ABHIJIT NIKHADE
GE Digital, SSO - 212599422

imagination at work

Case Study : Virtual Wind Shear Sensor

GE Analytics Engineer Program – Falls 2017

1. Introduction:

The world's electricity demand will grow by 50 percent over the next 20 years, and people want to get there by using reliable, affordable, and sustainable power. Renewable energy already started making its impact in global space considering the day-to-day increase in carbon foot print and dependency on non-renewable source of energy. Wind power is an immense source of renewable energy.

U.S. Geological Survey carried out a project to account the wind turbine installations across the country. The purpose of this project is to provide a publically available, spatially referenced, national dataset of onshore wind turbine locations and their corresponding facility information and turbine technical specifications. The project compiled wind turbine information from the Federal Aviation Administration (FAA) Digital Obstacle File, as well as other manually digitized turbine locations. The below map (Fig 1.1) shows different capacity wind turbines installed at various geographic locations across United States.

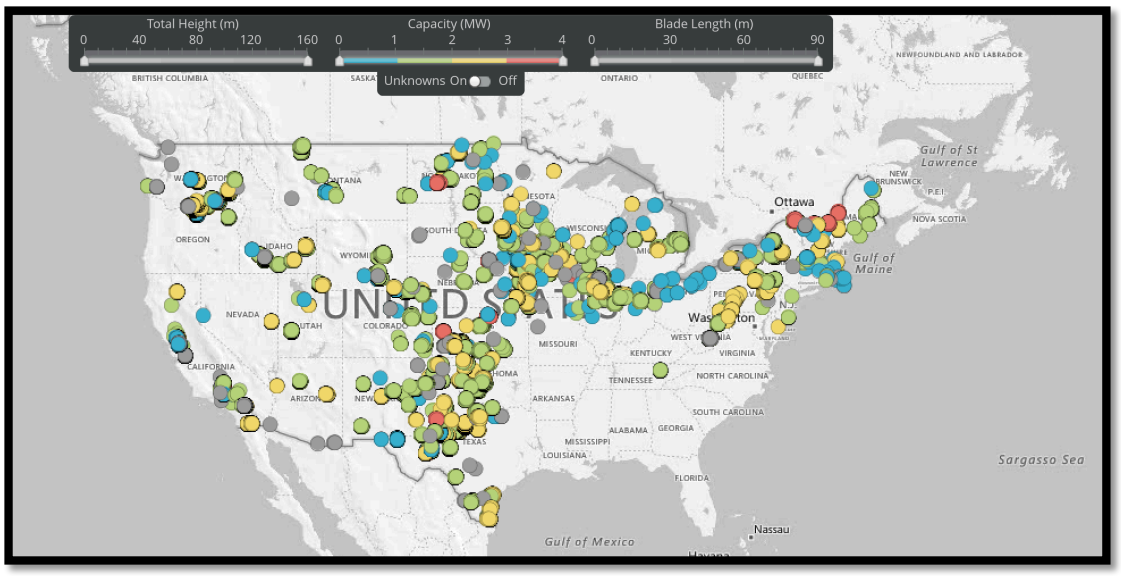

Fig 1.1 Wind turbines install base in United States

Data Point Verification: Verification of the turbine position was done by visual interpretation using high-resolution aerial imagery. Wind farm facility location data from various sources were collected and used to search for and digitize turbines. The database, current through July 2013, has ~47,000 wind turbine records

The wind industry relies on wind shear which is an important factor in the generation of wind energy. It is the main reason why wind turbines are taller than the tallest building. The tall designs are able to capture the stronger wind aloft during extreme wind shear conditions. Wind shear is the difference in wind speed by height. The higher the wind shear, the higher the wind speeds. The wind profile power law is a relationship between the wind speeds at one height, and those at another. The power law is often used in wind power assessments where wind speeds at the height of a turbine (>~ 50 meters) must be estimated from near surface wind observations (~10 meters), or where wind speed data at various heights must be adjusted to a standard height prior to use.

The wind profile power law relationship is:

$$\frac{V(h)}{V(HH)} = \left(\frac{h}{HH}\right)^{\alpha}$$

where $V(h)$ is the wind speed (in meters per second) at height h (in meters), and $V(HH)$ is the known wind speed at a reference height HH. The exponent (α) is an empirically derived coefficient (power i.e. wind shear) that varies dependent upon the stability of the atmosphere. For neutral stability conditions $\boldsymbol{\alpha}$, is approximately **1/7**, or **0.143**.

In order to estimate the wind speed at certain height h, the relationship would be rearranged to:

$$V(h) = V(HH) * \left(\frac{h}{HH}\right)^{\alpha}$$

2. GE Business Case:

To obtain wind shear, 5 wind speed (LIDAR) sensors need to be installed at different altitudes on a costly met mast. As of today, GE doesn't consider wind shear (α) to optimize the power output from wind turbine. Instead, GE considers load sensors installed at different heights on hub to sense loading parameters (nodding, pitching, yawing and other loading parameters) in order to estimate wind shear (α).

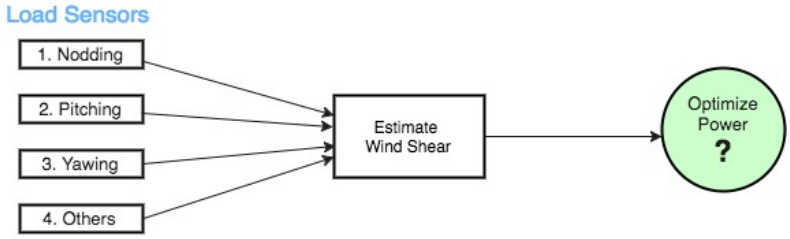

Fig 2.1 Business Case Approach

The analysis of field data has shown that wind speed follows 3 main types of profiles (Fig 2.2)

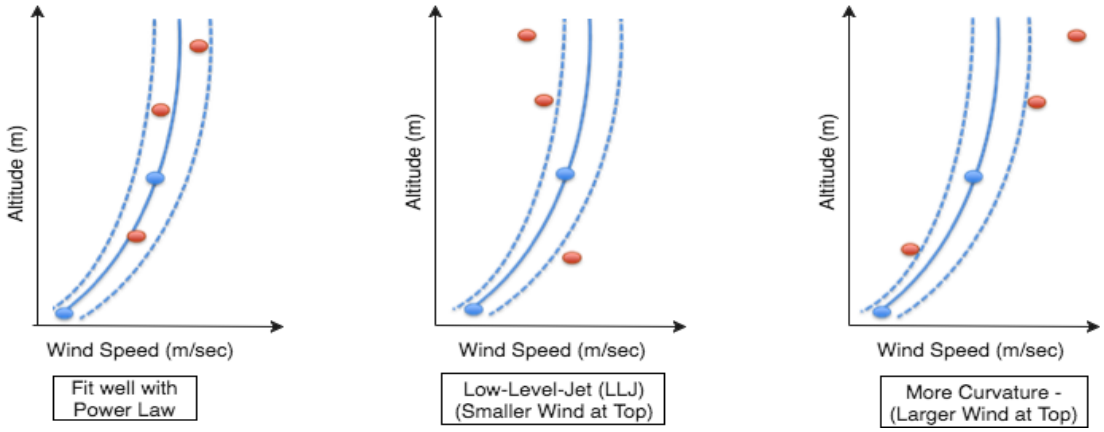

Fig 2.2 Wind Speed profiles

To obtain the ground truth of the wind speed profile, GE Wind built an expensive met mast and installed 5 wind speed sensors at altitudes of 38m, 58m, 78.7m, 103m and 122m as shown in the figure. Field engineers have found that about 80% of the wind speed profile follows the Power Law. Many speed sensors are equipped at the height (78.7m) and lower tip height (38 m). Therefore, it is customary in the industry to use these two speeds to calculate α.

$$V_{38m} = V_{78.7m}\left(\frac{38}{78.7}\right)^{\alpha}$$

$$\alpha = \frac{ln\left(\frac{V_{38m}}{V_{78.7m}}\right)}{\ln\left(\frac{38}{78.7}\right)}$$

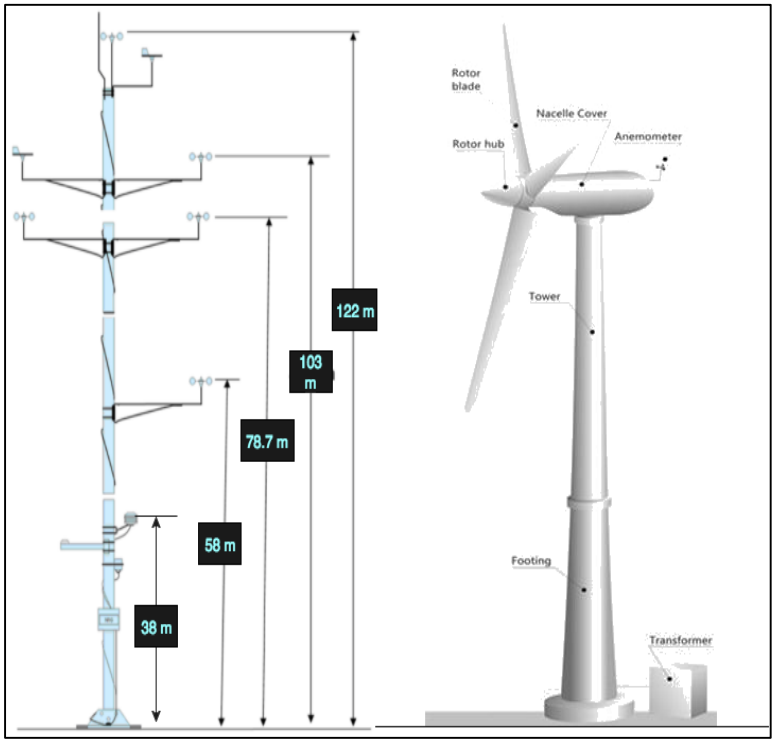

Fig 2.3 Met mast for 5 wind sensors

3. Data:

1. Summary:

- The data has been collected from 1-turbine
- We have the data of 12 days with every 2 minutes interval from June 17th, 2015 to June 28th, 2015.

2. Limitations:

- The study is based on data collected from just 1 turbine. The technical specifications of turbines located at same or different geographical location varies a lot. The data collected from different turbines at various location would have been much better for study.
- The study is based on the data collected for only 12 calendar days out of 365 days. Here we are talking about 3.29% of total data available for a year which is much lesser.
- The location of the turbine from where the data has been collected is missing. Average wind speed varies with geographic location, wind turbines generate more electricity as wind speed increases, and less as wind speed decreases. Wind consistency also varies with geographic location, the longer the wind blows at a given speed, the more a wind turbine in that area will generate.
- During the winter, in some locations, wind speed increases which affects wind power production. As you can see, the winter months are when wind speeds are the highest and

the hottest months which are the summer months are those the lowest wind speeds. Especially July and August.

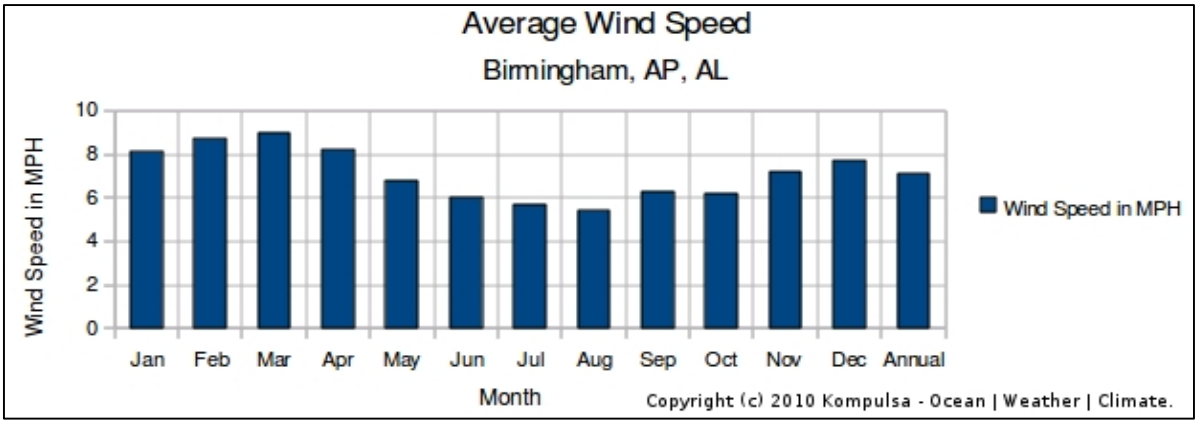

Fig 3.1 Average wind speed (12 months)

3. Metadata:

	Column Name	Description	Unit
Wind Speed Sensor	m38	Wind Speed at height of 38 m	m/sec
	m58	Wind Speed at height of 58 m	m/sec
	m78	Wind Speed at height of 78 m	m/sec
	m103	Wind Speed at height of 103 m	m/sec
	m122	Wind Speed at height of 122 m	m/sec
X variable (Input)	RPM_0P	0P component of RPM	1/min
	nodd_0P	0P component of nodding moment	Nm
	nodd_3C	3P (cosine) component of nodding moment	Nm
	nodd_3S	3P (sine) component of nodding moment	Nm
	pitch_d_0P	0P component of d-component of pitch angle	
	pitch_q_0P	0P component of q-component of pitch angle	
	pitch_d_3C	3P (cosine) component of d-component of pitch angle	
	pitch_d_3S	3P (sine) component of d-component of pitch angle	
	yaw_0P	0P component of yawing moment	Nm
	yaw_3C	3P (cosine) component of yawing moment	Nm
	yaw_3S	3P (sine) component of yawing moment	Nm
	P_el	0P component of electrical power	kW
	V_estim	0P component of MBC estimated wind speed	m/sec
	pitch_col_0P	0P component of collective pitch angle	
Y variable	ShearTypeClass	0: Power Law, 1: LLJ, 2:Flat, 3: Others	

4. Methods & Analysis

We have our data ready to consume in order to get the statistical analysis and insights. I will use R-language for working with the given data. R language gives us a flexibility to solve the complex problems. After spending good amount of time I decided to use some packages (libraries) for my detailed analysis. I used R Studio (IDE for R language development projects) for coding. Hereby I am explaining the step by step approach for our analysis.

1. **Setting up the working environment in R.**

```
# Installing the required libraries
install.packages('Hmisc')
install.packages('scales')
install.packages('ggplot2')
install.packages('GGally')
install.packages('corpcor')
install.packages('psych')
install.packages('glmnet')
install.packages('car')
install.packages('Metrics')
install.packages('randomForest')
install.packages('caret')
install.packages('mlbench')
install.packages('e1071')

# Invoking the libraries
library(Hmisc)
library(glmnet)
library(psych)
library(scales)
library(ggplot2)
library(GGally)
library(corpcor)
library(plyr)
library(MASS)
library(car)
library(randomForest)
library(caret)
library(mlbench)
library(e1071)

# Setting up the working directory
setwd("/Users/[redacted]/Documents/Analytics_Engineer_Program")
getwd()
```

2. **Reading the given data for the case study and getting the summary of the data.**

```
# Reading the data and getting the summary of the data
input_data <-
read.csv(file="/Users/[redacted]/Documents/Analytics_Engineer_Program/WindFarm_2min_Analy
ticsEngineer_wWS.csv", header = TRUE, na.strings = "?")
summary(input_data)
str(input_data)
```

We get the summary of all the parameters (columns) present in the data set. Also function str(<data>) would give us the data type of given data. The detailed summary of the data is shown in the figure 4.1. The summary gives the Minimum value, Maximum value, Median, Mean, 1st and 3rd quartile mean values for each parameter given in the data. We don't have any missing values in our data. Otherwise we may have to fill the missing values with the mean value of respective parameters from the data.

Fig 4.2 shows the data summary and the data type for all the parameters present in the given data. All the parameters except datetime and ShearTypeClass have numeric values.

```
> summary(input_data)
          datetime          m38               m58               m78              m103              m122        
 6/17/15 10:30:    1   Min.   : 0.4803   Min.   : 1.402   Min.   : 2.653   Min.   : 2.740   Min.   : 2.888  
 6/17/15 10:32:    1   1st Qu.: 6.1797   1st Qu.: 7.298   1st Qu.: 8.059   1st Qu.: 8.629   1st Qu.: 8.901  
 6/17/15 10:34:    1   Median : 7.6165   Median : 8.907   Median : 9.954   Median :10.866   Median :11.379  
 6/17/15 10:36:    1   Mean   : 7.7035   Mean   : 8.908   Mean   : 9.851   Mean   :10.730   Mean   :11.261  
 6/17/15 10:38:    1   3rd Qu.: 9.1330   3rd Qu.:10.493   3rd Qu.:11.649   3rd Qu.:12.725   3rd Qu.:13.484  
 6/17/15 10:40:    1   Max.   :15.9780   Max.   :17.478   Max.   :18.867   Max.   :18.845   Max.   :18.637  
 (Other)      :8040                                                                                        
     RPM_0P           nodd_0P           nodd_3C           nodd_3S          pitch_d_0P           pitch_q_0P        
 Min.   : 7.961   Min.   :-1593.1   Min.   :-267.87   Min.   :-183.16   Min.   :-1.5862000   Min.   :-1.4400000  
 1st Qu.:13.135   1st Qu.: -715.3   1st Qu.:  -6.37   1st Qu.:  37.97   1st Qu.:-0.0015374   1st Qu.:-0.0002527  
 Median :14.312   Median : -691.7   Median :  27.96   Median :  78.90   Median : 0.0000882   Median : 0.0030696  
 Mean   :13.447   Mean   : -612.9   Mean   :  48.31   Mean   :  93.96   Mean   : 0.4143185   Mean   : 0.1122205  
 3rd Qu.:14.325   3rd Qu.: -445.5   3rd Qu.:  80.82   3rd Qu.: 131.31   3rd Qu.: 0.9278875   3rd Qu.: 0.2510875  
 Max.   :14.618   Max.   :  206.0   Max.   : 652.53   Max.   : 561.63   Max.   : 2.4330000   Max.   : 1.4930000  

   pitch_d_3C           pitch_d_3S            yaw_0P            yaw_3C            yaw_3S       
 Min.   :-0.5965300   Min.   :-0.5467800   Min.   :-387.52   Min.   :-201.440   Min.   :-274.14  
 1st Qu.:-0.1335550   1st Qu.:-0.0131448   1st Qu.:  13.12   1st Qu.:   7.167   1st Qu.: -67.37  
 Median :-0.0299060   Median :-0.0000204   Median :  27.15   Median :  42.675   Median : -38.36  
 Mean   :-0.0764763   Mean   :-0.0006305   Mean   :  58.31   Mean   :  57.116   Mean   : -43.51  
 3rd Qu.:-0.0000303   3rd Qu.: 0.0104415   3rd Qu.: 111.30   3rd Qu.:  91.483   3rd Qu.: -14.47  
 Max.   : 0.1876000   Max.   : 0.3537400   Max.   : 489.13   Max.   : 487.160   Max.   : 208.73  

      P_el            V_estim        pitch_col_0P      ShearTypeClass
 Min.   :   3.384   Min.   : 2.756   Min.   : 0.0402   Min.   :0.0000  
 1st Qu.:1015.375   1st Qu.: 7.935   1st Qu.: 0.2385   1st Qu.:0.0000  
 Median :1770.100   Median : 9.793   Median : 1.7398   Median :0.0000  
 Mean   :1592.617   Mean   : 9.584   Mean   : 2.5916   Mean   :0.2923  
 3rd Qu.:2283.875   3rd Qu.:11.290   3rd Qu.: 4.0163   3rd Qu.:0.0000  
 Max.   :2359.400   Max.   :16.457   Max.   :15.1020   Max.   :3.0000  
```

Fig 4.1 Summary of Case Study Data

```
> str(input_data)
'data.frame':   8046 obs. of  21 variables:
 $ datetime      : Factor w/ 8046 levels "6/17/15 10:30",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ m38           : num  4.47 4.46 4.51 4.44 4.12 ...
 $ m58           : num  4.61 4.68 4.87 4.66 4.27 ...
 $ m78           : num  4.87 4.65 4.73 4.82 4.39 ...
 $ m103          : num  5.09 5.01 5.04 5.03 4.72 ...
 $ m122          : num  5.48 5.38 5.24 5.09 4.91 ...
 $ RPM_0P        : num  8.13 8.01 8.01 8.01 8.01 ...
 $ nodd_0P       : num  -86.91 -64.31 -33.83 8.63 73.74 ...
 $ nodd_3C       : num  78.8 64.5 63.6 52.4 26.2 ...
 $ nodd_3S       : num  -33.65 -4.81 14.09 32.03 29.24 ...
 $ pitch_d_0P    : num  -6.38e-05 -1.10e-04 -1.37e-04 -2.03e-03 -2.23e-03 ...
 $ pitch_q_0P    : num  -0.000327 -0.00048 -0.000342 -0.000502 -0.000666 ...
 $ pitch_d_3C    : num  -2.94e-04 -2.33e-05 -3.74e-05 -1.64e-04 -3.12e-05 ...
 $ pitch_d_3S    : num  -9.51e-06 5.83e-05 1.47e-05 -3.59e-04 -4.18e-04 ...
 $ yaw_0P        : num  -27.9 35.1 43.1 11.5 67.4 ...
 $ yaw_3C        : num  30.37 84.53 55.07 4.98 -4.07 ...
 $ yaw_3S        : num  -19.66 47.69 22.49 -14.64 -6.43 ...
 $ P_el          : num  176 138 156 139 128 ...
 $ V_estim       : num  4.68 4.33 4.41 4.36 4.23 ...
 $ pitch_col_0P  : num  0.205 0.305 0.326 0.251 0.259 ...
 $ ShearTypeClass: int  0 0 0 0 0 0 0 0 0 0 ...
```

Fig 4.2 Data Summary (Data Types)

3. Descriptive Analytics of the Predictors (X variables)

```
# Descriptive Analytics of the predictors (X)
count(input_data, 'ShearTypeClass')
predict1 <- input_data[ ,c(7:21)]

predict1[which(predict1$ShearTypeClass == 0) , 15 ] <- 'Shear0'
predict1[which(predict1$ShearTypeClass == 1) , 15 ] <- 'Shear1'
predict1[which(predict1$ShearTypeClass == 2) , 15 ] <- 'Shear2'
predict1[which(predict1$ShearTypeClass == 3) , 15 ] <- 'Shear3'
summary(predict1)
str(predict1)
```

As we observed in fig 4.2, ShearTypeClass has been detected as continues integer . We would like to consider ShearTypeClass as a factor. The summary of the ShearTypeClass data points is shown in the fig. 4.3. Observations for Shear3 type are quite sparse.

ShearTypeClass	freq
Shear0	6346
Shear1	1177
Shear2	394
Shear3	129

Fig 4.3 Shear Type Class

```
#  Analyze variables for multicollinearity

pairs.panels(predict1[ ,c(1:14)], col="red", gap=0)
```

Use of the function pairs.panels(<data>) gives the variable matrix for the 14 predictors. With the help of this matrix we can analyse variables for Normality of Distribution, Multiple Collinearity, Extreme Values and Homoscedasticity (even distribution of residuals). Here we'll only make a brief visual inspection of vars. The matrix shows in fig 4.4.

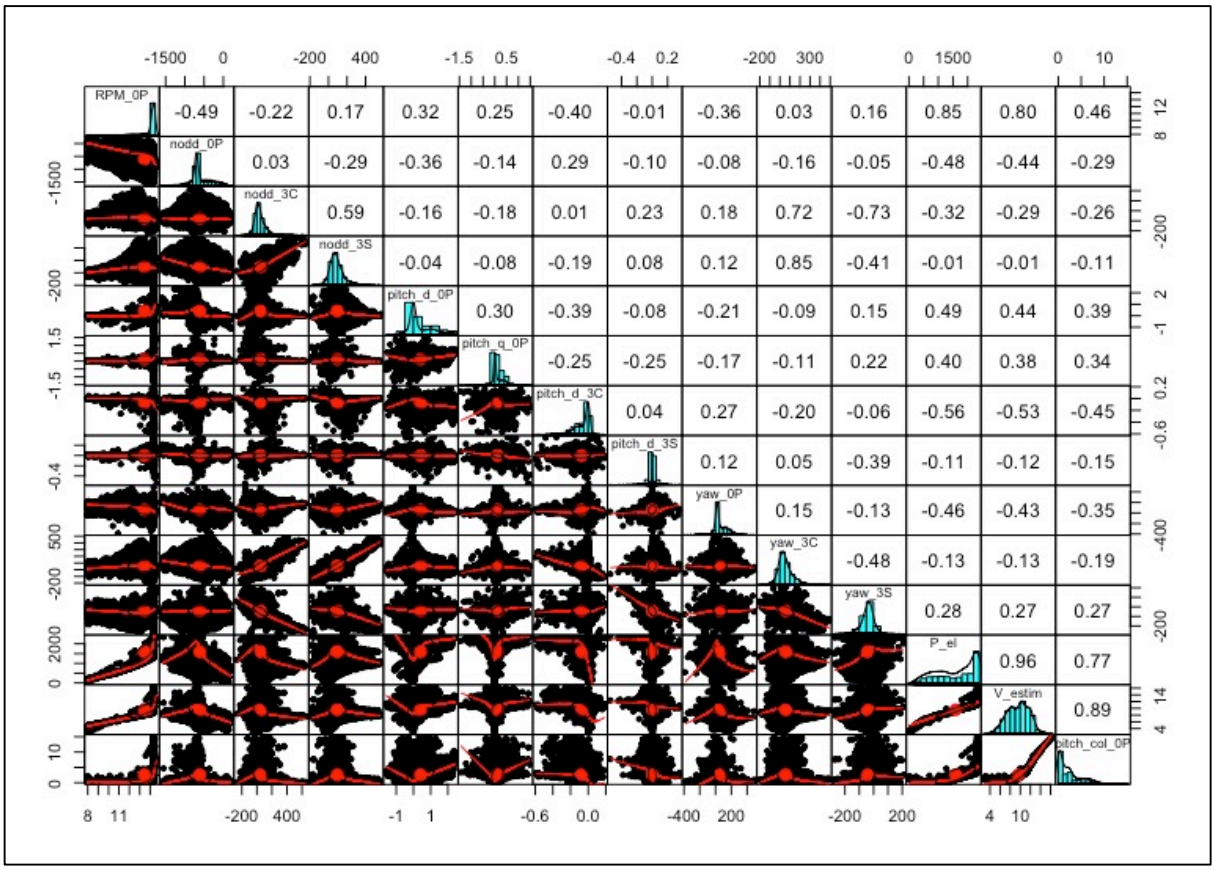

Fig 4.4 Multicollinearity Analysis of Variables

Looking at the matrix we can say that variables RPM_0P, pitch_d_3C, P_el, pitch_col_0P are skewed. We should be dealing with these variables. In order to have a good regression model, all variables should have the normal distribution. The variables should be independent.

The pairs (RPM_0P, P_el), (RPM_0P, V_estim), (P_el, V_estim), (P_el, pitch_col_0P) and (V_estim, pitch_col_0P) are highly correlated. This multicollinearity analysis of variance should be soley for the purpose of visual representation.

Now, we will see the boxplot of some variables for the clear understanding of the different shear classes. The code to get pox plot is as follows.

```
# box plots of some variables showing clear distinction for different shear classes
predict1$ShearTypeClass <- as.factor((predict1$ShearTypeClass))
ggplot(data = predict1, aes(x=ShearTypeClass, y=pitch_d_0P)) + geom_boxplot(aes(fill=ShearTypeClass))
ggplot(data = predict1, aes(x=ShearTypeClass, y=nodd_0P)) + geom_boxplot(aes(fill=ShearTypeClass))
ggplot(data = predict1, aes(x=ShearTypeClass, y=pitch_col_0P)) + geom_boxplot(aes(fill=ShearTypeClass))
```

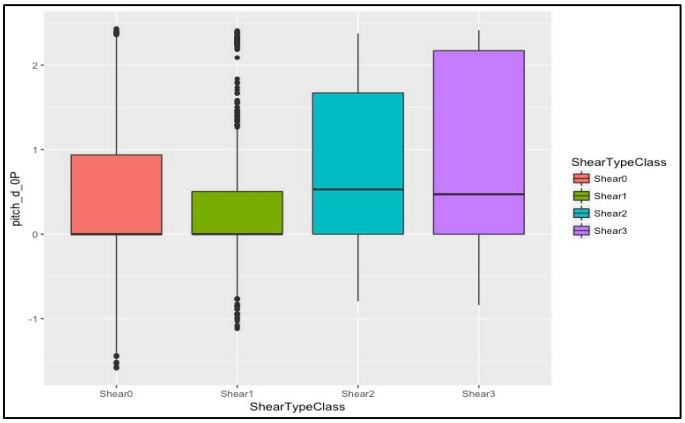

Fig 4.5 ShearTypeClass vs pitch_d_0P

We see the box plot of ShearTypeClass vs pitch_d_0P in fig 4.5. We see there are good number of outliers or extreme values for shear class 1. We see some extreme values for shear class 0 as well. Median for shear class 0 and 1 is almost the same. Also, we see there is a good data spread for shear class 2 and 3 as compared to shear class 0 and 1.

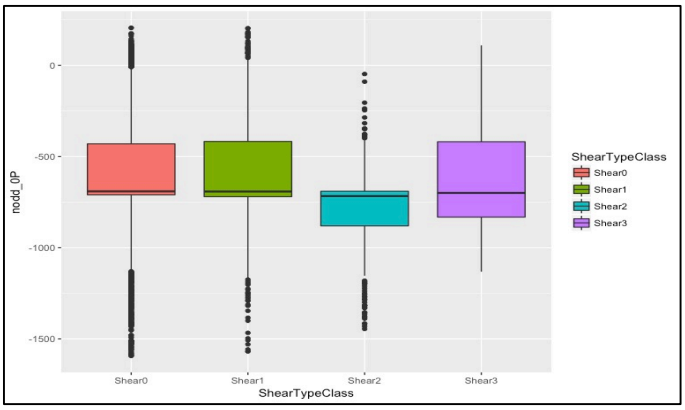

Fig 4.6 ShearTypeClass vs nodd_0P

Fig 4.6 shows the box plot of ShearTypeClass vs nodd_0P. We see the outliers for shear class 0, 1 and 2. Box plot shows the clear distinction of data points for shear class 1 vs shear class 2. Median for all the shear classes is about the same. Shear class 3 has good spread of data among all. We really can't tell much about the data just by looking at the box plots, but it gives us an overview of the data spread across the different shear classes.

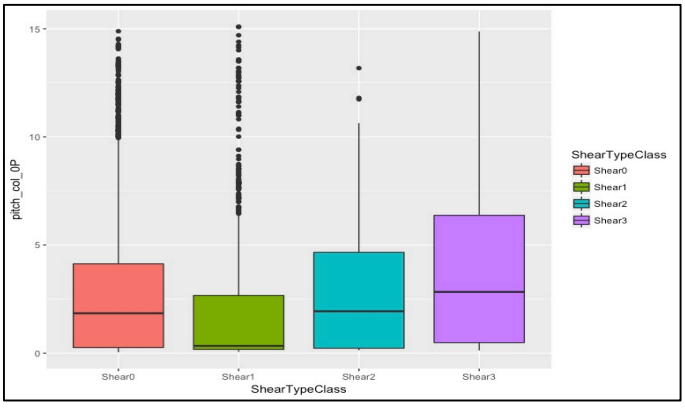

Fig 4.7 ShearTypeClass vs pitch_col_0P

Fig 4.7 shows the box plot of ShearTypeClass vs pitch_col_0P. This box plot gives us good understanding of the distinct median for all of the 4 ShearTypeClass. Shear type 0 and 1 have extreme values present in the dataset.

4. Classifying the Wind Speed Profile (PART A)

After spending some time on data analysis of the provided data, I decided to compare the result of different classifiers. I used Linear Discriminant Analysis (LDA) and Random Forest method for shear type classification. We will walk through the steps involved in classification. Going with the general thumb rule, let's take 60% of the available data as a training data set. We will assign remaining 20% data for testing the model and remaining 20% data for validation.

```
############ Part A: Classifying wind speed profile ############
# Training Data: 60%, Testing Data: 20%, Validation Data: 20%
set.seed(123)
indices = sample(1:nrow(predict1), size =0.6*nrow(predict1))
train_data <- predict1[indices, ]
predict2 <- predict1[-indices, ]
indices2 <- sample(1:nrow(predict2), size=0.5*nrow(predict2))
test_data <- predict2[indices2 , ]
val_data <- predict2[-indices2, ]
str(train_data)
```

A. Method 1: Linear Discriminant Analysis

Originally developed by R.A. Fisher in 1936 to classify subjects into one of the two clearly defined groups. It was later expanded to classify subjects into more than two groups. Helps to find linear combination of original variables that provide the best possible separation between the groups. The basic purpose is to estimate relationship between a single categorical development variable and a set of quantitative independent variables.

Linear Discrimination Analysis (LDA) is a well-established machine learning technique for predicting categories. Its main advantages, compared to other classification algorithms such as neural networks and random forests, are that the model is interpretable and that prediction is easy.

```
#Method-1 : Linear Discriminant Analysis#
lda_fit = lda(ShearTypeClass ~ . , data= train_data)
lda_fit
lda_pred=predict(lda_fit, test_data)
lda_pred
pred_class=lda_pred$class
table(pred_class, test_data$ShearTypeClass)
```

After going through the code above we get prior probabilities of all the groups, Proportion of trace and predictor class table as follows:

Shear 0 : 79%
Shear 1 : 14%
Shear 2 : 5%
Shear 3 : 2%

```
Proportion of trace:
    LD1     LD2     LD3
0.9179 0.0656 0.0165
```

pred_class	Shear0	Shear1	Shear2	Shear3
Shear0	1232	89	56	19
Shear1	37	153	0	4
Shear2	12	0	6	0
Shear3	1	0	0	0

The "proportion of trace" is the percentage separation achieved by each discriminant function. For shear class classification data, we get values 92%, 6.6% and 1.7%. Therefore, the first discriminant function (LD1) does achieve a very good separation between the four groups (four cultivars). Also, the second and third discriminant function does improve the separation of the groups by quite a bit (6.6% and 1.7%), so is it worth using those discriminant functions as well. Therefore, to achieve a good separation of the groups (cultivars), it is necessary to use all the three discriminant functions.

After this step, we will see the overall accuracy of the model as well as accuracy of model for each of the shear class. This time, we will test our model against testing data which is new for our trained model.

```
##Overall Accuracy of model = 85 %
acc1 <- mean(pred_class == test_data$ShearTypeClass)
print(paste('Overall Accuracy of the model in % = ' , acc1*100 ))

#Class_wise accuracy
acc2 <- (table(pred_class, test_data$ShearTypeClass) [1,1])/(nrow(test_data[
which(test_data$ShearTypeClass == 'Shear0'), ] ))
acc3 <- (table(pred_class, test_data$ShearTypeClass) [2,2])/(nrow(test_data[
which(test_data$ShearTypeClass == 'Shear1'), ] ))
acc4 <- (table(pred_class, test_data$ShearTypeClass) [3,3])/(nrow(test_data[
which(test_data$ShearTypeClass == 'Shear2'), ] ))
acc5 <- (table(pred_class, test_data$ShearTypeClass) [4,4])/(nrow(test_data[
which(test_data$ShearTypeClass == 'Shear3'), ] ))

print(paste('Predicted Accuracy Of class- Shear0 in % =', acc2*100))
print(paste('Predicted Accuracy Of class- Shear1 in % =', acc3*100))
print(paste('Predicted Accuracy Of class- Shear2 in % =', acc4*100))
print(paste('Predicted Accuracy Of class- Shear3 in % =', acc5*100))
```

We get the following result after testing our model against test data

Overall Accuracy of the LDA model : 87%
Predicted Accuracy of Shear 0 type : 96%
Predicted Accuracy of Shear 1 type : 63%
Predicted Accuracy of Shear 2 type : 10%
Predicted Accuracy of Shear 3 type : 0%

For Shear type 0, model is predicting the values really well, while for shear class 2 and 3, model couldn't able to predict.

B. Method 2: Random Forest for Classification

Random Forest is an ensemble learning method for classification and regression operate by constructing a multitude of decision trees. It is reasonably fast and very easy to use. It handles sparse and missing data quite well. Random Forest method of classification overcomes problem with over fitting. It has an effective method for estimating missing data and maintains accuracy when a large portion of data is missing. Let's work on classification based on Random Forest Classification method.

```
## Method-2 :Random Forest for ShearClassType Classification ##
#calulating mtry
mtry1 = floor(sqrt(ncol(train_data)))

#Run the random forest model with ntree= 250 corrsponding to mtry = 3
set.seed(123)
model_rf <- randomForest(ShearTypeClass ~ . , data= train_data , mtry = 3 , ntree = 250)
model_rf  #OOB estimate of  error rate: 9.45%
```

Here we are calculating the mtry (for selecting best possible variables / predictors) for random forest by floor method. We get the value of mtry as 3. After creating the model based on mtry value of 3 and ntree 250 we get the following result:

```
        OOB estimate of  error rate: 9.45%
Confusion matrix:
       Shear0 Shear1 Shear2 Shear3 class.error
Shear0   3754     49     10      1  0.01573152
Shear1    147    544      0      3  0.21613833
Shear2    178      1     65      0  0.73360656
Shear3     59      8      0      8  0.89333333
```

We get the OOB estimate of error rate as 9.45% without tuning the model. To improve the accuracy of the model, I would identify a better value of parameters mtry and ntree using grid search. The following coding steps shows the tuning of the model for improved values of mtry and ntree.

```
## Using validation set to identify better value of mtry using gridsearch
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
set.seed(123)
tunegrid <- expand.grid(.mtry=c(3:12))
rf_gridsearch <- train(ShearTypeClass ~ ., data=val_data, method="rf", metric="Accuracy",
tuneGrid=tunegrid, trControl=control)
print(rf_gridsearch)
plot(rf_gridsearch) # We got the best Accuracy at mtry = 4
```

We get the result for mtry as follows (Fig 4.8 & 4.9)

```
1610 samples
  14 predictor
   4 classes: 'Shear0', 'Shear1', 'Shear2', 'Shear3'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 1449, 1448, 1448, 1449, 1449, 1449, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   3    0.8778516  0.6131561
   4    0.8786850  0.6195349
   5    0.8776459  0.6185091
   6    0.8774440  0.6188122
   7    0.8780625  0.6226543
   8    0.8782632  0.6256174
   9    0.8766081  0.6201955
  10    0.8759960  0.6188764
  11    0.8766132  0.6218654
  12    0.8766145  0.6229223

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 4.
```

Fig 4.8 mtry using grid search

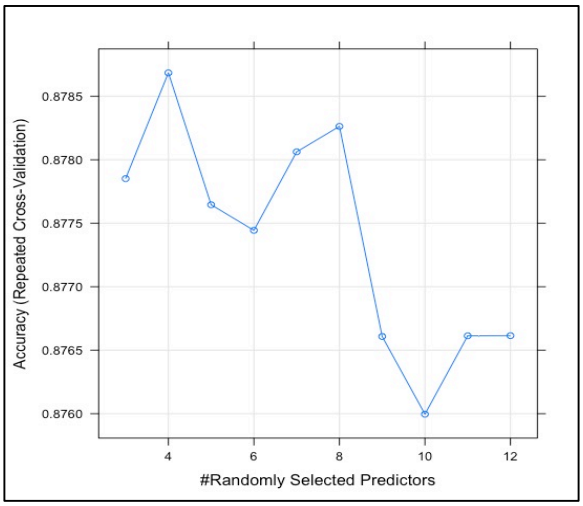

Fig. 4.9 mtry plot

From the above result, we can conclude that the best possible value of predictor is 4 in order to get the optimal model. Now as we know the best possible value of mtry, we will identify the best possible value of ntree at mtry=4. We can do this by selecting manual values of ntree in order to get optimal out of the selection.

```
# Manual Search using validation set to identify best value of "ntree" at mtry=4
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=4)
modellist <- list()
for (ntree in c(300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200)) {
  set.seed(123)
  fit <- train(ShearTypeClass ~ ., data=val_data, method="rf", metric="Accuracy",
tuneGrid=tunegrid, trControl=control, ntree=ntree)
  key <- toString(ntree)
  modellist[[key]] <- fit
}
# compare results
results <- resamples(modellist)
summary(results)
dotplot(results)
```

We get the following result out of this.

```
Models: 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200
Number of resamples: 30

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
300  0.8385093 0.8635649 0.8761598 0.8784792 0.8925347 0.9187500    0
400  0.8385093 0.8653356 0.8757764 0.8784818 0.8942450 0.9130435    0
500  0.8385093 0.8695652 0.8761598 0.8778555 0.8925347 0.9068323    0
600  0.8385093 0.8655395 0.8761598 0.8786876 0.8942450 0.9187500    0
700  0.8385093 0.8689538 0.8757764 0.8784779 0.8942450 0.9068323    0
800  0.8385093 0.8689538 0.8757764 0.8780651 0.8925347 0.9068323    0
900  0.8385093 0.8695652 0.8757764 0.8780677 0.8909915 0.9068323    0
1000 0.8385093 0.8695652 0.8757764 0.8784831 0.8909915 0.9068323    0
1100 0.8385093 0.8695652 0.8757764 0.8780664 0.8909915 0.9068323    0
1200 0.8385093 0.8695652 0.8757764 0.8778606 0.8909915 0.9068323    0
```

Fig 4.10 Best Accuracy on ntree (Manual ntree Input)

After looking at mean values (fig 4.10), ntree 600 has the highest percentage of accuracy (mean value) amongst all ntree we have provided manually. So we will be considering the ntree value 600 and mtry value 4 in order to build the optimal model for classification.

```
# Rerun the random forest model with ntree= 600 corrsponding to mtry = 4
set.seed(123)
model_rf_1 <- randomForest(ShearTypeClass ~ . , data= train_data , mtry =4 , ntree=600)
model_rf_1  #OOB estimate of  error rate: 9.24%
```

After running the optimal model with ntree=600 and mtry=4, we get the following result.

```
        OOB estimate of  error rate: 9.24%
Confusion matrix:
       Shear0 Shear1 Shear2 Shear3 class.error
Shear0   3747     54     12      1  0.01756686
Shear1    140    552      0      2  0.20461095
Shear2    173      1     70      0  0.71311475
Shear3     55      8      0     12  0.84000000
```

We get the OOB estimate of error rate as 9.24% after tuning the model which slightly better than the model we created without considering optimized mtry and ntree values.

Now, we will run our optimized model on test data to see the overall as well as class wise accuracy of the model.

```
## Running on test data with type = 'class' ##
random_pd <- predict(model_rf_1, test_data, type= 'class')
tab <- table(prediction= random_pd, actual= test_data$ShearTypeClass)
tab

#overall Accuracy
sum(diag(tab))/sum(tab) # Almost 92 %

#Class_wise accuracy
acc2 <- (tab [1,1])/(nrow(test_data[ which(test_data$ShearTypeClass == 'Shear0'), ] ))
acc3 <- (tab [2,2])/(nrow(test_data[ which(test_data$ShearTypeClass == 'Shear1'), ] ))
acc4 <- (tab [3,3])/(nrow(test_data[ which(test_data$ShearTypeClass == 'Shear2'), ] ))
acc5 <- (tab [4,4])/(nrow(test_data[ which(test_data$ShearTypeClass == 'Shear3'), ] ))
```

After running the optimal model on test data, we get the following result and confusion matrix.

prediction	Shear0	Shear1	Shear2	Shear3
Shear0	1256	31	45	16
Shear1	20	209	2	3
Shear2	6	0	15	0
Shear3	0	2	0	4

Overall Accuracy of the Random Forest Model : 92% (87% in LDA Model)
Predicted Accuracy of Shear 0 type : 98% (96% in LDA Model)
Predicted Accuracy of Shear 1 type : 86% (63% in LDA Model)
Predicted Accuracy of Shear 2 type : 24% (10% in LDA Model)
Predicted Accuracy of Shear 3 type : 17% (0% in LDA Model)

Conclusion (PART A):

Random Forest classifier is capable of taking care of collinearity. Also, Overall Accuracy and class wise accuracy is better for Random Forest Classifier as compared to LDA. Random Forest could able to classify Wind Shear class for shear type 0 (Zero Law) and 1 (Level Level Jet) very well.

5. Estimation of Wind Shear (α) from Load Sensor Data (PART B)

Here we will try to estimate the actual α from load sensor data (14 parameters given in the data). We are considering the data which follows the power law (ShearTypeClass 0). We will calculate the value of α with the help of data available from the wind sensor (anemometer) which is at height 38 meters and 78.7 meters (Industry Standards). The following code will calculate the value of α for every data point associated with ShearTypeClass 0. Before applying any method we will assign 60% of total data (Shear Class 0 data) to train our model data and remaining 40% data for validation and testing of model.

```
######### Part B: Estimation of  α #########
# Training Data: 60%, Testing Data: 20%, Validation Data: 20%
# Calculation of Alpha α=( ln(V_38m/V_78.7m ))/( ln(38/78.7) )

set.seed(222)
alpha_data <- input_data[which(input_data$ShearTypeClass == 0), ]
alpha_data$alpha <- (( log(alpha_data$m38/alpha_data$m78))/( log(38/78.7) ))
idx <- sample(1:nrow(alpha_data), size =0.6*nrow(alpha_data))
alpha_train <- alpha_data[idx, c(7:20, 22) ]
non_train <- alpha_data[-idx, c(7:20, 22)]
idx2 <- sample(1:nrow(non_train), size =0.5*nrow(non_train))
alpha_val<- non_train[idx2, ]
alpha_test <- non_train[-idx2,]

# Mean Absolute Error Calculation Function
Mean_Abs_Error <- function(predicted_val, actual_val)
{
  mean(abs(actual_val - predicted_val))
}
```

We have defined the function for Mean Absolute Error calculation. I would like to evaluate couple of methods for estimation of α. I will evaluate Multiple Regression model without principle component analysis and Lasso Regression.

A. Method 1: Multiple Regression without Principal Component Analysis(PCA)

Multiple regression is an extension of simple linear regression. It is used when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable). As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables.

The step by step code for training and testing the multiple regression model is given below.

```
##Method-1: Linear Model without P.C.A.##
# Multiple regression model utilises a simple formula
#  α = B0 + B1*RPM_0P + B2*nodd_0P + B3*nodd_3C + B4*nodd_3S + B5*pitch_d_0P
#         + B6*pitch_q_0P + B7*pitch_d_3C + B8*pitch_d_3S + B9*yaw_0P
#         + B10*yaw_3C + B11*yaw_3S + B12*P_el + B13*V_estim + B14*pitch_col_0P

op1 <- lm(alpha ~ . , data= alpha_train) #Adjusted R-squared:  0.7911
summary(op1)
pred_lm <- predict(op1, alpha_test) #Predicting α in the test data set
pred_lm
Mean_Abs_Error(pred_lm, alpha_test$alpha)  #MAE = 0.06632481
mse(pred_lm, alpha_test$alpha)  #MSE = 0.008839699
```

The output of the model shows the following analysis:

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.216e-01  2.824e-02  14.925  < 2e-16 ***
RPM_0P       -6.123e-02  5.978e-03 -10.242  < 2e-16 ***
nodd_0P      -4.903e-04  7.289e-06 -67.272  < 2e-16 ***
nodd_3C       1.911e-04  3.622e-05   5.276 1.39e-07 ***
nodd_3S       1.069e-04  3.450e-05   3.098 0.001960 **
pitch_d_0P    1.467e-01  2.446e-03  59.989  < 2e-16 ***
pitch_q_0P   -2.460e-02  6.889e-03  -3.571 0.000361 ***
pitch_d_3C   -3.111e-03  2.571e-02  -0.121 0.903701
pitch_d_3S   -1.621e-01  3.635e-02  -4.460 8.43e-06 ***
yaw_0P        3.572e-05  2.242e-05   1.593 0.111145
yaw_3C        9.924e-05  4.640e-05   2.139 0.032530 *
yaw_3S       -2.670e-04  4.646e-05  -5.747 9.77e-09 ***
P_el         -1.232e-04  1.587e-05  -7.764 1.05e-14 ***
V_estim       6.565e-02  1.229e-02   5.342 9.71e-08 ***
pitch_col_0P -3.714e-02  4.894e-03  -7.588 4.06e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09448 on 3792 degrees of freedom
Multiple R-squared:  0.7919,    Adjusted R-squared:  0.7911
F-statistic:  1031 on 14 and 3792 DF,  p-value: < 2.2e-16
```

Most of the variables apart from pitch_d_3C, yaw_0P and yaw_3C shows significance for the regression model. Percentage of variance explained by the model i.e. adjusted R-square is around 79% which is really good. **Mean Absolute Error (MAE) is 0.06632481**. Mean Absolute Squared Error **(MSE) is 0.008839699** which is very low. This implies that linear model itself quite good without tuning it.

Let’s check for the collinearity amongst the factors in training data.

```
# Checking for Co-relation among factors
pairs.panels(alpha_train, col="red", gap=0)
```

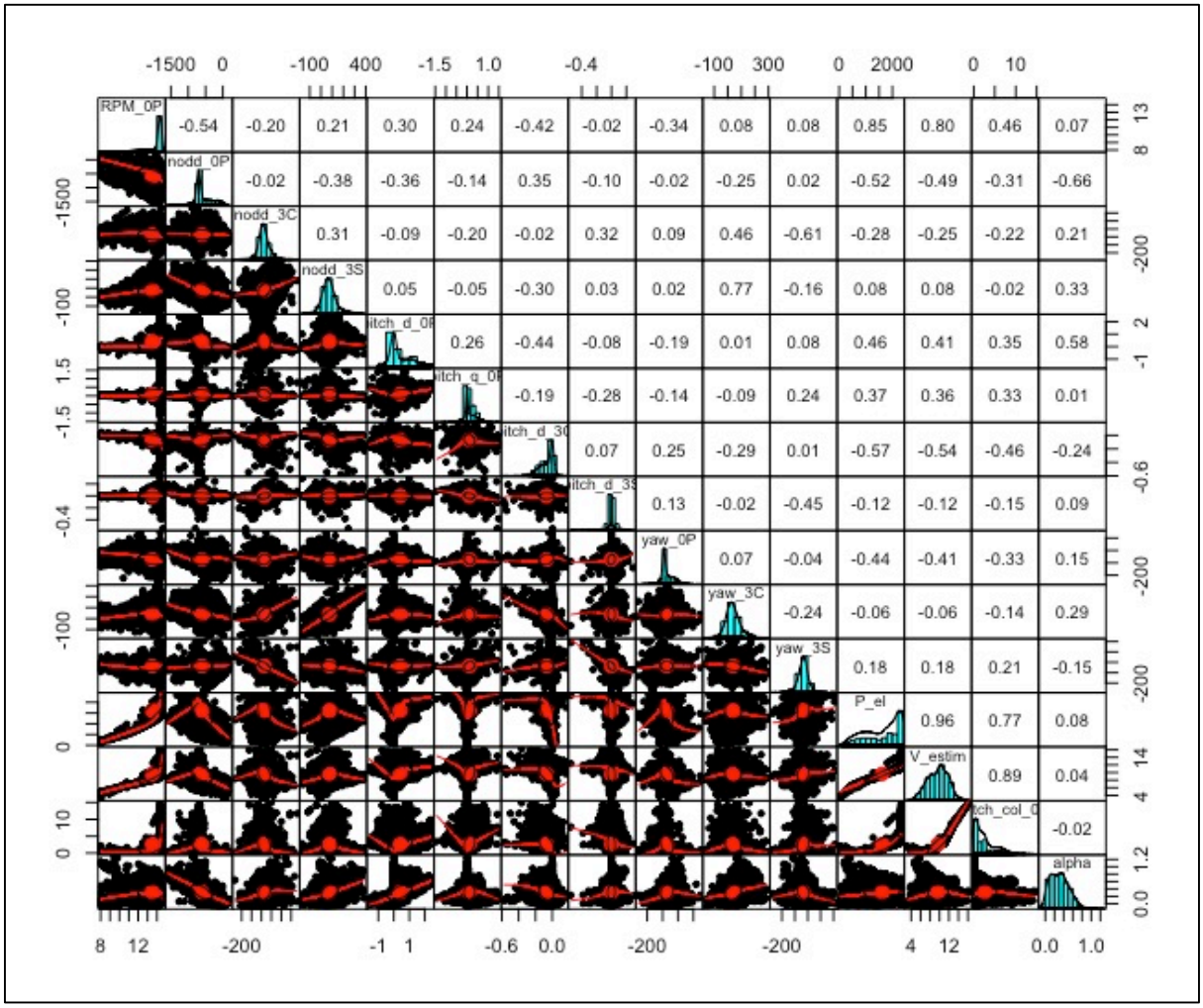

Fig 4.11 Multicollinearity Analysis of Variables

Based on the correlation of predictors, we can see that there not much overall co-relations when all predictors taken together, but there are pairs with very high co-relations. Pairs (RPM_0P - P_el), (RPM_OP — v_estim) , (nod_3s — yaw_3c), (P_el - pitch_col_0P), (P_el - V-estim), (v_estim — pitch_col_op) have high co-relation amongst them.

We can further tune the model by a stepwise selection of variables by backward elimination. To take care of multicollinearity we will now evaluate Lasso Regression.

B. Lasso Regression

The Lasso is a shrinkage and selection method for linear regression. It minimizes the usual sum of squared errors, with a bound on the sum of the absolute values of the coefficients. It has connections to soft-thresholding of wavelet coefficients, forward stagewise regression, and boosting methods. Lasso regression takes care of the co-relations.

```
## Method2 : Using Lasso Regression ##
trainx <- as.matrix.data.frame(alpha_train[ ,c(1:14)])
trainy <- alpha_train[ , 15]
cvfit = cv.glmnet(trainx, trainy, nfolds=20, family = "gaussian")
plot(cvfit)
```

The two different values of λ reflect two common choices for λ. The λmin is the one which minimizes out-of-sample loss in CV. The λ1se is the one which is the largest λ value within 1 standard error of λmin. One line of reasoning suggests using λ1se because it hedges against overfitting by selecting a larger λ value than the min. Which choice is best is context-dependent. Confidence intervals represent error estimates for the loss metric (red dots). They're computed using CV. The vertical lines show the locations of λmin and λ1se. The numbers across the top are the number of nonzero coefficient estimates.

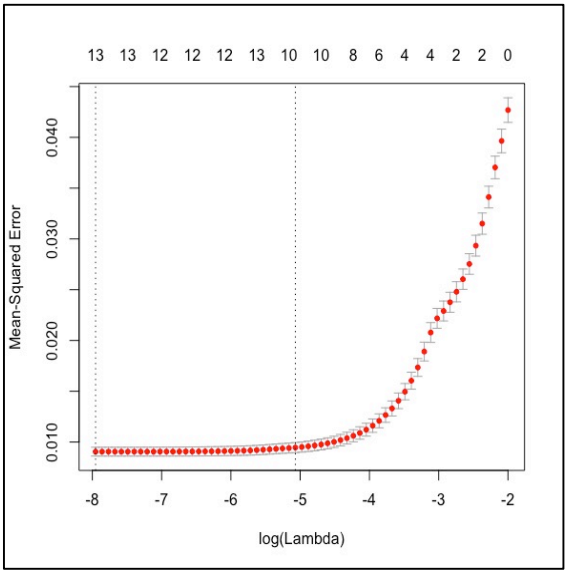

Fig 4.12 Lambda Value Analysis

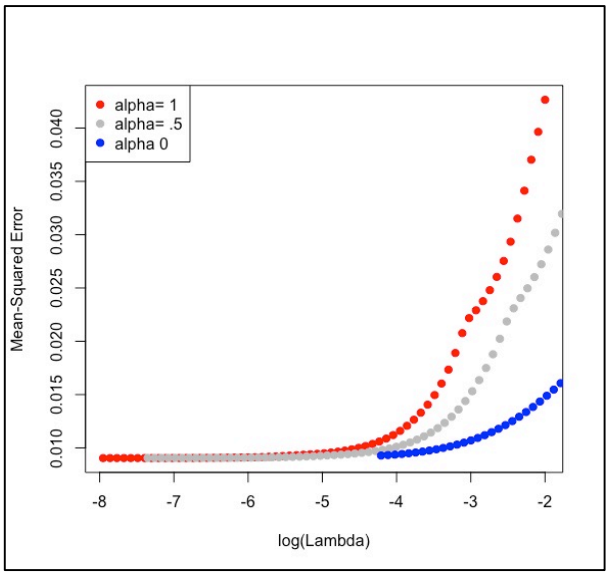

Fig 4.13 Evaluation of alpha at 1, 0.5 and 0

Now, we will see the best value for the alpha (α).

```
## Calculating the best value for alpha
cv1=cv.glmnet(trainx,trainy,alpha=1)
cv.5=cv.glmnet(trainx,trainy,alpha=.5)
cv0=cv.glmnet(trainx,trainy,alpha=0)

plot(cv1);plot(cv.5);plot(cv0)
plot(log(cv1$lambda),cv1$cvm,pch=19,col="red",xlab="log(Lambda)",ylab=cv1$name)
points(log(cv.5$lambda),cv.5$cvm,pch=19,col="grey")
points(log(cv0$lambda),cv0$cvm,pch=19,col="blue")
legend("topleft",legend=c("alpha= 1","alpha= .5","alpha
0"),pch=19,col=c("red","grey","blue"))
```

Fig 4.13 shows the different lambda values for the alpha. It is clear from the above figure that value of alpha 1 serves the best need for the creating the best Lasso model. Using this value we will optimize our model for getting best estimates.

Following code shows the optimized Lasso model and coefficients with respect to this model. Gaussian family gives coefficients at two values (λmin and λ1se.).

```
## So best value we get is for alpha = 1. lets consider it for the calculation
cvfit = cv.glmnet(trainx, trainy, nfolds=20, family = "gaussian", alpha=1)
plot(cvfit)

## Finding the coefficients
coef(cvfit, s= 'lambda.min')
coef(cvfit, s= 'lambda.1se')
```

We see the final graph for the optimized model as follows.

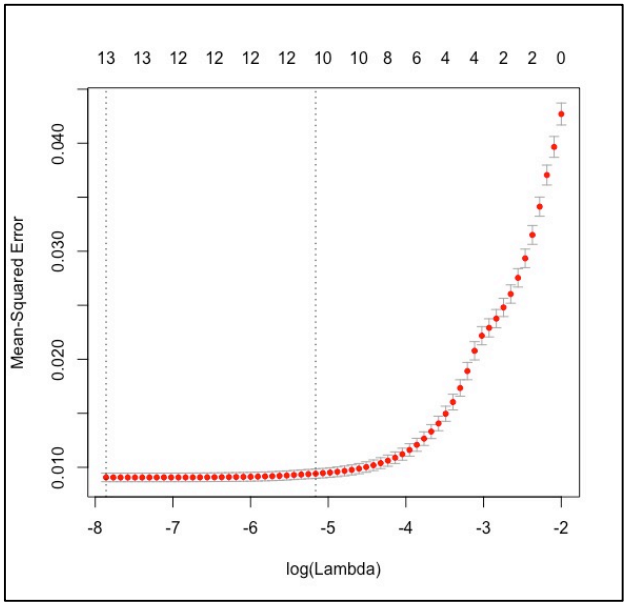

Fig 4.14 Optimized Model MSE vs Lambda

From the coefficients, we calculated in above step, its yet not clear which one to pick out of those two. Therefore, we will consider validating lambda on both of the parameters.

```
#validating to identify best lambda
valx <- as.matrix.data.frame(alpha_val[ , c(1:14)] )
valy <- alpha_val [ , 15]
predict_alpha1 <- predict(cvfit, valx , type = "response", s = "lambda.min")
predict_alpha2 <- predict(cvfit, valx ,type = "response", s = "lambda.1se")
MSE3 <- mse(alpha_val$alpha , predict_alpha1) # MSE = 0.008133895
MSE4 <- mse(alpha_val$alpha , predict_alpha2) # MSE = 0.008384234
```

Looking at the result lambda.min gives less Mean Squared Error (MSE). So, we will use it for testing our model on test data.

```
#Testing on test data
testx <- as.matrix.data.frame(alpha_test[ , c(1:14)] )
testy <- alpha_test[ , 15]
predicted_test_alpha <- predict(cvfit, testx, type = "response", s = "lambda.min")
Mean_Error <- Mean_Abs_Error(predicted_test_alpha , alpha_test$alpha) # MAE = 0.06689367
mse (alpha_test$alpha, predicted_test_alpha) #MSE = 0.009007776

# Visualizing the actual vs predicted
# a,b are calculated using lm (alpha_test$alpha ~ predicted_test_alpha)
plot(predicted_test_alpha, alpha_test$alpha, xlab="Predicted",ylab="Actual", col= "blue", pch
= 20)
abline(a=-0.022231,b=1.058436, col="Red", lty= 6, lwd= 2)
```

Conclusion (PART B):

The result shows that **Mean Absolute Error (MAE) is 0.06689367**. Mean Absolute Squared Error **(MSE) is 0.009007776** which is very low. Although, the results for Linear Regression model were quite superior to Lasso. It doesn't show much difference in error. Both the linear and Lasso regression models have almost same Mean Absolute Error. Still we should opt for Lasso Regression Model as it will take care of the inbuilt collinearity of the predictors.

The graph for Predicted vs Actual value of wind shear is shown in the figure below.

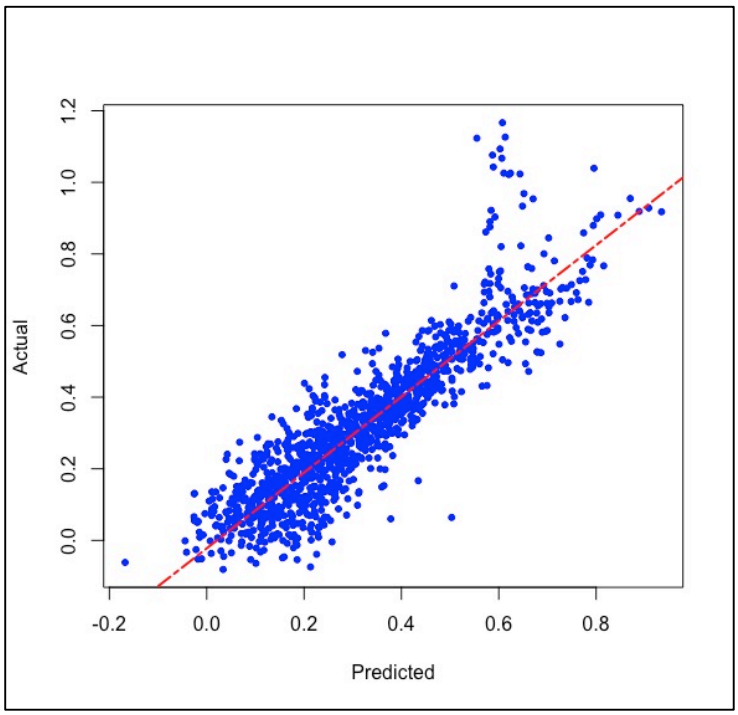

Fig 4.15 Wind Shear (Predicted vs Actual)

From the fig 4.15, it shows clear linear distribution of actual vs predicted shear speed. Although we see some outliers (extreme values) in the data, still we should consider developing the virtual wind shear sensor with right amount of data and considering the seasonality of geography.

5. Conclusion:

1. The Industrial Internet is a digital network connecting, collecting and analyzing machine data. GE believes that the Industrial Internet could add $10 to $15 trillion to global GDP in efficiency gains over the next two decades. Predictive modelling will gain significant amount of share in driving the innovation and drilling down the operating cost for next generation wind turbines.

2. We built the predictive model based on the limited available data without considering most of the external factors which may prove decisive while getting the results closer to the actual. We can use other factors such as GIS data to co-relate with the atmospheric condition of wind farm.

3. As of today, we are using meteorological masts – met towers for measurement of wind speed. We know that there is reasonable amount of distance between wind turbine and met mast. So, it's quite tough to estimate the minor changes in speed profile with respect to limited number of sensors available on met mast for wind speed calculation.

4. With a limited amount of data our classifier predicted wind shear class quite well. Also, we succeeded in predicting the wind shear with minimal error. GE Wind should consider launching NPI program to offer a "Virtual Shear Sensor"

6. References:

1. GE Reports: Wind in the Cloud? How the Digital Wind Farm Will Make Wind Power 20 Percent More Efficient https://www.ge.com/reports/post/119300678660/wind-in-the-cloud-how-the-digital-wind-farm-will-2/
2. U.S. Geological Survey Data Visualization: https://energy.usgs.gov/OtherEnergy/WindEnergy.aspx#4312358-data
3. Onshore Industrial Wind Turbine Locations Data for the United States: https://pubs.usgs.gov/ds/817/
4. https://windwisema.org/wind_shear-turbine_noise-faq/
5. Wind profile power law: https://en.wikipedia.org/wiki/Wind_profile_power_law
6. Hierarchical Communication Network Architectures for Offshore Wind Power Farms: http://www.mdpi.com/1996-1073/7/5/3420/htm
7. Wind Turbine Generator Technologies: https://www.intechopen.com/books/advances-in-wind-power/wind-turbine-generator-technologies
8. https://www.kompulsa.com/factors-affecting-wind-turbine-performance/
9. http://www.ewea.org/fileadmin/files/library/publications/reports/Economics_of_Wind_Energy.pdf
10. https://www.gerenewableenergy.com/wind-energy/turbines
11. https://people.duke.edu/~rnau/rsquared.htm
12. http://web.mit.edu/windenergy/windweek/Presentations/Wind%20Energy%20101.pdf
13. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
14. http://statweb.stanford.edu/~tibs/lasso.html

Reference Diagram for Closer look on Pitching, Nodding and Yawing: