Predict Failure

Suresh Ooty

Table of Contents

Predict Failures

An additional *DUMMY* attribute *LeadToFailure* was added to the 2015 dataset. This variable will have *TRUE* if there was a shutdown identified in next 60 minutes (only forward) using the timestamped order. This transformaiton was achieved using an **Alteryx** workflow.

Note: It was observed by the researcher that there are regular shutdowns of the plant during the holiday season such as New year, May 1st and Christmas. Initially it was considered that these holidays shall not be included in to the logic of constructing *LeadToFailure* variable. But, a reasonable second thought that, if there is a shutdown either intentional or non-intentional, the sequence of shut down steps shall be in similar pattern.

Rule on LeadToFailure: For this variable to be TRUE, both *Production* & *Hay_out_waste* should be '0'. This was validated with the SME.

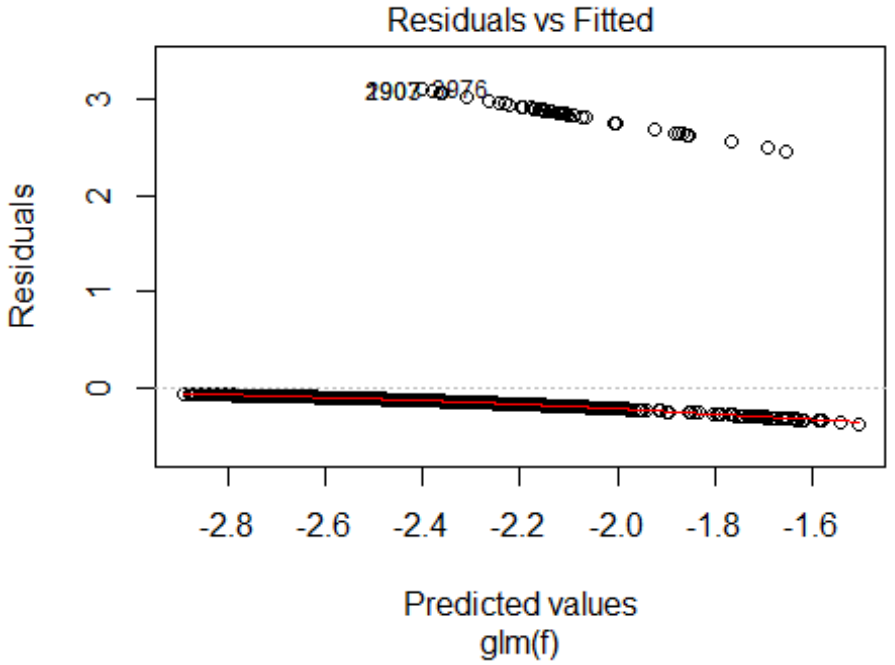
Logistic Regression

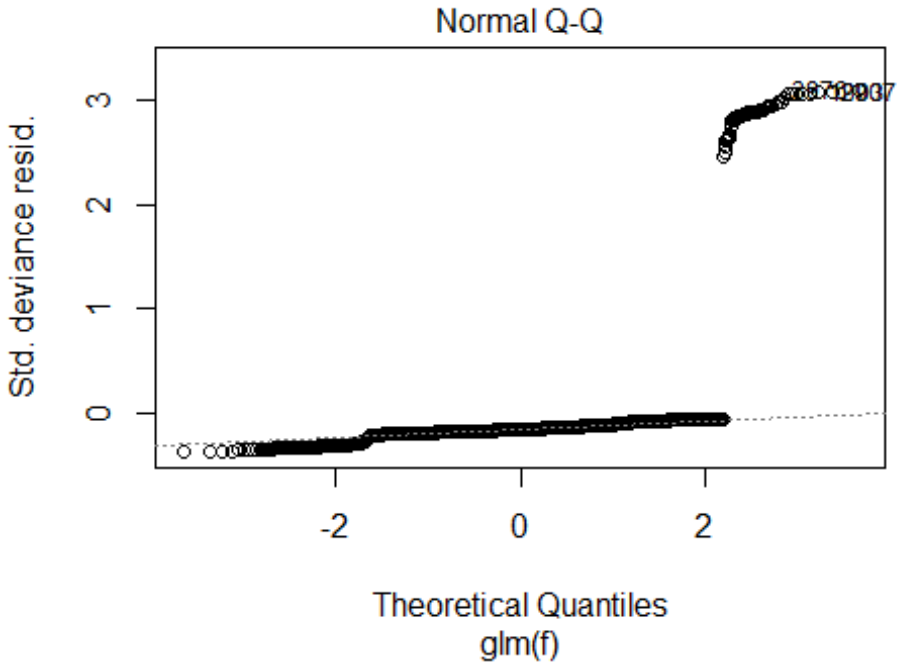
;

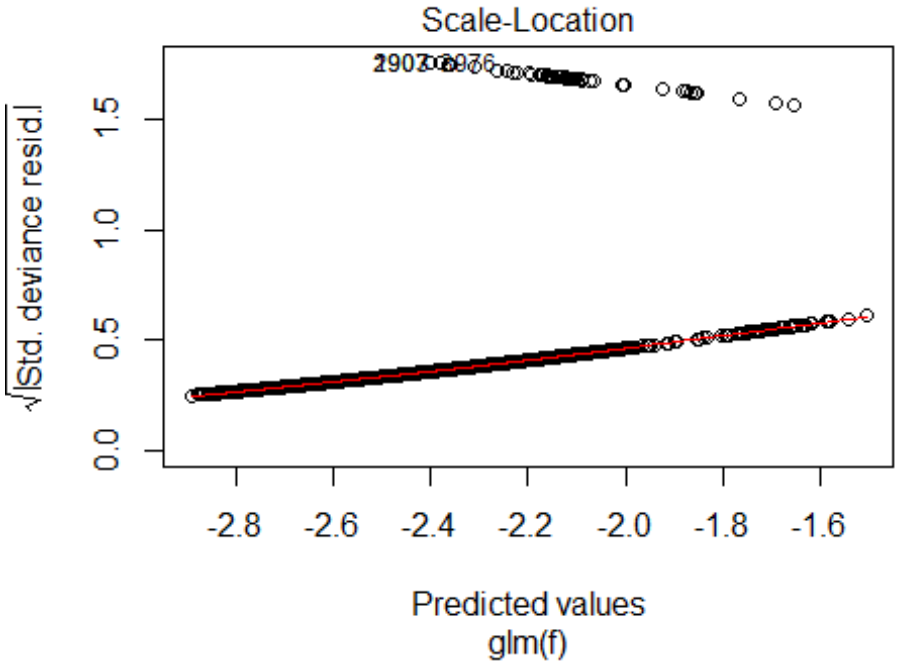
Using the min-max transformation dataset

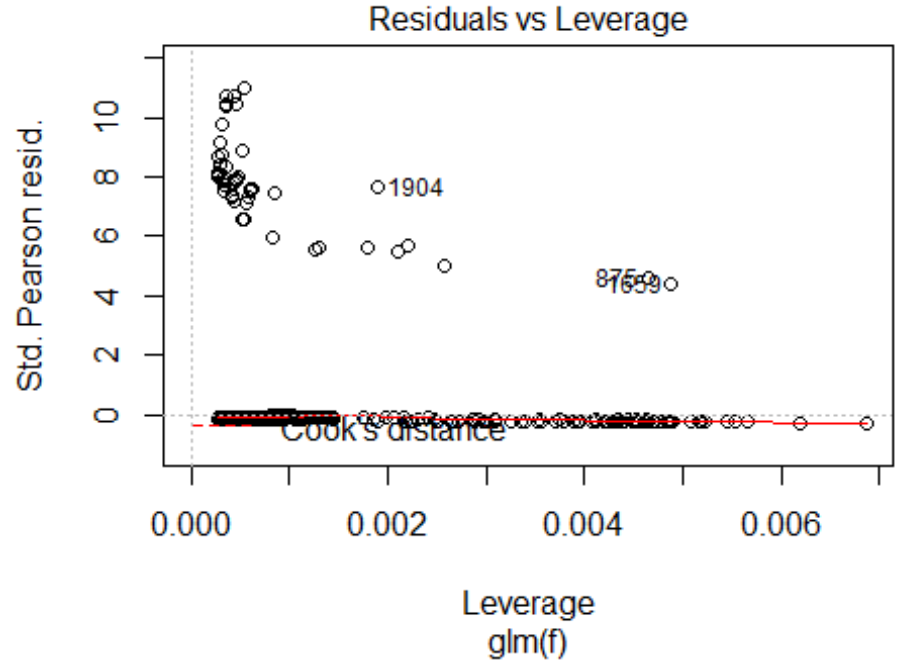
```
mins <- apply(data_15, 2, min)
maxs <- apply(data_15, 2, max)
```

```r
scaled_data <- as.data.frame(scale(data_15, center = mins,scale = maxs -
mins))
# first 4000 lines have 200 records with leadToFailure = TRUE
train_ <- scaled_data[1:4000,]
test_ <- scaled_data[4001:15000,]
L2F <- test_$LeadToFailure

n <- names(train_)
# the predictors were finalized after few iterations
f <- as.formula(paste("LeadToFailure ~", paste(n[!n %in% c("LeadToFailure",
"WatMCon","NatGCon","Hay_out_waste","WatGCon","CmpACon","EleCon","Production"
)], collapse = " + ")))
logR.mdl <- glm(f,data = train_,family = binomial(link = "probit"))
summary(logR.mdl)

##
## Call:
## glm(formula = f, family = binomial(link = "probit"), data = train_)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3698  -0.1834  -0.1593  -0.1293   3.0987
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.2902     0.2646  -8.654  < 2e-16 ***
## SteCon       -0.7883     0.2627  -3.001  0.00269 **
## WatWGen       1.0463     0.3716   2.815  0.00487 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 580.78  on 3999  degrees of freedom
## Residual deviance: 561.31  on 3997  degrees of freedom
## AIC: 567.31
##
## Number of Fisher Scoring iterations: 8

plot(logR.mdl)
```


Residuals vs Fitted
Residuals
Predicted values
glm(f)


Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(f)






```r
p<- predict(logR.mdl,newdata = test_,type = "response")

#pr <- prediction(p,L2F)
#prf <- performance(pr, measure = "tpr", x.measure = "fpr")
```

```
#plot(prf)
#auc <- performance(pr, measure = "auc")
#auc <- auc@y.values[[1]]
#auc
```

Observations: The residual plots (Q-Q) indicate the obvious outliers, which need to be removed if this study need to be improved. The ROC curver area is about 65%

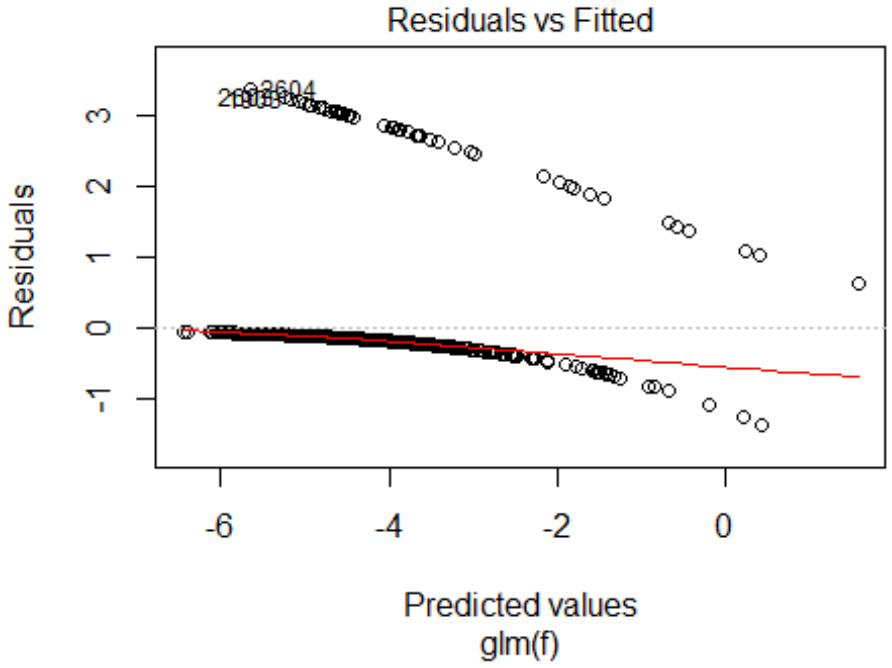
PLEASE NOTE THAT THE PREDICTION FUNCTION STARTED FAILING FEW MINUTES BEFORE SUBMISSION :-(65% was observed earlier.

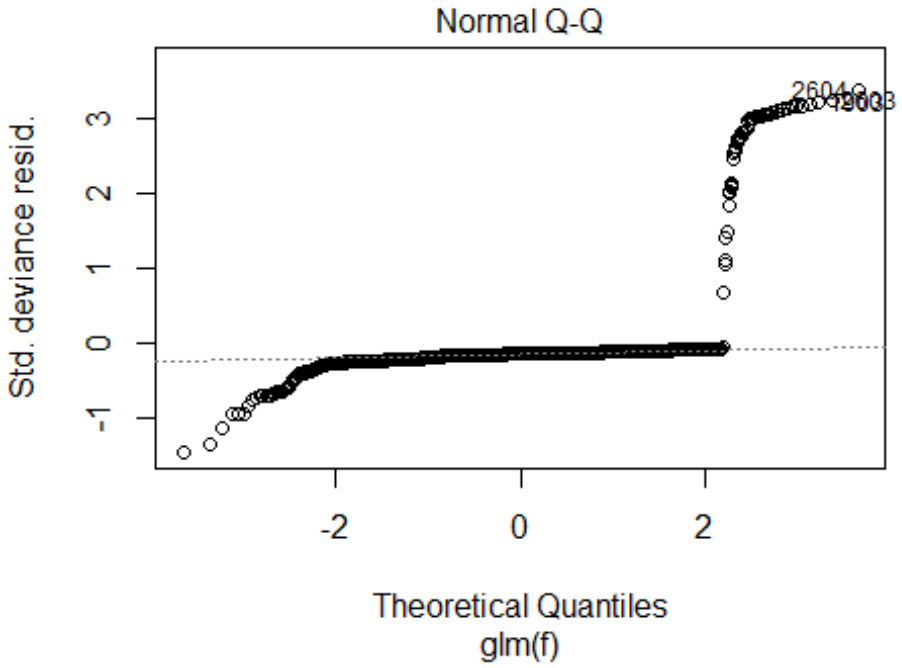
Using the LOG transformed dataset

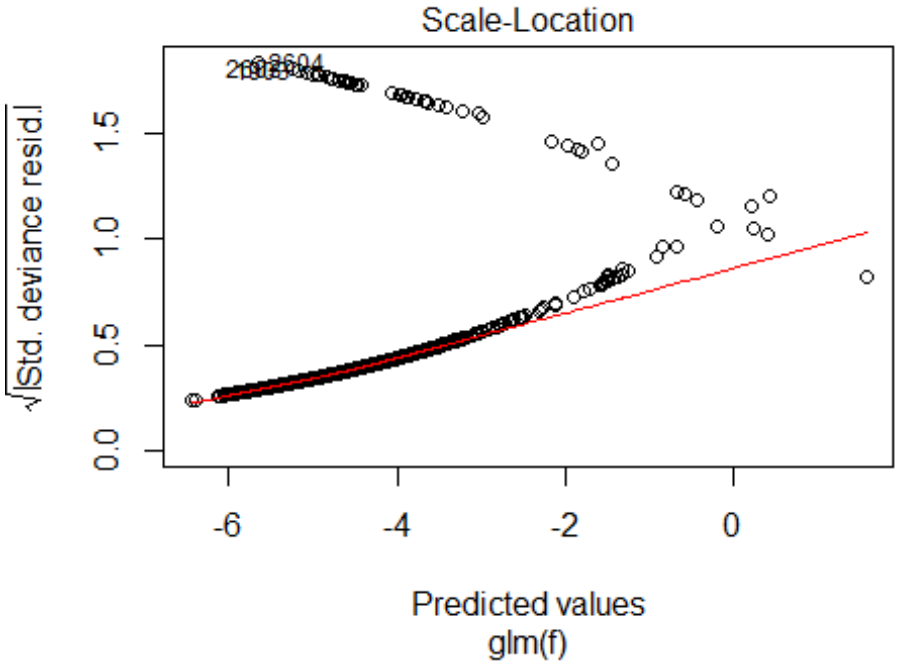
```
log.data_15 <- cbind.data.frame("LeadToFailure"=data_15$LeadToFailure,
log(data_15[,2:10]+1))
log.train_ <- log.data_15[1:4000,]
log.test_ <- log.data_15[4001:15000,]

n <- names(log.train_)
# the equation was finalized after few iterations
f <- as.formula(paste("LeadToFailure ~", paste(n[!n %in%
c("LeadToFailure","WatGCon","WatMCon","CmpACon","WatWGen")], collapse = " +
")))
logR.mdl <- glm(f,data = log.train_,family = binomial(link = "logit"))
summary(logR.mdl)

##
## Call:
## glm(formula = f, family = binomial(link = "logit"), data = log.train_)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3667  -0.1532  -0.1360  -0.1179   3.3646
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -18.01303    5.87079  -3.068  0.00215 **
## Production     -6.67506    1.15440  -5.782 7.37e-09 ***
## Hay_out_waste   0.16914    0.05872   2.881  0.00397 **
## EleCon          2.48466    0.99550   2.496  0.01256 *
## NatGCon        -0.40548    0.19998  -2.028  0.04260 *
## SteCon          0.41468    0.18389   2.255  0.02413 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 580.78  on 3999  degrees of freedom
## Residual deviance: 513.14  on 3994  degrees of freedom
## AIC: 525.14
```

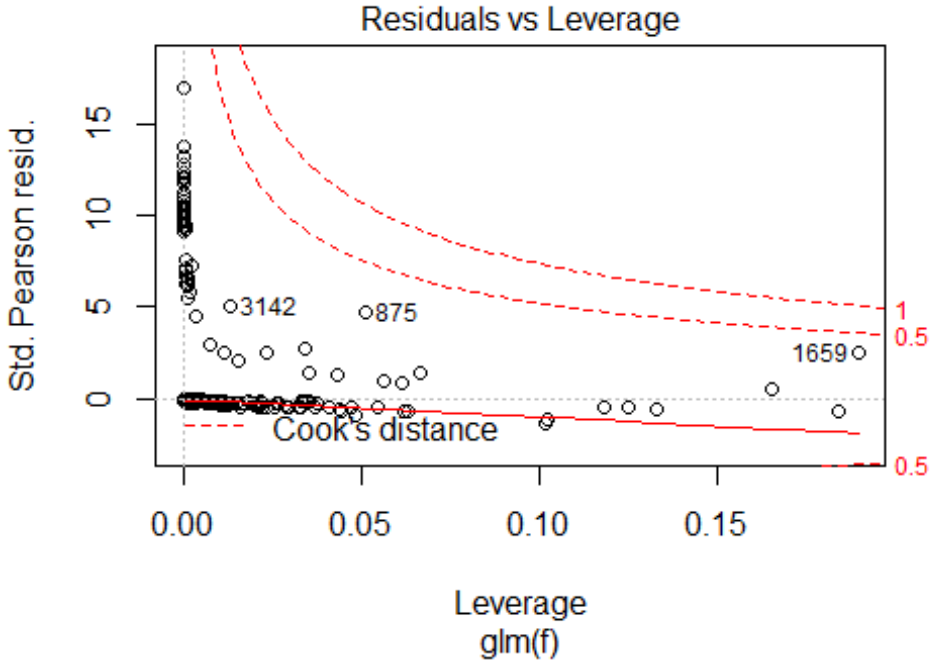
```
##
## Number of Fisher Scoring iterations: 7
```

```
plot(logR.mdl)
```










```
p<- predict(logR.mdl,newdata = log.test_,type = "response")
#pr <- prediction(p,log.test_$LeadToFailure)
#prf <- performance(pr, measure = "tpr", x.measure = "fpr")
#plot(prf)
```

```
#auc <- performance(pr, measure = "auc")
#auc <- auc@y.values[[1]]
#auc
```

Notes on Log Transformed dataset: It has been observed consistently throughout the LOG transformed data has performed better. However, the dataset still contains outliers that needs to be dealt with.

Conclusion on Logistic Model: Using both type of Min-max transformed dataset, with different iterations on link functions with 'probit','logit' & 'cloglog' the ROC curve had about 65% area covered in the plot. However, when the LOG transformed dataset was applied with similar iteration using the link functions, the ROC area is much better at 75%.

PLEASE NOTE THAT THE PREDICTION FUNCTION STARTED FAILING FEW MINUTES BEFORE SUBMISSION :-(75% was observed earlier.

- It is important to note that the "Train" dataset has about 4000 observations, that is throughout the year including the holiday shutdowns. This is more or less trying to solve a Time-series failure prediction, using Logistic Regression. The logistic regression model would not have assumed any auto correlations between each observation.
- Like it was noted in the 'Regression on 3 partitions', there seems to be multiple patterns hiding inside the sample dataset.

Notes for Future: Like the slicing of data based on the Production variable at different MT values, there shall be multiple *SEASONAL* slices applied on the dataset, like summer, winter,etc., or even monthly.

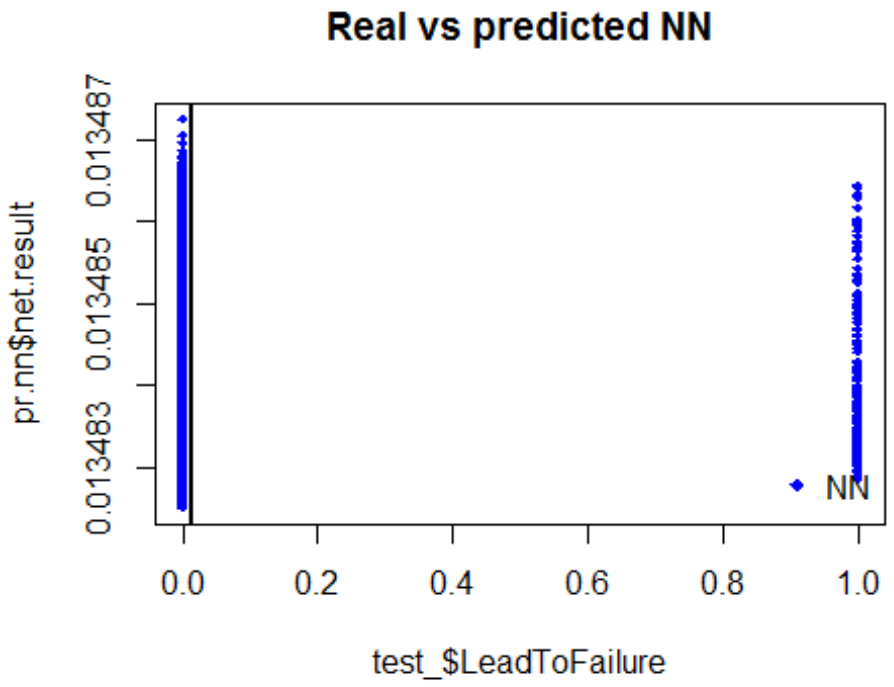
Neural Network

A similar approach, i.e., trying to predict the logistic outcome of failures on the same dataset, lead to few more insights and conclusions.

Using the min-max transformation dataset

```
n <- names(train_)
f <- as.formula(paste("LeadToFailure ~", paste(n[!n %in% "LeadToFailure"], 
collapse = " + ")))
nn <- neuralnet(f, data = train_, hidden = c(7,5,3,1),linear.output = 
FALSE,lifesign = "minimal", threshold = 0.1)

## hidden: 7, 5, 3, 1    thresh: 0.1    rep: 1/1    steps:      10  error: 
27.12199 time: 0.08 secs

plot(nn)

# predict using the test set
# leaving the y, passing the rest of the variables
```

```
pr.nn <- compute(nn,test_[,2:10])
plot(test_$LeadToFailure,pr.nn$net.result,col='blue',main='Real vs predicted
NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='blue', bty='n')
```



__Notes:__ High error values (~27) and the predicted values are always close 0. The model did not predict the FAILURE not even once.

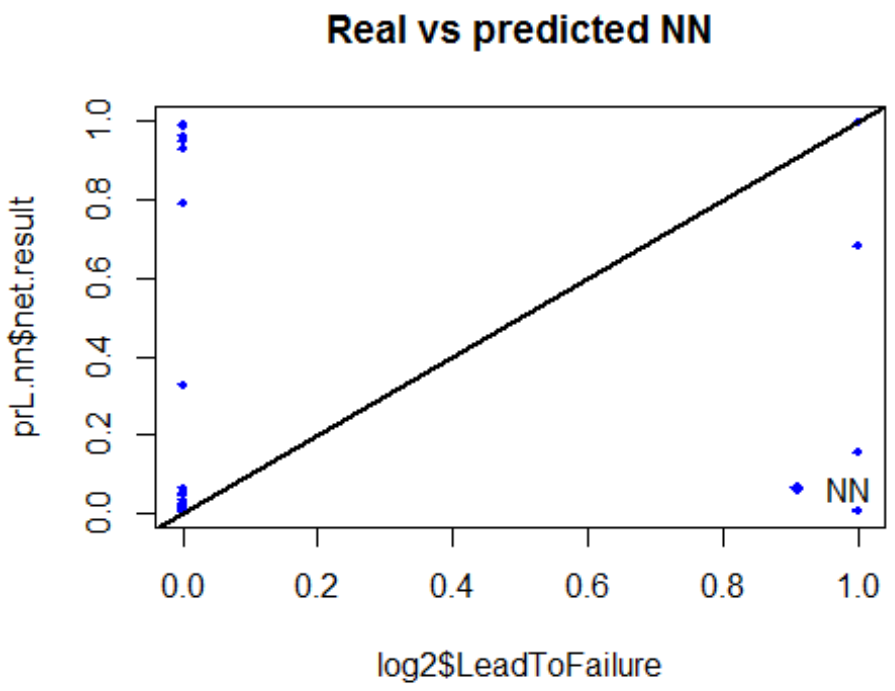
Using the LOG transformed dataset

```
log.data_15 <- cbind.data.frame("LeadToFailure"=data_15$LeadToFailure,
log(data_15[,2:10]+1))
# An observation was made that when the dataset size was brought down to 2000
, from 4000 the error came down drastically.
log.train_ <- log.data_15[1:2000,]
log.test_ <- log.data_15[4001:6000,]
n <- names(log.train_)
f <- as.formula(paste("LeadToFailure ~", paste(n[!n %in%
c("LeadToFailure","WatGCon","WatMCon","CmpACon","WatWGen")], collapse = " +
")))
nn <- neuralnet(f, data = log.train_, hidden = 4,linear.output =
FALSE,lifesign = "minimal", threshold = 0.1)

## hidden: 4    thresh: 0.1    rep: 1/1    steps:     110    error: 9.98407
time: 0.15 secs

plot(nn)
```

__Notes:__ The error came down.

Predicting the failure using the Neural net

```
#dropping unwanted columns from test data
log2 <- log.test_[,-4]
log2 <- log2[1:6]
prL.nn <- compute(nn,log2[,2:6])
MSE.nn <- sum((log2$LeadToFailure - prL.nn$net.result)^2)/nrow(log2)
print(paste("Mean Square Error in NN = ",MSE.nn))

## [1] "Mean Square Error in NN =  0.0154897282424603"

# plot the results
plot(log2$LeadToFailure,prL.nn$net.result,col='blue',main='Real vs predicted
NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='blue', bty='n')
```



Observations: LOG transformed data (with reduced size) behave very similar or close to the Min-Max transformed dataset. But the predicted values have improved very little but still always close to ZERO, meaning that Neural network did not predict FAILURE (TRUE) condition at all.

Conclusions

Overall, the models used here were not satisfactory (even if it has ROC values of 75%) in my opinion. And the seasonality & trend influences are very evident throughout. Moreover, using *DUMMY* variable that leads to a failure is an idea, not a robust method to predict failures in a large pool of dataset like this. A very deep analysis is necessary to realign the approach in predicting the failure. This could be possible by adding more attributes given at machine level with more sophisticated application of methods.