matrix, the second most recent 'element' in the second row and so on. Therefore, the oldest 'element' is put in the last row.*

Description

Creates a matrix to be used for calculating distances. The most recent 'element' is put in the first row of the matrix, the second most recent 'element' in the second row and so on. Therefore, the oldest 'element' is put in the last row.

Usage

```
knn_elements(y, d)
```

Arguments

y	A matrix.
d	Length of each of the 'elements'.

Value

A matrix to be used for calculating distances.

`knn_forecast` *Predicts next value of the time series using k-nearest neighbors algorithm.*

Description

Predicts next value of the time series using k-nearest neighbors algorithm.

Usage

```
knn_forecast(
  y,
  k,
  d,
  distance = "euclidean",
  weight = "proportional",
  v = 1,
  threads = 1,
  h = 1
)
```

Arguments

`y`	A time series or a trained kNN model generated by the knn_param_search_function. In case that a model is provided the rest of parameters will be ignored and all of them will be taken from the model.
`k`	Number of neighbors.
`d`	Length of each of the 'elements'.
`distance`	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, camberra and others. For more information about the supported metrics check the values that 'method' argument of function parDist (from parallelDist package) can take as this is the function used to calculate the distances. Link to package info: https://cran.r-project.org/web/packages/parallelDist Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "camberra", "chord".
`weight`	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proportional, average, linear. **proportional** the weight assigned to each neighbor is inversely proportional to its distance **average** all neighbors are assigned with the same weight **linear** nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.

v	Variable to be predicted if given multivariate time series.
threads	Number of threads to be used when parallelizing, default is 1
h	Temporal horizon of the prediction (only value 1 is implemented). This parameter is present only for compatibility with the forecast package.

Value

The predicted value.

Examples

```
knn_forecast(AirPassengers, 5, 2)
knn_forecast(LakeHuron, 3, 6)
```

---

knn_param_search *Searches for the optimal values of k and d for a given time series. First, values corresponding to instants from initial + 1 to the last one are predicted. The first value predicted, which corresponds to instant initial + 1, is calculated using instants from 1 to instant initial; the second value predicted, which corresponds to instant initial + 2, is predicted using instants from 1 to instant initial + 1; and so on until last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1. Finally, the error is evaluated between the predicted values and the real values of the series. This version of the optimization function uses a parallelized distances calculation function, and the computation of the predicted values is done parallelizing by the number of d's.*

Description

Searches for the optimal values of k and d for a given time series. First, values corresponding to instants from initial + 1 to the last one are predicted. The first value predicted, which corresponds to instant initial + 1, is calculated using instants from 1 to instant initial; the second value predicted, which corresponds to instant initial + 2, is predicted using instants from 1 to instant initial + 1; and so on until last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1. Finally, the error is evaluated between the predicted values and the real values of the series. This version of the optimization function uses a parallelized distances calculation function, and the computation of the predicted values is done parallelizing by the number of d's.

Usage

```
knn_param_search(
  y,
  k,
  d,
  initial = NULL,
```

```
  distance = "euclidean",
  error_measure = "MAE",
  weight = "proportional",
  v = 1,
  threads = 1
)
```

Arguments

y	A time series.
k	Values of k's to be analyzed.
d	Values of d's to be analyzed.
initial	Variable that determines the limit of the known past for the first instant predicted.
distance	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, camberra and others. For more information about supported metrics check the values that 'method' argument of function parDist (from parallelDist package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/web/packages/parallelDist Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "camberra", "chord".
error_measure	Type of metric to evaluate the prediction error. Five metrics supported: **ME** Mean Error **RMSE** Root Mean Squared Error **MAE** Mean Absolute Error **MPE** Mean Percentage Error **MAPE** Mean Absolute Percentage Error
weight	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proportional , average, linear. **proportional** the weight assigned to each neighbor is inversely proportional to its distance **average** all neighbors are assigned with the same weight **linear** nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.
v	Variable to be predicted if given multivariate time series.
threads	Number of threads to be used when parallelizing, default is 1

Value

A matrix of errors, optimal k and d. All tested ks and ks and all the used metrics.

Examples

```
knn_param_search(AirPassengers, 1:5, 1:3)
knn_param_search(LakeHuron, 1:10, 1:6)
```

knn_past *Predicts values of the time series using k-nearest neighbors algorithm. Values corresponding to instants from initial + 1 to the last one are predicted. The first value predicted, which corresponds to instant initial + 1, is calculated using instants from 1 to instant initial; the second value predicted, which corresponds to instant initial + 2, is predicted using instants from 1 to instant initial + 1; and so on until the last value, which corresponds to instant n (length of the time series), is predicted using instants from 1 to instant n - 1.*

Description

Predicts values of the time series using k-nearest neighbors algorithm. Values corresponding to instants from initial + 1 to the last one are predicted. The first value predicted, which corresponds to instant initial + 1, is calculated using instants from 1 to instant initial; the second value predicted, which corresponds to instant initial + 2, is predicted using instants from 1 to instant initial + 1; and so on until the last value, which corresponds to instant n (length of the time series), is predicted using instants from 1 to instant n - 1.

Usage

```
knn_past(
  y,
  k,
  d,
  initial = NULL,
  distance = "euclidean",
  weight = "proportional",
  v = 1,
  threads = 1
)
```

Arguments

y	A time series or a trained kNN model generated by the knn_param_search_function. In case that a model is provided the rest of parameters will be ignored and all of them will be taken from the model.
k	Number of neighbors.
d	Length of each of the 'elements'.
initial	Variable that determines the limit of the known past for the first instant predicted.
distance	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, camberra and others. For more information about supported metrics check the values that 'method' argument of function parDist (from parallelDist package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/web/packages/parallelDist Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "camberra", "chord".

`weight` Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proportional, average, linear.

proportional the weight assigned to each neighbor is inversely proportional to its distance

average all neighbors are assigned with the same weight

linear nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.

`v` Variable to be predicted if given multivariate time series.

`threads` Number of threads to be used when parallelizing, default is 1

Value

The predicted value.

Examples

```
knn_past(AirPassengers, 5, 2)
knn_past(LakeHuron, 3, 6)
```

Index