Preventive maintenance using data science

BK

Sunday, December 20, 2015

Business context

- A turbo charger can fail expensively if not maintained.
- The conditions for such failure are to be determined and prevented through timely warning.
- Final outcome is to display an alert, given current operating conditions.

Predictors of turbo failure

Early information on failure conditions indicates the following potential candidates,
- Total number of hours of operation since prior service
- Ambient temperature (degree C)
- Turbine bearing temperature (degree C)
- Particulate dust in the environment (ppm)
- Operating speed (rpm)
- Turbo pressure (mbar)
- Application (marine or mining)

Data sources

Embedded devices/ IoT
- Total number of hours of operation since prior service
- Ambient temperature (degree C)
- Turbine bearing temperature (degree C)
- Particulate dust in the environment (ppm)
- Operating speed (rpm)
- Turbo pressure (mbar)

Manual data enhancement
- Application (factor, LOV: marine, mining)
- Failure indicator
- Dust proxy

Data features

- There is a correlation between two of the regressors - "Turbo Pressure" & "Speed", so one can be ignored

- The "Dust" data might be hard for an IoT device to produce. Instead, a proxy "dustProxy" can be created that used from the manually applied "Application" data

- One of the predictors "Application" is a factor variable while the others are numeric

Analytics problem

A logistic regression model was chosen to determine failure conditions. Being parametric, it provides insight into why failure occurs (which a classification model would not provide)

Data mock-up

```
# Loading requisite libraries
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2

#library(dplyr)
#library(ggplot2)



# Creating sample data.Note data enhancement/ proxy data creation for dust.
set.seed(11)
priorService <- sort(runif(10000,min=0,max=10000))
ambTemp <- sort(rnorm (10000, mean=15, sd=5))
bearTemp <- sort(rep(25:26,each=5000))
dust <- sort(rnorm (10000, mean=100, sd=20))
dustProxy <- rep(100,10000)
speed <- sort(rnorm (10, mean=10000, sd=200))
pressure <- (speed^2)/100000
application <- sample(c("marine","mining"), 10000, replace=T)
dustProxy[application=="marine"]=0
failure <- rep("NO",10000)

# Adding failure conditions
failure[priorService > 7000 & ambTemp > 30]="YES"
failure[dust > 100 & dustProxy > 0]="YES"
failure[speed > 17000]="YES"

# Assembling the data frame
turbo <- data.frame(priorService,ambTemp,dustProxy,speed,pressure,failure)
str(turbo)

## 'data.frame':    10000 obs. of  6 variables:
##  $ priorService: num  0.564 0.941 1.093 5.183 5.246 ...
##  $ ambTemp     : num  -6 -5.71 -5.24 -3.71 -3.41 ...
##  $ dustProxy   : num  100 0 0 0 100 0 100 0 100 100 ...
##  $ speed       : num  9730 9737 9745 9883 9940 ...
##  $ pressure    : num  947 948 950 977 988 ...
##  $ failure     : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...

# Check to see failure data appears correctly in data frame
table(turbo$failure)

##
##   NO  YES
## 7462 2538

table(failure)

## failure
##   NO  YES
## 7462 2538

# slicing data
inTrain <- createDataPartition(y=turbo$failure,p=0.75,list=F)
training <- turbo[inTrain,]
testing <- turbo[-inTrain,]
str(training)

## 'data.frame':    7501 obs. of  6 variables:
##  $ priorService: num  0.941 1.093 5.183 5.246 10.787 ...
##  $ ambTemp     : num  -5.71 -5.24 -3.71 -3.41 -2.34 ...
##  $ dustProxy   : num  0 0 0 100 100 100 100 100 0 0 ...
##  $ speed       : num  9737 9745 9883 9940 10059 ...
##  $ pressure    : num  948 950 977 988 1012 ...
##  $ failure     : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
```

Model training, accuracy check prediction

It is seen that all the regressors chosen are statistically significant. So, the none hypothesis that the regressor does not affect the outcome can be rejected.
A further test using the Confusion Matrix on the test data shows a satisfactory accuracy of 99%. Please note that this test data has not been used to train the model. So, model performance with new data is deemed to be satisfactory.
Note: The model is generating warnings that need to be addressed. Most likely related to the data mock up. More time is needed.

```
# training the model on training data and checking its accuracy on testing
data
modelFitglm <- train(failure~.,data=training,method="glm", family=binomial)
summary(modelFitglm$finalModel)

##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
##   0.00     0.00     0.00     0.00     8.49
```

```
##
## Coefficients:
##                  Estimate Std. Error    z value Pr(>|z|)
## (Intercept)  -1.926e+16  4.310e+09   -4469745   <2e-16 ***
## priorService  4.869e+11  1.302e+03  373880100   <2e-16 ***
## ambTemp       1.622e+13  7.522e+05   21566508   <2e-16 ***
## dustProxy     3.737e+13  1.550e+04 2410610330   <2e-16 ***
## speed         2.546e+12  8.678e+05    2934372   <2e-16 ***
## pressure     -1.266e+13  4.367e+06   -2898786   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8498.74  on 7500  degrees of freedom
## Residual deviance:  432.52  on 7495  degrees of freedom
## AIC: 444.52
##
## Number of Fisher Scoring iterations: 20
```

```
pred <- predict(modelFitglm, testing)
confusionMatrix(pred, testing$failure)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   NO  YES
##        NO  1865    2
##        YES    0  632
##
##                Accuracy : 0.9992
##                  95% CI : (0.9971, 0.9999)
##     No Information Rate : 0.7463
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9968
##          Pos Pred Value : 0.9989
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7463
##          Detection Rate : 0.7463
##    Detection Prevalence : 0.7471
##       Balanced Accuracy : 0.9984
##
##        'Positive' Class : NO
##
```

Failure warning

Embedded devices typically carry embedded data (e.g. in a car, air/fuel mixture to use depending on load, speed etc.). In the same way, predictions of failure for many possible combinations of predictors can first be generated using the prediction function above. Then, these can be stored on the embedded device. When failure conditions are encountered, the device will warn the operator. Two samples, one each for "YES" and "NO" for alert are shown.

It is understood that "hours to failure" might be better. This model can be extended to achieve the same.

Or for simplicity's sake, a factor of safety can be added to all predictors so the operator is warned before failure conditions are reached.

```
# demo of failure warning scenario with sample data
testing[2486,]

##      priorService  ambTemp dustProxy    speed pressure failure
## 9957     9956.487 27.92051       100 10058.77 1011.789     YES

operatorAlert <- pred[2486]
operatorAlert

## [1] YES
## Levels: NO YES

# demo of NO failure warning scenario with sample data
testing[2485,]

##      priorService  ambTemp dustProxy    speed pressure failure
## 9944     9944.891 27.47141         0 9882.824  976.702      NO

operatorAlert <- pred[2485]
operatorAlert

## [1] NO
## Levels: NO YES
```