Majority-class classification anomaly detector: Supervised anomaly detection using a majority-class baseline to illustrate the hanc\_ml interface. Events correspond to predicted positive class above the threshold.

Objectives: This Rmd shows a simple supervised baseline for anomaly classification using hanc\_ml with a Majority Classifier (cla\_majority). It predicts the most frequent class in training. Steps: load packages/data, visualize, preprocess (split + normalize), define and fit, detect, evaluate, and plot results.

# Install Harbinger (only once, if needed)  
#install.packages("harbinger")

# Load required packages  
library(daltoolbox)  
library(harbinger)

# Load example datasets bundled with harbinger  
data(examples\_anomalies)

# Use the "tt" time series (labeled)  
dataset <- examples\_anomalies$tt  
  
head(dataset)

## serie event  
## 1 1.0000000 FALSE  
## 2 0.9689124 FALSE  
## 3 0.8775826 FALSE  
## 4 0.7316889 FALSE  
## 5 0.5403023 FALSE  
## 6 0.3153224 FALSE

# Plot the time series  
har\_plot(harbinger(), dataset$serie)



# Data preprocessing: split and scale  
  
  
train <- dataset[1:80,]  
test <- dataset[-(1:80),]  
  
norm <- minmax()  
norm <- fit(norm, train)  
train\_n <- transform(norm, train)  
summary(train\_n)

## serie event   
## Min. :0.0000 Mode :logical   
## 1st Qu.:0.2859 FALSE:76   
## Median :0.5348 TRUE :4   
## Mean :0.5221   
## 3rd Qu.:0.7587   
## Max. :1.0000

# Define Majority Classifier for events (hanc\_ml + cla\_majority)  
model <- hanc\_ml(cla\_majority("event", c("FALSE", "TRUE")))

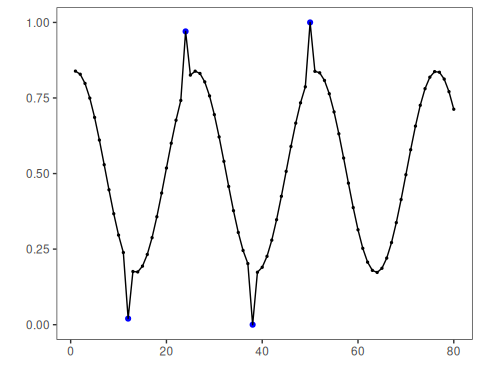
# Fit the model on training data  
model <- fit(model, train\_n)  
detection <- detect(model, train\_n)  
print(detection |> dplyr::filter(event==TRUE))

## [1] idx event type   
## <0 rows> (or 0-length row.names)

# Evaluate training performance  
evaluation <- evaluate(model, detection$event, as.logical(train\_n$event))  
print(evaluation$confMatrix)

## event   
## detection TRUE FALSE  
## TRUE 0 0   
## FALSE 4 76

# Plot training detections  
 har\_plot(model, train\_n$serie, detection, as.logical(train\_n$event))



# Prepare test data (apply same scaler)  
 test\_n <- transform(norm, test)

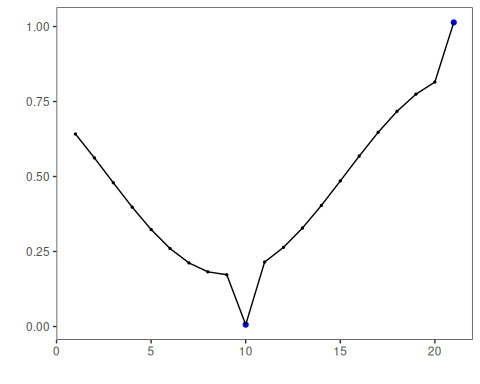
# Detect and evaluate on test data  
 detection <- detect(model, test\_n)  
  
 print(detection |> dplyr::filter(event==TRUE))

## [1] idx event type   
## <0 rows> (or 0-length row.names)

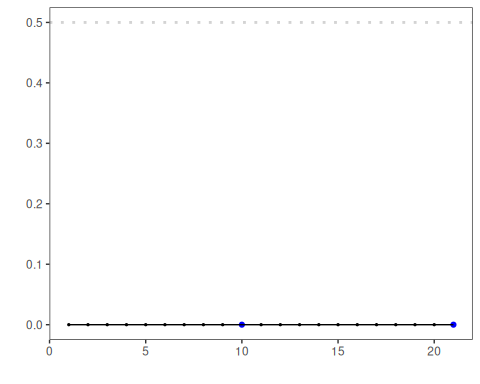
evaluation <- evaluate(model, detection$event, as.logical(test\_n$event))  
 print(evaluation$confMatrix)

## event   
## detection TRUE FALSE  
## TRUE 0 0   
## FALSE 2 19

# Plot test detections  
 har\_plot(model, test\_n$serie, detection, as.logical(test\_n$event))



# Plot residual scores and threshold  
har\_plot(model, attr(detection, "res"), detection, test\_n$event, yline = attr(detection, "threshold"))



References - Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. - Hyndman, R. J., Athanasopoulos, G. (2021). Forecasting: Principles and Practice. OTexts.