Using R and H2O Isolation Forest to predict car battery failures.

2019-May-24

This is a study about what might be if car makers start using machine learning in our cars to predict falures.

*# Loading libraries*  
suppressWarnings( suppressMessages( **library**( h2o ) ) )   
suppressWarnings( suppressMessages( **library**( data.table ) ) )  
suppressWarnings( suppressMessages( **library**( plotly ) ) )  
suppressWarnings( suppressMessages( **library**( DT ) ) )  
  
*# Reading data file*  
*# Data from: https://www.kaggle.com/yunlevin/levin-vehicle-telematics*  
dataFileName = "/Development/Analytics/AnomalyDetection/AutomovileFailurePrediction/v2.csv"  
carData = fread( dataFileName, skip=0, header = TRUE )  
carBatteryData = data.table( TimeStamp = carData$timeStamp  
 , BatteryVoltage = as.numeric( carData$battery )   
 )  
rm(carData)  
  
*# Data cleaning, filtering and conversion*  
carBatteryData = na.omit( carBatteryData ) *# Keeping just valid Values*  
  
*# According to this article:*   
*# https://shop.advanceautoparts.com/r/advice/car-maintenance/car-battery-voltage-range*  
*#*  
*# A perfect voltage ( without any devices or electronic systems plugged in )*   
*# is between 13.7 and 14.7V.*   
*# If the battery isn’t fully charged, it will diminish to 12.4V at 75%,*   
*# 12V when it’s only operating at 25%, and up to 11.9V when it’s completely discharged.*   
*#*  
*# Battery voltage while a load is connected is much slower*  
*# it should be something between 9.5V and 10.5V*   
*#*  
*# This value interval ensures that your battery can store and deliver enough*   
*# current to start your car and power all your electronics and electric devices*   
*# without any difficulty*  
  
carBatteryData = carBatteryData[BatteryVoltage >= 9.5] *# Filtering voltages greater or equal to 9.5*  
carBatteryData$TimeStamp = as.POSIXct( paste0( substr(carBatteryData$TimeStamp,1,17),"00" ) )  
carBatteryData = unique(carBatteryData) *# Removing duplicate voltage readings*  
carBatteryData = carBatteryData[order(TimeStamp)]  
  
  
*# spliting all data, using the last date as testing data and the rest for training.*  
lastDate = max( as.Date( format( carBatteryData$TimeStamp, "%Y-%m-%d" ) ) )  
trainingData = carBatteryData[ as.Date( format( carBatteryData$TimeStamp, "%Y-%m-%d" ) ) != lastDate ]  
testingData = carBatteryData[ as.Date( format( carBatteryData$TimeStamp, "%Y-%m-%d" ) ) == lastDate ]  
  
  
  
*################################################################################*  
*# Creating Anomaly Detection Model*  
*################################################################################*  
  
 h2o.init( nthreads = -1, max\_mem\_size = "5G" )

##   
## H2O is not running yet, starting it now...  
##   
## Note: In case of errors look at the following log files:  
## C:\Users\LaranIkal\AppData\Local\Temp\Rtmp6lTw4H/h2o\_LaranIkal\_started\_from\_r.out  
## C:\Users\LaranIkal\AppData\Local\Temp\Rtmp6lTw4H/h2o\_LaranIkal\_started\_from\_r.err  
##   
##   
## Starting H2O JVM and connecting: Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 1 seconds 899 milliseconds   
## H2O cluster timezone: America/Mexico\_City   
## H2O data parsing timezone: UTC   
## H2O cluster version: 3.24.0.2   
## H2O cluster version age: 1 month and 7 days   
## H2O cluster name: H2O\_started\_from\_R\_LaranIkal\_tzd452   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 4.44 GB   
## H2O cluster total cores: 8   
## H2O cluster allowed cores: 8   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## H2O API Extensions: Amazon S3, Algos, AutoML, Core V3, Core V4   
## R Version: R version 3.6.0 (2019-04-26)

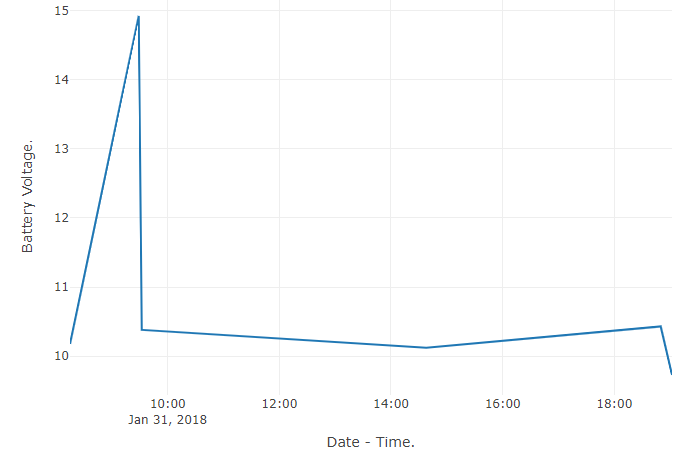
h2o.no\_progress() *# Disable progress bars for Rmd*  
 h2o.removeAll() *# Cleans h2o cluster state.*

## [1] 0

*# Convert the training dataset to H2O format.*  
 trainingData\_hex = as.h2o( trainingData[,2], destination\_frame = "train\_hex" )  
   
 *# Build an Isolation forest model*  
 trainingModel = h2o.isolationForest( training\_frame = trainingData\_hex  
 , sample\_rate = 0.1  
 , max\_depth = 32  
 , ntrees = 100  
 )  
   
 *# According to H2O doc:*   
 *# http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/if.html*  
 *#*  
 *# Isolation Forest is similar in principle to Random Forest and is built on the basis of decision trees.*   
   
 *# Isolation Forest creates multiple decision trees to isolate observations.*  
 *#*   
 *# Trees are split randomly, The assumption is that:*  
 *#*   
 *# IF ONE UNIT MEASUREMENTS ARE SIMILAR TO OTHERS,*  
 *# IT WILL TAKE MORE RANDOM SPLITS TO ISOLATE IT.*  
 *#*   
 *# The less splits needed, the unit is more likely to be anomalous.*  
 *#*   
 *# The average number of splits is then used as a score.*  
  
 *# Calculate score for training dataset*  
 score <- h2o.predict( trainingModel, trainingData\_hex )  
 result\_pred <- as.vector( score$predict )  
  
  
*################################################################################*  
*# Setting threshold value for anomaly detection.*  
*################################################################################*  
  
 *# Setting desired threshold percentage.*  
 threshold = .995 *# Let's say we have 99.5% voltage values correct*  
   
 *# Using avobe threshold to get score limit to filter anomalous voltage readings.*  
 scoreLimit = round( quantile( result\_pred, threshold ), 4 )  
   
  
   
*################################################################################*  
*# Get anomalous voltage readings from testing data, using model and scoreLimit got using training data.*  
*################################################################################*  
  
 *# Convert testing data frame to H2O format.*  
 testingDataH2O = as.h2o( testingData[,2], destination\_frame = "testingData\_hex" )  
   
 *# Get score using training model*  
 testingScore <- h2o.predict( trainingModel, testingDataH2O )  
  
 *# Add row score at the beginning of testing dataset*  
 testingData = cbind( RowScore = round( as.vector( testingScore$predict ), 4 ), testingData )  
  
 *# Check if there are anomalous voltage readings from testing data*  
 anomalies = testingData[ testingData$RowScore > scoreLimit, ]

*# Here there is and additional filter to ensure maintenance recommendation*  
 *# If there are more than 3 anomalous voltage readings, display an alert.*  
 **if**( dim( anomalies )[1] > 3 ) {   
 cat( "Show alert on car display: Battery got anomalous voltage readings, it is recommended to take it to service." )  
   
 plot\_ly( data = anomalies  
 , x = ~TimeStamp  
 , y = ~BatteryVoltage  
 , type = 'scatter'  
 , mode = "lines"  
 , name = 'Anomalies') %>%  
 layout( yaxis = list( title = 'Battery Voltage.' )  
 , xaxis = list( categoryorder='trace', title = 'Date - Time.' )  
 )  
 }

## Show alert on car display: Battery got anomalous voltage readings, it is recommended to take it to service.

[](https://i0.wp.com/1.bp.blogspot.com/-U4cUb9n8teo/XOht39VKTdI/AAAAAAAABs0/JGBJMr8whxcm6Z9OF2n5BP1ZiLy7u-lJQCLcBGAs/s1600/AnomaliesChart.PNG?ssl=1)

*if( dim( anomalies )[1] > 3 ) {*   
 *DT::datatable(anomalies[,c(2,3)], rownames = FALSE )*  
*}*

*Show*

*entries*

*Search:*

|  |
| --- |
| **TimeStamp** | **BatteryVoltage** |
|  |  |
|  |  |
| 2018-01-31T14:15:00Z | 10.175 |
|  |  |
| 2018-01-31T15:29:00Z | 14.88 |
|  |  |
| 2018-01-31T15:29:00Z | 14.92 |
|  |  |
| 2018-01-31T15:32:00Z | 10.38 |
|  |  |
| 2018-01-31T20:38:00Z | 10.12 |
|  |  |
| 2018-02-01T00:50:00Z | 10.43 |
|  |  |
| 2018-02-01T01:02:00Z | 9.727 |
|  |  |

*Showing 1 to 7 of 7 entries*

[*Previous1Next*](https://www.blogger.com/null)

*Using this approach we may prevent failures on cars, not only for batteries but for many cases when sensors are used.*