NORTH CAROLINA STATE UNIVERSITY

ISE789/OR791 – Data Science for Industrial & System Engineers

Spring 2019

Predicting Remaining Battery Life of Edge 540 Aircrafts

Authors:

Pragun Gupta

Muhammad Ali Haider

Vikram Patil

Advisor:

Dr. Xiaolei Fang

NC STATE UNIVERSITY

April 26, 2019

Contents

1. Introduction

Due to technological advancements and more awareness about renewable energy, electric vehicles are becoming increasingly common. However, the dream of an electric commercial aircraft is still far from reality. There are many challenges that are presented when a commercial aircraft is powered using electric propulsion system, for example, the absence of a practical cooling system, the weight of the engines etc. Yet the major limiting factor is the battery, especially estimating the remaining battery operating time and subsequently the remaining flying time of the aircraft. It is imperative to predict remaining flying time in order to alert the flyer to initiate landing procedure before the battery's State of Charge (SOC) is two minutes away from falling below the safe 30% threshold. There is an ongoing research at the National Aeronautics and Space Administration (NASA) to address this issue.

2. Project Objective

This project aims to build a predictive model to forecast remaining battery life for Edge 540 aircraft using measures of aircraft state and battery SOC.

3. Dataset

The data used for this analysis is from ground test experiments conducted by NASA on Edge 540 aircrafts. These are aerobatic aerial vehicles and their sub-scale version was used for these experiments. There is a large amount of data available for these experiments. For our analysis, we have used data from 9 experiments, which is approximately 4 million rows.

3.1 Variables

The dataset consists of 18 variables describing three major types of measurements, explained below:

1) Remaining flying time estimate for the aircraft – This parameter is the response variable in our analysis. It is measured in seconds.
2) Aircraft states – Three variables namely Revolutions per minute (RPM), Forward Motor Controlled Sensor and After Motor Controlled Sensor measure the aircraft state at a given point of time.
3) Battery SOC estimates – There are 14 variables describing the state of charge of different batteries used in the aircraft. Batteries' state of charge has been described by measures of their voltage, current, and temperature at a given point of time.

4. Data Analysis

4.1 Data Preprocessing

4.1.1 Feature Engineering

Since the response variable in our dataset is dependent on functions of the predictor parameters, functional data analysis (FDA) seems like a natural option. FDA is a collection of statistical techniques specifically developed to analyze curve data (Frøslie, 2013). However, Yogev (2014) mentions that it is a common practice to use simple summary measures, such as the area under the curve (AUC) to obtain information from the functions. Similarly, for our analysis we have used estimated AUC for independent variables to predict the remaining battery life.

The notion behind using AUC to determine remaining battery life is that all the predictors must be a function of power and responsible for battery consumption over time. Since the area under the power - time graph provides an estimate of energy consumption, the area under the predictor - time curve will yield the proportion of battery consumed over time. Applying this methodology on all the predictors provide information about the amount of battery consumed over time.

Thus, we performed feature engineering on the parameters by taking their cumulative sum and used the subsequent values for prediction modeling. Figure 1 shows the plot of the RPM curve against time. Figure 2 depicts the cumulative sum of RPM over time.

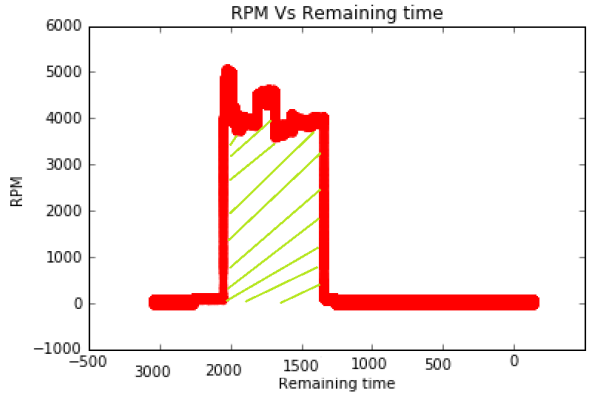


Figure 1: RPM response over time

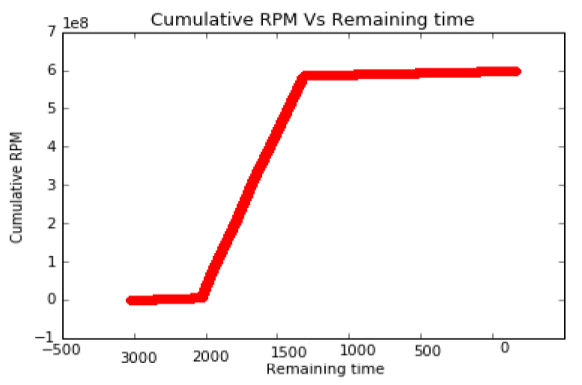


Figure 2: Cumulative sum of RPM over time

4.1.2 Correlation among the variables

Since many predictors are functions of voltage, current, and temperature; a high correlation was expected among the parameters. Figure 3 shows the correlation plot. We can observe that the predictors are highly correlated with each other. It can also be noted that all the predictors are positively correlated. There is a high correlation between the first seven predictors and also between the remaining ten predictors. Therefore, two major clusters in the predictor variables have been identified which can be used for dimensionality reduction.

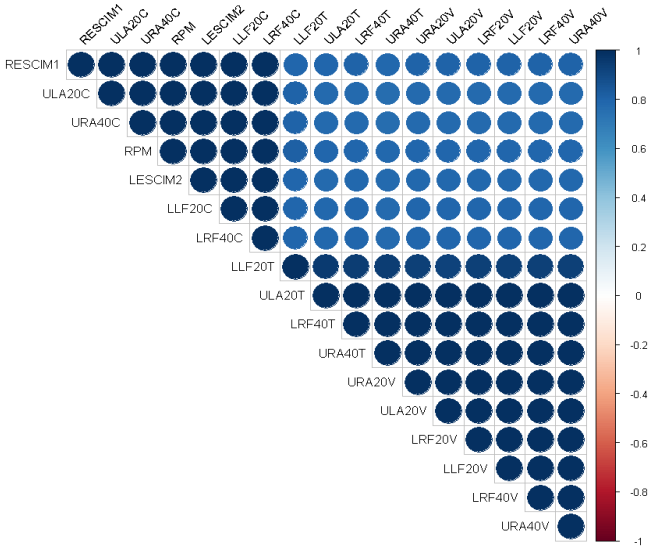


Figure 3: Correlation plot between covariates.

4.1.3 Principal component analysis (PCA)

To reduce the dimensionality of the dataset, PCA is performed to determine the amount of variance that can be captured with a lower number of parameters. Figure 4 shows an elbow plot of the PCA results. We can note that the first two principal components captured 99% of the variance. Since our dataset has 4 million data points, the dimensionality reduction from 17 variables to just 2 variables by PCA means a significant reduction in computational cost.

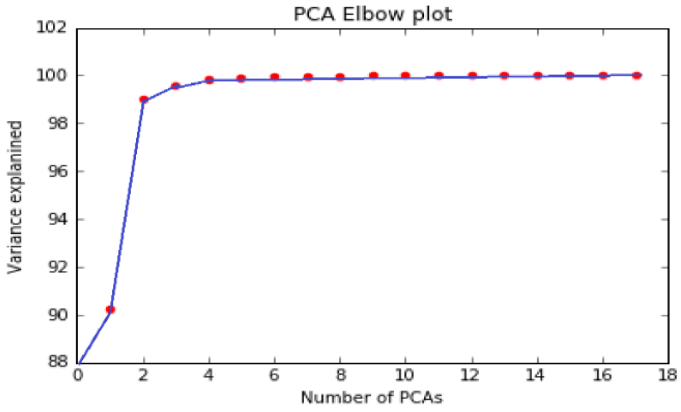


Figure 4: Cumulative variance explained by number of principal components.

4.2 Prediction modeling

After performing feature engineering and PCA, the dataset was prepared for applying prediction algorithms. For our analysis, we used two prediction models - Random forest and Neural network. Their results have been discussed below:

4.2.1 Random Forest

For the Random Forest model, we used 50 decision trees and 2 principal components. Additionally, we used data from 8 experiments as the training dataset and 1 experiment as the test dataset. This model performed well on the training set, with an MSE of 27431.40. However, it showed some deviation from the actual values on the test set, as depicted in Figure 5.

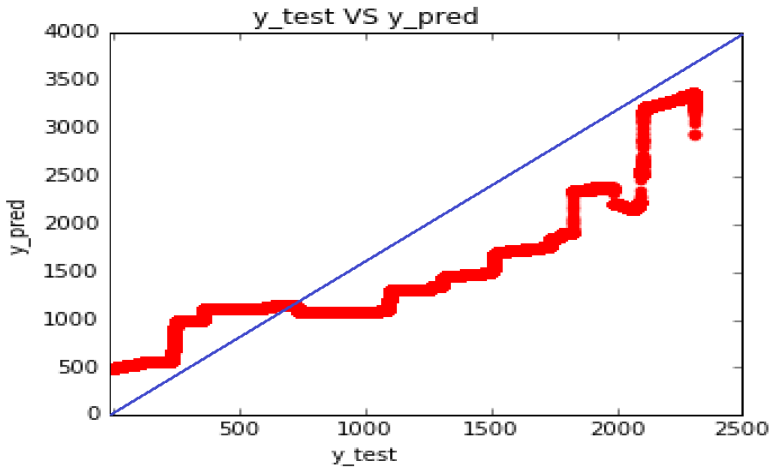


Figure 5: Prediction values vs actual values of remaining flight time using Random Forest model.

4.2.2 Neural Network

The distribution of the response variable is shown in Figure 6. This distribution is unique as it has positive values with uniform distribution for initial range and exponential distribution at

higher values. Clearly, generalized linear models of known distributions may not be suitable for modeling this response. Additionally, from the scientific understanding of battery and flight technologies, it is expected to have a non-linear relationship between covariates including many significant higher order interactions. Therefore, a neural network is considered for building a predictive model. As neural network can perform variable selection on its own by assigning appropriate weights, the first five principal components which explain 99.999% of the variance are used as input predictors for this model. The number of hidden layers and nodes are varied to identify a better fitting model.

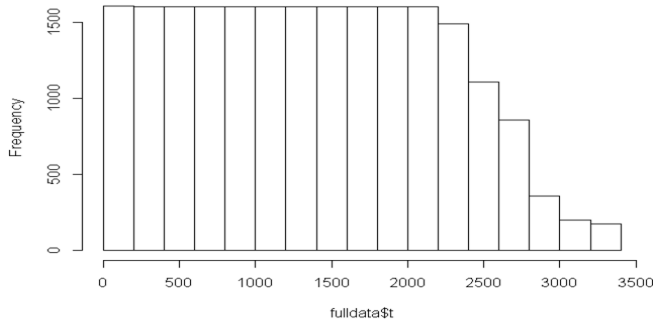
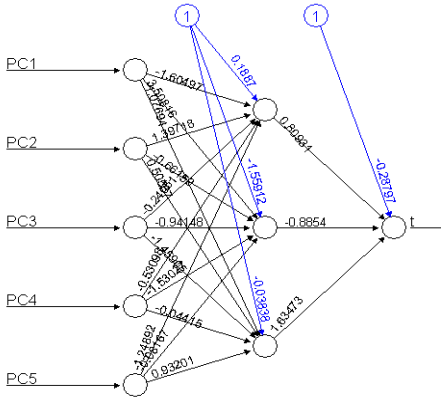
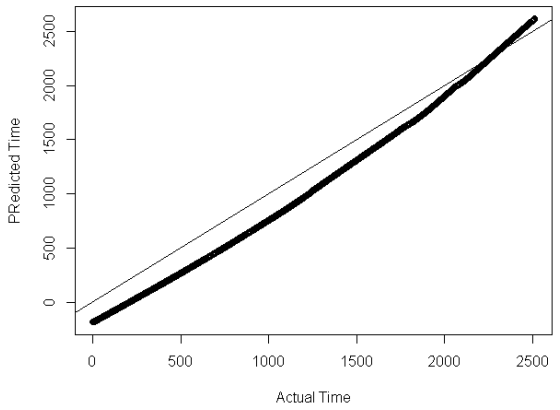


Figure 6: Histogram of remaining flight time response.

A total of four neural network configurations were considered. The first model with a single hidden layer of 3 nodes is built. The neural network model with weights and bias on this model after training is shown in Figure 7-(a) and prediction on a new battery-flight data with this model is shown in Figure 7-(b). It can be observed that this model under-predicts most of the



(a) Weights and Bias in Neural Net

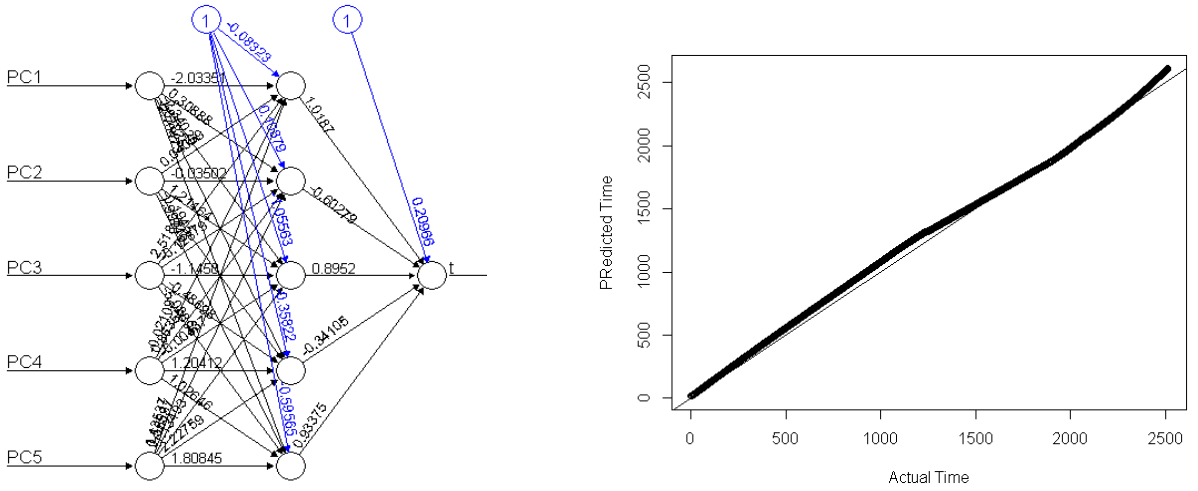


(b) Predicted time vs actual time response

Figure 7: Results with single hidden layer of 3 nodes neural network model.

time with a small over-prediction towards the end. The MSE on prediction is 35763 for this model.

Furthermore, the second model with a single hidden layer of 5 nodes is trained. The additional nodes in the model would capture additional complex relationship and interactions between predictors. Figure 8-(a) shows the weights and bias from the trained neural net and prediction plot for this model. It can be observed that prediction accuracy increases significantly from the previous model of 3 nodes. The prediction time follows the actual time very closely all the



(a) Weights and Bias in Neural Net

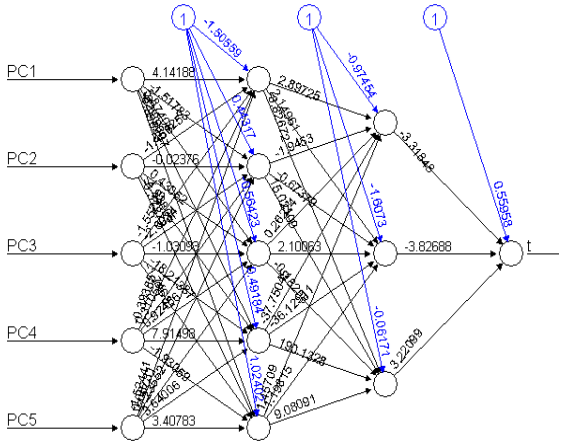
(b) Predicted time vs actual time response

Figure 8: Results with single hidden layer of 5 nodes neural network model.

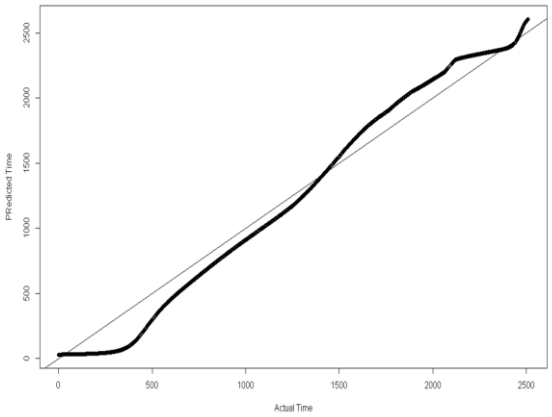
time. The MSE of the prediction is reduced to 2717. Figure 8-(b) clearly illustrates the significant reduction in prediction error.

Further, complex deep learning neural net configurations were also explored. The neural net with two hidden layers with the first layer of 5 nodes and the second layer of 1 node is trained. It is observed that prediction accuracy decreases for the model with a single layer of 5 nodes with MSE value of 11896. Also, a model with the first layer of 5 nodes and the second layer of 3 nodes showed poor prediction performance with MSE of 15709. The weights and bias with and prediction results this model is shown in Figure 9. It is observed that prediction is significantly worse with 2-layer model than single layer 5 nodes model. Although training error decreased with the complex neural network, prediction error was observed to be least with the

model having a single layer of 5 nodes. Therefore, this model is the best model with the neural network modeling approach.



(a) Weights and Bias in Neural Net



(b) Predicted time vs actual time response

Figure 9: Results with 2 Layer neural network of 5 nodes in first layer and 3 nodes in second layer

4.2.3 Model Comparison

The prediction performance of Random Forest and Neural Network is compared in terms of prediction accuracy, computational time, and robustness of the model in handling higher dimensional data. Prediction MSE for the best Neural Network model was 2717 whereas Prediction MSE for Random forest model was 27431. Clearly, the neural network showed significantly smaller prediction error. However, the computational time required for training neural network was significantly higher (about 3 times) even with a reduced data set. Additionally, neural network can handle feature selection whereas Random forest requires a feature selection process prior to model building. Therefore, the top two principal components were used for random forest whereas the top five principal components were considered for neural network models. Neural network models are more robust in handling high-dimensional data than random forest. Additionally, random forest model was implemented in Python whereas neural network was built in R. Clearly, there are differences in the data, programming approach, the computational infrastructure used for both the modeling approaches, therefore it is not a fair comparison. However, just based on MSE of prediction, a neural network model with a single hidden layer of 5 nodes is observed to the best model for prediction.

5. Conclusions

This project addressed the problem of building a predictive model to forecast the remaining battery operating time and subsequently remaining flying time for an electric powered aircraft, using the measures of aircraft state and battery SOC. After feature engineering and Principal Component Analysis, two predictive models were developed – Random Forest and Neural Network. Prediction MSE for the best Neural Network model was 2717 whereas Prediction MSE for Random forest model was 27431. Purely based on MSE, Neural network performed better and yielded better predictions of remaining flying time. Thus, with further research and development of other models such as functional regression, prediction of remaining flying time estimate can be made more accurate.

6. References

Frøslie, K. F., Røislien, J., Qvigstad, E., Godang, K., Bollerslev, J., Voldner, N., & Veierød, M. B. (2013). Shape information from glucose curves: functional data analysis compared with traditional summary measures. BMC medical research methodology, 13(1), 6.

Yogev Y, Ben-Haroush A, Chen R, Rosenn B, Hod M, Langer O: Diurnal glycemic profile in obese and normal weight nondiabetic pregnant women. Am J Obstet Gynecol. 2004, 191: 949-953. 10.1016/j.ajog.2004.06.059.

C. Kulkarni, E. Hogge, C. Quach and K. Goebel "HIRF Battery Data Set", NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA

Hogge, Edward F., Brian M. Bole, Sixto L. Vazquez, Jose R. Celaya, Thomas H. Strom, Boyd L. Hill, Kyle M. Smalling, and Cuong C. Quach. "Verification of a remaining flying time prediction system for small electric aircraft." (2015).

Accessed from https://techcrunch.com/2018/07/08/the-electric-aircraft-is-taking-off/ on April 22, 2019.

7. Appendix

4.5.1 Python Code

```
# Regression Template

# Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing the dataset
dataset14 = pd.read_csv('HIRF14.csv')
X_HIRF14 = dataset14.iloc[:, :-1].values
y_HIRF14 = dataset14.iloc[:, 17].values
y_HIRF14 = y_HIRF14[-1] - y_HIRF14
X_HIRF14 = np.cumsum(X_HIRF14,axis=0)

dataset15 = pd.read_csv('HIRF15.csv')
X_HIRF15 = dataset15.iloc[:, :-1].values
y_HIRF15 = dataset15.iloc[:, 17].values
y_HIRF15 = y_HIRF15[-1] - y_HIRF15
X_HIRF15 = np.cumsum(X_HIRF15,axis=0)

dataset16 = pd.read_csv('HIRF16.csv')
X_HIRF16 = dataset16.iloc[:, :-1].values
y_HIRF16 = dataset16.iloc[:, 17].values
y_HIRF16 = y_HIRF16[-1] - y_HIRF16
X_HIRF16 = np.cumsum(X_HIRF16,axis=0)

dataset17 = pd.read_csv('HIRF17.csv')
X_HIRF17 = dataset17.iloc[:, :-1].values
y_HIRF17 = dataset17.iloc[:, 17].values
y_HIRF17 = y_HIRF17[-1] - y_HIRF17
X_HIRF17 = np.cumsum(X_HIRF17,axis=0)

dataset18 = pd.read_csv('HIRF18.csv')
X_HIRF18 = dataset18.iloc[:, :-1].values
y_HIRF18 = dataset18.iloc[:, 17].values
y_HIRF18 = y_HIRF18[-1] - y_HIRF18
X_HIRF18 = np.cumsum(X_HIRF18,axis=0)

dataset19 = pd.read_csv('HIRF19.csv')
X_HIRF19 = dataset19.iloc[:, :-1].values
y_HIRF19 = dataset19.iloc[:, 17].values
y_HIRF19 = y_HIRF19[-1] - y_HIRF19
X_HIRF19 = np.cumsum(X_HIRF19,axis=0)
```

```
dataset20 = pd.read_csv('HIRF20.csv')
X_HIRF20 = dataset20.iloc[:, :-1].values
y_HIRF20 = dataset20.iloc[:, 17].values
y_HIRF20 = y_HIRF20[-1] - y_HIRF20
X_HIRF20 = np.cumsum(X_HIRF20,axis=0)

dataset21 = pd.read_csv('HIRF21.csv')
X_HIRF21 = dataset21.iloc[:, :-1].values
y_HIRF21 = dataset21.iloc[:, 17].values
y_HIRF21 = y_HIRF21[-1] - y_HIRF21
X_HIRF21 = np.cumsum(X_HIRF21,axis=0)


X = np.concatenate((X_HIRF14, X_HIRF15, X_HIRF16, X_HIRF17, X_HIRF18, X_HIRF19, X_HIRF20))
y = np.concatenate((y_HIRF14, y_HIRF15, y_HIRF16, y_HIRF17, y_HIRF18, y_HIRF19, y_HIRF20))

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
X_HIRF21 = sc.transform(X_HIRF21)

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 17)
X = pca.fit_transform(X)
X_HIRF21 = pca.transform(X_HIRF21)
explained_variance = pca.explained_variance_ratio_

# Fitting the Randdom Forest regression to the data set
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators =50, random_state = 0)
regressor.fit(X,y)

# Predicting a new result
y_pred = regressor.predict(X_HIRF21)

#Test accuracy
d = (y_pred - y_HIRF21)
d = np.square(d)
d = sum(d)
d/3896162

#Visualizing the linear regression results
```

```
plt.scatter(y_HIRF21,y_pred, color = 'red')
plt.title('y_test VS y_pred')
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()

# Elbow plot for PCA
"""cumper = np.cumsum(explained_variance,axis=0)/sum(explained_variance) *100
number = np.arange(18)
number = np.delete(number, 0)
plt.scatter(number,explained_variance, color = 'red')
plt.title('PCA Elbow plot')
plt.xlabel('Number of PCAs')
plt.ylabel('Variance explanined')
plt.show()"""
```

4.5.2 R Markdown Code

```
---
title: "OR791_Project1_ver2"
author: "Vikram Patil"
date: "April 20, 2019"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
This is a R markdown code for PCA and Neural Network modelling for Prediction of Remaining Flight time for BAttery operated Aircraft.
Loading Data
```{r}
library(rmatio)
library(dplyr)
fulldata<-data.frame()
for ( i in 14:16) {
orgdat<-read.mat(paste("./Data/HIRF",i,".mat",sep = ""))$data
orgdat<- orgdat[!(names(orgdat) %in% c("BHM"))]
dat  <-  as.data.frame(matrix(unlist(orgdat), nrow=length(unlist(orgdat[1]))))
names(dat)<-names(orgdat)
cumdat<-cumsum(dat)
cumdat$t<-dat$t
dat<-cumdat
rm(cumdat)
red_n<-200
dat<-aggregate(dat,list(rep(1:(nrow(dat)%/%red_n+1),each=red_n,len=nrow(dat))),mean)[-
1]
```
```

```
dat$t<-dat$t[nrow(dat)]-dat$t
fulldata<-rbind(fulldata,dat)
}
#rm(dat,orgdat)

fulldata<-na.omit(fulldata)
```
Performing PCA
```{r}
#library(GGally)
#ggpairs(data = HIRF5, columns = 1:3)
#cormat<-cor(HIRF5)
#install.packages('corrplot')
cormat<-cor(fulldata[-1])
library(corrplot)
corrplot(cormat,type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)

pca.out<-prcomp(fulldata[-1],retx = T, center = T, scale. = T)
summary(pca.out)
nPCs<-5
PC_loadings<-pca.out$rotation[,1:nPCs]

### Useful PCA data
pca_useful<-as.data.frame(pca.out$x)
pca_useful<-pca_useful[,1:nPCs]
pca_useful$t<-fulldata$t

maxs <- apply(pca_useful, 2, max)
mins <- apply(pca_useful, 2, min)
scaled_pca <- as.data.frame(scale(pca_useful, center = mins, scale = maxs - mins))
```
Neural NEtwork
```{r}
library(neuralnet)
nnet.out<-neuralnet(t ~ PC1+ PC2+ PC3+PC4+PC5, data = scaled_pca[1:5000,],hidden =
c(3),
                    linear.output = T, stepmax = 1e+06)
plot(nnet.out)
```
Creating Prediction Data
```{r}
newdata<-data.frame()
for ( i in 22) {
orgdat<-read.mat(paste("./Data/HIRF",i,".mat",sep = ""))$data
orgdat<- orgdat[!(names(orgdat) %in% c("BHM"))]
```

```
dat  <-  as.data.frame(matrix(unlist(orgdat), nrow=length(unlist(orgdat[1]))))
names(dat)<-names(orgdat)

cumdat<-cumsum(dat)
cumdat$t<-dat$t

dat<-cumdat
rm(cumdat)

red_n<-200
dat<-aggregate(dat,list(rep(1:(nrow(dat)%/%red_n+1),each=red_n,len=nrow(dat))),mean)[-
1]
dat$t<-dat$t[nrow(dat)]-dat$t

newdata<-rbind(newdata,dat)
}
#rm(dat,orgdat)

newdata<-na.omit(newdata)
pred_real_time<-newdata$t
new_scaled_data<-scale(newdata[-1],center = pca.out$center, scale = pca.out$scale)
test_pc_data<-new_scaled_data%*%PC_loadings
test_pc_data<-as.data.frame(test_pc_data)
test_pc_data$t<-newdata$t

test_pc_scaled<-scale(test_pc_data, center = mins, scale = maxs - mins)
test_pc_scaled<-as.data.frame(test_pc_scaled)
```
##Predicting
```{r}

pred_scaled_time<-compute(nnet.out,test_pc_scaled[,1:nPCs])
plot(pred_scaled_time$net.result,test_pc_scaled$t)+abline(0,1)

pred_time<-pred_scaled_time$net.result*(maxs[nPCs+1]-mins[nPCs+1])+mins[nPCs+1]
plot(pred_real_time,pred_time,xlab = "Actual Time", ylab = "PRedicted Time", main =
"Prediction with Neural Network")+abline(0,1)

plot(pred_time, type="l", col="black", xlim = c(0,2500))

lines(pred_real_time, type = "l", col="green",xlim = c(0,2500))

pred_error<-pred_real_time-pred_time
mse_pred<-(t(pred_error)%*%pred_error)/nrow(pred_error)
mse_pred
```