Package: EventDetectR (via r-universe)

September 1, 2024

Version 0.3.6

Title Event Detection Framework

Description Detect events in time-series data. Combines multiple well-known R packages like 'forecast' and 'neuralnet' to deliver an easily configurable tool for multivariate event detection.

Encoding UTF-8

LazyData yes

Type Package

ByteCompile TRUE

BugReports https://github.com/frehbach/EventDetectR/issues

URL https://github.com/frehbach/EventDetectR

Depends R (>= 3.1.0)

Imports imputeTS, forecast, ggplot2, gridExtra, neuralnet, Rdpack

RdMacros Rdpack

Suggests testthat, utils, caret, e1071

License GPL-3

RoxygenNote 7.1.1

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

RemoteUrl https://github.com/frehbach/eventdetectr

RemoteRef HEAD

RemoteSha 7c4cc4a39455416f2b44c8572ddf3a962babab e

Contents

`EventDetectR-package` *EventDetectR-package description*

Description

Detect events/ anomalies in time-series data.

Details

The EventDetectR package enables detection of events/ anomalies in multivariate time-series data. It combines multiple well-known R packages like 'forecast, 'neuralnet' to deliver an easily configurable tool for event detection.

References

Chandrasekaran S, Rebolledo M, Bartz-Beielstein T (2020). "EventDetectR – An Open-Source Event Detection System." Institute for Data Science, Engineering, and Analytics, TH Köln, Steinmüllerallee 1, 51643 Gummersbach, Germany. https://nbn-resolving.org/urn:nbn:de:hbz:832-cos4-9232.

`buildEDModel` *build Event Detection Model*

Description

Builds an event detection object (edObject) containing all models and configurations that are used to detect events in given data.

Usage

```
buildEDModel(
  x,
  dataPrepators = "ImputeTSInterpolation",
  dataPreparationControl = list(),
  buildModelAlgo = "ForecastETS",
  buildForecastModelControl = list(),
  buildNeuralNetModelControl = list(),
  postProcessors = "bedAlgo",
  postProcessorControl = list(),
  ignoreVarianceWarning = FALSE,
  oldModel = NULL
)
```

Arguments

x	data.frame containing initial data on which the model will be fitted. Data should be free of events. The data should not include a timestamp column
dataPrepators	string or vector of strings, that defines which preparators to use. Lists are not accepted. Usage Example: dataPreparators = "ImputeTSInterpolation" results in the usage of imputeTS::na.interpolation as a data preparator. All possible preparators are listed via: getSupportedPreparations() Can also be set to NULL in order to shut off data preparation
dataPreparationControl	list, control-list containing all additional parameters that shall be passed to the dataPreparators.
buildModelAlgo	string, model name to be used. All possible preparators are listed via: getSupportedModels().
buildForecastModelControl	list, control-list containing all additional parameters that shall be passed to forecast modeling algorithm
buildNeuralNetModelControl	list, control-list containing all additional parameters that shall be passed to the neuralnet modeling algorithm
postProcessors	string or vector of strings, that defines which postProcessors to use. Lists are not accepted. Usage Example: postProcessors = "bedAlgo" results in the usage of bed as a event postProcessing tool. All possible preparators are listed via: getSupportedPostProcessors() Can also be set to NULL in order to shut off data postProcessing
postProcessorControl	list, control-list containing all additional parameters that shall be passed to the postProcessirs.
ignoreVarianceWarning	Ignores the continously appearing warning for missing variance in some variable columns given a smaller windowSize
oldModel	If another model was previously fitted it can be passed to the next model fit. By doing so the eventHistory is preserved

Value

model, event detection object (edObject) containing all models and configurations that are used to detect events in given data.

Examples

```
## build a simple event detection model with standard configuration
x <- stationBData[100:200,-1]
buildEDModel(x,ignoreVarianceWarning = TRUE)

## Set up a more complex event detection model defining some additional configuration
buildEDModel(x, buildModelAlgo = "ForecastArima",ignoreVarianceWarning = TRUE)

 ## Set up a multivariate neuralnetwork model
buildEDModel(x, buildModelAlgo = "NeuralNetwork",ignoreVarianceWarning = TRUE)
```

---

detectEvents *detectEvents in a given data.frame*

Description

detectEvents builds a prediction model (edObject) on the first 'windowSize' points of the given data x. The next 'nIterationRefit' data-points are classified as 'Event' or not. The window is moved iteratively and the next models are fitted. The first 'windowSize' points will always be classified as no Event and should only contain 'clean' data

Usage

```
detectEvents(
  x,
  windowSize = 100,
  nIterationsRefit = 1,
  verbosityLevel = 0,
  dataPrepators = "ImputeTSInterpolation",
  dataPreparationControl = list(),
  buildModelAlgo = "ForecastETS",
  buildForecastModelControl = list(),
  buildNeuralNetModelControl = list(),
  postProcessors = "bedAlgo",
  postProcessorControl = list(),
  ignoreVarianceWarning = TRUE
)
```

Arguments

x	data.frame, data which shall be classified as event or not
windowSize	amount of data points to consider in each prediction model

`nIterationsRefit`
: amount of points into the future which will be predicted without fitting a new model. E.g. if nIterationsRefit = 10 then the next five dataPoints are classified without refitting.

`verbosityLevel`
: Print output of function progress. 0 -> No output, 1 -> every 100th model building iteration, 2 -> every 10th, 3 -> every iteration

`dataPrepators`
: string or vector of strings, that defines which preparators to use. Lists are not accepted. Usage Example: dataPreparators = "ImputeTSInterpolation" results in the usage of imputeTS::na.interpolation as a data preparator. All possible preparators are listed via: getSupportedPreparations()

`dataPreparationControl`
: list, control-list containing all additional parameters that shall be passed to the dataPreparators.

`buildModelAlgo`
: string, model name to be used. All possible preparators are listed via: getSupportedModels().

`buildForecastModelControl`
: list, control-list containing all additional parameters that shall be passed to the forecast modelling algo.

`buildNeuralNetModelControl`
: list, control-list containing all additional parameters that shall be passed to the neuralnet modelling algo.

`postProcessors`
: string or vector of strings, that defines which postProcessors to use. Lists are not accepted. Usage Example: postProcessors = "bedAlgo" results in the usage of bed as a event postProcessing tool. All possible preparators are listed via: getSupportedPostProcessors()

`postProcessorControl`
: list, control-list containing all additional parameters that shall be passed to the postProcessirs.

`ignoreVarianceWarning`
: Ignores the continously appearing warning for missing variance in some variable columns given a smaller windowSize

Value

edsResults edObject, list of results. $classification -> data.frame containing the T/F event classification

Examples

```
## Run event detection with default settings:
def <- detectEvents(x = stationBData[1:100,-1])


## Refit the model at every new datapoint,
## have someoutput with verbosityLevel = 2 and ignore
## the variance warning
ed <- detectEvents(stationBData[1:110,-1],nIterationsRefit = 1,
                   verbosityLevel = 2,ignoreVarianceWarning = TRUE)
```

```
## Switch to another model: Arima
ed2 <- detectEvents(stationBData[1:110,-1],nIterationsRefit = 1,
                    verbosityLevel = 0,ignoreVarianceWarning = TRUE,
                    buildModelAlgo = "ForecastArima")

   ## Switch to multivariate model: NeuralNetwork
ed3 <- detectEvents(stationBData[1:110,-1],nIterationsRefit = 1, buildModelAlgo = "NeuralNetwork")
```

`geccoIC2018Test` *geccoIC2018Test*

Description

2018s Test set of the gecco industrial challenge - http://www.spotseven.de/gecco/gecco-challenge/

`geccoIC2018Train` *geccoIC2018Train*

Description

2018s train set of the gecco industrial challenge - http://www.spotseven.de/gecco/gecco-challenge/

`getSupportedModels` *getSupportedModels*

Description

Get a list of all data modelling methods that are currently supported in package 'eventDetectR'.

Usage

```
getSupportedModels()
```

Value

allSupportedModels a list of strings with each supported method name. The strings can be copied and used in calls to 'eventDetect' or 'buildEDModel'

Examples

```
models <- getSupportedModels()
```

getSupportedPostProcessors *getSupportedPostProcessors*

Description

Get a list of all data postprocessing methods that are currently supported in package 'eventDetectR'.

Usage

```
getSupportedPostProcessors()
```

Value

allSupportedPostProcessors a list of strings with each supported method name. The strings can be copied and used in calls to 'eventDetect' or 'buildEDModel'

Examples

```
preps <- getSupportedPostProcessors()
```

getSupportedPreparations *getSupportedPreparations*

Description

Get a list of all data preparation methods that are currently supported in package 'eventDetectR'.

Usage

```
getSupportedPreparations()
```

Value

allSupportedPreparations a list of strings with each supported method name. The strings can be copied and used in calls to 'eventDetect' or 'buildEDModel'

Examples

```
preps <- getSupportedPreparations()
```

plot.edObject *Plot an Event Detection Object*

Description

Plot an Event Detection Object

Usage

```
## S3 method for class 'edObject'
plot(x, varsToPlot = names(edObject$classification), ...)
```

Arguments

x	edObject
varsToPlot	vars
...	Additional parameters

Value

A Plot

print.edObject *Print an Event Detection Object*

Description

Prints the last classification results for an event detection object. If 'nLast' (integer) is given, it specifies the amount of rows to be printed.

Usage

```
## S3 method for class 'edObject'
print(x, ...)
```

Arguments

x	edObject, the event detection object that shall be printed
...	any additional parameters

qualityStatistics *qualityStatistics*

Description

Wrapper function for caret::confusionMatrix. qualityStatistics calculates statistics for judging the quality of the eventDetection based on the fitted edModel and a reference dataset

Usage

```
qualityStatistics(edObject, reference)
```

Arguments

edObject	The eventdetection object you obtain by running 'detectEvents'
reference	true/false vector, reference vector based on labeled data: which datapoints are real events.

Value

list, Confusion Matrix and Statistics

Examples

```
train <- geccoIC2018Train[15000:17000,]
edObject <- detectEvents(train[,-c(1,11)],windowSize = 1000,
                nIterationsRefit = 500,verbosityLevel = 2,
                postProcessorControl = list(nStandardDeviationseventThreshold = 3))
qualityStatistics(edObject, train$EVENT)
```

simulateEvents *Imposes simulated events on the top of the data*

Description

Simulates Events on columns of a data frame or a matrix by applying different transformations. The events of type sinusoidal, square, binomial or ramp can be used.

Usage

```
simulateEvents(
  Data,
  Params,
  Event_type,
  Event_strength = NULL,
  Start_index = NULL,
  Event_duration = NULL,
  Percentage = NULL
)
```

Arguments

Data	Data frame or matrix containing the data to which the events will be introduced
Params	Numeric vector or vector of strings indicating the column names (in case Data is a data frame) or the column numbers (in case Data is a matrix) of the parameters in which an event will be simulated
Event_type	String vector indicating which type of transformation the parameters will undergo. Current valid options include sinusoidal, square, ramp and slowsinusoidal. If Params contains more that one element and Event_type only contains one element the same transformation will be applied to all given Params
Event_strength	(Optional) Numeric Vector indicating the amplitude. Only valid for sinusoidal and square transformations. When specified for other type of transformations it will have no effect. However it must have the same number of elements as Params.
Start_index	Numeric, indicates the index where the event should start
Event_duration	Numeric, indicates the number of steps the transformation should last. Default is 100
Percentage	(Optional) Numeric value from 0 to 1. Alternative input indicating the percentage of data that should be affected by the transformation. Either Event_duration or Percentage should be especified.

Value

Matrix or data frame containing the selected columns with simulated events

Examples

```
#Generate event of type sinusoidal and ramp on two columns of the stationBData data set
simupar<-c("B_PH_VAL","B_TEMP_VAL")
SimulatedEvents<-simulateEvents(stationBData,
                                simupar,Event_type = c("sinusoidal","ramp"),
                                Start_index = 2500)

#When specifiying Event_strength the lenght of the vector needs to match the number
#of elements in Params.
SimulatedEvents<-simulateEvents(stationBData,
                                simupar,Event_type = c("sinusoidal","ramp"),
```

```
                                Start_index = 2500,
                                Percentage = 0.2,
                                Event_strength = c(4,1))
```

---

stationBData *stationBData*

Description

Data for package testing purposes

Index