Package 'utsf'

October 14, 2024

Title Univariate Time Series Forecasting

Version 1.0.0

Description An engine for univariate time series forecasting using different regression models in an autoregressive way. The engine provides an uniform interface for applying the different models. Furthermore, it is extensible so that users can easily apply their own regression models to univariate time series forecasting and benefit from all the features of the engine, such as preprocessings or estimation of forecast accuracy.

Maintainer Francisco Martinez <fmartin@ujaen.es>

License MIT + file LICENSE

URL https://github.com/franciscomartinezdelrio/utsf

BugReports https://github.com/franciscomartinezdelrio/utsf/issues

Imports Cubist, FNN, forecast, ggplot2, ipred, methods, ranger, rpart, vctsfr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Maria Pilar Frias-Bustamante [aut] (<https://orcid.org/0000-0001-6886-0953>), Francisco Martinez [aut, cre, cph] (<https://orcid.org/0000-0002-5206-1898>)

Repository CRAN

Date/Publication 2024-10-14 09:30:02 UTC

Contents

`autoplot.utsf` *Create a ggplot object from an* `utsf` *object*

Description

Plot the time series and its associated forecast.

Usage

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

Arguments

`object`	An object of class `utsf`.
`...`	additional parameter.

Value

The `ggplot` object representing a plotting of the time series and its forecast.

Examples

```
f <- forecast(AirPassengers, h = 12, lags = 1:12, method = "rf")
library(ggplot2)
autoplot(f)
```

build_examples *Build the training examples*

Description

Build the training examples for a regressive model to forecast a time series using lagged values of the series as autoregressive features.

Usage

```
build_examples(timeS, lags)
```

Arguments

timeS	The time series.
lags	An integer vector with the lags used as feature vector in decreasing order.

Value

A list with two fields: 1) a matrix with the features of the examples and 2) a vector with the targets of the examples

Examples

```
build_examples(ts(1:5), lags = 2:1)
```

differences *Specifying the order of differences*

Description

This function is used to specify that the time series will be preprocessed using first differences.

Usage

```
differences(n = -1)
```

Arguments

n	An integer specifying the order of differences to be applied. If the default (-1) is used, the order of differences needed by the time series will be computed by the forecast::ndiffs() function.

Value

An integer with the order of differences to be applied.

Examples

```
differences(1)
```

`forecast` *Train an univariate time series forecasting model and make forecasts*

Description

This function trains a model from the historical values of a time series using an autoregressive approach: the targets are the historical values and the features of the targets their lagged values. Then, the trained model is used to predict the future values of the series using a recursive strategy.

Usage

```
forecast(
  timeS,
  h,
  lags = NULL,
  method = "knn",
  param = NULL,
  efa = NULL,
  tuneGrid = NULL,
  preProcess = list("additive")
)
```

Arguments

`timeS`	A time series of class `ts` or a numeric vector.
`h`	A positive integer. Number of values to be forecast into the future, i.e., forecast horizon.
`lags`	An integer vector, in increasing order, expressing the lags used as autoregressive variables. If the default value (`NULL`) is provided, a suitable vector is chosen.
`method`	A string indicating the method used for training and forecasting. Allowed values are: • `"knn"`: k-nearest neighbors (the default) • `"rt"`: regression trees • `"mt"`: model trees • `"bagging"` • `"rf"`: random forest. See details for a brief explanation of the models. It is also possible to use your own regression model, in that case a function explaining how to build your model must be provided, see the vignette for further details.
`param`	A list with parameters for the underlying function that builds the model. If the default value (`NULL`) is provided, the model is fitted with its default parameters. See details for the functions used to train the models.

efa	A character value indicating the kind of method used to estimate the forecast accuracy of the model using the time series. If the default value (NULL) is provided, no estimation is done. Possible values are "rolling" and "fixed", indicating if rolling or fixed origin evaluation is done. To estimate forecast accuracy the last h values of the time series are used as test set and the previous values as training set.
tuneGrid	A data frame with possible tuning values. The columns are named the same as the tuning parameters. The estimation of forecast accuracy is done as explained for the efa parameter. Rolling or fixed origin evaluation is done according to the value of the efa parameter (fixed if NULL). The best combination of parameters is used to train the model with all the historical values of the time series.
preProcess	A list indicating the preprocessings or transformations. Currently, the length of the list must be 1 (only one preprocessing). If NULL no preprocessing is applied. The element of the list is a character value indicating what transformation is applied. By default ("additive") an additive transformation is done. It is also possible a multiplicative transformation ("multiplicative"). These transformations are recommended if the time series has a trend. Also, taking differences is allowed using the differences() function.

Details

The functions used to build and train the model are:

- KNN: In this case no model is built and the function FNN::knn.reg() is used to predict the future values of the time series.
- Regression trees: Function rpart::rpart() to build the model and the method rpart::predict.rpart() associated with the trained model to forecast the future values of the time series.
- Model trees: Function Cubist::cubist() to build the model and the method Cubist::predict.cubist() associated with the trained model to forecast the future values of the time series.
- Bagging: Function ipred::bagging() to build the model and the method ipred::predict.regbagg() associated with the trained model to forecast the future values of the time series.
- Random forest: Function ranger::ranger() to build the model and the method ranger::predict.ranger() associated with the trained model to forecast the future values of the time series.

Value

An S3 object of class utsf, basically a list with, at least, the following components:

ts	The time series being forecast.
features	A data frame with the features of the training set. The column names of the data frame indicate the autoregressive lags.
targets	A vector with the targets of the training set.
lags	An integer vector with the autoregressive lags.
model	The regression model used recursively to make the forecast.
pred	An object of class ts and length h with the forecast.

efa	This component is included if forecast accuracy is estimated. A vector with estimates of forecast accuracy according to different forecast accuracy measures.
tuneGrid	This component is included if the tuneGrid parameter has been used. A data frame in which each row contains estimates of forecast accuracy for a combination of tuning parameters.

Examples

```
## Forecast time series using k-nearest neighbors
f <- forecast(AirPassengers, h = 12, method = "knn")
f$pred
library(ggplot2)
autoplot(f)

## Using k-nearest neighbors changing the default k value
forecast(AirPassengers, h = 12, method = "knn", param = list(k = 5))$pred

## Using your own regression model

# Function to build the regression model
my_knn_model <- function(X, y) {
  structure(list(X = X, y = y), class = "my_knn")
}
# Function to predict a new example
predict.my_knn <- function(object, new_value) {
  FNN::knn.reg(train = object$X, test = new_value, y = object$y)$pred
}
forecast(AirPassengers, h = 12, method = my_knn_model)$pred

## Estimating forecast accuracy of the model
f <- forecast(UKgas, h = 4, lags = 1:4, method = "rf", efa = "rolling")
f$efa

## Estimating forecast accuracy of different tuning parameters
f <- forecast(UKgas, h = 4, lags = 1:4, method = "knn", tuneGrid = expand.grid(k = 1:5))
f$tuneGrid

## Forecasting a trending series
# Without any preprocessing or transformation
f <- forecast(airmiles, h = 4, method = "knn", preProcess = NULL)
autoplot(f)

# Applying the additive transformation (default)
f <- forecast(airmiles, h = 4, method = "knn")
autoplot(f)
```

`predict.utsf` *Prediction from* `utsf` *objects*

Description

Predict the class of a new observation based on the model associated with the `utsf` object

Usage

```
## S3 method for class 'utsf'
predict(object, new_value, ...)
```

Arguments

`object`	object of class `utsf`.
`new_value`	a data frame with one row of a new observation.
`...`	further arguments passed to or from other methods.

Value

a numeric value with the forecast.

Index