MIDTERM

INFO 7390: Advances in Data Sciences/Architecture
Team 9

Bhavesh Patel
Ila Nigam
Ayushi Srivastava

Table of Contents

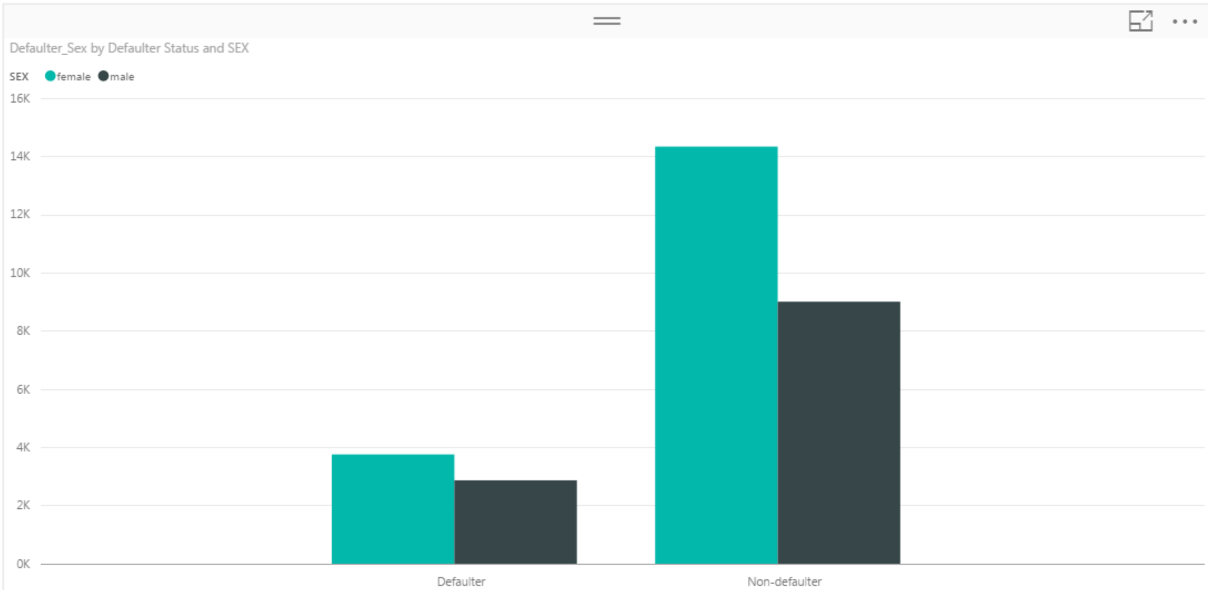
Problem1- Credit Card Clients

GOAL

1. The data set contains instances and 23 attributes consisting of gender, education profile, marital status, age, history of statement balance, payment status and binary status of default (1 or 0).
2. The goal of the problem is to clean the data and infer various relations between the predictor variables. Using logistic regression, neural network and classification tree we have to classify the data and predict the values.
3. We have to build different models to evaluate different trends and compare the sensitivity, Specification and plot confusion matrix, ROC Curve and Lift Charts for all the classification method.

Power PI and Inferences

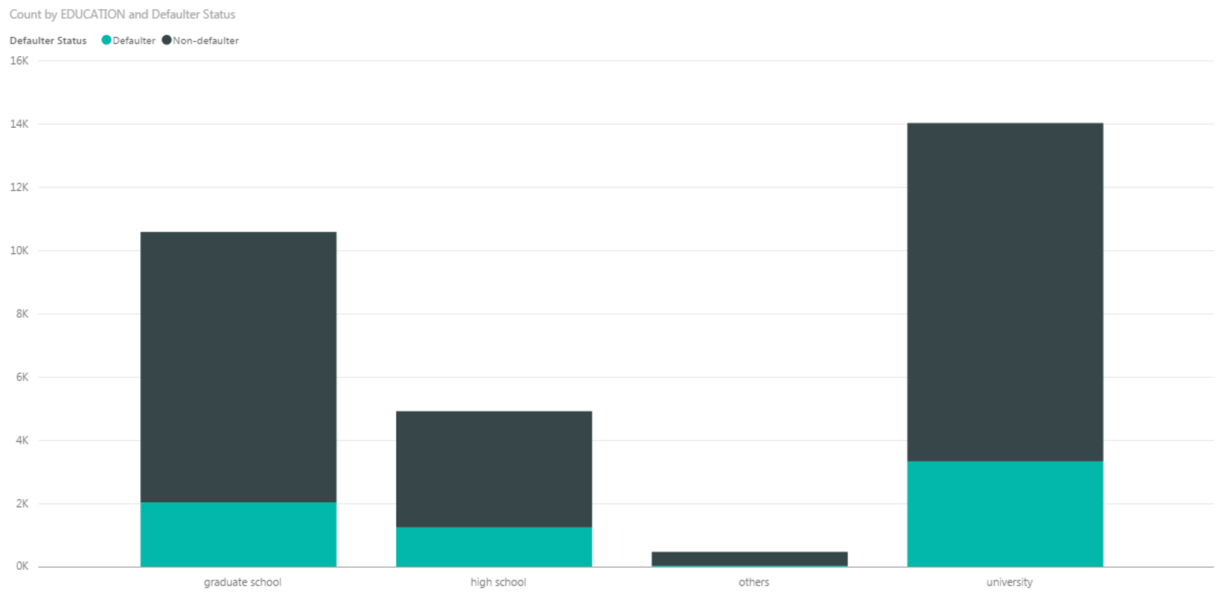
Gender profile of defaults payment vs Non-Default



Gender profile of defaults payment vs Non-Default

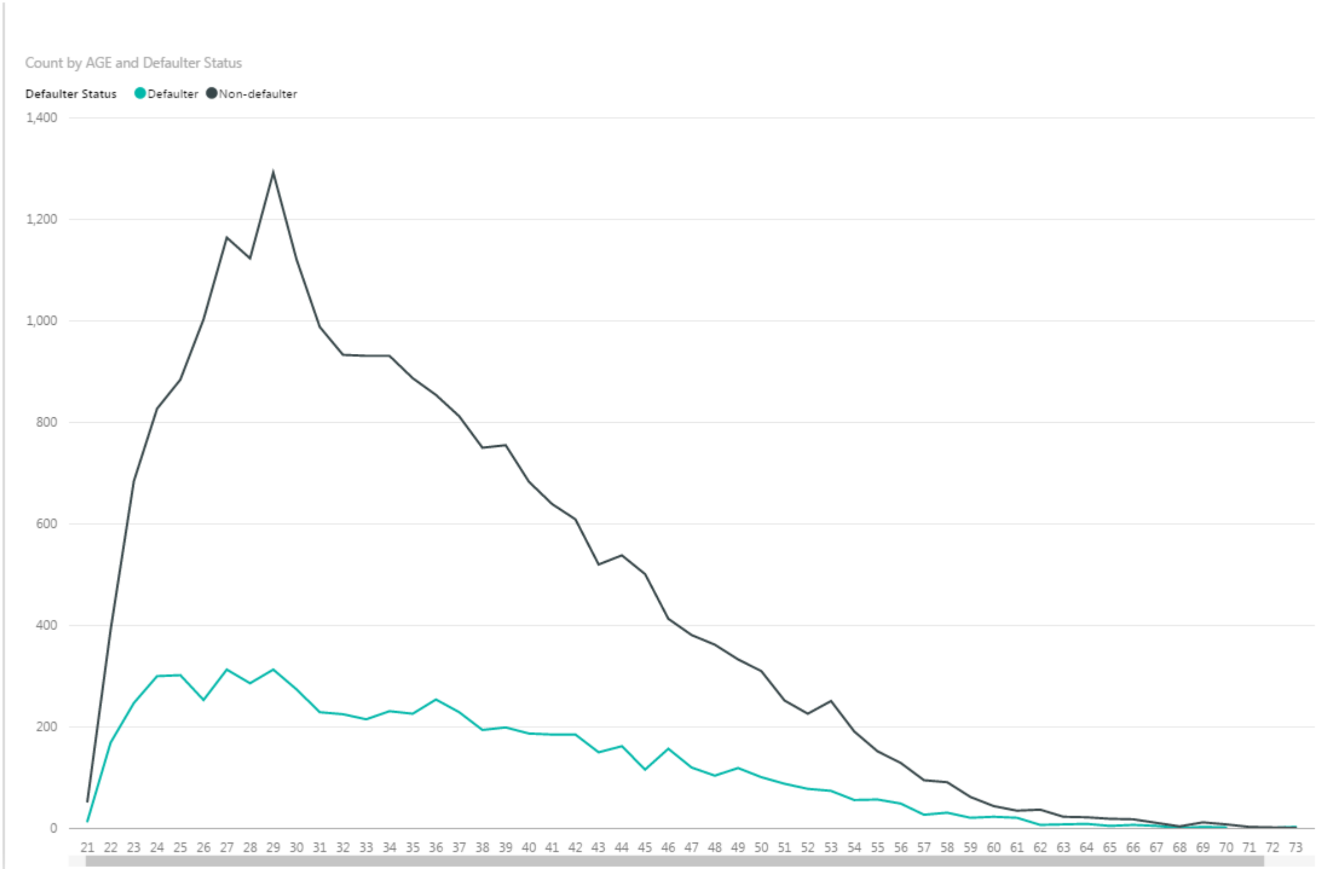
- ✓ *We can see from the above bar chart, number of females is higher than males for both default and no default payment categories, but we cannot see their respective*

Education profile of defaults payment vs Non-Default



- ✓ We can see from above bar graph, ratio of customers having graduate school is higher in default payment category as compared to no-default and ratio of customers having university education is higher in No Default Payment than Default category

Density vs. Age Profile

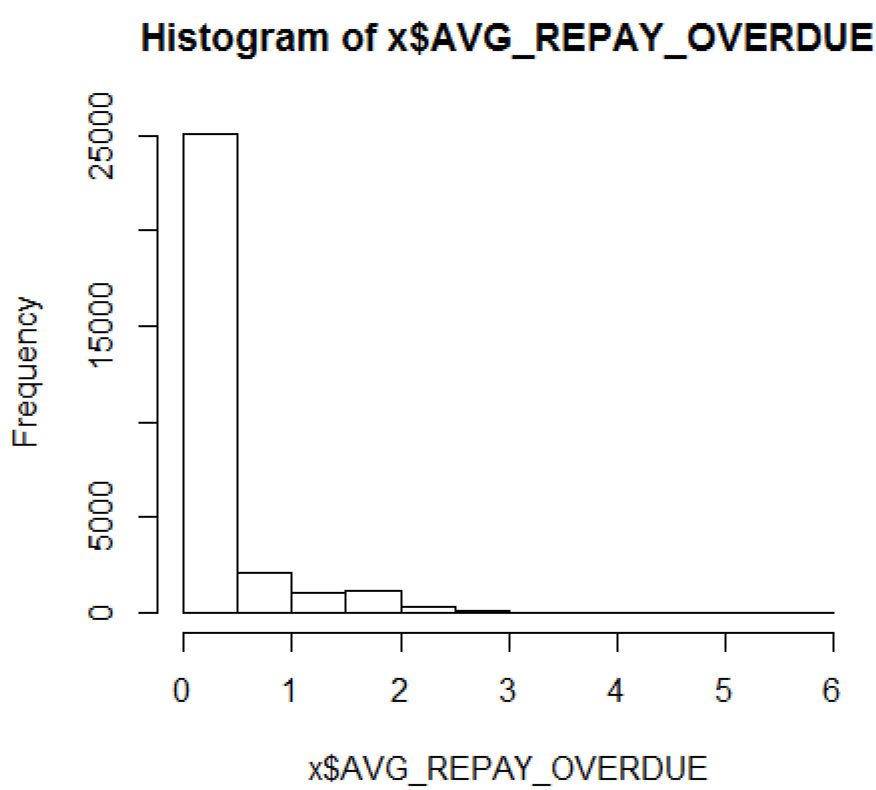


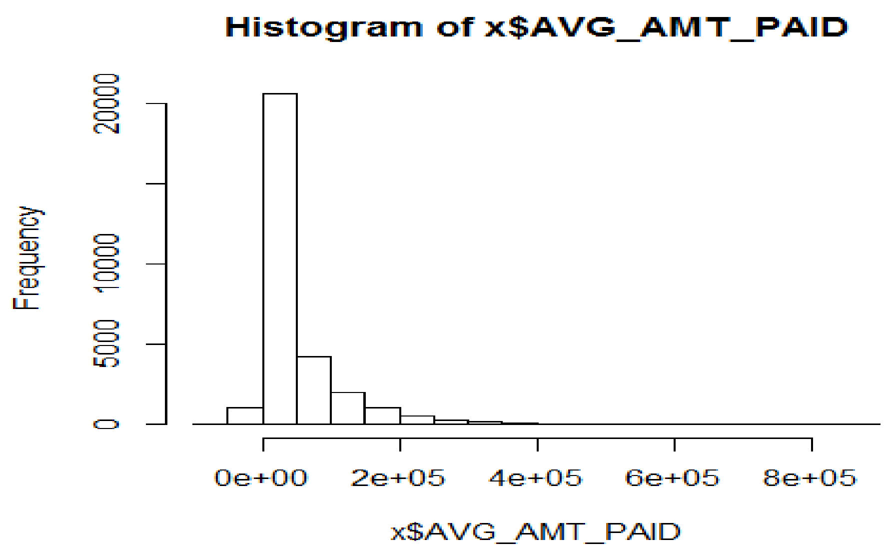
- ✓ *We can see clearly see that age peaks around 28-29 years in default category and it has lower peak around 27-28 years in no default category.*

Logistic Regression

1. To go ahead with logistic regression model we first cleaned the data. Removed the outliers. Plotted various graphs such as Histograms, Box plot to maintain the integrity of data.
2. Next we sampled the data into Test and Train to model the train data and then evaluate our model on based of Test data set.
3. In Logistic regression we did GLM modeling with variables : LIMIT_BAL+SEX+MARRIAGE+AVG_REPAY_OVERDUE
4. Threshold of 0.27 was set to classify data into 0 or 1 and predictions were made as per the analysis.
5. Graphs were plotted to evaluate the performance of the models

Predictor analysis



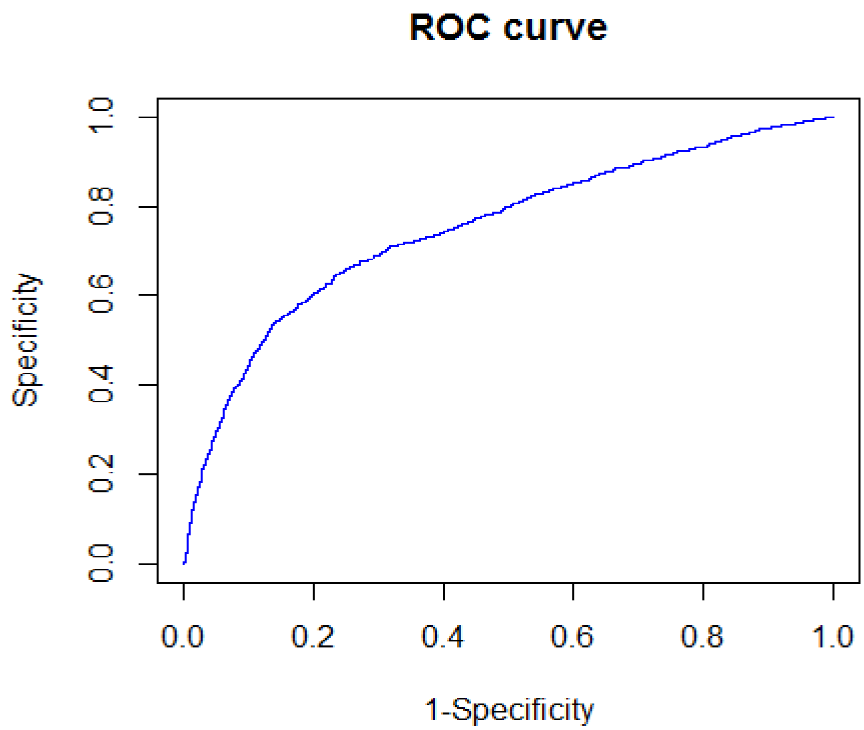



```r
#Sampling the data
smp_size <- floor(0.75 * nrow(x))
set.seed(123)
train_ind <- sample(seq_len(nrow(x)), size = smp_size)

train <- x[train_ind, ]
test <- x[-train_ind, ]
names(test)
#Build Logistic Regression
glm.fit <- glm(Y~LIMIT_BAL+SEX+MARRIAGE+AVG_REPAY_OVERDUE,data=train, family=binomial(link="logit"
summary(glm.fit)

prob <- predict(glm.fit, newdata=test, type="response")
pred <- rep(1,length(prob))
pred[prob<=0.27] <- 0
```

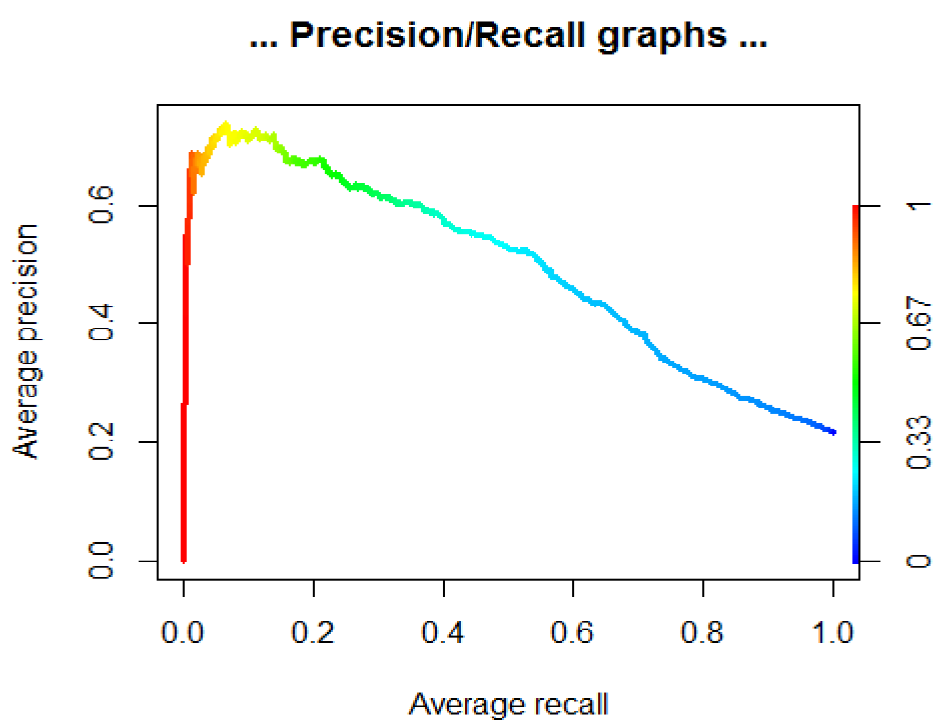
```
glm(formula = Y ~ LIMIT_BAL + SEX + MARRIAGE + AVG_REPAY_OVERDUE +
    AVG_BILL_REPAY + AVG_AMT_PAID, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3625  -0.6358  -0.5635  -0.3821   3.3093

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)       -9.269e-01  8.052e-02 -11.511  < 2e-16 ***
LIMIT_BAL         -1.270e-06  1.564e-07  -8.118 4.75e-16 ***
SEX               -1.511e-01  3.330e-02  -4.537 5.71e-06 ***
MARRIAGE          -1.653e-01  3.156e-02  -5.238 1.63e-07 ***
AVG_REPAY_OVERDUE  1.278e+00  2.910e-02  43.941  < 2e-16 ***
AVG_BILL_REPAY    -3.215e-05  3.552e-06  -9.051  < 2e-16 ***
AVG_AMT_PAID       1.201e-06  3.138e-07   3.826  0.00013 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 27008  on 25499  degrees of freedom
Residual deviance: 23551  on 25493  degrees of freedom
AIC: 23565

Number of Fisher Scoring iterations: 5
```

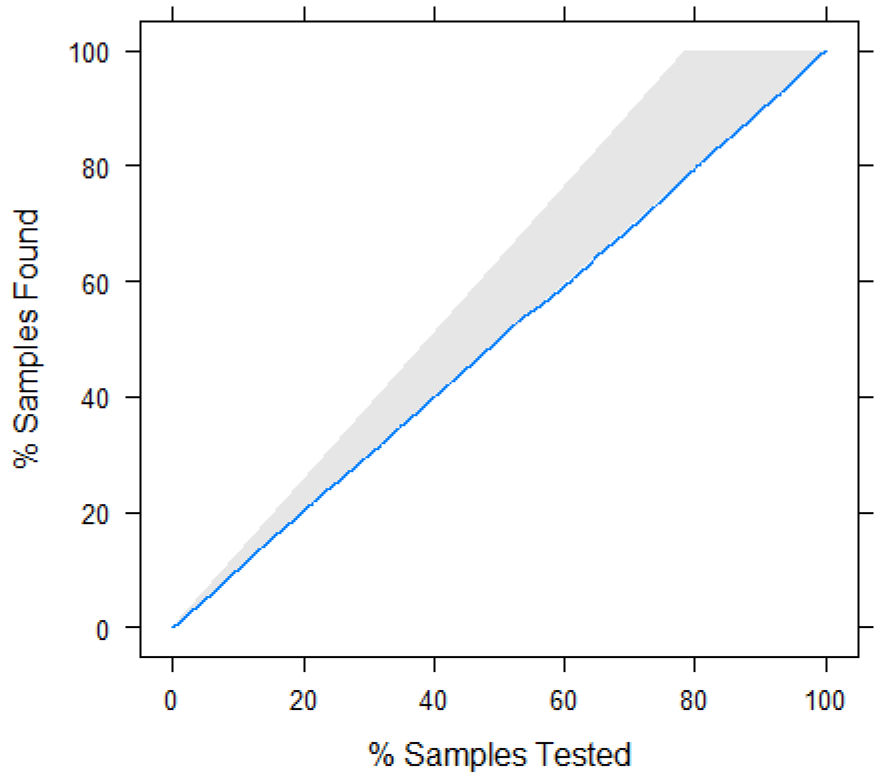
```
> #Plotting ROC Curve
> #install.packages("ROCR")
> library(ROCR)
> prediction <- prediction(prob,test$Y)
> performance <- performance(prediction, meas
> plot(performance, main="ROC curve",xlab="1-
> # ROC area under the curve
> prediction <- prediction(prob,test$Y)
> auc.tmp <- performance(prediction,"auc")
> auc <- as.numeric(auc.tmp@y.values)
> print(auc)
[1] 0.7521183
>
```

```
> confusionMatrix(pred,test$Y)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3177  541
         1  351  431

               Accuracy : 0.8018
                 95% CI : (0.7898, 0.8133)
    No Information Rate : 0.784
    P-Value [Acc > NIR] : 0.00183

                  Kappa : 0.3701
 Mcnemar's Test P-Value : 2.481e-10

            Sensitivity : 0.9005
            Specificity : 0.4434
         Pos Pred Value : 0.8545
         Neg Pred Value : 0.5512
             Prevalence : 0.7840
         Detection Rate : 0.7060
   Detection Prevalence : 0.8262
      Balanced Accuracy : 0.6720

       'Positive' Class : 0
```


ROC curve
Specificity
1-Specificity
0.0
0.2
0.4
0.6
0.8
1.0

:


... Precision/Recall graphs ...
Average precision
Average recall
0.0
0.2
0.4
0.6
0.8
1.0
0
0.33
0.67
1



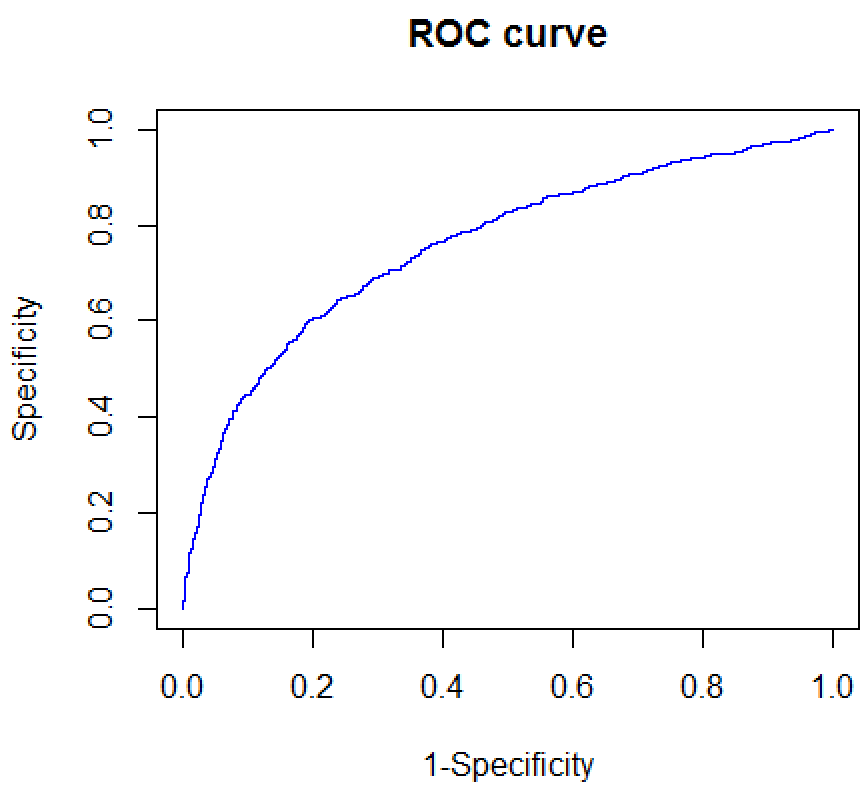
Neural Networks

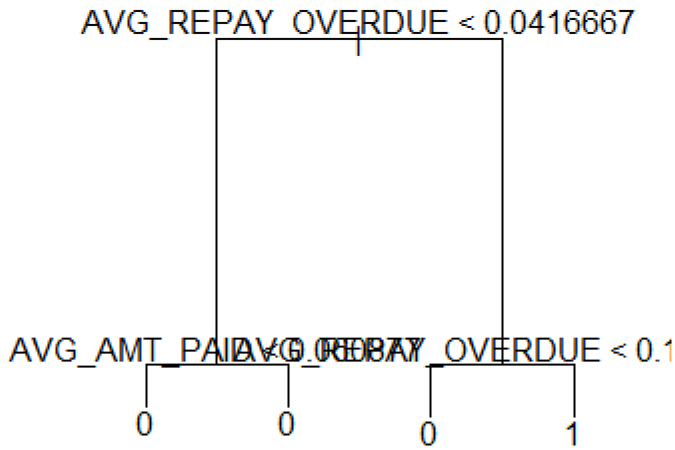
1. To go ahead with Neural Networks model we first cleaned the data. Removed the outliers. Plotted various graphs such as Histograms, Box plot to maintain the integrity of data.
2. Next we sampled the data into Test and Train to model the train data and then evaluate our model on based of Test data set.
3. In Neural Networks we did modeling with variables : LIMIT_BAL+SEX+MARRIAGE+AGE+AVG_REPAY_OVERDUE+AVG_AMT_PAID+AVG_BILL_REPAY
4. Graphs were plotted to evaluate the performance of the models

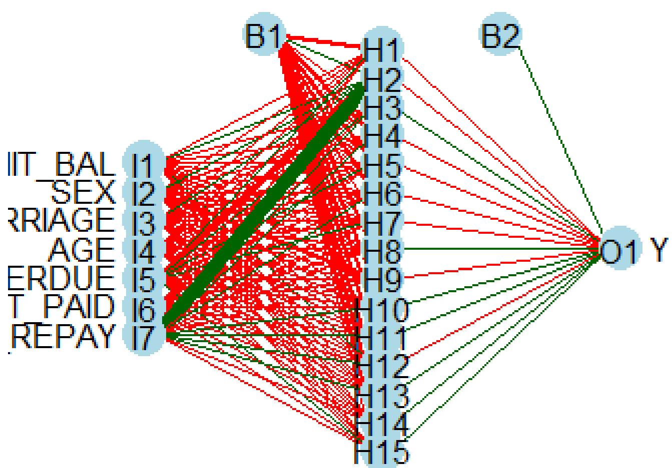
```
library(NeuralNetTools)

maxs <- apply(x, 2, max)
mins <- apply(x, 2, min)

scaled <- as.data.frame(scale(x, center = mins, scale = maxs - mins))

index <- sample(1:nrow(x),round(0.90*nrow(x)))
train_ <- scaled[index,]
test_ <- scaled[-index,]
test_$Y <- as.factor(test_$Y)
train_$Y <- as.factor(train_$Y)
# library(neuralnet)
# n <- names(train_)
# f <- as.formula(paste("Y ~", paste(n[!n %in% "Y"], collapse = " + ")))
# nn <- neuralnet(f,data=train_,hidden=c(4,3))


#Build the model

fitnn <- nnet(Y~LIMIT_BAL+SEX+MARRIAGE+AGE+AVG_REPAY_OVERDUE+AVG_AMT_PAID+AVG_BILL_REPAY,
              data=train_, size=15,hess = T, dk=5e-4, maxit = 200)
summary(fitnn)

#Predict for test data
pred = predict(fitnn, newdata=test_, type="class")
str(test_)
```

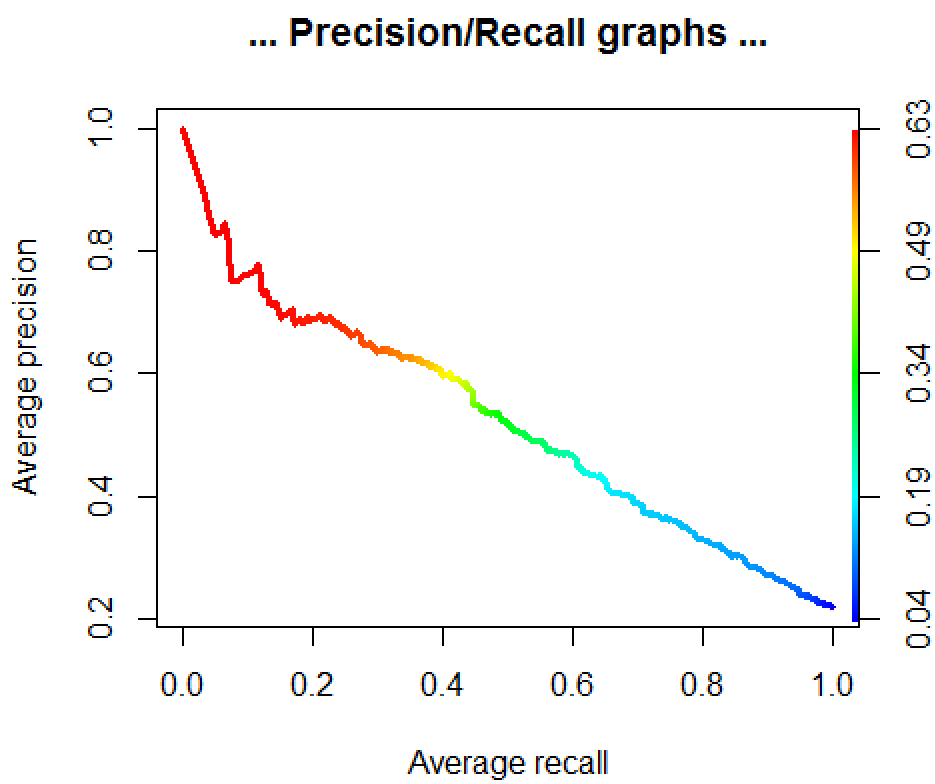
```
> # ROC area under the curve
> pred = predict(fitnn, newdata=test_, type="raw")
> prediction <- prediction(pred,test_$Y)
> auc.tmp <- performance(prediction,"auc")
> auc <- as.numeric(auc.tmp@y.values)
> print(auc)
[1] 0.7595364
>
```

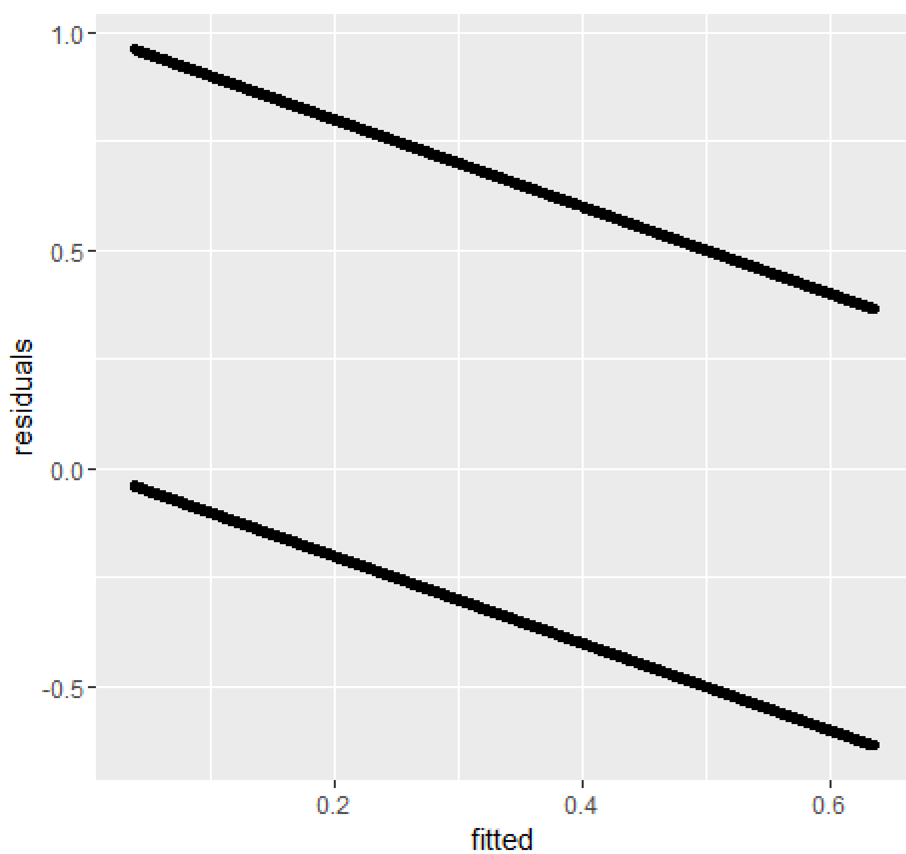
```
Loading required package: ggplot2
> library(e1071)
> confusionMatrix(pred,test_$Y)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2170  397
         1  170  263

               Accuracy : 0.811
                 95% CI : (0.7965, 0.8249)
    No Information Rate : 0.78
    P-Value [Acc > NIR] : 1.691e-05

                  Kappa : 0.3717
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9274
            Specificity : 0.3985
         Pos Pred Value : 0.8453
         Neg Pred Value : 0.6074
             Prevalence : 0.7800
         Detection Rate : 0.7233
   Detection Prevalence : 0.8557
      Balanced Accuracy : 0.6629

       'Positive' Class : 0
```

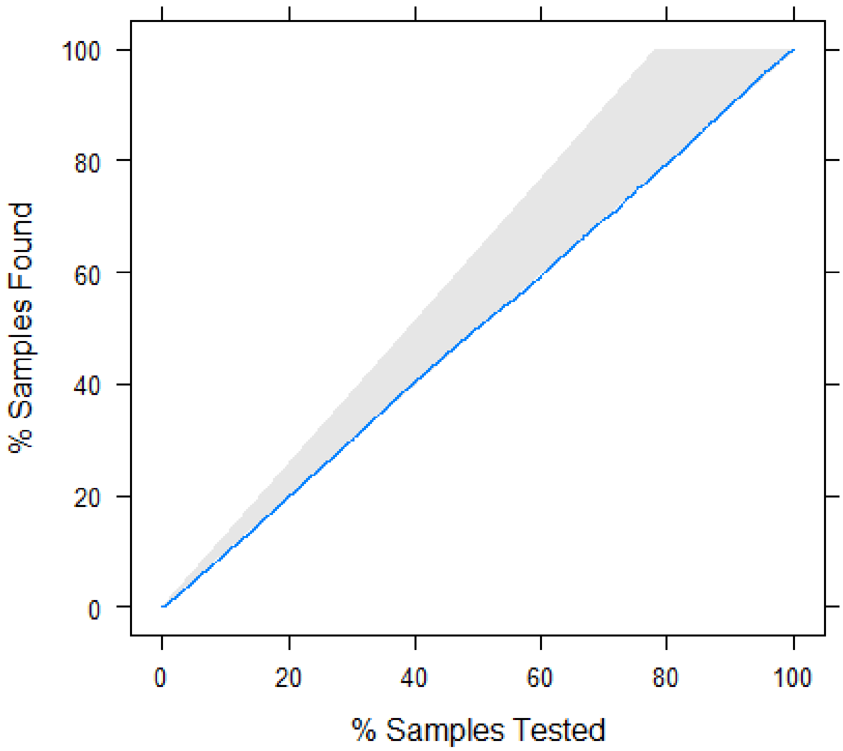


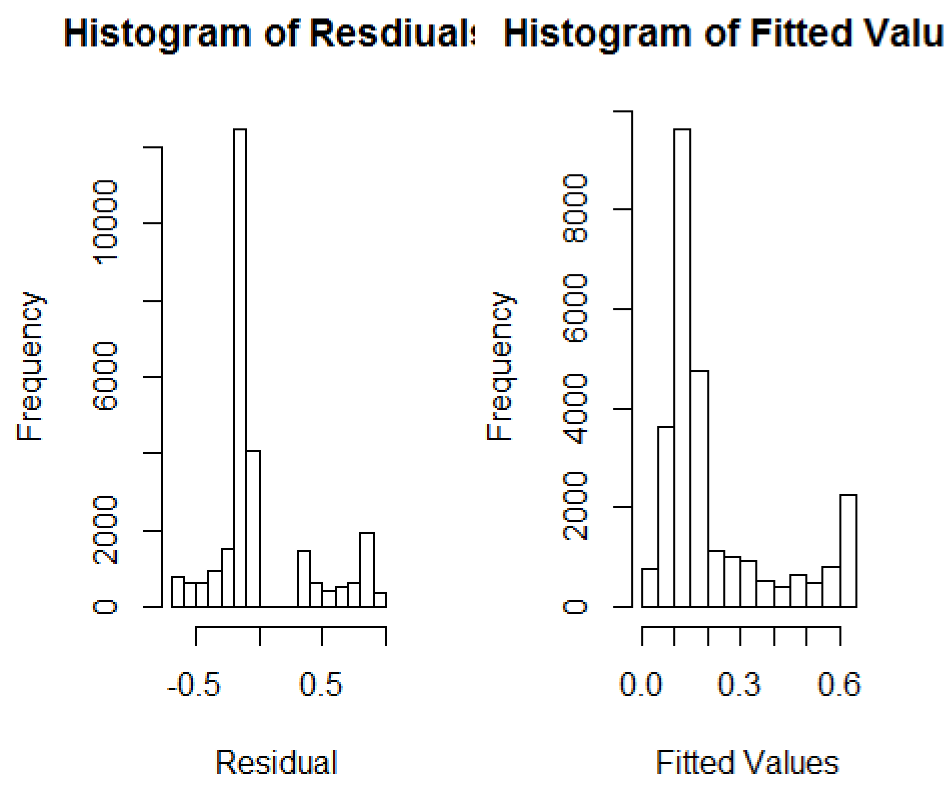

AVG_REPAY_OVERDUE < 0.0416667
0
0
0
1


B1
B2
I1
I2
I3
I4
I5
I6
I7
H1
H2
H3
H4
H5
H6
H7
H8
H9
H10
H11
H12
H13
H14
H15
O1
Y
SEX
AGE


... Precision/Recall graphs ...
Average precision
Average recall
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
0.04
0.19
0.34
0.49
0.63


residuals
fitted
1.0
0.5
0.0
-0.5
0.2
0.4
0.6


100
80
60
40
20
0
% Samples Found
0
20
40
60
80
100
% Samples Tested



Classification Tree

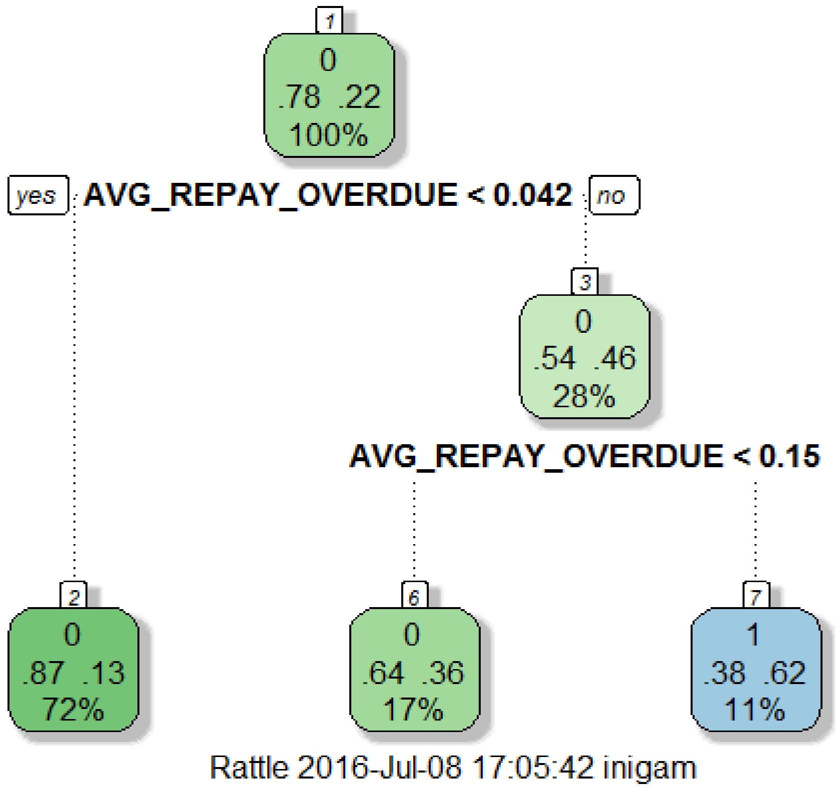
1. To go ahead with classification model we first cleaned the data. Removed the outliers. Plotted various graphs such as Histograms, Box plot to maintain the integrity of data.
2. Next we sampled the data into Test and Train to model the train data and then evaluate our model on based of Test data set.
3. In classification we did modeling with variables all variables
4. Graphs were plotted to evaluate the performance of the models

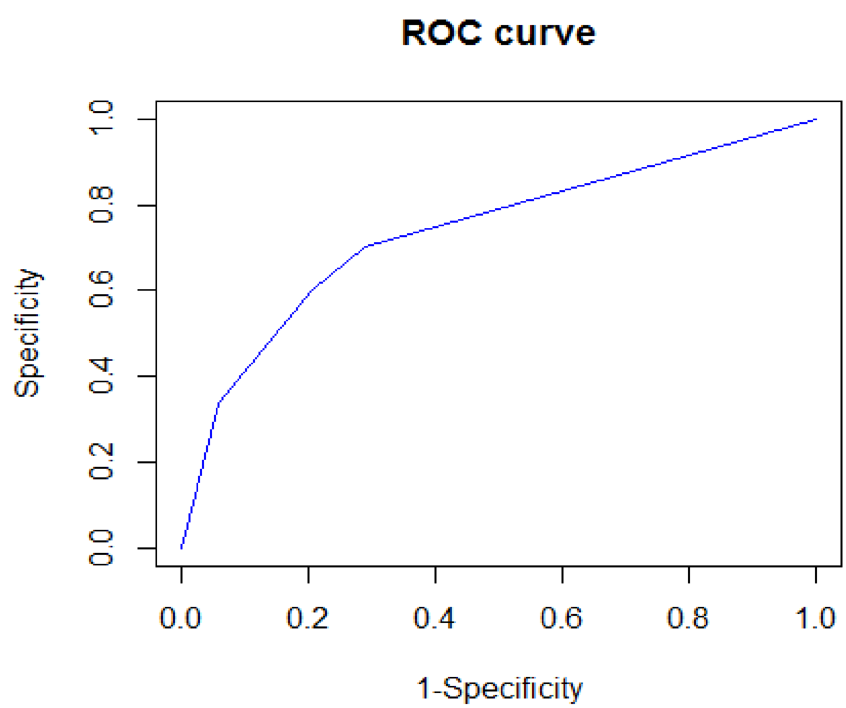
```
library(tree)
tree.train = tree(Y~.,data=train_)
summary(tree.train)

#Display the tree structure and node labels
plot(tree.train)
text(tree.train, pretty =0) #Pretty=0 includes the category names
```

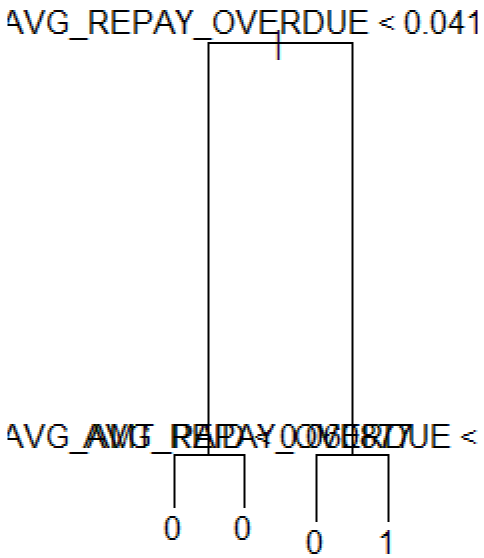
```
> # ROC area under the curve
> tree.pred = predict(tree.train, test_, type ="vector")
> prediction <- prediction(tree.pred[,2],test_$Y)
> auc.tmp <- performance(prediction,"auc")
> auc <- as.numeric(auc.tmp@y.values)
> print(auc)
[1] 0.7388666
> confusionMatrix(tree.pred,test_$Y)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2208  437
         1  132  223

               Accuracy : 0.8103
                 95% CI : (0.7958, 0.8242)
    No Information Rate : 0.78
    P-Value [Acc > NIR] : 2.511e-05

                  Kappa : 0.3374
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9436
            Specificity : 0.3379
         Pos Pred Value : 0.8348
         Neg Pred Value : 0.6282
             Prevalence : 0.7800
         Detection Rate : 0.7360
   Detection Prevalence : 0.8817
      Balanced Accuracy : 0.6407

       'Positive' Class : 0
```

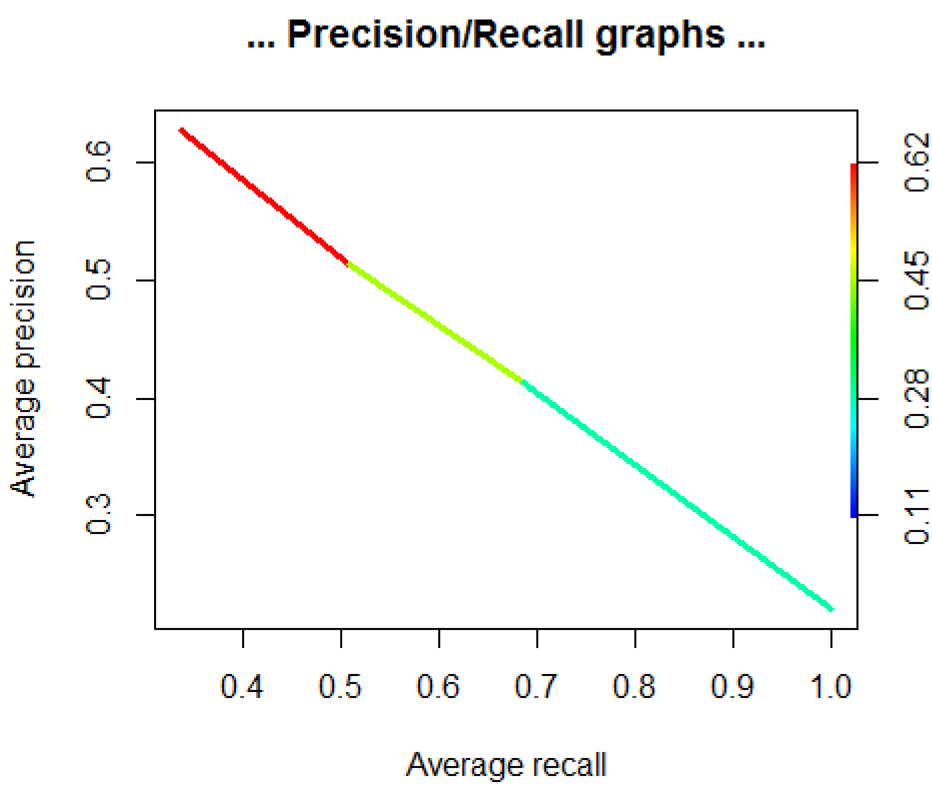
```
tree.pred    0    1
        0 2208  437
        1  132  223
```

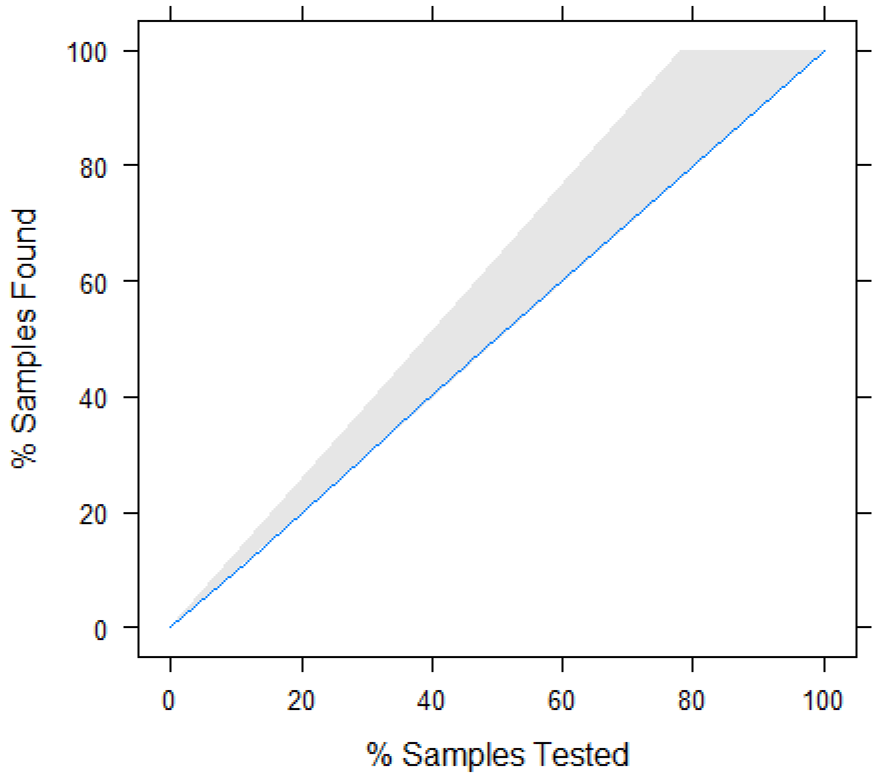
```
Classification tree:
tree(formula = Y ~ ., data = train_)
Variables actually used in tree construction:
[1] "AVG_REPAY_OVERDUE" "AVG_AMT_PAID"
Number of terminal nodes:  4
Residual mean deviance:  0.902 = 24350 / 27000
Misclassification error rate: 0.1939 = 5236 / 27000
>
```


1
0
.78 .22
100%
yes
AVG_REPAY_OVERDUE < 0.042
no
3
0
.54 .46
28%
AVG_REPAY_OVERDUE < 0.15
2
0
.87 .13
72%
6
0
.64 .36
17%
7
1
.38 .62
11%
Rattle 2016-Jul-08 17:05:42 inigam


ROC curve
Specificity
0.0
0.2
0.4
0.6
0.8
1.0
1-Specificity


AVG_REPAY_OVERDUE < 0.041
0
0
0
1


... Precision/Recall graphs ...
Average precision
0.3
0.4
0.5
0.6
Average recall
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0.11
0.28
0.45
0.62



Discussion

- ✓ **With the results derived above we can conclude that Neural Network has best Performance metrics, the sensitivity and specificity hold best results along with high range of accuracy**
- ✓ **Also, the area under ROC curve and lift curve analysis supports neural network model the most accurate model on given dataset.**
- ✓ **In neural network, the prediction of default payment for next month was predicted agrees with the default payment for next month provided with dataset, can be proved with confusion matrix.**

***

Problem2- Advertisements

GOAL

1. The dataset contains a set of advertisements found on Internet. These advertisements are mostly described in image geometries as well as phrases contained in the URL, the image's URL, alt text, the anchor text, and places near the anchor text. The dataset is a two-class data set. In other word, its class attribute is Boolean, which is either "ad" or "nonad", corresponds to "It is an advertisement" and "It is not an advertisement". There are 3279 instances, where 2821 instances are "nonads" and 458 instances are "ads". There are 1558 attributes, where 3 attributes are continuous and others are binary.
2. Our Goal is to clean the data, process it to make sure the outliers are removed and then apply logistic regression, neural network and classification tree to predict and classify the model
3. We have to build different models to evaluate different trends and compare the sensitivity, Specification and plot confusion matrix, ROC Curve and Lift Charts for all the classification method.

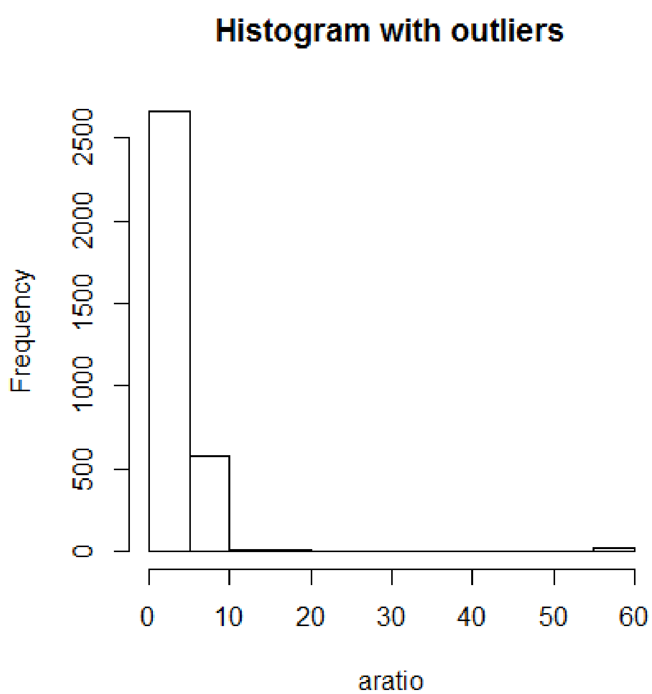
Data Cleaning

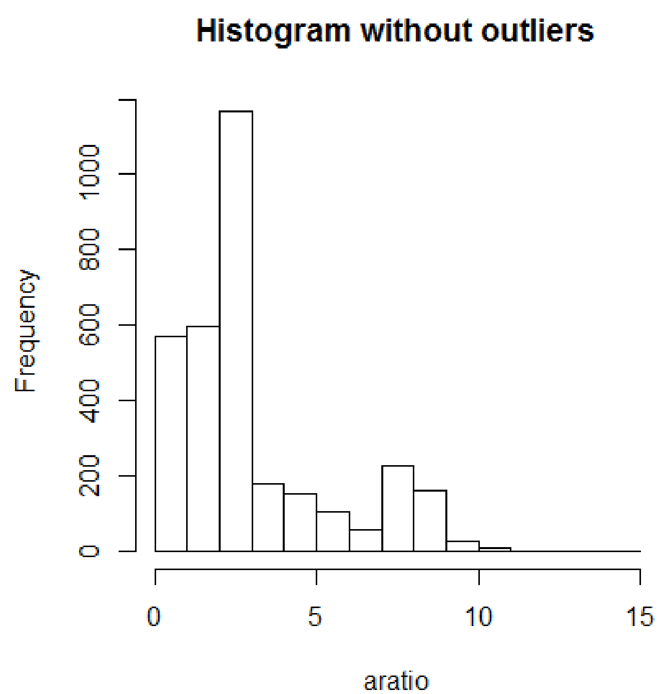
1. It was observed 28% of continuous data was missing which are dimensions of advertisements and area ratio. There are various results to fill the missing data.
   - ✓ Replace by mean
   - ✓ Replace by median
   - ✓ Replace with the previous value using locf

   Approach:- Replacing the values with locf will not make sense as previous values has no relation with each other as each ad is different. Replacing the values with mean or median works best so as to make sure the data set is concentrated towards a common area. We choose mean to replace it and plotted histogram.

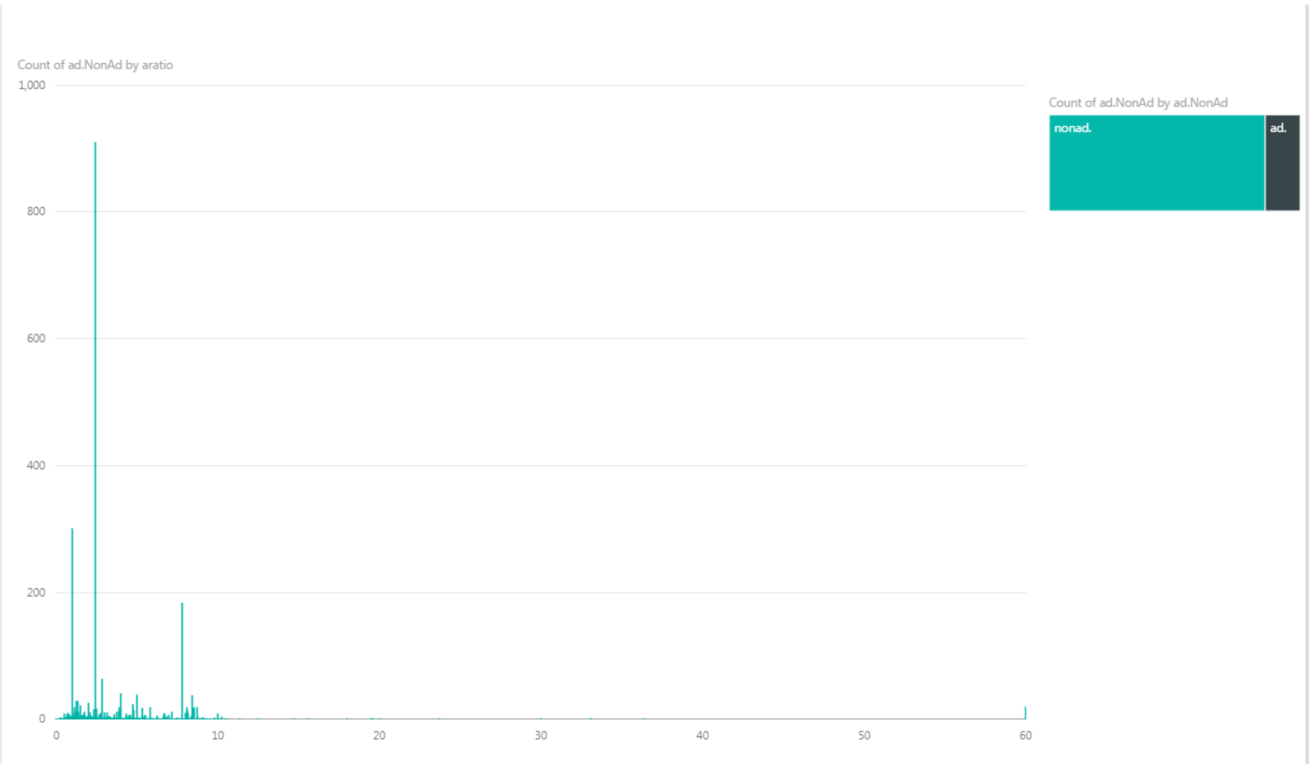
2. Removing the outliers: - Plotted histogram as shown below to see the range to data where they lie. We choose frequency of aspect ratio as a base as it is calculated using height and width. From the histogram as well as from power BI, it's clear that the data is more frequently occurring with aspect ratio of less than 12. So we remove the data which had aspect ratio greater than 12.

```
#Replace ? with NA
addata$height[addata$height == "   ?"] <- NA
addata$width[addata$width == "   ?"] <- NA
addata$aratio[addata$aratio == "     ?"] <- NA

#Convert charcters to numerics to calculate mean
addata$ad.NonAd <- as.factor(addata$ad.NonAd)
addata$height <- as.numeric(addata$height)
addata$width <- as.numeric(addata$width)
addata$aratio <- as.numeric(addata$aratio)

#calculating mean
meanHeight <- mean(addata$height, na.rm=TRUE)
meanWidth <- mean(addata$width, na.rm=TRUE)

#replacing missing data with mean
addata$height[is.na(addata$height)] = meanHeight
addata$width[is.na(addata$width)] = meanWidth
addata$aratio[is.na(addata$aratio)] = meanWidth/meanHeight
```

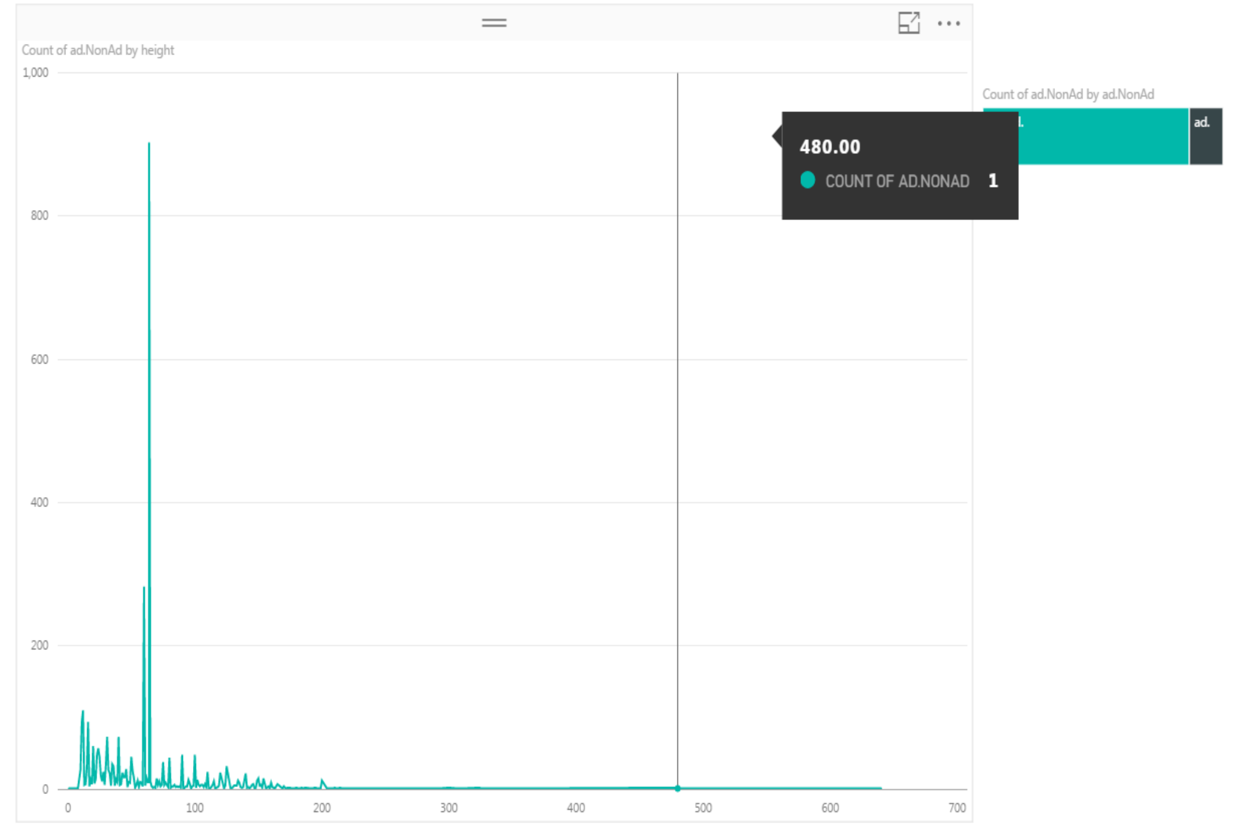




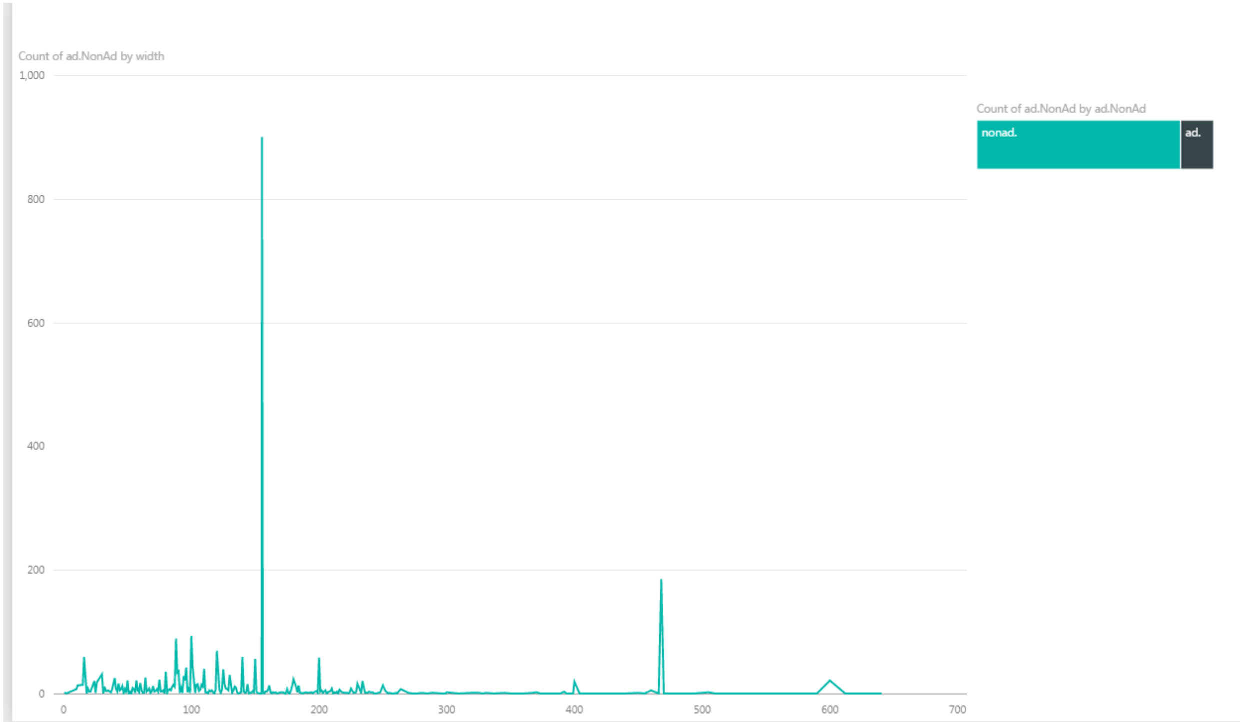
Power PI and Inferences



- ✓ *The above graph shows count of Non-Ads/Ads by Area Ratio. We can depict that Non add whose total count is 2820 and major of the non-ads have high are ratio less than 12 and all of the Ads lie in area ratio below 12.*



- ✓ *We can conclude that almost all non-ads also lie with under height of 200 and Ads have kind of constant height throughout so we can conclude that almost all the Ads have same height.*



- ✓ *We can observe that all Ads have variety of distribution when it comes to height but in case of widths it has distribution throughout.*

- ✓ ***Using tool Power BI we can conclude the with height, width and area ratio which are continuous data set in our problem, we can conclude that all the Ads have area ratio less than 12. So we can consider aspect ratio to predict whether an Advertisement is Ad or Non-Ad***

Logistic Regression

1. To go ahead with logistic regression model we first cleaned the data. Next we sampled the data into Test and Train to model the train data and then evaluate our model on based of Test data set. We have taken 75% train data but according to a research study, its better to take 90% train data and 10% test data as we don't want a ad to be classified as non-ad as it is very signification according to business scenario.
2. In Logistic regression we did GLM modeling with variables : url.ad+alt.click+url.images.home+height+ancurl.com+ ancurl.exe+url.ads+alt.net+width+ancurl.click+ancurl.http.www+aratio
3. Threshold of 0.5 probability was set to classify data into add or non-ad and predictions were made as per the analysis.
4. Graphs were plotted to evaluate the performance of the models

```
#Sampling the data
smp_size <- floor(0.75 * nrow(addata))
set.seed(123)
train_ind <- sample(seq_len(nrow(addata)), size = smp_size)

train <- addata[train_ind, ]
test <- addata[-train_ind, ]

#Build Logistic Regression
glm.fit <- glm(ad.NonAd~url.ad+alt.click+url.images.home+height+ancurl.com+
                 ancurl.exe+url.ads+alt.net+width+ancurl.click+ancurl.http.www,
               data=train, family=binomial(link="logit"))
summary(glm.fit)

prob <- predict(glm.fit, newdata=test, type="response")
pred <- rep("nonad.",length(prob))
pred[prob<=0.5] <- "ad."
```

```
> confusionMatrix(pred,test$ad.NonAd)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.    107      8
    nonad.  15    682

               Accuracy : 0.9716749
                 95% CI : (0.9578003, 0.9819614)
    No Information Rate : 0.8497537
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.886388
 Mcnemar's Test P-Value : 0.2109029

            Sensitivity : 0.8770492
            Specificity : 0.9884058
         Pos Pred Value : 0.9304348
         Neg Pred Value : 0.9784792
             Prevalence : 0.1502463
         Detection Rate : 0.1317734
   Detection Prevalence : 0.1416256
      Balanced Accuracy : 0.9327275

       'Positive' Class : ad.
```

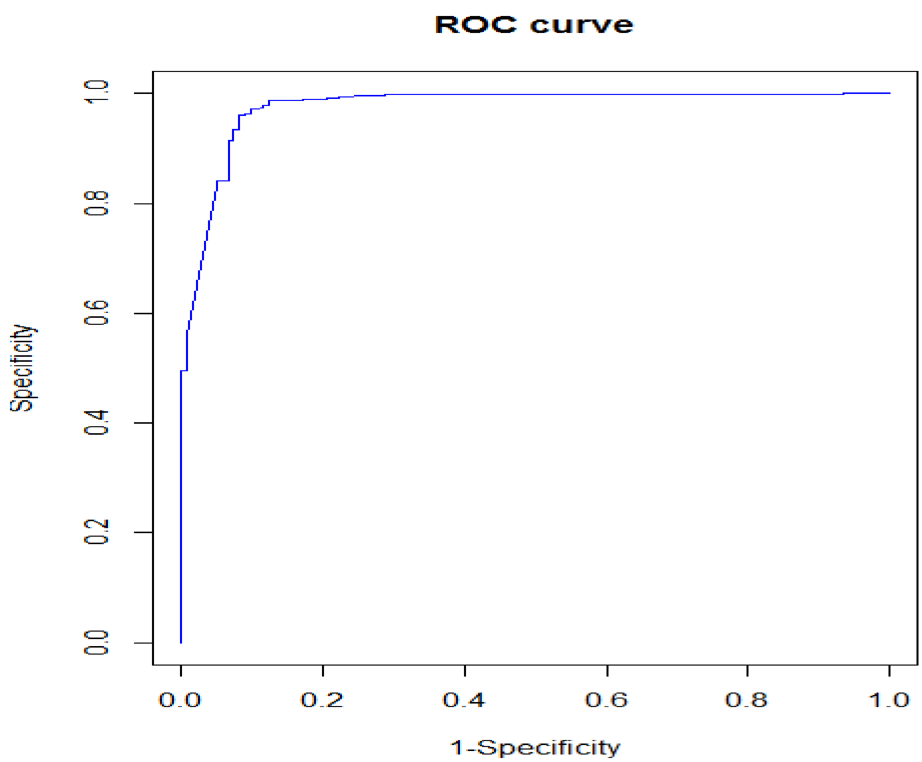
AUC:-

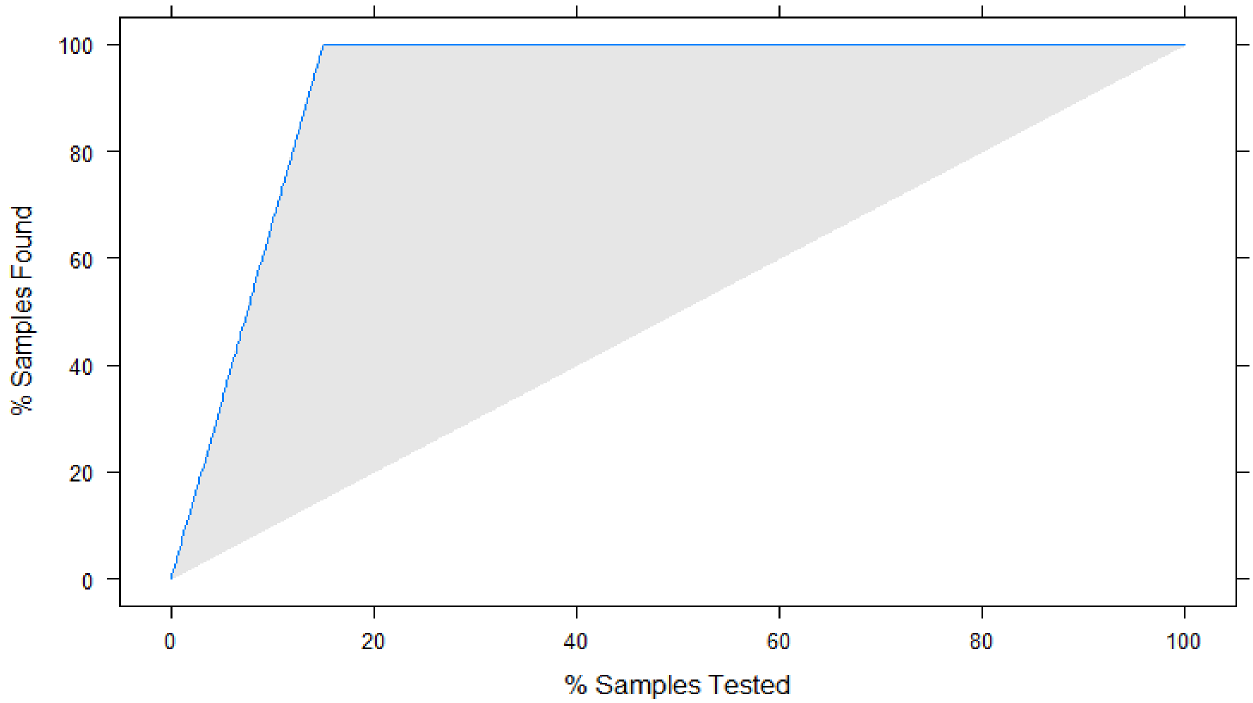
From a random classifier you can expect as many true positives as false positives. That's the dashed line on the plot. AUC score for the case is 0.5. A score for a perfect classifier would be 1.

```
> # ROC area under the curve
> prediction <- prediction(prob,test$ad.NonAd)
> auc.tmp <- performance(prediction,"auc")
> auc <- as.numeric(auc.tmp@y.values)
> print(auc)
[1] 0.9760572583
```

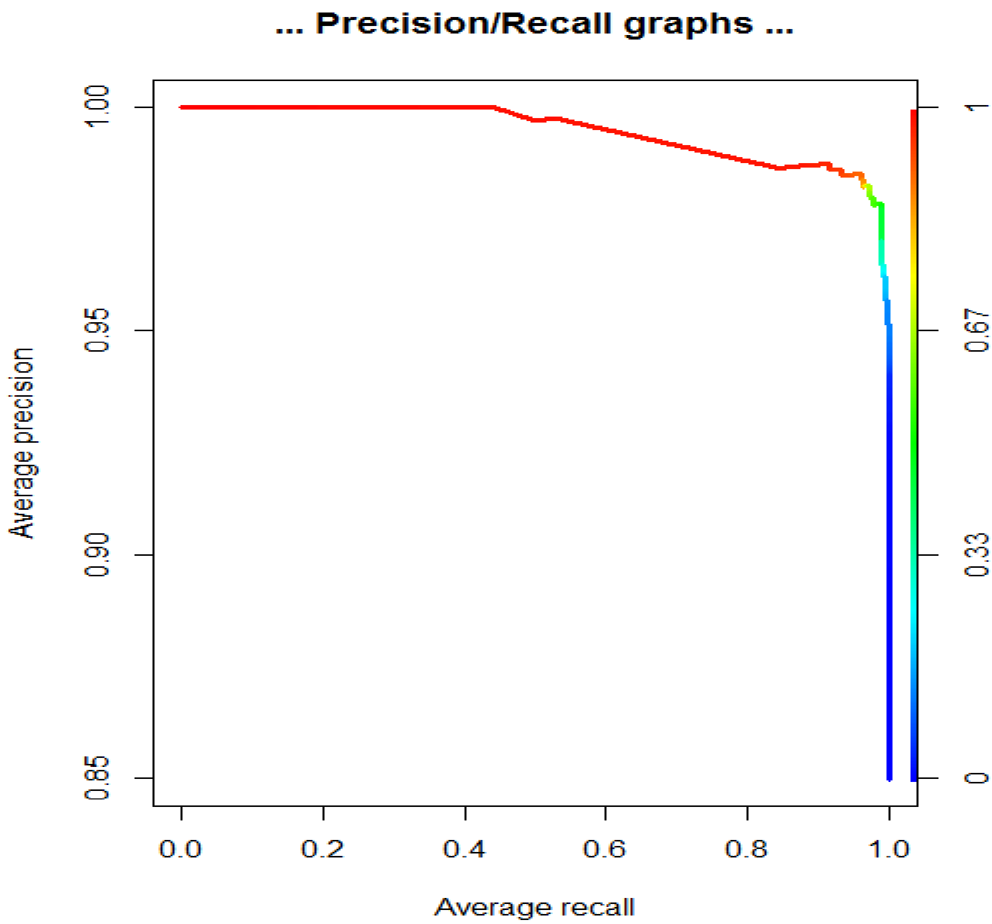
ROC:-

In statistics, a **receiver operating characteristic** (**ROC**), or **ROC curve**, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, or recall in machine learning. The false-positive rate is also known as the fall-out and can be calculated as (1 - specificity). **An ideal ROC is the one which is closer to top left corner as below.**





100
80
60
40
20
0
% Samples Found
0
20
40
60
80
100
% Samples Tested



Classification Tree

1. To go ahead with classification model we first cleaned the data. Removed the outliers. Plotted various graphs such as Histograms, Box plot to maintain the integrity of data.
2. Next we sampled the data into Test and Train to model the train data and then evaluate our model on based of Test data set.
3. In classification we did modeling with variables all variables
4. Graphs were plotted to evaluate the performance of the models

```
#Use variables to fit a classification tree
library(tree)
tree.train = tree(ad.NonAd~.,data=train)
summary(tree.train)


#Display the tree structure and node labels
plot(tree.train)
text(tree.train, pretty =0) #Pretty=0 includes the category names

#FancyRPlot
library(rpart)
library(rattle)
tree <- rpart(ad.NonAd~.,data=train,method="class")
fancyRpartPlot(tree)

tree.pred = predict(tree.train, test, type ="class")
table(tree.pred, test$ad.NonAd)
```

```
> confusionMatrix(tree.pred,test$ad.NonAd)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.    106     16
    nonad.  16    674

               Accuracy : 0.9605911
                 95% CI : (0.9448196, 0.972891)
    No Information Rate : 0.8497537
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.8456641
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8688525
            Specificity : 0.9768116
         Pos Pred Value : 0.8688525
         Neg Pred Value : 0.9768116
             Prevalence : 0.1502463
         Detection Rate : 0.1305419
   Detection Prevalence : 0.1502463
      Balanced Accuracy : 0.9228320

       'Positive' Class : ad.
```

```
> summary(tree.train)

Classification tree:
tree(formula = ad.NonAd ~ ., data = train)
Variables actually used in tree construction:
 [1] "width"           "ancurl.com"      "url.ads"         "ancurl.click"    "url.ad"          "alt.click"
 [7] "url.images.home" "alt.net"         "aratio"          "height"
Number of terminal nodes:  12
Residual mean deviance:  0.2214256 = 536.7356 / 2424
Misclassification error rate: 0.02873563 = 70 / 2436
```

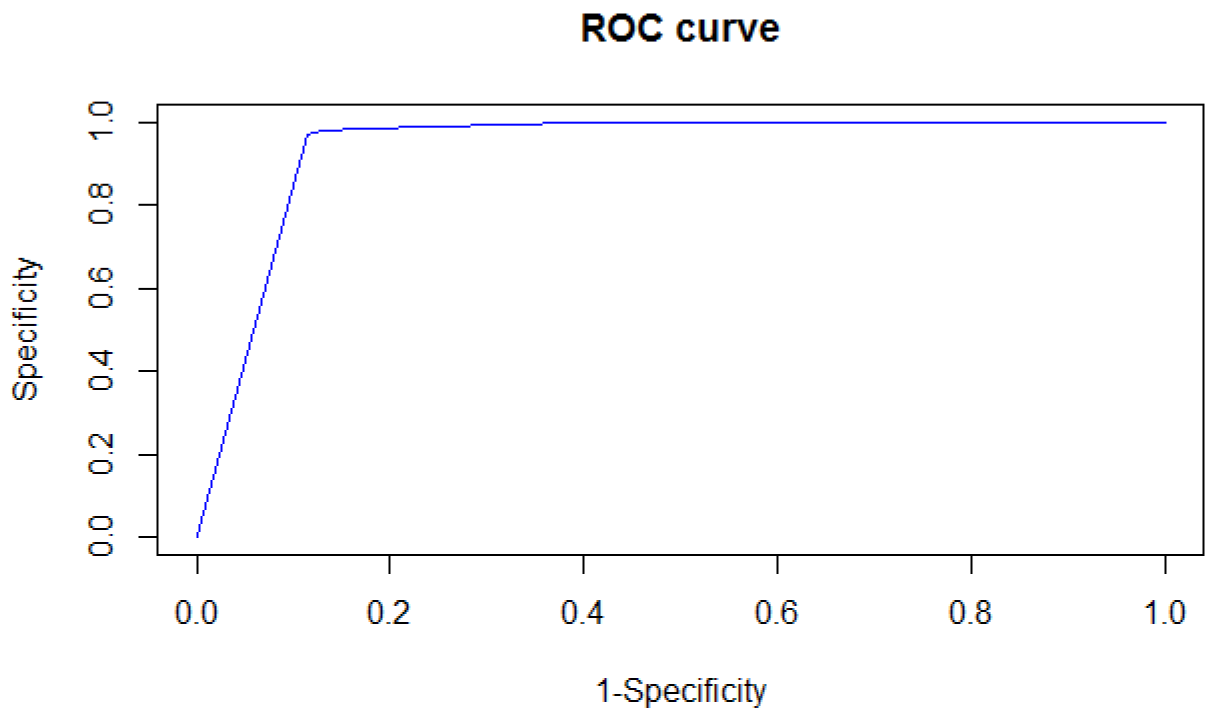
AUC:-

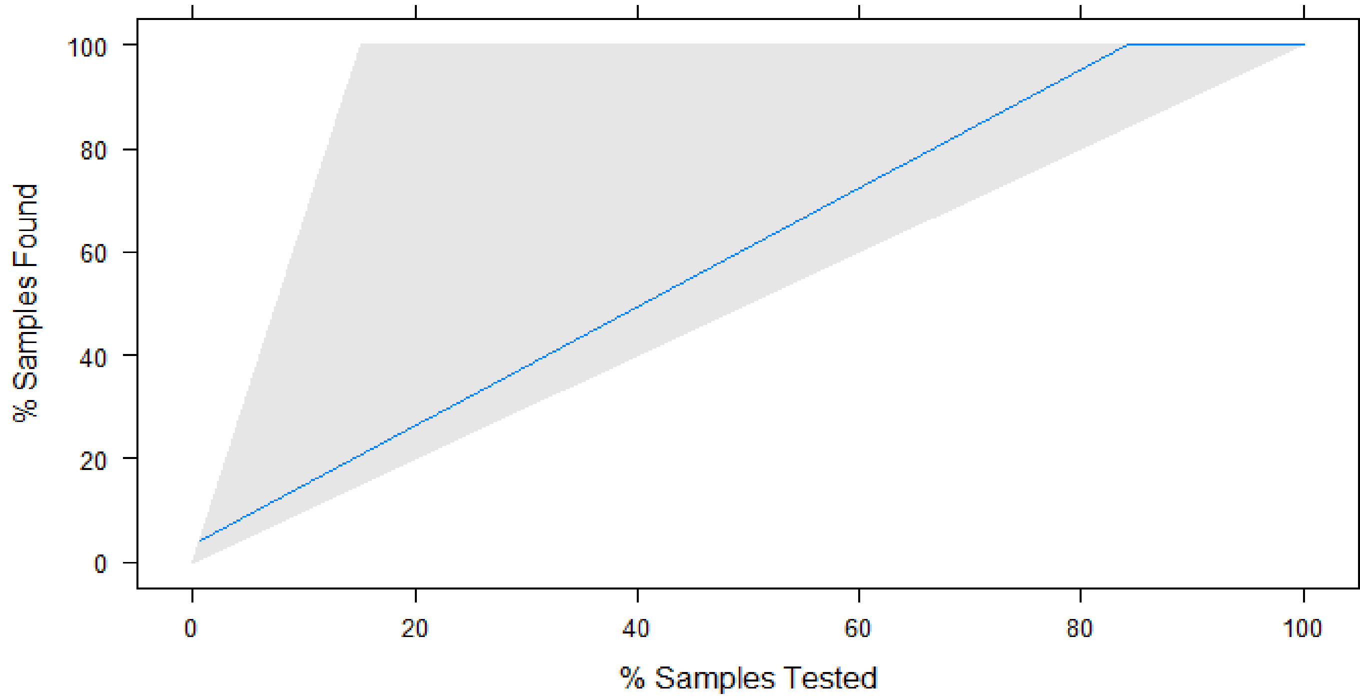
From a random classifier you can expect as many true positives as false positives. That's the dashed line on the plot. AUC score for the case is 0.5. A score for a perfect classifier would be 1.

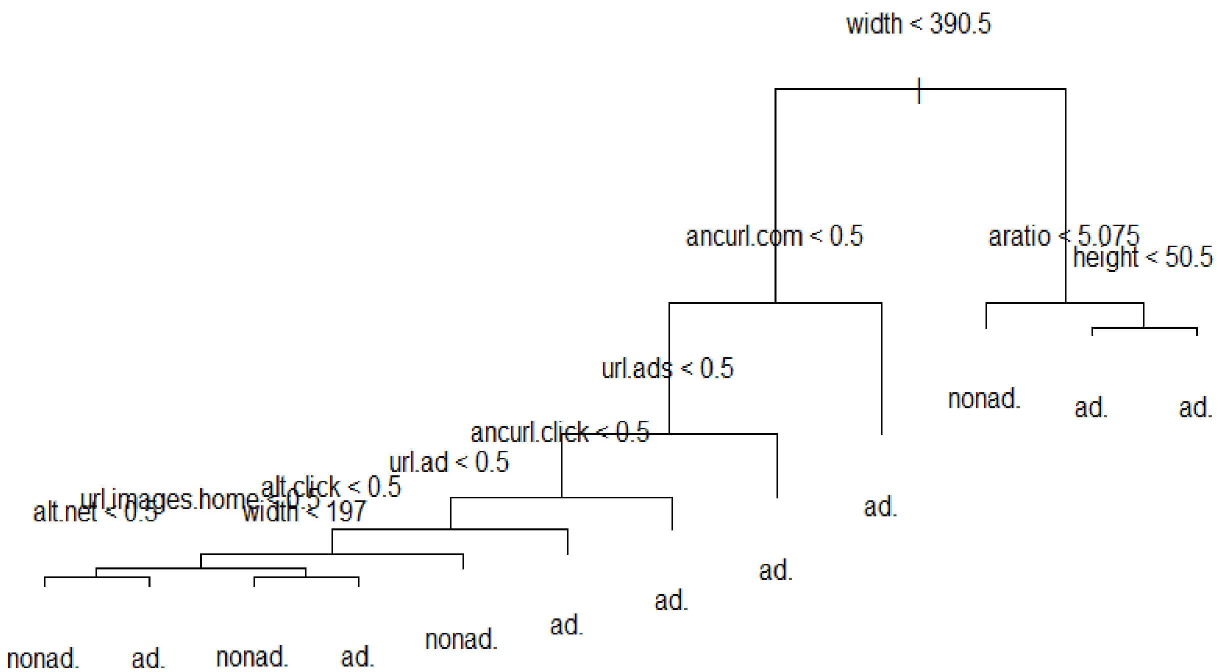
```
> prediction <- prediction(tree.pred[,2],test$ad.NonAd)
> auc.tmp <- performance(prediction,"auc")
> auc <- as.numeric(auc.tmp@y.values)
> print(auc)
[1] 0.9377464956
```

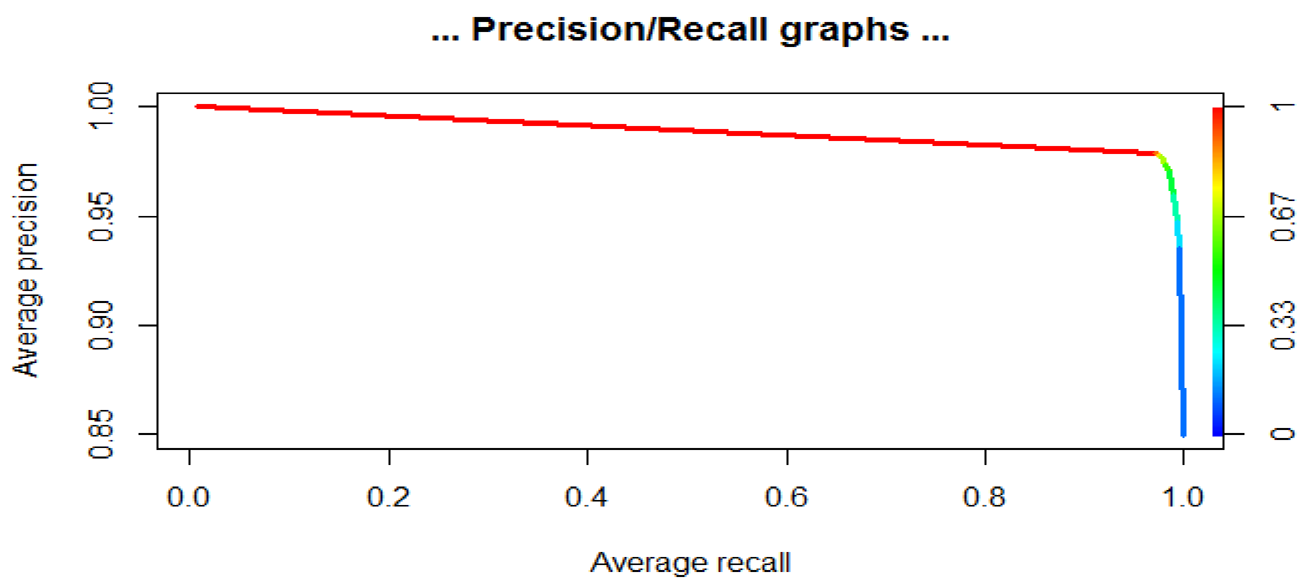
ROC:-

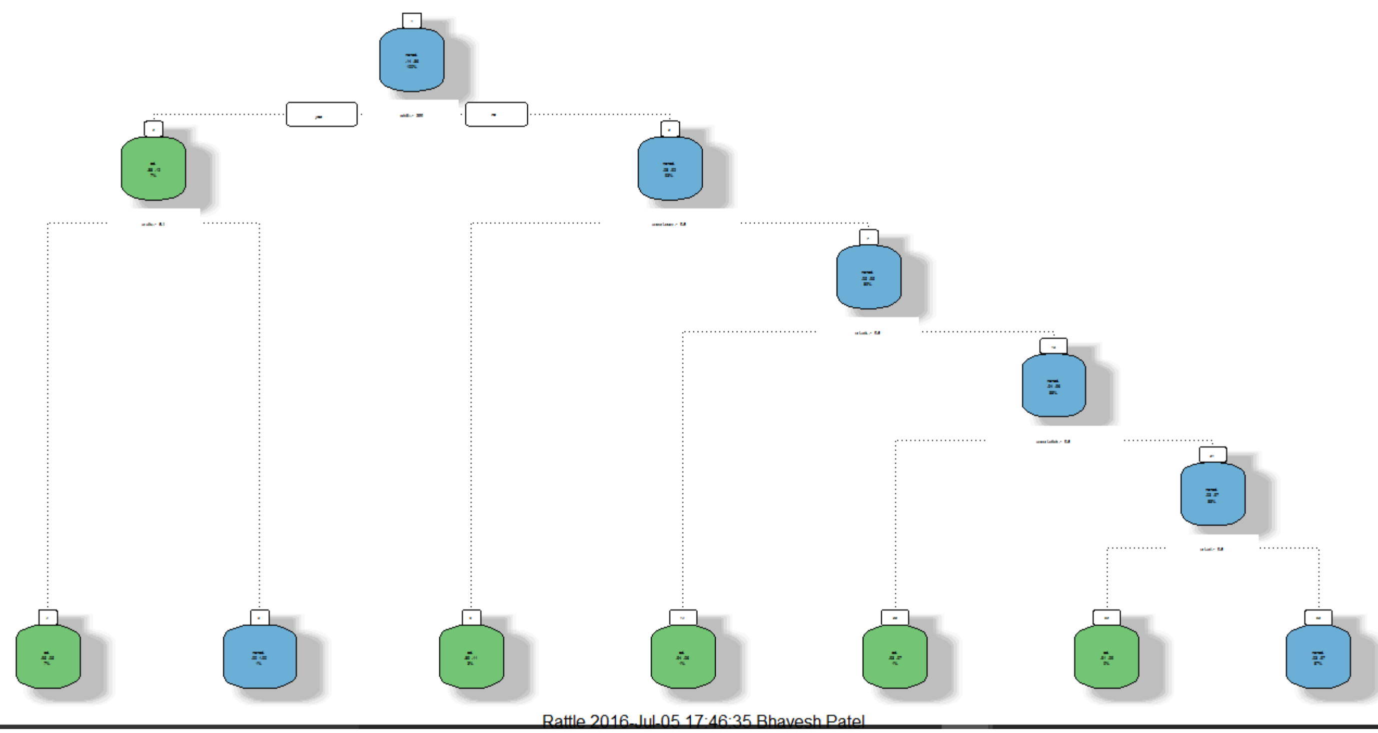
In statistics, a **receiver operating characteristic** (**ROC**), or **ROC curve**, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, or recall in machine learning. The false-positive rate is also known as the fall-out and can be calculated as (1 - specificity). **An ideal ROC is the one which is closer to top left corner as below.**


ROC curve
Specificity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
1-Specificity


% Samples Found
% Samples Tested
0
20
40
60
80
100


width < 390.5
ancurl.com < 0.5
aratio < 5.075
height < 50.5
url.ads < 0.5
ancurl.click < 0.5
url.ad < 0.5
alt.click < 0.5
url.images.home < 0.5
width < 197
alt.net < 0.5
nonad.
ad.
nonad.
ad.
nonad.
ad.
ad.
ad.
ad.
nonad.
ad.
ad.


... Precision/Recall graphs ...
Average precision
1.00
0.95
0.90
0.85
0.0
0.2
0.4
0.6
0.8
1.0
Average recall
1
0.67
0.33
0


Rattle 2016-Jul-05 17:46:35 Bhavesh Patel

Neural Networks

1. To go ahead with Neural Networks model we first cleaned the data. Removed the outliers. Plotted various graphs such as Histograms, Box plot to maintain the integrity of data.
2. In Neural Networks we did modeling with variables : url.ad+alt.click+url.images.home+ancurl.com+height+ancurl.exe+ url.ads+alt.net+width+ancurl.click+ancurl.http.www
3. Graphs were plotted to evaluate the performance of the models

```
#Neural Network
library(nnet)
library(NeuralNetTools)

#Build the model
fitnn <- nnet(ad.NonAd~url.ad+alt.click+url.images.home+ancurl.com+height+ancurl.exe+
                url.ads+alt.net+width+ancurl.click+ancurl.http.www,data=train, size=4,
               hess = T, dk=15e-4, maxit = 200)
summary(fitnn)

#Predict for test data
pred = predict(fitnn, newdata=test, type="class")

#Confusion Matrix
library(caret)
library(e1071)
confusionMatrix(pred,test$ad.NonAd)
```

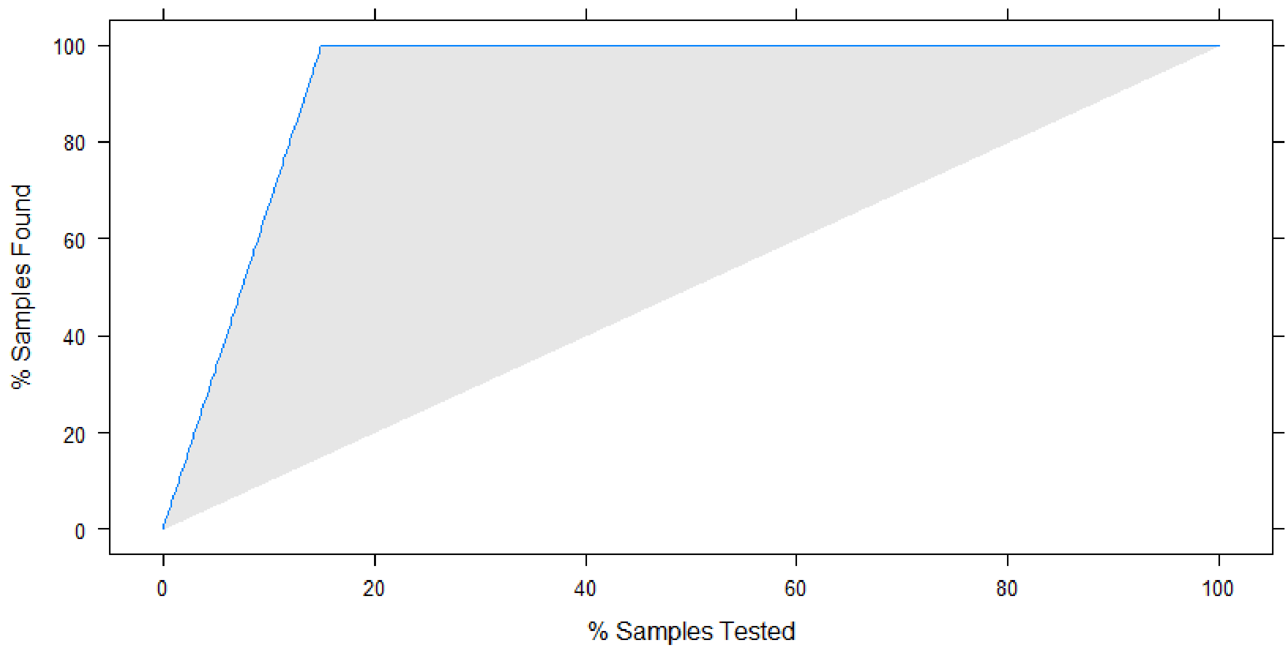
```
> summary(fitnn)
a 11-4-1 network with 53 weights
options were - entropy fitting
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1
  -2.27   -2.44   -0.62    0.42    0.23   -5.11   -0.48   -0.62   -0.68   -4.07   -0.33   -0.62
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2
   4.32   -8.57   -3.82   -3.28  -50.34    0.03   -5.47  -23.38    6.46   -0.02  -40.14  -21.67
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3 i10->h3 i11->h3
  -2.04    0.01    0.66   -0.42   -0.09   -1.97   -0.65   -0.56   -0.11   -2.38   -0.66   -0.46
  b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i9->h4 i10->h4 i11->h4
  -4.21   11.62   -0.23   -0.01    0.35   -1.06    0.48   -0.25   -0.37   -2.28    0.41   -0.34
  b->o  h1->o  h2->o  h3->o  h4->o
 -2.90  -1.70   7.03  -2.24 -11.55
```

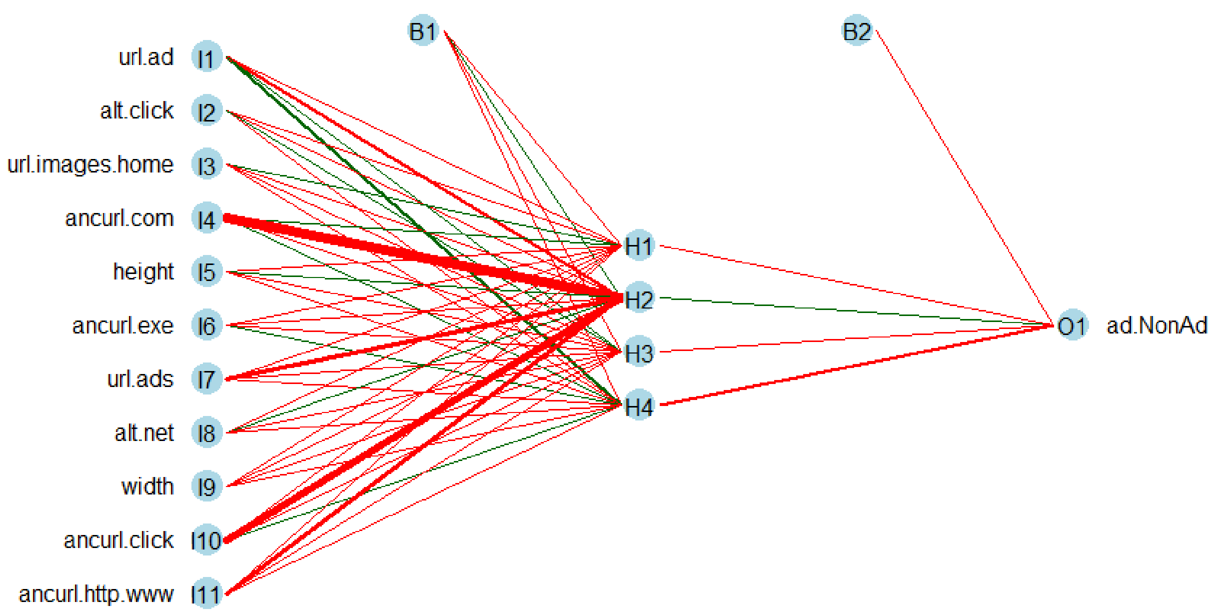
AUC:-

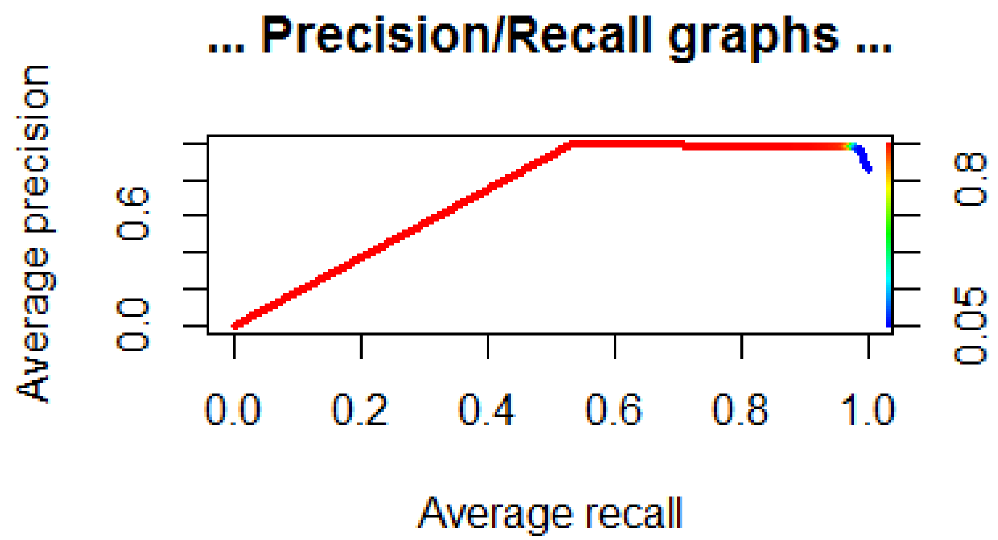
From a random classifier you can expect as many true positives as false positives. That's the dashed line on the plot. AUC score for the case is 0.5. A score for a perfect classifier would be 1.

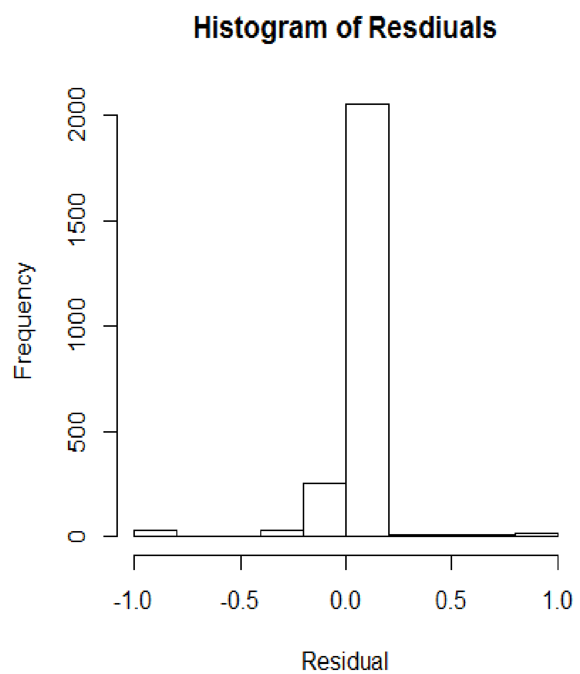
```
> # ROC area under the curve
> pred = predict(fitnn, newdata=test, type="raw")
> prediction <- prediction(pred,test$ad.NonAd)
> auc.tmp <- performance(prediction,"auc")
> auc <- as.numeric(auc.tmp@y.values)
> print(auc)
[1] 0.9602934189
```

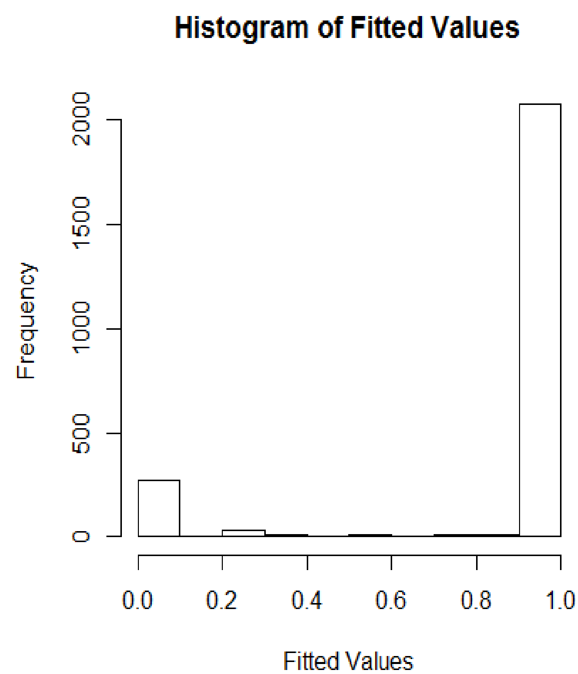
```
> confusionMatrix(pred,test$ad.NonAd)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.    109     17
    nonad.  13    673

               Accuracy : 0.9630542
                 95% CI : (0.9476756, 0.9749366)
    No Information Rate : 0.8497537
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.8572366
 Mcnemar's Test P-Value : 0.5838824

            Sensitivity : 0.8934426
            Specificity : 0.9753623
         Pos Pred Value : 0.8650794
         Neg Pred Value : 0.9810496
             Prevalence : 0.1502463
         Detection Rate : 0.1342365
   Detection Prevalence : 0.1551724
      Balanced Accuracy : 0.9344025

       'Positive' Class : ad.
```

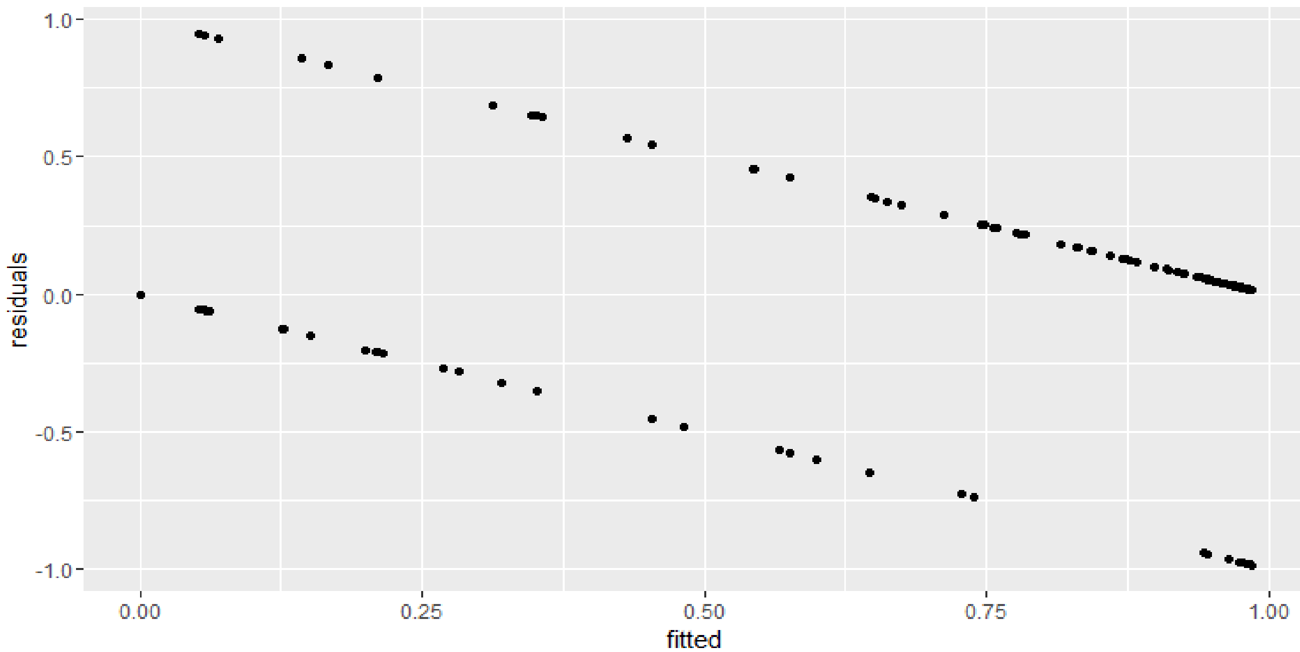

% Samples Found
% Samples Tested
0
20
40
60
80
100


B1
B2
url.ad
alt.click
url.images.home
ancurl.com
height
ancurl.exe
url.ads
alt.net
width
ancurl.click
ancurl.http.www
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
H1
H2
H3
H4
O1
ad.NonAd


... Precision/Recall graphs ...
Average precision
0.0
0.6
0.05
0.8
0.0
0.2
0.4
0.6
0.8
1.0
Average recall


Histogram of Resdiuals
Frequency
0
500
1000
1500
2000
-1.0
-0.5
0.0
0.5
1.0
Residual


Histogram of Fitted Values
Frequency
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0
Fitted Values



Discussion

- ✓ **With the results derived above we can conclude that Classification Tree has good Performance metrics, the sensitivity and specificity hold best results along with high range of accuracy**
- ✓ **Also, the area under ROC curve and lift curve analysis supports Classification Tree the most accurate model on given dataset.**
- ✓ **In Classification Tree, the prediction of ad and non-ad classification is given in the confusion matrix.**

Problem3- Wind Forecasting

GOAL

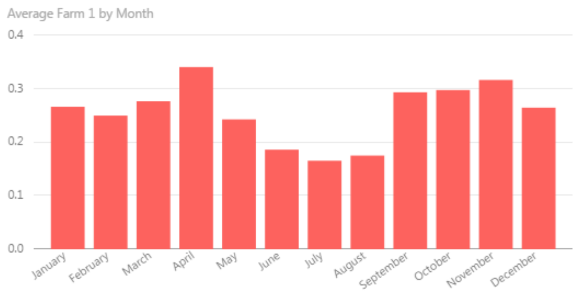
1. The dataset contains wind power forecast value for seven different warms over 4.5 years of date.
2. Our Goal is to clean the data, process it to make sure the data is clean and without outliners. There is test and train data based on date filter. We have to make model using linear regression, neural network and Regression tree to forecast the data in Test data for missing data.
3. We have to build different models to evaluate different trends and compare the performance metric for all the models.

Data Cleaning

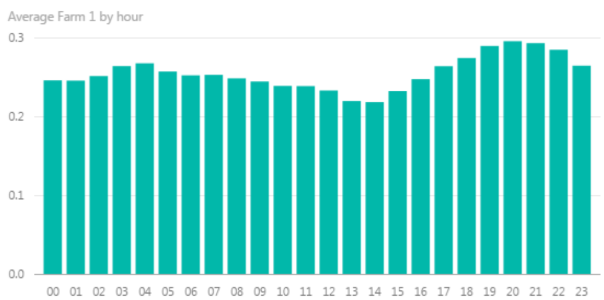
1. It was observed data had date with time stamp and there were multiple entries with same timestamp so we took mean of all the entries for Single time stamp.
2. Histogram was plotted to see outliers and using Cook's Method we removed outliers.
3. The complicated part of data cleaning was we have to separate and refine the data with its time stamp for all the seven farm files and then later we had to merge all the Wp(wind power) into one single file with Timestamps.

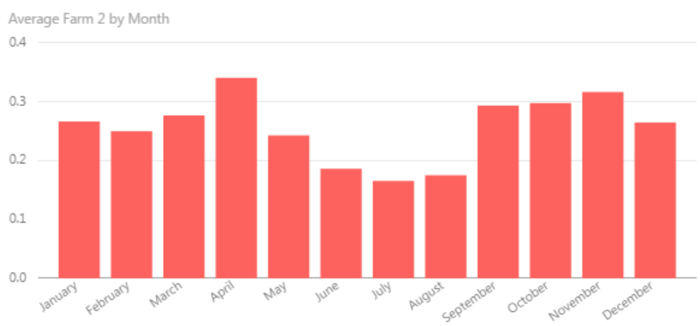
```
for(i in 1:7){

  assign(paste0("wf"), read.csv(paste("windforecasts_wf",
                                      i, ".csv", sep=""),colClasses = c("character", "numeric",

  wf$date <- as.POSIXct(wf$date,format = "%Y%m%d%H")
  wf$actualDate <- wf$date + (wf$hors * 3600)

  wf <- wf[,c(3:7)]
  meanU <- mean(wf$u, na.rm=TRUE)
  meanV <- mean(wf$v, na.rm=TRUE)
  meanWS <- mean(wf$ws, na.rm=TRUE)
  meanWD <- mean(wf$wd, na.rm=TRUE)

  wf$u[is.na(wf$u)] = meanU
  wf$v[is.na(wf$v)] = meanV
  wf$ws[is.na(wf$ws)] = meanWS
  wf$wd[is.na(wf$wd)] = meanWD

  wf <- aggregate(x = wf[c("u","v","ws","wd")],
                  FUN = mean,
                  by = list(Date = wf$actualDate))


  wf$hour <- as.factor(format(wf$Date, "%H"))
  wf$month <- as.factor(format(wf$Date, "%m"))
  wf$year <- as.factor(format(wf$Date, "%Y"))
  wf$OnlyDate <- as.Date(format(wf$Date))

  assign(paste0("wf", i), wf)
}
```

Power B I and Inferences

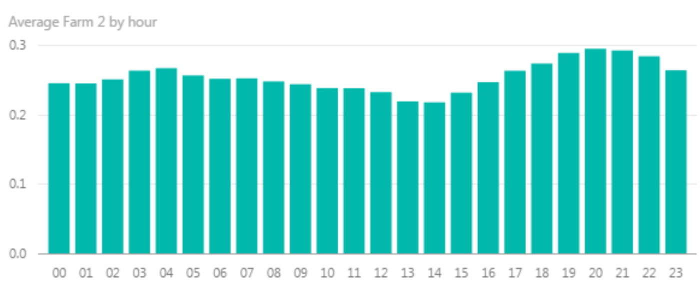


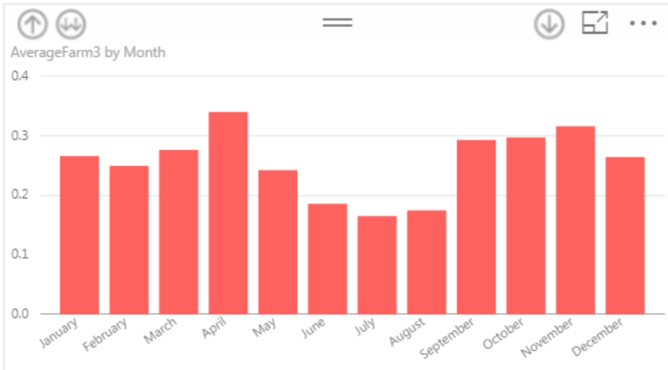
FARM 1





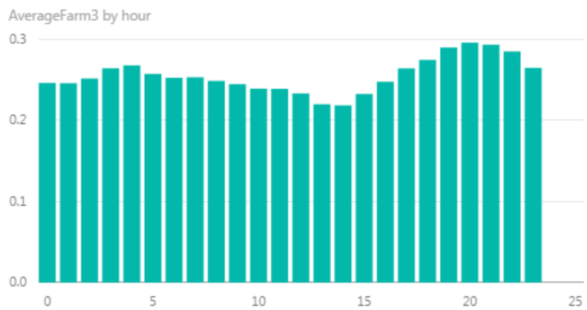
FARM 2



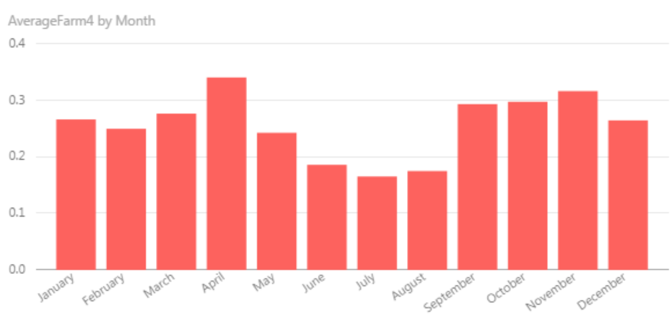


FARM 3

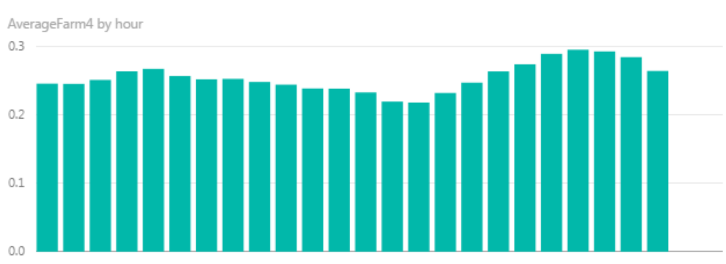
year
- 2009
- 2010
- 2011
- 2012



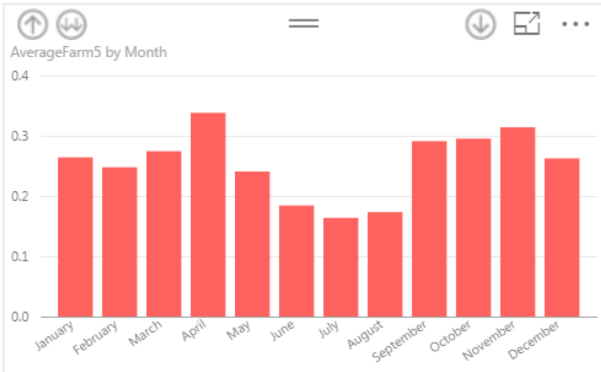
FARM 4



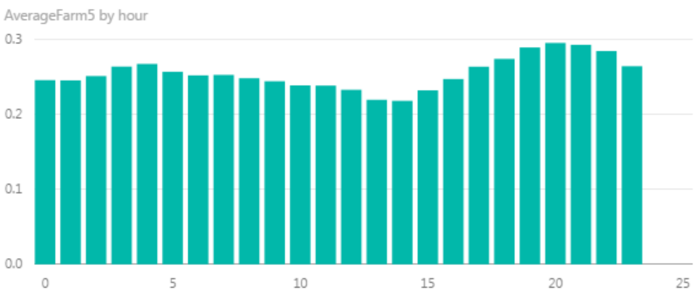
year
- 2009
- 2010
- 2011
- 2012



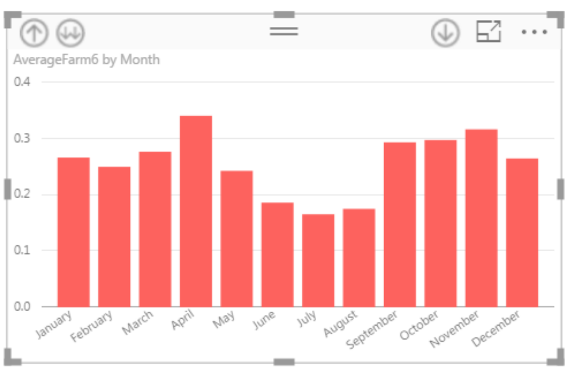
FARM 5



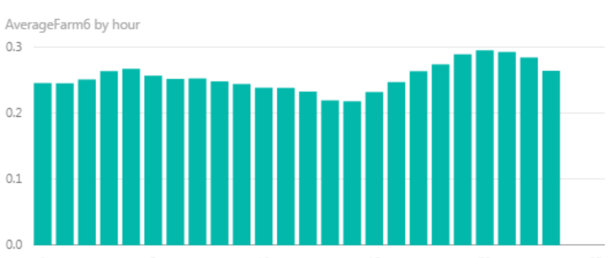
year
- 2009
- 2010
- 2011
- 2012



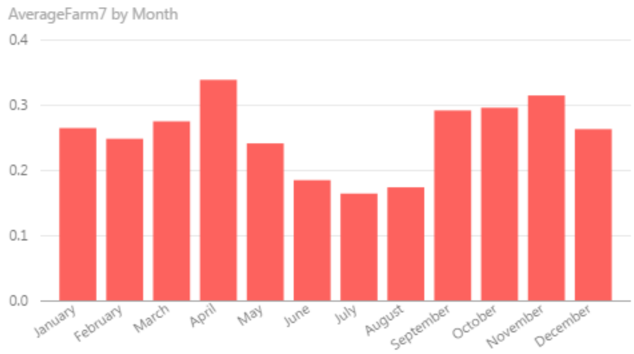
FARM 6



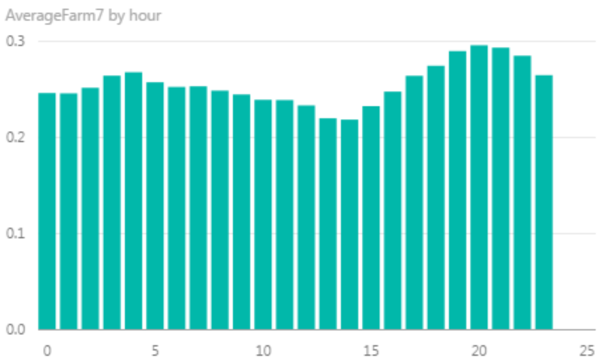
year
- 2009
- 2010
- 2011
- 2012



FARM 7



year
- 2009
- 2010
- 2011
- 2012



- ✓ *As per the above graphs we can see a pattern in the Average wind power to month and Hours for all the 7 farms.*
- ✓ *The Wind power is maximum in April month and in month of May-September the wind power decreases.*
- ✓ *The wind power increases from September to November.*
- ✓ *On per hour basis the wind power is observed to be maximum in evening Hours and decreases around afternoon 1pm to 3pm.*
- ✓ *The wind speed and wind direction and zonal and meridional wind components also have a =n impact on the output which is wind power and temperature, hour, month such factors impact the predictors u,v,ws,wd*

Linear Regression

1. To go ahead with linear regression model we first cleaned the data and removed outliers with cook's method. Next we sampled the data into Test and Train to model the train data and then evaluate our model on based of Test data set.
2. In linear regression we did modeling for all 7 farms and the models for all the farms were different.
3. Feature Selection was done all the farms and model was applied based on the different predictors.
4. Performance metrics and accuracy was calculated for all seven models.

Wind Farm 1

```
lm.fit1=lm(wp1~.-Date-OnlyDate-month-year,data=model1)
summary(lm.fit1)
pred = predict(lm.fit1, test1)
accuracy(pred, test1$wp1)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -7.603e-02  5.239e-03 -14.513  < 2e-16 ***
hour        -5.711e-03  2.030e-04 -28.141  < 2e-16 ***
u           -4.473e-03  8.016e-04  -5.580 2.45e-08 ***
v            8.367e-03  4.974e-04  16.822  < 2e-16 ***
ws           1.026e-01  8.835e-04 116.081  < 2e-16 ***
wd          -1.449e-04  2.539e-05  -5.706 1.18e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1552 on 13127 degrees of freedom
Multiple R-squared:  0.5746,    Adjusted R-squared:  0.5744
F-statistic:  3546 on 5 and 13127 DF,  p-value: < 2.2e-16

> pred = predict(lm.fit1, test1)
> accuracy(pred, test1$wp1)
                ME      RMSE       MAE MPE MAPE
Test set 0.0356211 0.1773012 0.1339909 NaN  Inf
```

Wind Farm 2

```
lm.fit2=lm(wp2~.-Date-OnlyDate-month-u, data=model2)
summary(lm.fit2)
library(forecast)
pred = predict(lm.fit2, test2)
accuracy(pred, test2$wp2)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.264e-01  7.130e-03  -17.73   <2e-16 ***
hour        -7.131e-03  2.207e-04  -32.31   <2e-16 ***
v            1.086e-02  5.005e-04   21.70   <2e-16 ***
ws           1.138e-01  8.387e-04  135.68   <2e-16 ***
wd           5.754e-04  1.659e-05   34.69   <2e-16 ***
year        -4.491e-02  3.232e-03  -13.89   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.17 on 13064 degrees of freedom
Multiple R-squared:  0.5999,	Adjusted R-squared:  0.5997
F-statistic:  3918 on 5 and 13064 DF,  p-value: < 2.2e-16

> library(forecast)
> pred = predict(lm.fit2, test2)
> accuracy(pred, test2$wp2)
                 ME      RMSE       MAE MPE MAPE
Test set 0.06563687 0.1819857 0.1387788 NaN  Inf
```

Wind Farm 3

```
lm.fit3=lm(wp3~.-Date-OnlyDate-year-month-wd, data=model3)
summary(lm.fit3)
library(forecast)
pred = predict(lm.fit3, test3)
accuracy(pred, test3$wp3)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.1357297  0.0040748  -33.31   <2e-16 ***
hour        -0.0070628  0.0002145  -32.92   <2e-16 ***
u           -0.0058941  0.0004309  -13.68   <2e-16 ***
v            0.0111887  0.0004329   25.84   <2e-16 ***
ws           0.1205050  0.0006978  172.70   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1677 on 13100 degrees of freedom
Multiple R-squared:  0.7129,	Adjusted R-squared:  0.7128
F-statistic:  8130 on 4 and 13100 DF,  p-value: < 2.2e-16

> library(forecast)
> pred = predict(lm.fit3, test3)
> accuracy(pred, test3$wp3)
                 ME      RMSE       MAE MPE MAPE
Test set 0.02410656 0.1862717 0.1407924 NaN  Inf
```

Wind Farm 4

```
lm.fit4=lm(wp4~.-Date-OnlyDate-year-u-month, data=model4)
summary(lm.fit4)
library(forecast)
pred = predict(lm.fit4, test4)
accuracy(pred, test4$wp4)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.714e-01  4.935e-03 -34.728   <2e-16 ***
hour        -7.181e-03  2.016e-04 -35.628   <2e-16 ***
v            1.233e-02  3.937e-04  31.327   <2e-16 ***
ws           1.143e-01  6.661e-04 171.630   <2e-16 ***
wd           1.531e-04  1.586e-05   9.652   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1563 on 13125 degrees of freedom
Multiple R-squared:  0.703,	Adjusted R-squared:  0.703 
F-statistic:  7768 on 4 and 13125 DF,  p-value: < 2.2e-16

> library(forecast)
> pred = predict(lm.fit4, test4)
> accuracy(pred, test4$wp4)
                 ME      RMSE       MAE MPE MAPE
Test set 0.02954673 0.1707248 0.1306941 NaN  Inf
```

Wind Farm 5

```
lm.fit5=lm(wp5~.-Date-OnlyDate-month+I(u^2)+I(wd^2), data=model5)
summary(lm.fit5)
library(forecast)
pred = predict(lm.fit5, test5)
accuracy(pred, test5$wp5)
```

```
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3.168e-01  9.746e-03 -32.508  < 2e-16 ***
hour        -3.489e-03  2.006e-04 -17.396  < 2e-16 ***
u           -1.255e-02  7.848e-04 -15.995  < 2e-16 ***
v            1.607e-02  7.015e-04  22.903  < 2e-16 ***
ws           1.076e-01  9.915e-04 108.566  < 2e-16 ***
wd           6.542e-04  9.895e-05   6.611 3.97e-11 ***
year         6.499e-02  2.929e-03  22.187  < 2e-16 ***
I(u^2)       2.110e-03  1.375e-04  15.342  < 2e-16 ***
I(wd^2)     -2.599e-06  2.842e-07  -9.144  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1566 on 13066 degrees of freedom
Multiple R-squared:  0.6835,	Adjusted R-squared:  0.6833 
F-statistic:  3527 on 8 and 13066 DF,  p-value: < 2.2e-16

> library(forecast)
> pred = predict(lm.fit5, test5)
> accuracy(pred, test5$wp5)
                  ME      RMSE       MAE MPE MAPE
Test set -0.05783443 0.1918051 0.1555689 NaN  Inf
```

Wind Farm 6

```
lm.fit6=lm(wp6~.-Date-OnlyDate-year-u, data=model6)
summary(lm.fit6)
library(forecast)
pred = predict(lm.fit6, test6)
accuracy(pred, test6$wp6)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.898e-01  5.455e-03 -34.797  < 2e-16 ***
hour        -4.279e-03  1.871e-04 -22.874  < 2e-16 ***
v            7.734e-03  3.159e-04  24.482  < 2e-16 ***
ws           9.892e-02  5.608e-04 176.412  < 2e-16 ***
wd           1.009e-04  1.462e-05   6.901 5.42e-12 ***
month        1.029e-03  3.907e-04   2.633  0.00847 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1457 on 13095 degrees of freedom
Multiple R-squared:  0.7125,	Adjusted R-squared:  0.7124
F-statistic:  6490 on 5 and 13095 DF,  p-value: < 2.2e-16

> library(forecast)
> pred = predict(lm.fit6, test6)
> accuracy(pred, test6$wp6)
                 ME      RMSE       MAE MPE MAPE
Test set 0.04763217 0.1678523 0.1241318 NaN  Inf
>
```

Wind Farm 7

```
lm.fit7=lm(wp7~.-Date-OnlyDate-u-year-month, data=model7)
summary(lm.fit7)
library(forecast)
pred = predict(lm.fit7, test7)
accuracy(pred, test7$wp7)
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.559e-01  4.963e-03  -51.56   <2e-16 ***
hour        -2.944e-03  1.948e-04  -15.11   <2e-16 ***
v            1.159e-02  2.891e-04   40.10   <2e-16 ***
ws           9.700e-02  5.048e-04  192.17   <2e-16 ***
wd           2.647e-04  1.512e-05   17.50   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1521 on 13092 degrees of freedom
Multiple R-squared:  0.7437,	Adjusted R-squared:  0.7436
F-statistic:  9497 on 4 and 13092 DF,  p-value: < 2.2e-16

> library(forecast)
> pred = predict(lm.fit7, test7)
> accuracy(pred, test7$wp7)
                 ME     RMSE       MAE MPE MAPE
Test set 0.01396243 0.164149 0.1257243 NaN  Inf
>
```

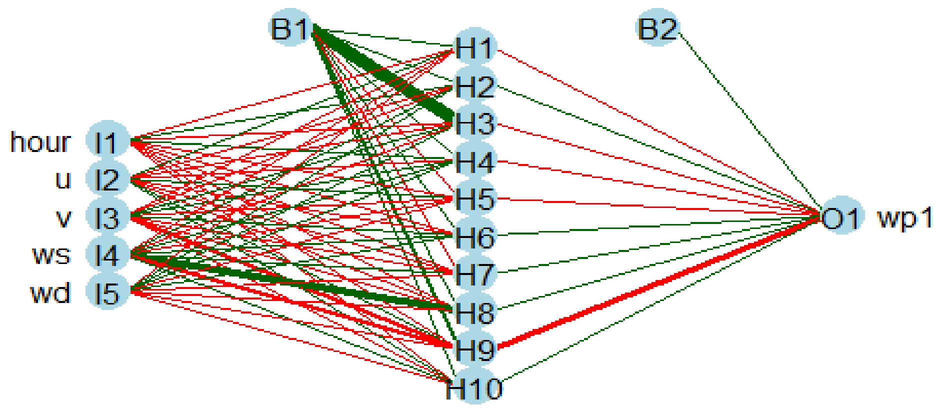
Neural Network

1. To go ahead with neural network model we first cleaned the data and removed outliers with cook's method.
2. In Neural network we did modeling for all 7 farms and the models for all the farms were different.
3. Feature Selection was done all the farms and model was applied based on the different predictors.
4. Performance metrics RMSE was calculated for all seven models.

Wind Farm 1

```
library(forecast)
library(ROCR)
fml<- as.formula("wp1~.-Date-OnlyDate-month-year");
res <- nnet(fml, data=model1,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test1,type="raw")
pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test1$wp1)^2))
mape <- mean(abs((test1$wpl - pred)/test1$wp1))*100
```

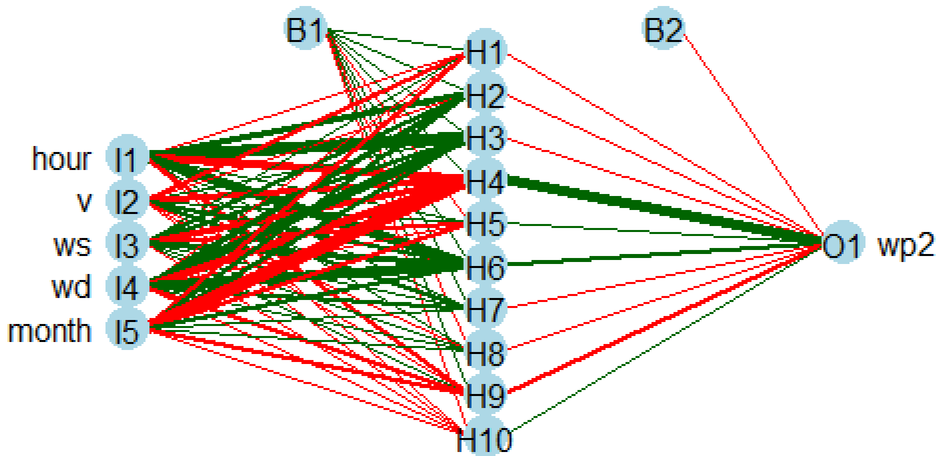
RMSE: 0.173577



Wind Farm 2

```
fml<- as.formula("wp2~.-Date-OnlyDate-year-u");
res <- nnet(fml, data=model2,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test2,type="raw")
pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test2$wp2)^2))
mape <- mean(abs((test2$wp2 - pred)/test2$wp2))*100
..
```

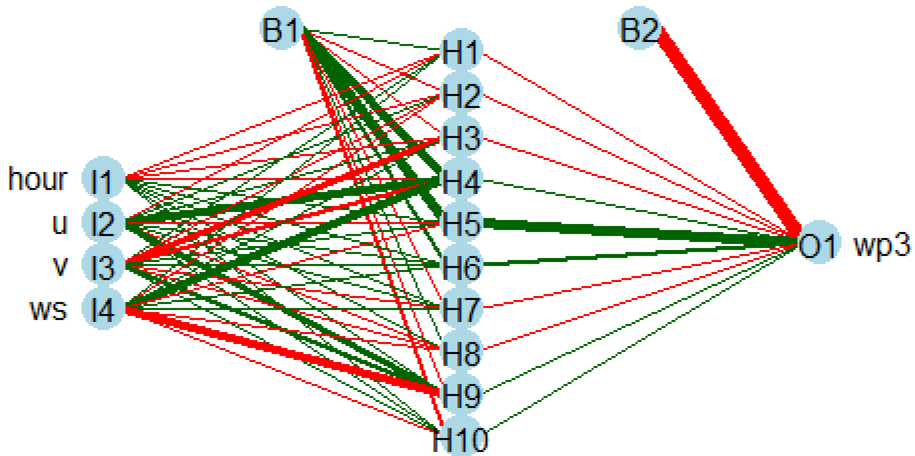
RMSE : 0.1663406



Wind Farm 3

```
#------------------------------
fml<- as.formula("wp3~.-Date-OnlyDate-year-month-wd");
res <- nnet(fml, data=model3,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test3,type="raw")
pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test3$wp3)^2))
mape <- mean(abs((test3$wp3 - pred)/test3$wp3))*100
rmse
```

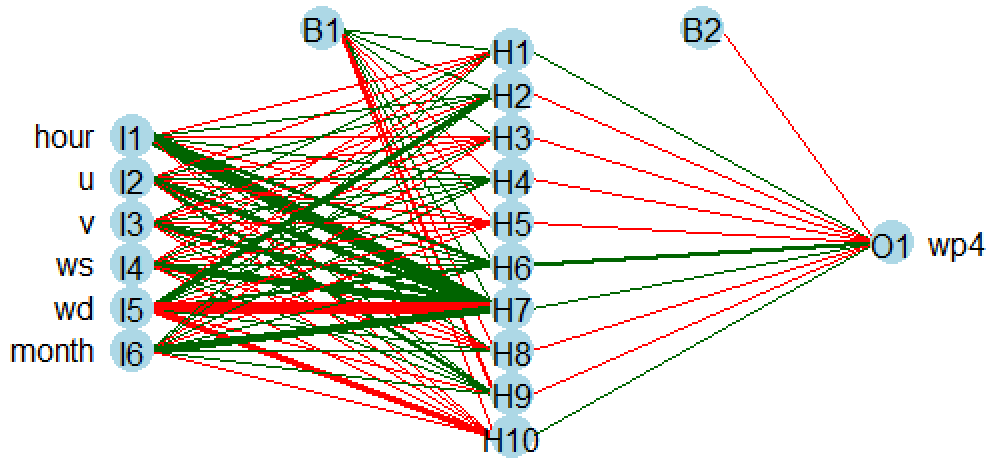
RMSE : 0.183047



Wind Farm 4

```
..
fml<- as.formula("wp4~.-Date-OnlyDate-year-month");
res <- nnet(fml, data=model4,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=1
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test4,type="raw")
pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test4$wp4)^2))
mape <- mean(abs((test4$wp4 - pred)/test4$wp4))*100
rmse
```

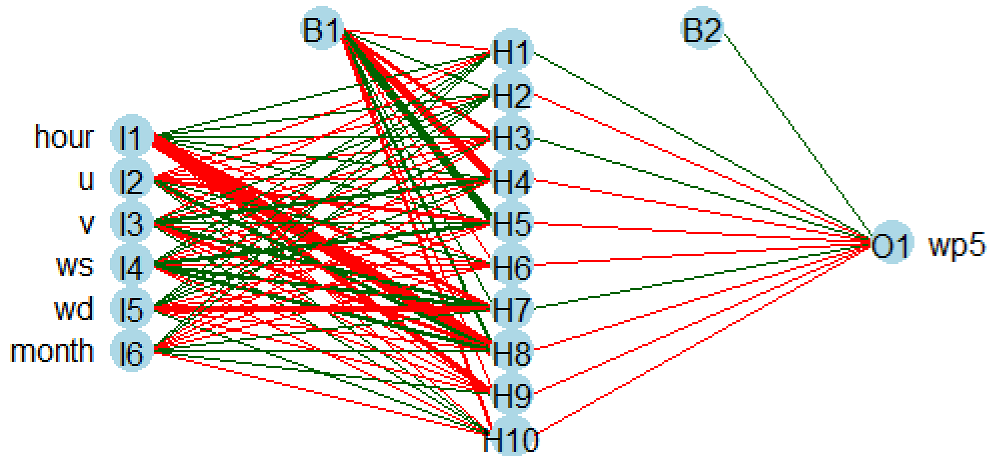
RMSE : 0.1718888



Wind Farm 5

```
fml<- as.formula("wp5~.-Date-OnlyDate-year");
res <- nnet(fml, data=model5,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=1
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test5,type="raw")
pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test5$wp5)^2))
mape <- mean(abs((test5$wp5 - pred)/test5$wp5))*100
rmse
```

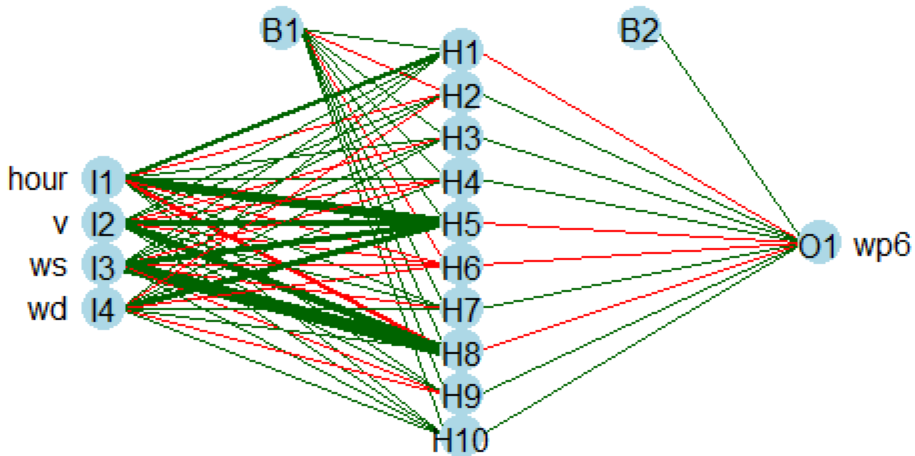
RMSE : 0.1897593



Wind Farm 6

```
fml<- as.formula("wp6~.-Date-OnlyDate-year-month-u");
res <- nnet(fml, data=model6,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test6,type="raw")
pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test6$wp6)^2))
mape <- mean(abs((test6$wp6 - pred)/test6$wp6))*100
rmse
```

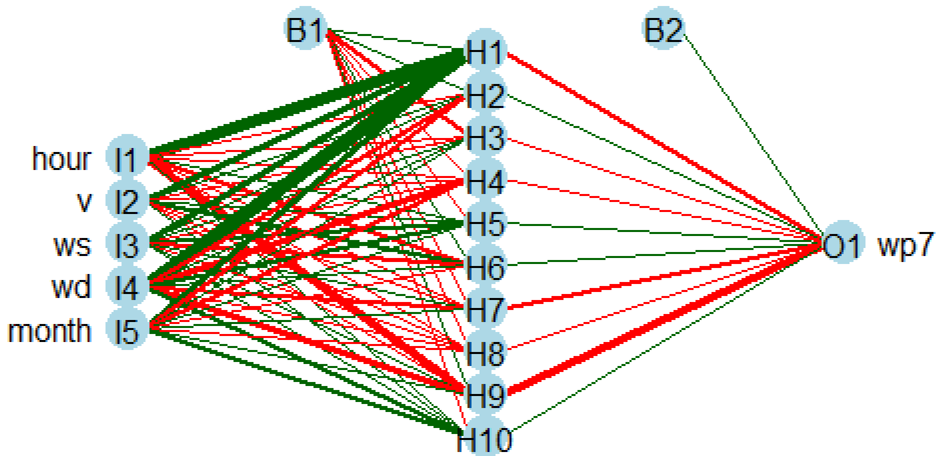
RMSE : 0.1664821



Wind Farm 7

```
#--------------------------------
fml<- as.formula("wp7~.-Date-OnlyDate-year-u");
res <- nnet(fml, data=model7,size=10, linout=TRUE, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
summary(res)

#plot
plotnet(res,circle_col = "lightblue", circle_cex = 3,pos_col = "darkgreen", neg_col = "red")

#predict
pred<-predict(res, newdata=test7,type="raw")
#pred <- round(pred, digits = 3)
rmse <- sqrt(mean((pred- test7$wp7)^2))
mape <- mean(abs((test7$wp7 - pred)/test7$wp7))*100
rmse
```

RMSE : 0.1601374



Regression Tree

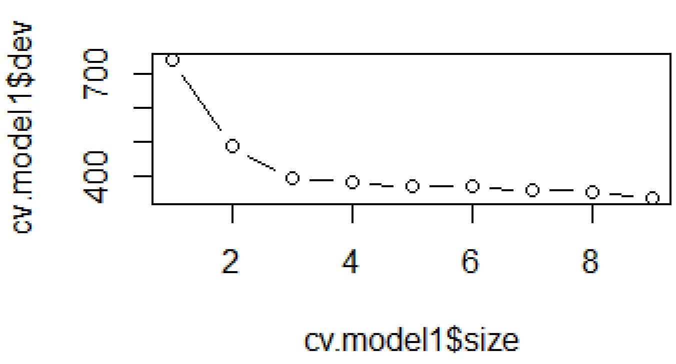
1. To go ahead with regression model we first cleaned the data and removed outliers with cook's method.
2. In regression tree we did modeling for all 7 farms and the models for all the farms were different.
3. Feature Selection was done all the farms and model was applied based on the different predictors.
4. Performance metrics and accuracy was calculated for all seven models.

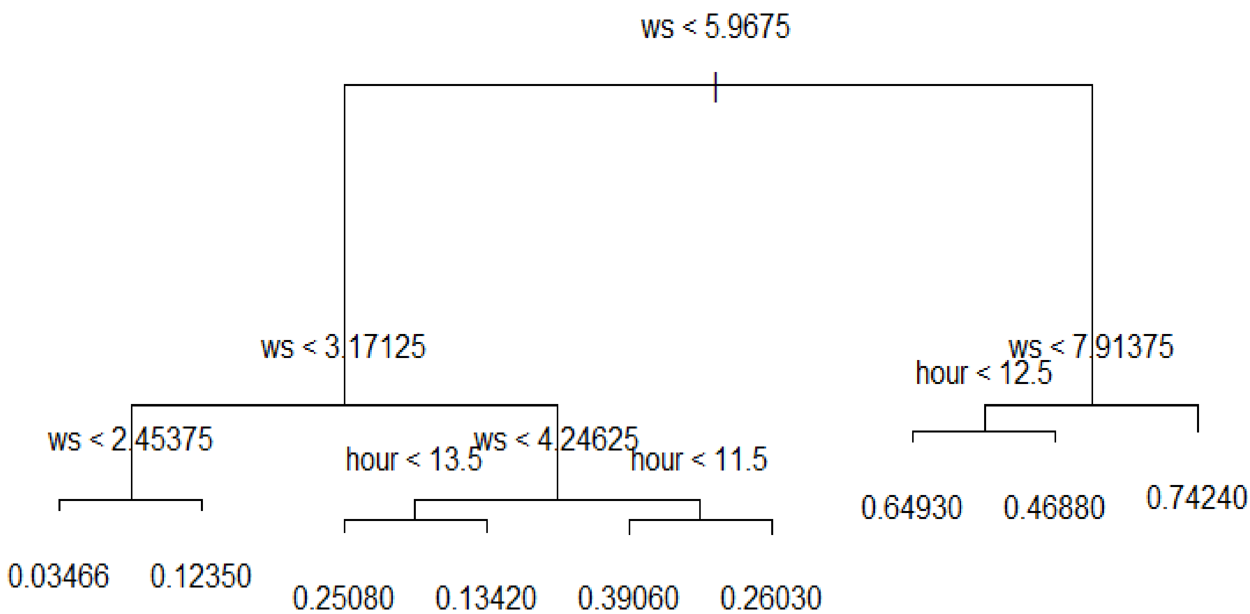
Wind Farm 1

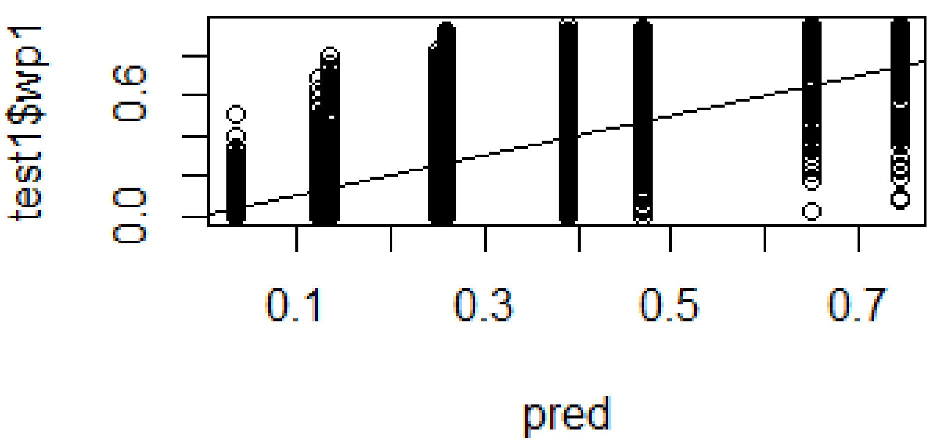
```
library(tree)
tree.model1 = tree(wp1~.-Date-OnlyDate,model1)
summary (tree.model1)

#plot
plot (tree.model1)
text (tree.model1, pretty = 0)
cv.model1 = cv.tree (tree.model1)
plot (cv.model1$size, cv.model1$dev, type='b')

#predict
pred=predict (tree.model1, newdata =test1)
plot(pred,test1$wp1)
abline (0,1)
accuracy(pred,test1$wp1)
```

```
> summary (tree.model1)

Regression tree:
tree(formula = wp1 ~ . - Date - OnlyDate, data = model1)
Variables actually used in tree construction:
[1] "ws"   "hour"
Number of terminal nodes:  9
Residual mean deviance:  0.02474 = 324.7 / 13120
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.63740 -0.10280 -0.02917  0.00000  0.08458  0.70780
```

```
> accuracy(pred,test1$wp1)
                 ME      RMSE       MAE  MPE MAPE
Test set 0.03795856 0.1835736 0.1358228 -Inf  Inf
```




ws < 5.9675
ws < 3.17125
ws < 7.91375
hour < 12.5
ws < 2.45375
hour < 13.5
ws < 4.24625
hour < 11.5
0.64930
0.46880
0.74240
0.03466
0.12350
0.25080
0.13420
0.39060
0.26030

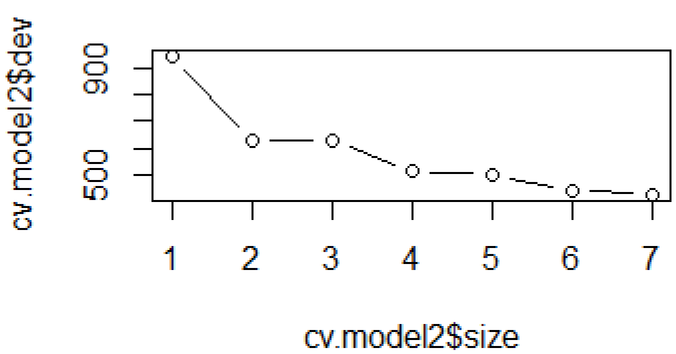

test1$wp1
0.0
0.6
0.1
0.3
0.5
0.7
pred

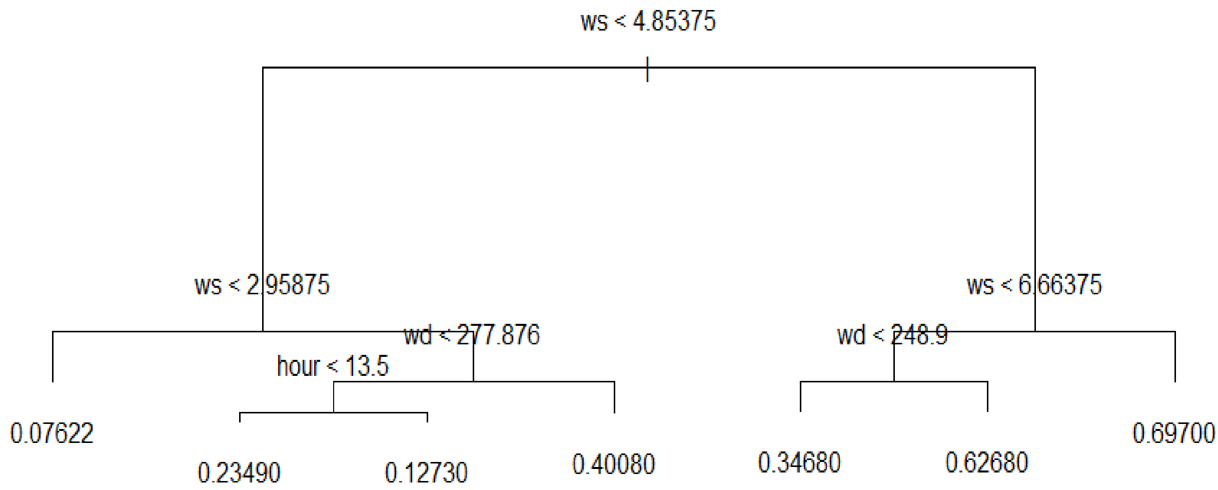
Wind Farm 2

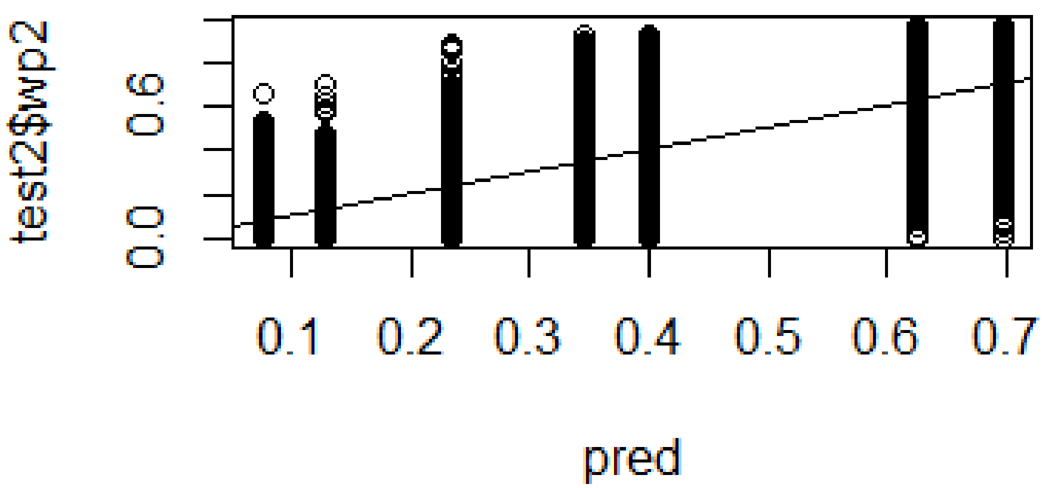
```
library(tree)
tree.model2 = tree(wp2~.-Date-OnlyDate,model2)
summary (tree.model2)

#plot
plot (tree.model2)
text (tree.model2, pretty = 0)
cv.model2 = cv.tree (tree.model2)
plot (cv.model2$size, cv.model2$dev, type='b')

#predict
pred=predict (tree.model2, newdata =test2)
plot(pred,test2$wp2)
abline (0,1)
accuracy(pred,test2$wp2)
```

```
> summary (tree.model2)

Regression tree:
tree(formula = wp2 ~ . - Date - OnlyDate, data = model2)
Variables actually used in tree construction:
[1] "ws"   "wd"   "hour"
Number of terminal nodes:  7
Residual mean deviance:  0.03208 = 419 / 13060
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.63900 -0.08776 -0.03726  0.00000  0.09103  0.76480
```

```
> accuracy(pred,test2$wp2)
                  ME      RMSE       MAE  MPE MAPE
Test set -0.01006036 0.1774975 0.1347039 -Inf  Inf
>
```




ws < 4.85375
ws < 2.95875
ws < 6.66375
wd < 277.876
wd < 248.9
hour < 13.5
0.07622
0.23490
0.12730
0.40080
0.34680
0.62680
0.69700

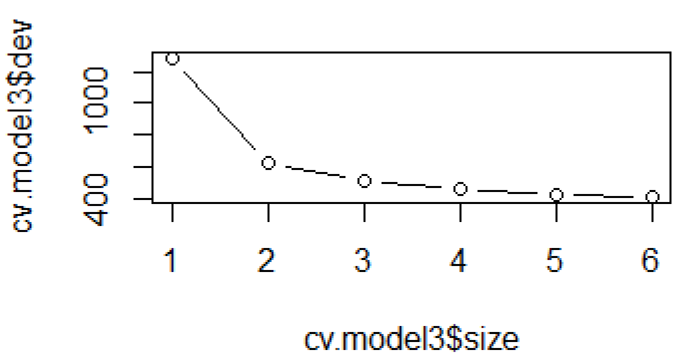

test2$wp2
0.0
0.6
0.1
0.2
0.3
0.4
0.5
0.6
0.7
pred

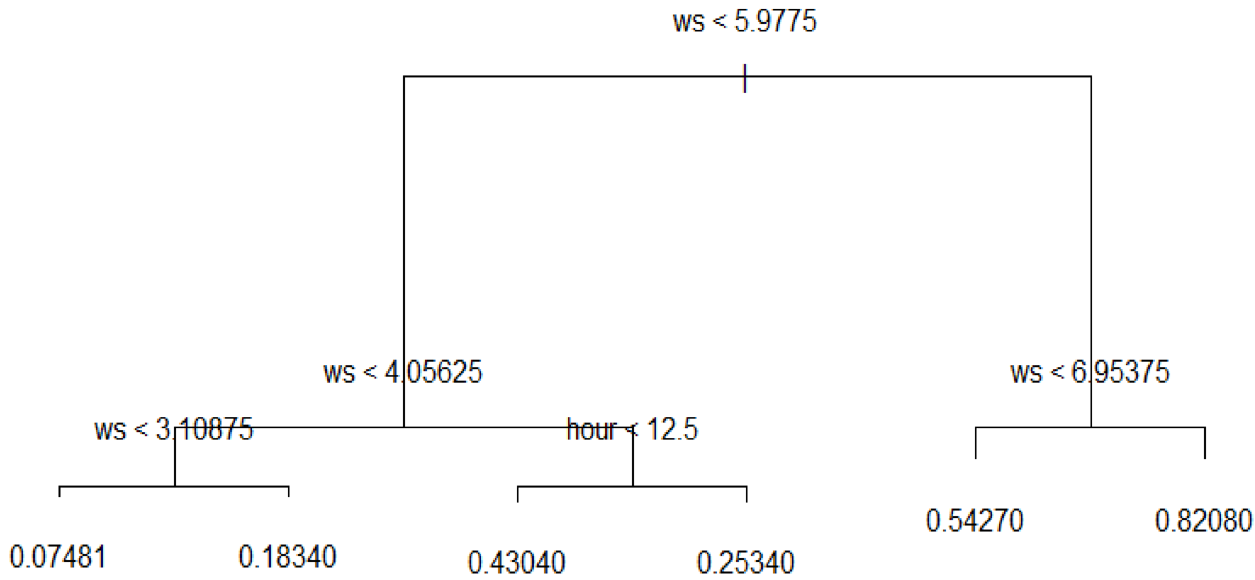
Wind Farm 3

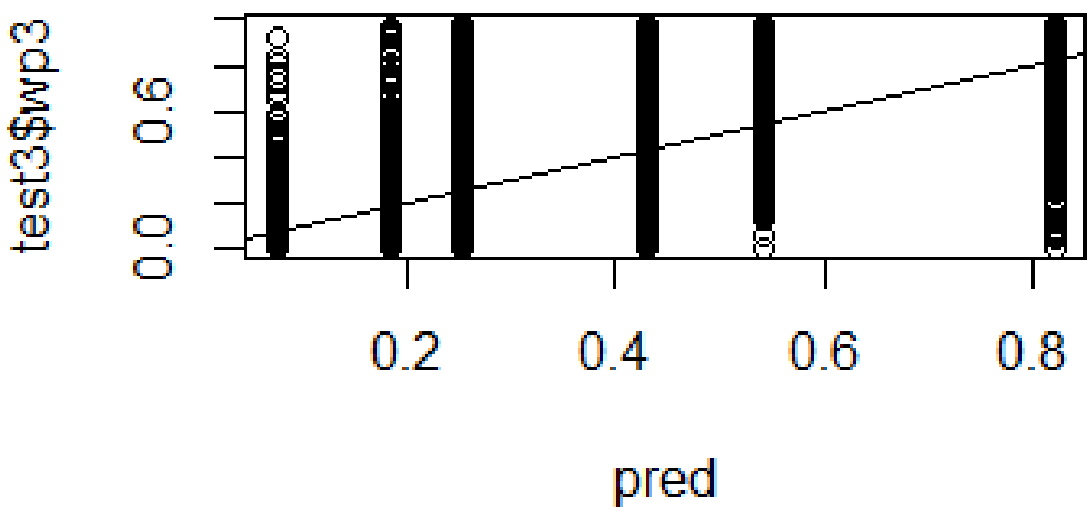
```
#RegressionTree
library(tree)
tree.model3 = tree(wp3~.-Date-OnlyDate,model3)
summary (tree.model3)

#plot
plot (tree.model3)
text (tree.model3, pretty = 0)
cv.model3 = cv.tree (tree.model3)
plot (cv.model3$size, cv.model3$dev, type='b')

#predict
pred=predict (tree.model3, newdata =test3)
plot(pred,test3$wp3)
abline (0,1)
accuracy(pred,test3$wp3)
```

```
> summary (tree.model3)

Regression tree:
tree(formula = wp3 ~ . - Date - OnlyDate, data = model3)
Variables actually used in tree construction:
[1] "ws"   "hour"
Number of terminal nodes:  6
Residual mean deviance:  0.03109 = 407.2 / 13100
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.82080 -0.08545 -0.02581  0.00000  0.10320  0.81420
```

```
> accuracy(pred,test3$wp3)
                 ME      RMSE       MAE  MPE MAPE
Test set 0.02290298 0.2005844 0.1452925 -Inf  Inf
```




ws < 5.9775
ws < 4.05625
ws < 6.95375
ws < 3.10875
hour < 12.5
0.07481
0.18340
0.43040
0.25340
0.54270
0.82080

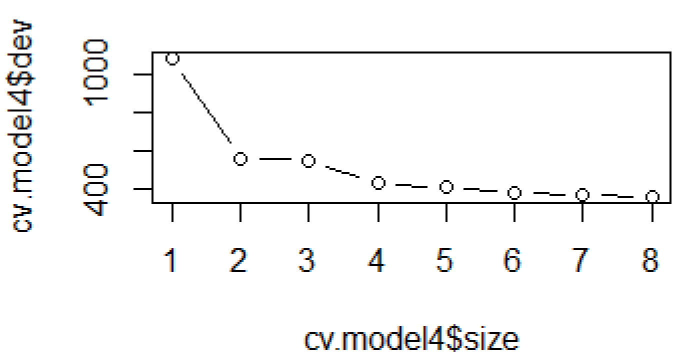

test3$wp3
0.0
0.6
0.2
0.4
0.6
0.8
pred

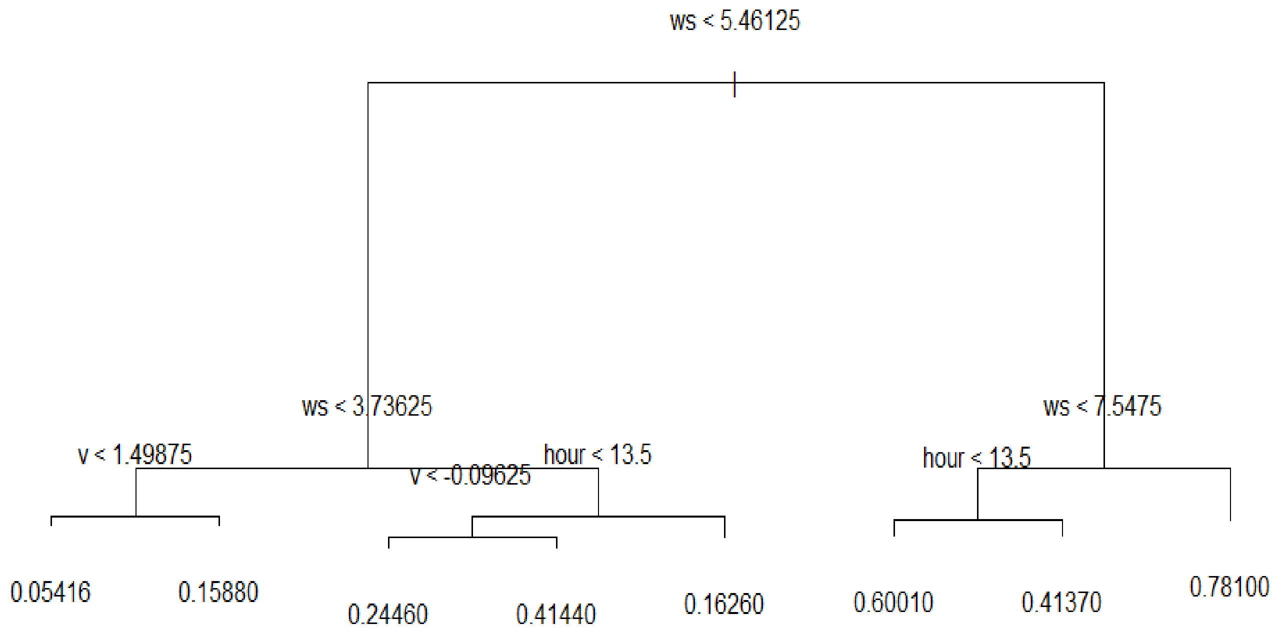
Wind Farm 4

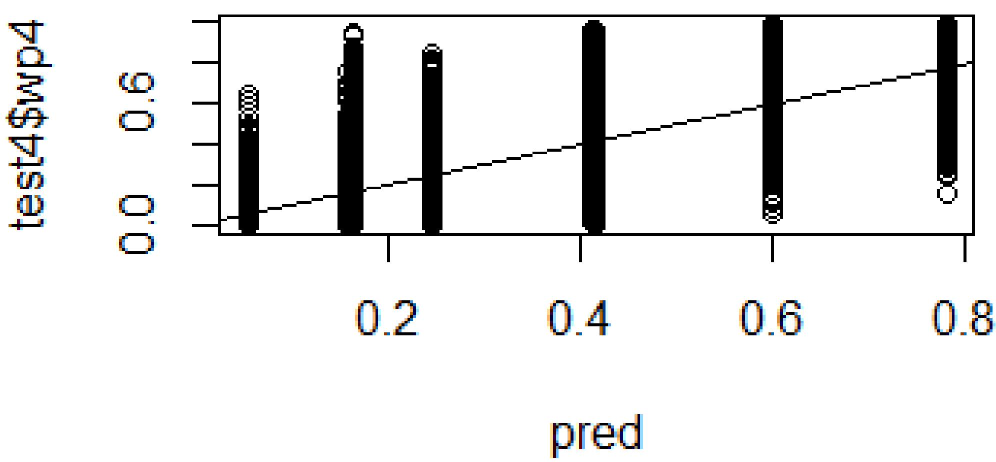
```
#RegressionTree
library(tree)
tree.model4 = tree(wp4~.-Date-OnlyDate,model4)
summary (tree.model4)

#plot
plot (tree.model4)
text (tree.model4, pretty = 0)

#predict
pred=predict (tree.model4, newdata =test4)
plot(pred,test4$wp4)
abline (0,1)
accuracy(pred,test4$wp4)
```

```
> summary (tree.model4)

Regression tree:
tree(formula = wp4 ~ . - Date - OnlyDate, data = model4)
Variables actually used in tree construction:
[1] "ws"   "v"    "hour"
Number of terminal nodes:  8
Residual mean deviance:  0.02583 = 338.9 / 13120
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.74200 -0.07824 -0.03369  0.00000  0.08884  0.70620
>
```

```
> accuracy(pred,test4$wp4)
                 ME      RMSE       MAE  MPE MAPE
Test set 0.02800967 0.1789393 0.1339365 -Inf  Inf
>
```




ws < 5.46125
ws < 3.73625
ws < 7.5475
v < 1.49875
v < -0.09625
hour < 13.5
hour < 13.5
0.05416
0.15880
0.24460
0.41440
0.16260
0.60010
0.41370
0.78100

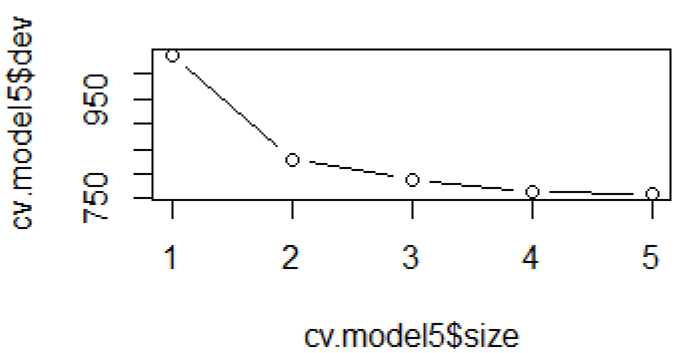

test4$wp4
0.0
0.6
0.2
0.4
0.6
0.8
pred

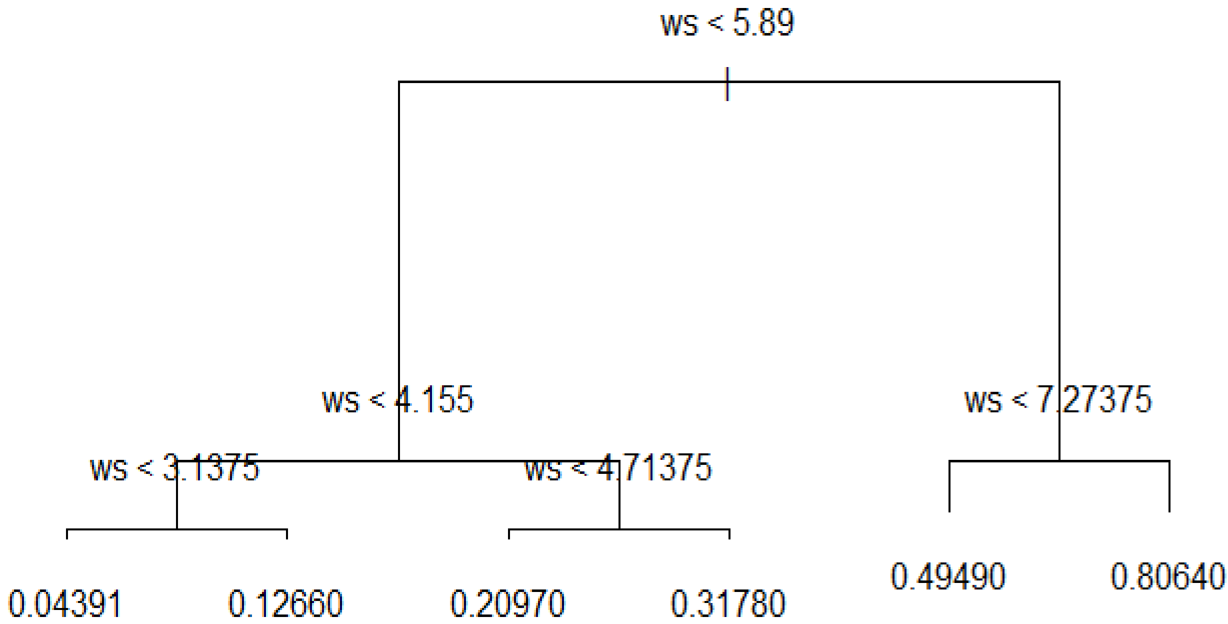
Wind Farm 5

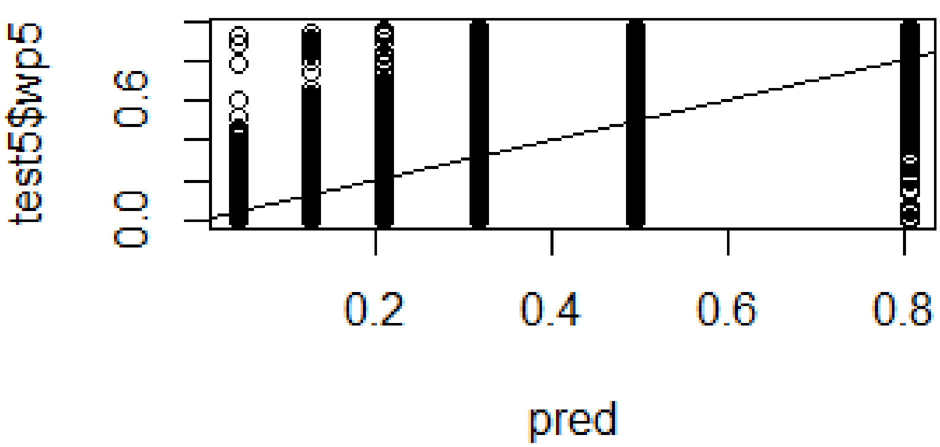
```
#RegressionTree
library(tree)
tree.model5 = tree(wp5~.-Date-OnlyDate,model5)
summary (tree.model5)

#plot
plot (tree.model5)
text (tree.model5, pretty = 0)


#predict
pred=predict (tree.model5, newdata =test5)
plot(pred,test5$wp5)
abline (0,1)
accuracy(pred,test5$wp5)
```

```
> summary (tree.model5)

Regression tree:
tree(formula = wp5 ~ . - Date - OnlyDate, data = model5)
Variables actually used in tree construction:
[1] "ws"
Number of terminal nodes:  6
Residual mean deviance:  0.02631 = 343.9 / 13070
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.80640 -0.08375 -0.02891  0.00000  0.07618  0.77840
>
```

```
> accuracy(pred,test5$wp5)
                ME      RMSE       MAE  MPE MAPE
Test set 0.0485396 0.1979785 0.1387474 -Inf  Inf
>
```




ws < 5.89
ws < 4.155
ws < 7.27375
ws < 3.1375
ws < 4.71375
0.04391
0.12660
0.20970
0.31780
0.49490
0.80640

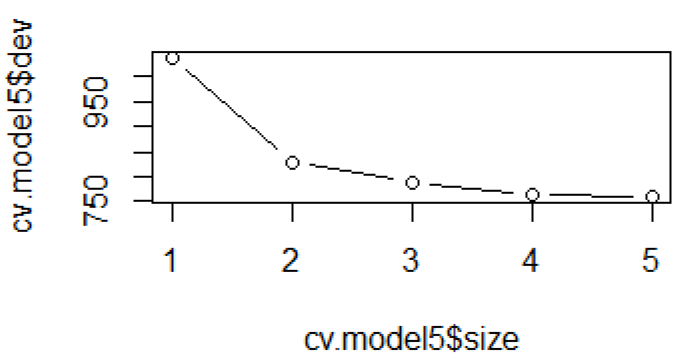

test5$wp5
0.0
0.6
0.2
0.4
0.6
0.8
pred

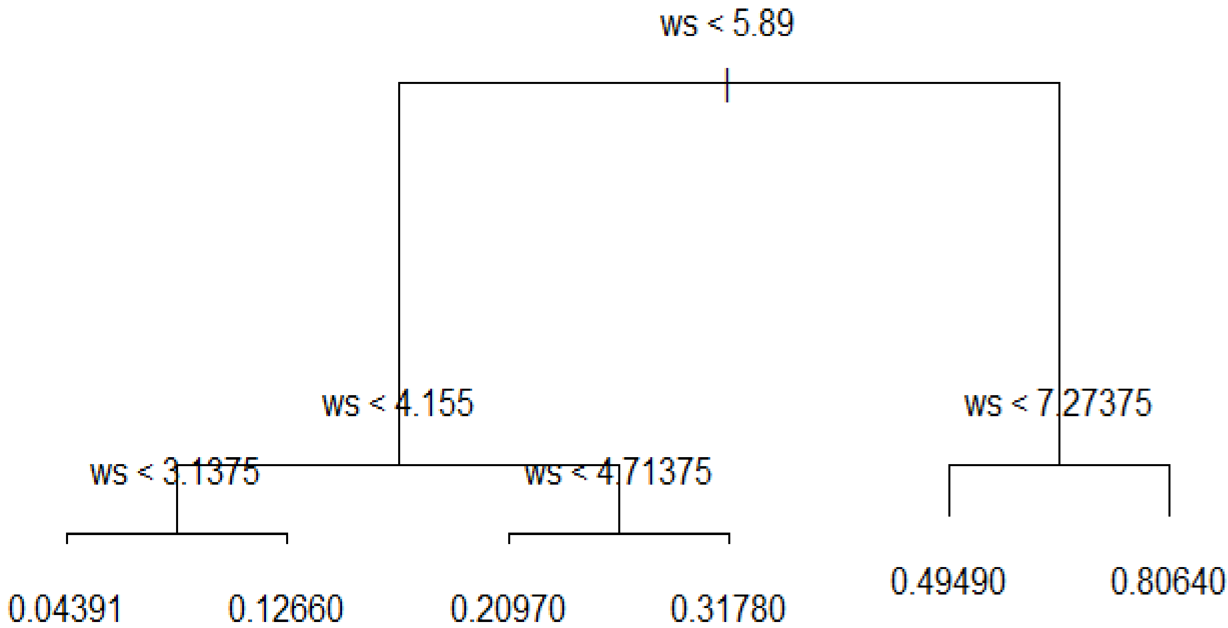
Wind Farm 6

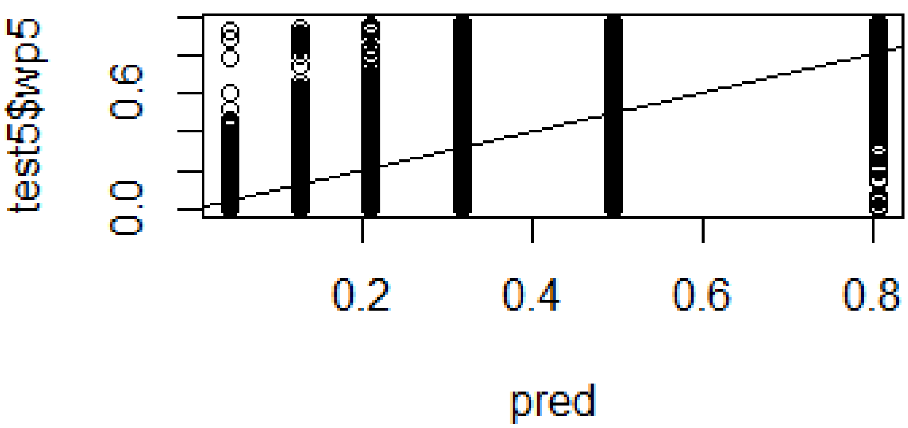
```
#RegressionTree
library(tree)
tree.model6 = tree(wp6~.-Date-OnlyDate,model6)
summary (tree.model6)

#plot
plot (tree.model6)
text (tree.model6, pretty = 0)


#predict
pred=predict (tree.model6, newdata =test6)
plot(pred,test6$wp6)
abline (0,1)
accuracy(pred,test6$wp6)
```

```
> summary (tree.model5)

Regression tree:
tree(formula = wp5 ~ . - Date - OnlyDate, data = model5)
Variables actually used in tree construction:
[1] "ws"
Number of terminal nodes:  6
Residual mean deviance:  0.02631 = 343.9 / 13070
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.80640 -0.08375 -0.02891  0.00000  0.07618  0.77840
>
```

```
> accuracy(pred,test5$wp5)
                ME     RMSE       MAE  MPE MAPE
Test set 0.0485396 0.1979785 0.1387474 -Inf  Inf
>
```




ws < 5.89
ws < 4.155
ws < 7.27375
ws < 3.1375
ws < 4.71375
0.04391
0.12660
0.20970
0.31780
0.49490
0.80640

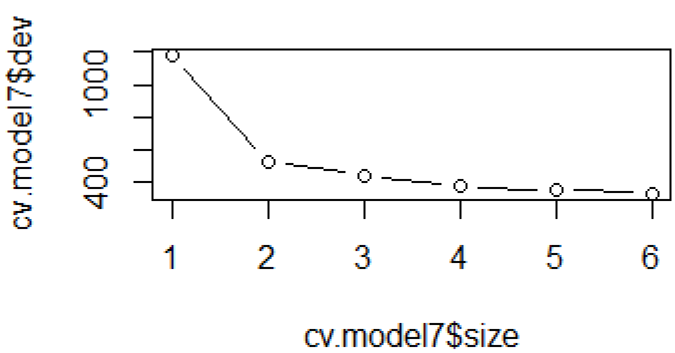

test5$wp5
0.0
0.6
0.2
0.4
0.6
0.8
pred

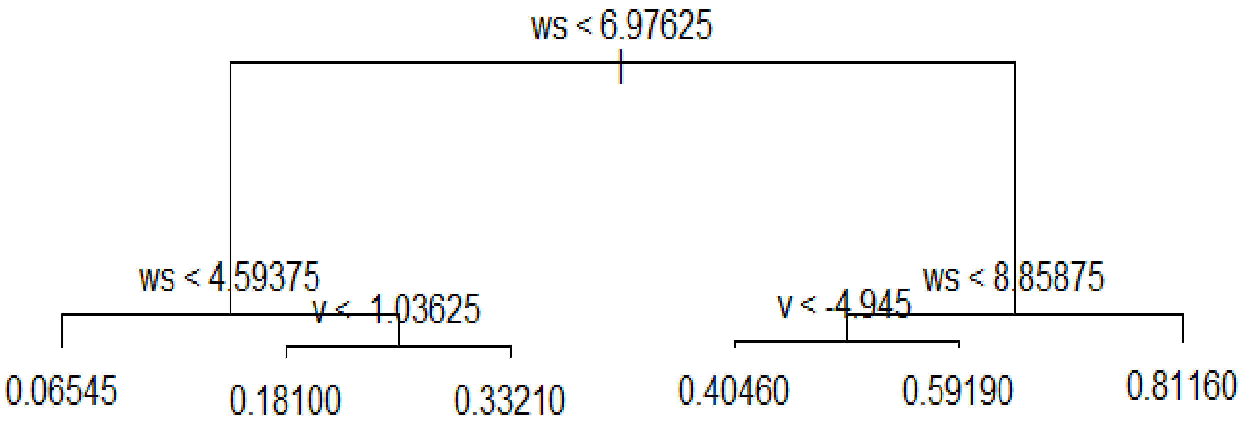
Wind Farm 7

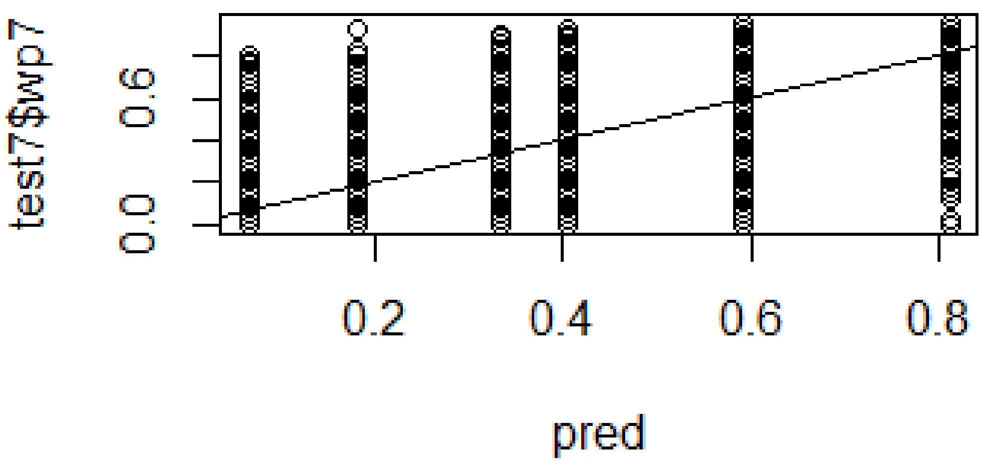
```
#RegressionTree
library(tree)
tree.model7 = tree(wp7~.-Date-OnlyDate,model7)
summary (tree.model7)

#plot
plot (tree.model7)
text (tree.model7, pretty = 0)
cv.model7 = cv.tree (tree.model7)
plot (cv.model7$size, cv.model7$dev, type='b')

#predict
pred=predict (tree.model7, newdata =test7)
plot(pred,test7$wp7)
abline (0,1)
accuracy(pred,test7$wp7)
```

```
> summary (tree.model7)

Regression tree:
tree(formula = wp7 ~ . - Date - OnlyDate, data = model7)
Variables actually used in tree construction:
[1] "ws" "v"
Number of terminal nodes:  6
Residual mean deviance:  0.02495 = 326.6 / 13090
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.71060 -0.06545 -0.04045  0.00000  0.09687  0.70300
> plot (tree.model7)
```

```
> accuracy(pred,test7$wp7)
                ME      RMSE       MAE  MPE MAPE
Test set 0.0117528 0.1671721 0.1240614 -Inf  Inf
>
```




ws < 6.97625
ws < 4.59375
v < 1.03625
v < -4.945
ws < 8.85875
0.06545
0.18100
0.33210
0.40460
0.59190
0.81160


test7$wp7
0.0
0.6
0.2
0.4
0.6
0.8
pred

Discussion

- ✓ **With the results derived above we can conclude that Multi Linear Regression has good Performance metrics.**
- ✓ **The forecast values had good varied predictions as compared to decision tree and neural network as these both divide the region and give a fixed values in when a data falls in a particular region**
- ✓ **Multilinear Regression works well in case of regression while neural network and decision tree works best in case of classification.**

Benchmark Output files: -

benchmark_nnet.csv benchmark_tree.csv benchmark_linearmodel.csv

***