# 181058 DSLab Final

Task 01

Reading the dataset

importing dataset into R

```
df = read.csv('regression_Data.csv')
head(df)
```

```
##      AT     V      AP    RH     PE
## 1 14.96 41.76 1024.07 73.17 463.26
## 2 25.18 62.96 1020.04 59.08 444.37
## 3  5.11 39.40 1012.16 92.14 488.56
## 4 20.86 57.32 1010.24 76.64 446.48
## 5 10.82 37.50 1009.23 96.62 473.90
## 6 26.27 59.44 1012.23 58.77 443.67
```

checking for its structure to find nore information about the given dataset

```
str(df)
```

```
## 'data.frame':    9568 obs. of  5 variables:
##  $ AT: num  14.96 25.18 5.11 20.86 10.82 ...
##  $ V : num  41.8 63 39.4 57.3 37.5 ...
##  $ AP: num  1024 1020 1012 1010 1009 ...
##  $ RH: num  73.2 59.1 92.1 76.6 96.6 ...
##  $ PE: num  463 444 489 446 474 ...
```

From the output, we can see that we have 9568 observations/ rows for 5 variables/column. We can also see that all are of types *num*

Cleaning the data

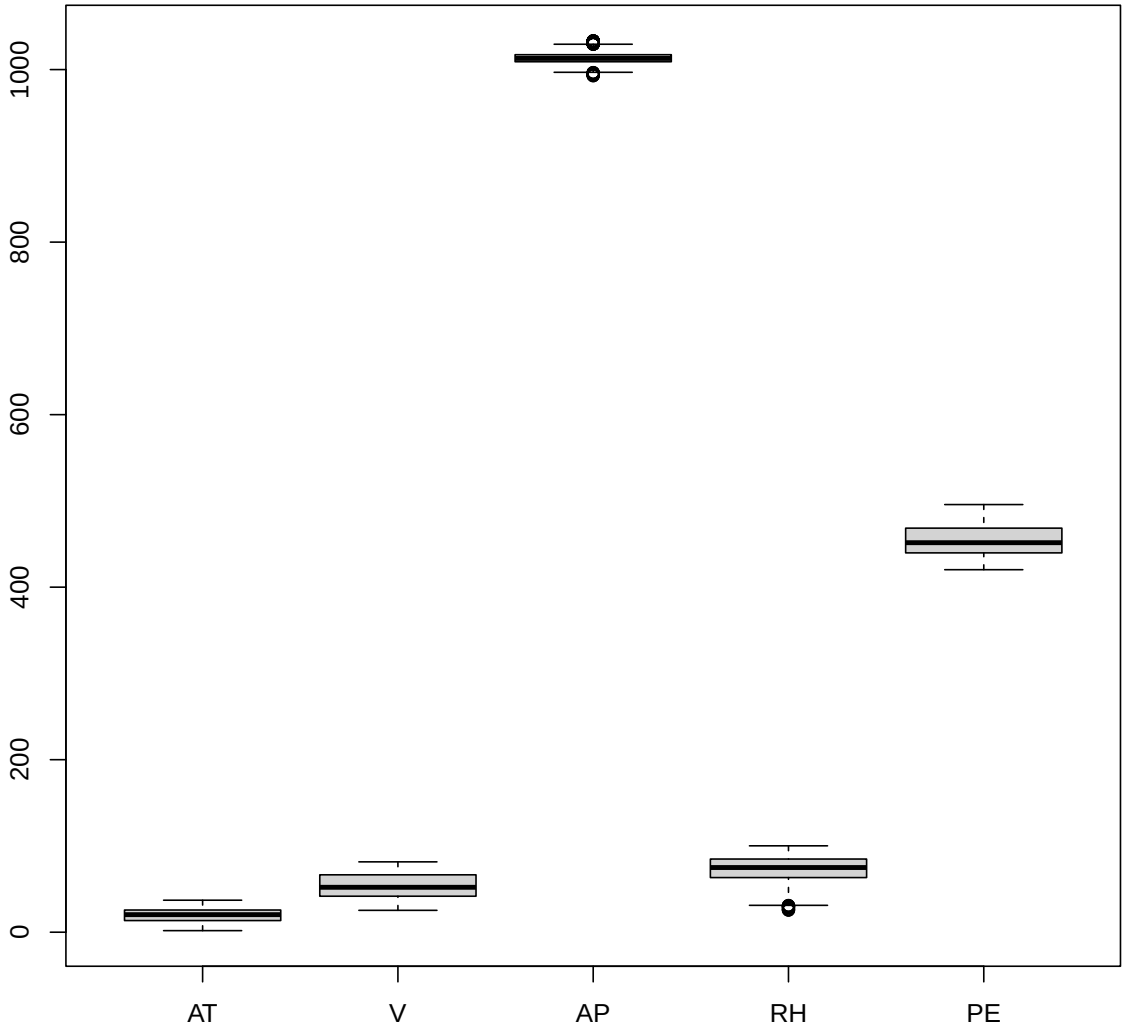
For cleaning the data, we need to

1. Finding outliers
2. Finding null values
3. Finding categorical values

Finding outliers

Best method is to use boxplot, values outside the plot are outliers

```r
boxplot(df)
```




From the figure, we don't seem to find any outlier, trying again, mathematically. 1. We could find mean and standard deviation, from where we may find Q1 and Q3, Q2 is mean Expression for outliers will be *Quartile = 1.5(Q3-Q1)* Values below mean and above max would be classified as outliers which should be removed.

```r
quartiles <- quantile(df$AT, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$AT)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
#finding outlier
df$AT[which(df$AT < Lower | df$AT > Upper)]
```

```
## numeric(0)
```

```
quartiles <- quantile(df$V, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$V)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
#finding outlier
df$V[which(df$V < Lower | df$V > Upper)]
```

```
## numeric(0)
```

```
quartiles <- quantile(df$AP, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$AP)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
#finding outlier
df$AP[which(df$AP < Lower | df$AP > Upper)]
```

```
##  [1] 1030.46 1033.25 1030.18 1031.10 1029.65 1029.80 1033.04 1033.30 1032.67
## [10] 1033.08  994.17  995.88  995.24 1031.55  996.32 1029.60 1030.30 1031.50
## [19] 1030.94  996.55 1030.83 1031.96  996.35 1031.01 1030.77 1032.98 1032.93
## [28] 1031.33  996.03 1030.86 1030.72 1033.19 1031.96 1030.42 1032.86 1032.77
## [37] 1029.63 1033.09 1030.82 1029.99 1031.80 1032.08  993.31 1033.14 1029.54
## [46] 1032.88 1029.70 1030.81 1030.00  995.45 1030.20 1032.72 1031.45 1032.14
## [55] 1029.91  995.02 1031.14  993.82 1032.83 1032.37 1032.16 1031.16 1030.72
## [64] 1029.90  994.60 1030.10 1029.61  993.74 1031.39 1031.20 1031.27 1030.38
## [73] 1031.97 1033.29 1031.58 1031.08  993.11 1031.75 1029.54 1029.70 1031.21
## [82] 1031.34 1030.68  992.89 1031.71 1031.21 1031.32 1030.61
```

```
quartiles <- quantile(df$RH, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$RH)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
#finding outlier
df$RH[which(df$RH < Lower | df$RH > Upper)]
```

```
##  [1] 25.89 28.16 29.43 26.67 29.86 30.59 25.56 30.83 28.81 26.30 30.34 30.99
```

```
quartiles <- quantile(df$PE, probs=c(.25, .75), na.rm = FALSE)
IQR <- IQR(df$PE)
Lower <- quartiles[1] - 1.5*IQR
Upper <- quartiles[2] + 1.5*IQR
#finding outlier
df$PE[which(df$PE < Lower | df$PE > Upper)]
```

```
## numeric(0)
```

So, we have outliers in AP and RH Removing them.

```
without_outlier <- subset(df, df$AP > Lower & df$AP < Upper)
without_outlier <- subset(df, df$RH > Lower & df$RH < Upper)

dim(without_outlier)
```

```
## [1] 0 5
```

```
outliers <- boxplot(df, plot = FALSE)$out
data_no_outlier <- df[-which(df %in% outliers)]
dim(data_no_outlier)
```

```
## [1] 9568    0
```

```
length(data_no_outlier)
```

```
## [1] 0
```

Task 02 Linear Regression

Plotting dependant and independant variables

Finding corelation between all variables

```
#installing corrplot
library(corrplot)
```

```
## corrplot 0.92 loaded
```

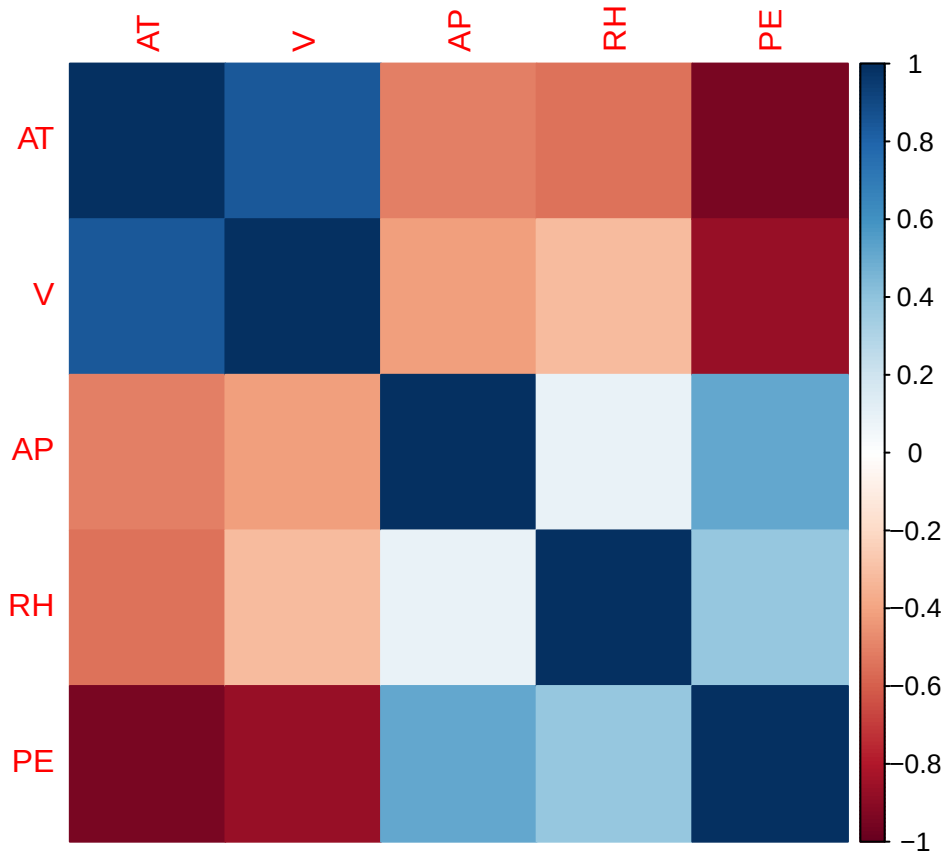
```
corr <- sapply(df, is.numeric)
corr
```

```
##   AT    V   AP   RH   PE
## TRUE TRUE TRUE TRUE TRUE
```

```
cor.data <- cor(df[, corr])
cor.data
```

```
##            AT          V          AP          RH         PE
## AT  1.0000000  0.8441067 -0.50754934 -0.54253465 -0.9481285
## V   0.8441067  1.0000000 -0.41350216 -0.31218728 -0.8697803
## AP -0.5075493 -0.4135022  1.00000000  0.09957432  0.5184290
## RH -0.5425347 -0.3121873  0.09957432  1.00000000  0.3897941
## PE -0.9481285 -0.8697803  0.51842903  0.38979410  1.0000000
```

```
corrplot(cor.data, method='color')
```




Splitting the data

```
#Splitting the data
library(caTools)
set.seed(2)
split <- sample.split(df, SplitRatio = 0.8)
train <- subset(df, split == 'TRUE')
test <- subset(df, split == 'FALSE')

dim(train)
```

```
## [1] 7655    5
```

```
dim(test)
```

```
## [1] 1913    5
```

Linear Regression, plotting and summary

```
#making a model
model <- lm(PE ~., data = train)
summary(model)
```

```
##
## Call:
## lm(formula = PE ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -43.392  -3.215  -0.126   3.221  17.351
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 452.890965  10.885387   41.605  < 2e-16 ***
## AT           -1.967311   0.017243 -114.096  < 2e-16 ***
## V            -0.238547   0.008201  -29.088  < 2e-16 ***
## AP            0.063818   0.010562    6.042 1.59e-09 ***
## RH           -0.158174   0.004723  -33.492  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.589 on 7650 degrees of freedom
## Multiple R-squared:  0.9272, Adjusted R-squared:  0.9271
## F-statistic: 2.435e+04 on 4 and 7650 DF,  p-value: < 2.2e-16
```

```
pred <- predict(model, test)
head(pred)
```

```
##        5       10       15       20       25       30
## 471.7830 473.0014 435.3944 472.2713 442.8040 435.2883
```

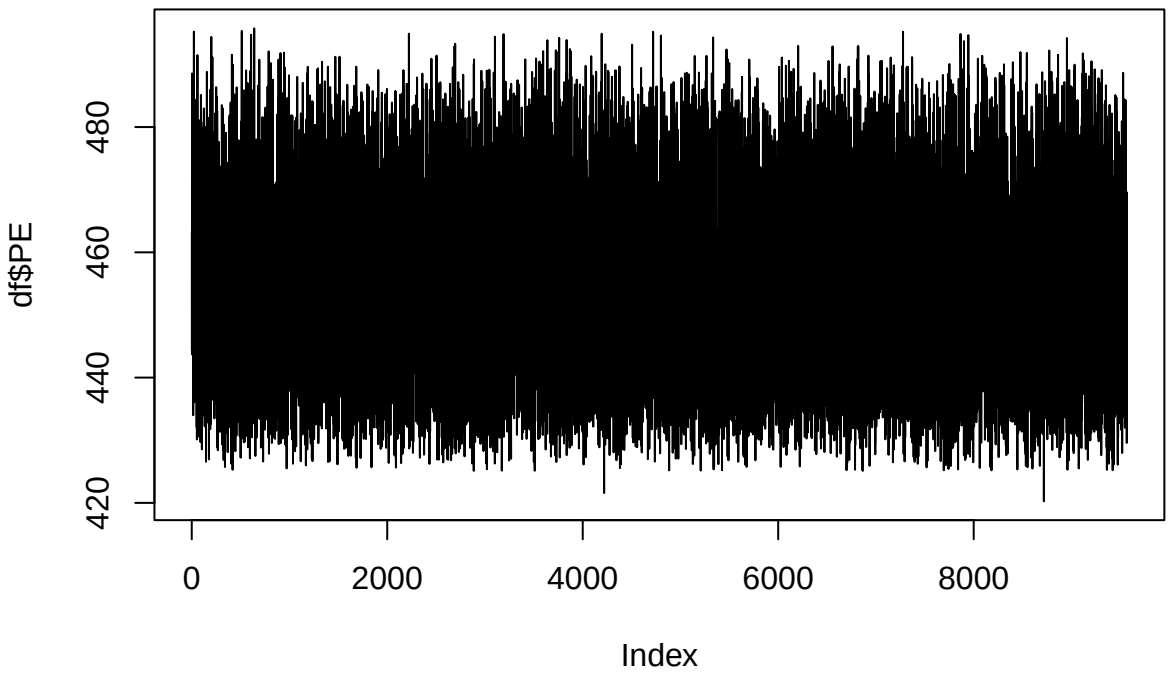
```
res <-residuals(model)
res <- as.data.frame(res)
head(res)
```

```
##          res
## 1 -4.0183796
## 2  0.2832469
## 3  5.1012716
## 4 -4.0479877
## 6  1.3373117
## 7  3.3948117
```

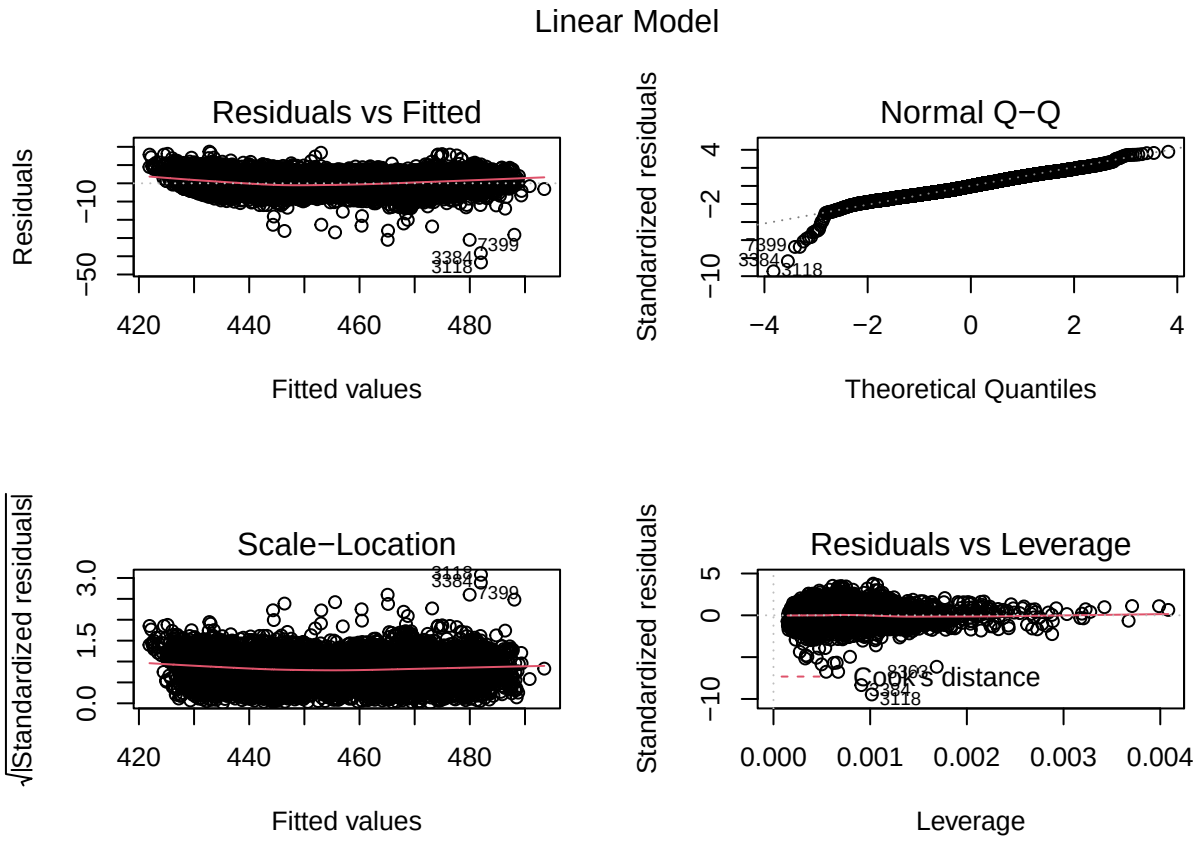
```
#Compairing with actual values
results <- cbind(pred, df$PE)
colnames(results) <- c('Predicted', 'Real')
results <- as.data.frame(results)
head(results)
```

```
##   Predicted   Real
## 1  471.7830 463.26
## 2  473.0014 444.37
## 3  435.3944 488.56
## 4  472.2713 446.48
## 5  442.8040 473.90
## 6  435.2883 443.67
```

```
plot(df$PE, type= 'l')
```




```
par(mfrow=c(2,2))
plot(model)
mtext("Linear Model",
      side = 3,
      line = - 1,
      outer = TRUE)
```

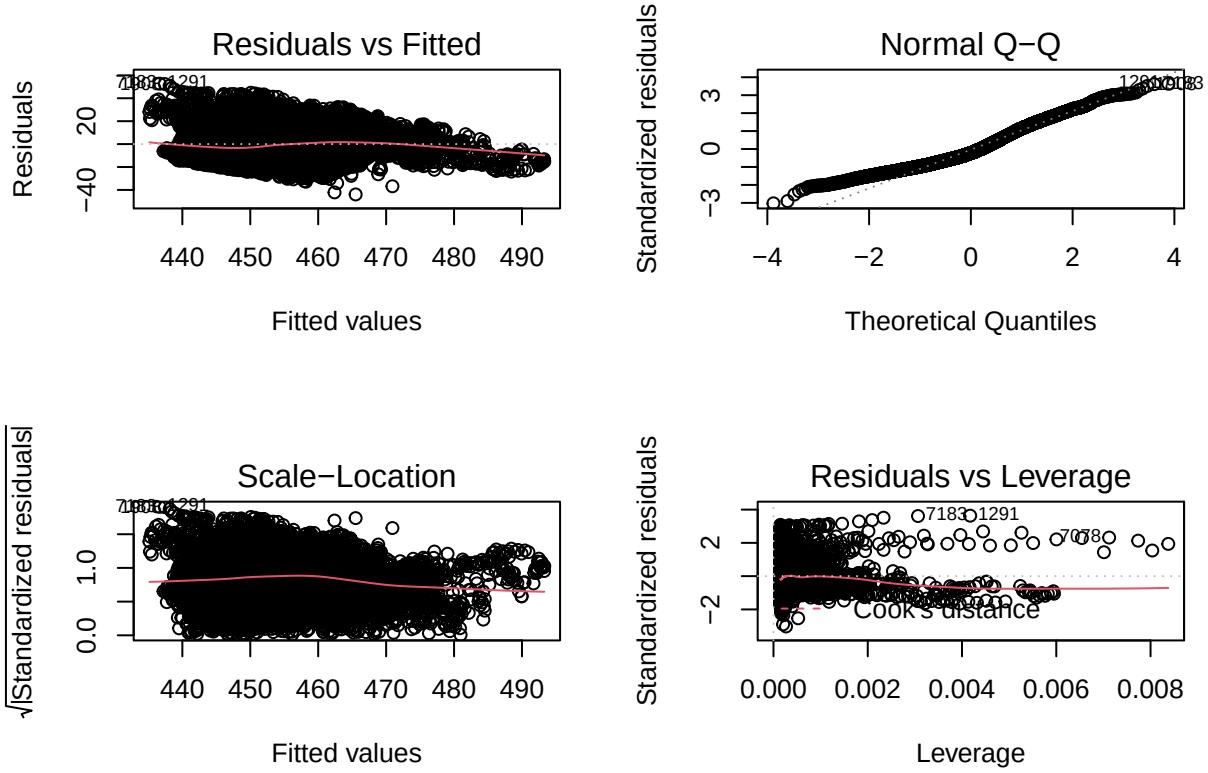



```
summary(head((model)))
```

```
##                Length Class  Mode
## coefficients        5   -none- numeric
## residuals        7655   -none- numeric
## effects          7655   -none- numeric
## rank                1   -none- numeric
## fitted.values    7655   -none- numeric
## assign              5   -none- numeric
```

Polynomial regression with degree 2 and 3 and summary

```
DegreeModel2 <- lm(df$PE ~ poly(df$AP, 2), data = train)
par(mfrow=c(2,2))
plot(DegreeModel2)
mtext("Degree Model 2",
      side = 3,
      line = - 1,
      outer = TRUE)
```

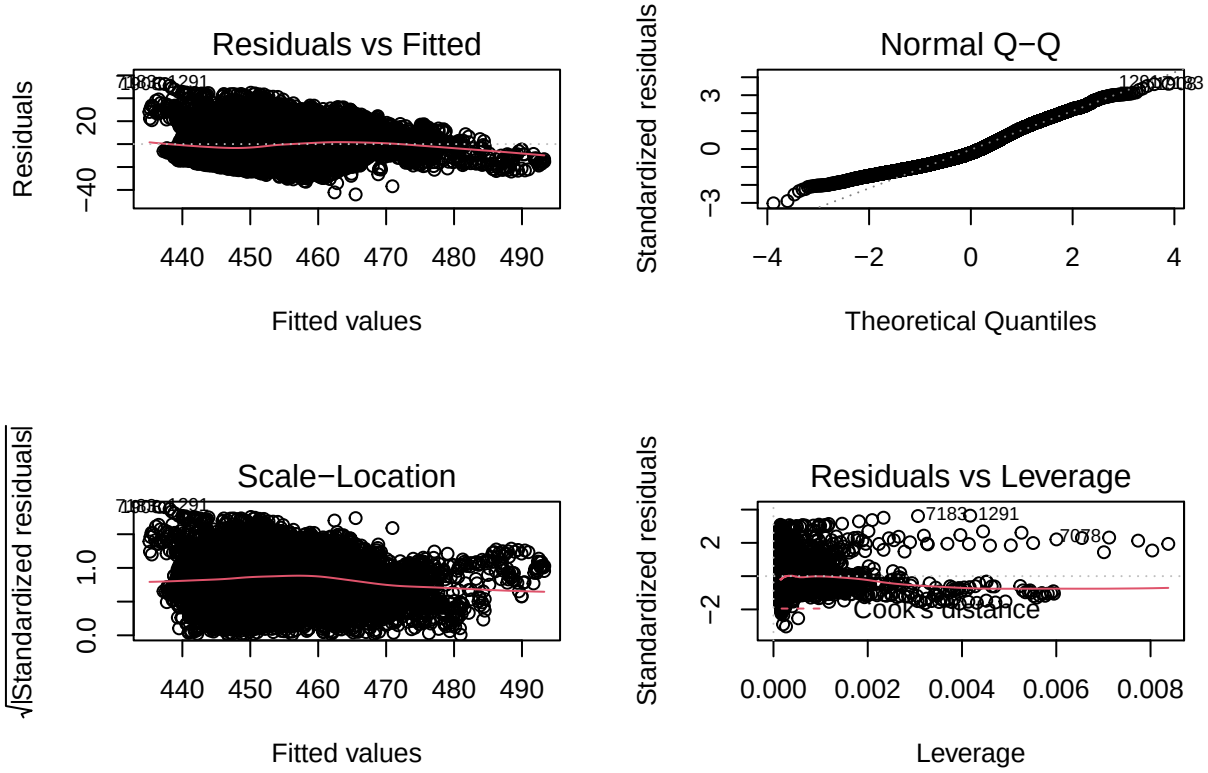



```
summary(head((DegreeModel2)))
```

```
##                Length Class  Mode
## coefficients        3   -none- numeric
## residuals        9568   -none- numeric
## effects          9568   -none- numeric
## rank                1   -none- numeric
## fitted.values    9568   -none- numeric
## assign              3   -none- numeric
```

```
DegreeModel3 <- lm(df$PE ~ poly(df$AP, 2), data = train)
par(mfrow=c(2,2))
plot(DegreeModel3)
mtext("Degree Model 3",
      side = 3,
      line = - 1,
      outer = TRUE)
```

Degree Model 3

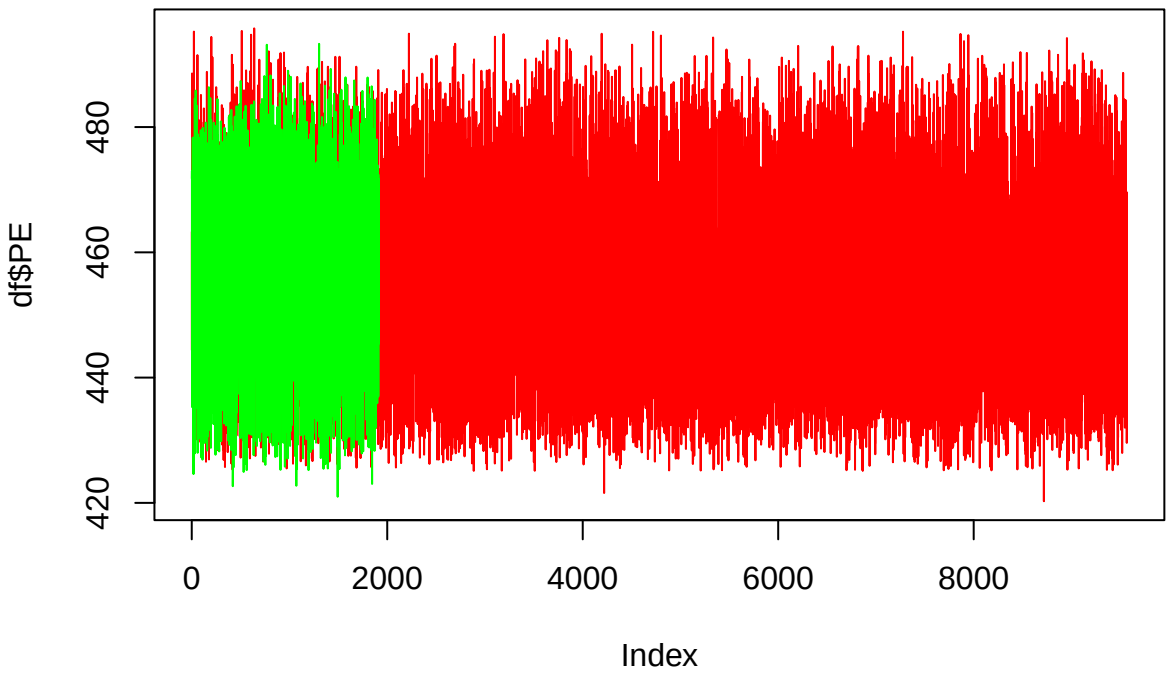



```
summary(head((DegreeModel3)))
```

```
##               Length Class  Mode
## coefficients     3   -none- numeric
## residuals     9568   -none- numeric
## effects       9568   -none- numeric
## rank             1   -none- numeric
## fitted.values 9568   -none- numeric
## assign           3   -none- numeric
```

Predicting the result using trained data

```
predicted_result <- predict(model, test)

res <- residuals(model)
res <- as.data.frame(res)

final_res <- cbind(predicted_result, df$PE)

colnames(final_res) <- c("Predicted","real")

final_res <- as.data.frame(final_res)

#plotting them
plot(df$PE, type = 'l', lty = 1.8, col = 'red')
lines(predicted_result, col='green')
```




```
#checking accuracy
rmse <- sqrt(mean(predicted_result - df$PE)^2)
paste("Accuracy is ", rmse)
```

```
## [1] "Accuracy is  0.212210824175547"
```