Package ‘iForecast’

November 18, 2021

Type Package
Title Machine Learning Time Series Forecasting
Version 1.0.4
Date 2021-11-18
Author Ho Tsung-wu
Maintainer Ho Tsung-wu <tsungwu@ntnu.edu.tw>
Description Compute both static and recursive time series forecasts of machine learning models.
License GPL (>= 2)
LazyData TRUE
LazyLoad yes
Depends R (>= 3.5), caret, zoo
Imports magrittr
Suggests data.table, DescTools, forecast, h2o, keras, kernlab, lubridate, tensorflow, tibble, timeSeries, timeDate, timetk
NeedsCompilation no
Repository CRAN
Date/Publication 2021-11-18 06:10:02 UTC

R topics documented:

  data-sets ....................................................... 2
  iForecast ...................................................... 2
  rollingWindows ................................................. 4
  ttsAutoML ....................................................... 5
  ttsCaret ....................................................... 6
  ttsLSTM ....................................................... 9

Index 11
### Economic and Financial Data Sets

**Description**

ES\_15m is 15-min realized absolute variance of e-mini S\&P 500. macrodata contains monthly US unemployment(unrate), ES\_Daily is daily realized absolute variance of e-mini S\&P 500. macrodata contains monthly US unemployment(unrate) and and year-to-year changes in three regional business cycle indices (OECD, NAFTA, and G7). bc contains monthly business cycle data, bc is binary indicator(0=recession, 1=boom) of Taiwan’s business cycle phases, IPI\_TWN is industrial production index of Taiwan, LD\_OECD, LD\_G7, and LD\_NAFTA are leading indicators of OECD, G7 and NAFTA regions; all four are monthly rate of changes.

**Usage**

```r
data(ES\_15m)
data(macrodata)
data(ES\_Daily)
data(bc)
```

**Value**

an object of class "zoo".

---

### iForecast

**Extract predictions and class probabilities from train objects**

**Description**

It generates both the static and recursive time series plots of machine learning prediction object generated by ttsCaret, ttsAutoML and ttsLSTM.

**Usage**

```r
iForecast(Model,newdata,type)
```

**Arguments**

- **Model**: Object of trained model.
- **newdata**: The dataset for prediction, the column names must be the same as the trained data.
- **type**: If type="staticfit", it computes the direct (static) forecasting values of insample model fit; if type="recursive", it computes the recursive (dynamic) forecasting values of insample model; for recursive forecasts, AR term is required.
Details
This function generates forecasts of ttsCaret, ttsAutoML, and ttsLSTM.

Value
prediction  The forecasted time series target variable. For binary case, it returns both probabilities and class.

Author(s)
Ho Tsung-wu <tsungwu@ntnu.edu.tw>, College of Management, National Taiwan Normal University.

Examples
# Cross-validation takes time, example below is commented.
## Machine Learning by library(caret)
## Case 1. Low frequency, regression type
data("macrodata")
dep <- macrodata[569:669, "unrate", drop=FALSE]
ind <- macrodata[569:669, -1, drop=FALSE]
train.end <- "2018-12-01"# Choosing the end dating of train
models <- c("svm","rf","rpart","gamboost","BstLm","bstSm","blackboost")[3]
type <- c("none","trend","season","both")[1]
Caret <- ttsCaret(y=dep, x=ind, arOrder=c(1), xregOrder=c(1),
method=models, tuneLength =1, train.end, type=type,
trControl= trainControl(method = "cv"))
testData1 <- window(Caret$data,start="2019-01-01",end=end(Caret$data))
P1 <- iForecast(Model=Caret,newdata=testData1,type="staticfit")
P2 <- iForecast(Model=Caret,newdata=testData1,type="recursive")

# Case 2. Low frequency, binary type
data(bc) #binary dependent variable, business cycle phases
dep=bc[,1,drop=FALSE]
ind=bc[-1]

train.end=as.character(rownames(dep))[as.integer(nrow(dep)*0.8)]
test.start=as.character(rownames(dep))[as.integer(nrow(dep)*0.8)+1]

#Caret = ttsCaret(y=dep, x=ind, arOrder=c(1), xregOrder=c(1), method=models,
#tuneLength =10, train.end, type=type)
#testData1=window(Caret$data,start=test.start,end=end(Caret$data))

#head(Caret$dataused)
#P1=iForecast(Model=Caret,newdata=testData1,type="staticfit")
#P2=iForecast(Model=Caret,newdata=testData1,type="recursive")
rollingWindows  

Rolling timeframe for time series analysis

Description

It extracts time stamp from a timeSeries object and separates the time into in-sample training and out-of-sample validation ranges.

Usage

rollingWindows(x, estimation="18m", by = "6m")

Arguments

x The time series matrix of dependent variable, with timeSeries or zoo format.
estimation The range of insample estimation period, the default is 18 months (18m), where the k-fold cross-section is performed.
by The range of out-of-sample validation/testing period, the default is 6 months (6m).

Details

This function is similar to the backtesting framework in portfolio analysis. Rolling windows fixes the origin and the training sample grows over time, moving windows can be achieved by placing window() on dependent variable at each iteration.

Value

window The time labels of from and to

Author(s)

Ho Tsung-wu <tsungwu@ntnu.edu.tw>, College of Management, National Taiwan Normal University.

Examples

data(macrodata)
dep=macrodata[,1,drop=FALSE]
ind=macrodata[,-1,drop=FALSE]
timeframe=rollingWindows(dep, estimation="300m", by = "6m")
FROM=timeframe$from
TO=timeframe$to
ttsAutoML

Train time series by automatic machine learning of h2o provided by H2O.ai

Description

It generates both the static and recursive time series plots of H2O.ai object generated by package h2o provided by H2O.ai.

Usage

ttsAutoML(y,x=NULL,train.end,arOrder=2,xregOrder=0,maxSecs=30)

Arguments

y  The time series vector of target variable, or the dependent variable, with zoo format, must have dimension. y can be either binary or continuous.

x  The time series matrix of input variables, or the independent variables, with zoo format.

train.end  The end date of training data, must be specified. The default dates of train.start and test.end are the start and the end of input data; and the test.start is the 1-period next of train.end.

arOrder  The autoregressive order of the target variable, which may be sequentially specified like arOrder=1:5; or discontinuous lags like arOrder=c(1,3,5); zero is not allowed.

xregOrder  The distributed lag structure of the input variables, which may be sequentially specified like xregOrder=1:5; or discontinuous lags like xregOrder=c(0,3,5); zero is allowed since contemporaneous correlation is allowed.

maxSecs  The maximal run time specified, in seconds. Default=20.
Details

This function calls the h2o.automl function from package h2o to execute automatic machine learning estimation. When execution finished, it computes two types of time series forecasts: static and recursive. The procedure of h2o.automl automatically generates a lot of time features.

Value

output Output object generated by train function of caret.
arOrder The autoregressive order of the target variable used.
data The dataset of imputed.
dataused The data used by arOrder, xregOrder

Author(s)

Ho Tsung-wu <tsungwu@ntnu.edu.tw>, College of Management, National Taiwan Normal University.

Examples

# Cross-validation takes time, example below is commented.
data("macrodata")
dep<-macrodata[,"unrate",drop=FALSE]
dind<-macrodata,-1,drop=FALSE]

# Choosing the dates of training and testing data
train.end<="2008-12-01"

#autoML of H2O.ai

#autoML <- ttsAutoML(y=dep,x=ind,train.end,arOrder=c(2,4),
xregOrder=c(0,1,3),maxSecs =30)
#testData2 <- window(autoML$dataused,start="2009-01-01",end=end(autoML$data))
#P1<-iForecast(Model=autoML,newdata=testData2,type="staticfit")
#P2<iForecast(Model=autoML,newdata=testData2,type="recursive")

tail(cbind(testData2[,1],P1))
tail(cbind(testData2[,1],P2))
Usage

ttsCaret(
  y,
  x=NULL,
  method,
  train.end,
  arOrder=2,
  xregOrder=0,
  type,
  tuneLength =10,
  trControl=trainControl(method = "cv"))

Arguments

- **y**: The time series vector of target variable, or the dependent variable, with `zoo` format, must have dimension. `y` can be either binary or continuous.
- **x**: The time series matrix of input variables, or the independent variables, with `zoo` format.
- **method**: The `train_model_list` of caret. While using this, make sure that the method allows regression. Methods in c("svm","rf","rpart","gamboost","BstLm","bstSm","blackboost") are feasible.
- **train.end**: The end date of training data, must be specified. The default dates of `train.start` and `test.end` are the start and the end of input data; and the `test.start` is the 1-period next of `train.end`.
- **arOrder**: The autoregressive order of the target variable, which may be sequentially specified like `arOrder=1:5`; or discontinuous lags like `arOrder=c(1,3,5)`; zero is not allowed.
- **xregOrder**: The distributed lag structure of the input variables, which may be sequentially specified like `xregOrder=0:5`; or discontinuous lags like `xregOrder=c(0,3,5)`; zero is allowed since contemporaneous correlation is allowed.
- **type**: The additional input variables. We have four selection: "none"=no other variables, "trend"=inclusion of time dummy, "season"=inclusion of seasonal dummies, "both"=inclusion of both trend and season. No default.
- **tuneLength**: The same as the length specified in train function of package caret.
- **trControl**: The same as `trainControl` function list in package caret. Default is `trainControl(method = "cv")`.

Details

This function calls the `train` function of package caret to execute estimation. When execution finished, we compute two types of time series forecasts: static and recursive.
Value

output       Output object generated by train function of caret.
arOrder      The autoregressive order of the target variable used.
data         The dataset of imputed.
dataused      The data used by arOrder, xregOrder, and type

Author(s)

Ho Tsung-wu <tsungwu@ntnu.edu.tw>, College of Management, National Taiwan Normal University.

Examples

# Cross-validation takes time, example below is commented.
## Machine Learning by library(caret)
library(zoo)
#Case 1. Low frequency
data("macrodata")
dep <- macrodata[569:669,"unrate",drop=FALSE]
ind <- macrodata[569:669,-1,drop=FALSE]
train.end <- "2018-12-01"# Choosing the end dating of train
models <- c("svm","rf","rpart","glm")[3]
type <- c("none","trend","season","both")[1]
Caret <- ttsCaret(y=dep, x=NULL, arOrder=c(1), xregOrder=c(1),
                  method=models, tuneLength =1, train.end, type=type,
                  trControl= trainControl(method = "cv"))
testData1 <- window(Caret$data,start="2019-01-01",end=end(Caret$data))
P1 <- iForecast(Model=Caret,newdata=testData1,type="staticfit")
P2 <- iForecast(Model=Caret,newdata=testData1,type="recursive")

#Case 2. High frequency
#head(ES_15m)
#head(ES_Daily)
#dep <- ES_15m #SP500 15-minute realized absolute variance
#ind <- NULL
#train.end <- as.character(rownames(dep))[as.integer(nrow(dep)*0.9)]

#models<-c("svm","rf","rpart","gamboost","BstLm","bstSm","blackboost")[1]
type<-c("none","trend","season","both")[1]
# Caret <- ttsCaret(y=dep, x=ind, arOrder=c(3,5), xregOrder=c(0,2,4),
#                  method=models, tuneLength =10, train.end, type=type,
#                  trControl= trainControl(method = "cv"))
#testData1<-window(Caret$data,start="2009-01-01",end=end(Caret$data))
#P1<-iForecast(Model=Caret,newdata=testData1,type="staticfit")
#P2<-iForecast(Model=Caret,newdata=testData1,type="recursive")
**ttsLSTM**

*Train time series by LSTM of tensorflow provided by kera*

**Description**

It generates both the static and recursive time series plots of deep learning LSTM object generated by package tensorflow provided by kera.

**Usage**

```r
ttsLSTM(y, x=NULL, train.end, arOrder=1, xregOrder=0, type, memoryLoops=10, shape=NULL, dim3=5)
```

**Arguments**

- `y` The time series vector of target variable, or the dependent variable, with zoo format, must have dimension. y can be only continuous, binary case is not allowed at this version.
- `x` The time series matrix of input variables, or the independent variables, with zoo format.
- `train.end` The end date of training data, must be specified. The default dates of train.start and test.end are the start and the end of input data; and the test.start is the 1-period next of train.end.
- `arOrder` The autoregressive order of the target variable, which may be sequentially specified like arOrder=1:5; or discontinuous lags like arOrder=c(1,3,5); zero is not allowed. Default is 1.
- `xregOrder` The distributed lag structure of the input variables, which may be sequentially specified like xregOrder=1:5; or discontinuous lags like xregOrder=c(0,3,5); zero is allowed since contemporaneous correlation is allowed.
- `type` The additional input variables. We have four selection: "none"=no other variables, "trend"=inclusion of time dummy, "season"=inclusion of seasonal dummies, "both"=inclusion of both trend and season. No default.
- `memoryLoops` Length of LSTM learning network loop, to achieve better learning results, this not is suggested to be the same as the length of data row. Default is 10.
- `shape` The second dimension of LSTM array. If NULL, then it will use the number of columns of complete dataset.
- `dim3` The third dimension of LSTM array. Default is 5.

**Details**

This function calls the function fit of package tensorflow to execute Long-Short Term Memory (LSTM) estimation. When execution finished, it computes two types of time series forecasts: static and recursive.
Value

output  Output object generated by train function of caret.
batch.size  The batch.size used for LSTM network.
k  The third dimension of array in LSTM network.
SHAPE  The shape size of array in LSTM network.
arOrder  he autoregressive order of the target variable used.
data  The dataset of used.
dataused  The data used by arOrder, xregOrder, and type

Author(s)

Ho Tsung-wu <tsungwu@ntnu.edu.tw>, College of Management, National Taiwan Normal University.

Examples

# Cross-validation takes time, example below is commented.
data("macrodata")
dep<-macrodata[,"unrate",drop=FALSE]
ind<-macrodata[,-1,drop=FALSE]

# Choosing the dates of training and testing data
train.end<="2008-12-01"

#RNN with LSTM network
#LSTM<-ttsLSTM(y=dep, x=ind, train.end,arOrder=c(2,4), xregOrder=c(1,4),
# memoryLoops=5, type=c("none","trend","season","both")[4])

testData3<-window(LSTM$dataused,start="2009-01-01",end=end(LSTM$data))
#P1<-iForecast(Model=LSTM,newdata=testData3,type="staticfit")
#P2<-iForecast(Model=LSTM,newdata=testData3,type="recursive")

tail(cbind(testData3[,1],P1,P2))
Index

* datasets
  data-sets, 2

bc (data-sets), 2

data-sets, 2

ES_15m (data-sets), 2
ES_Daily (data-sets), 2

iForecast, 2

macrodata (data-sets), 2

rollingWindows, 4

ttsAutoML, 5
ttsCaret, 6
ttsLSTM, 9