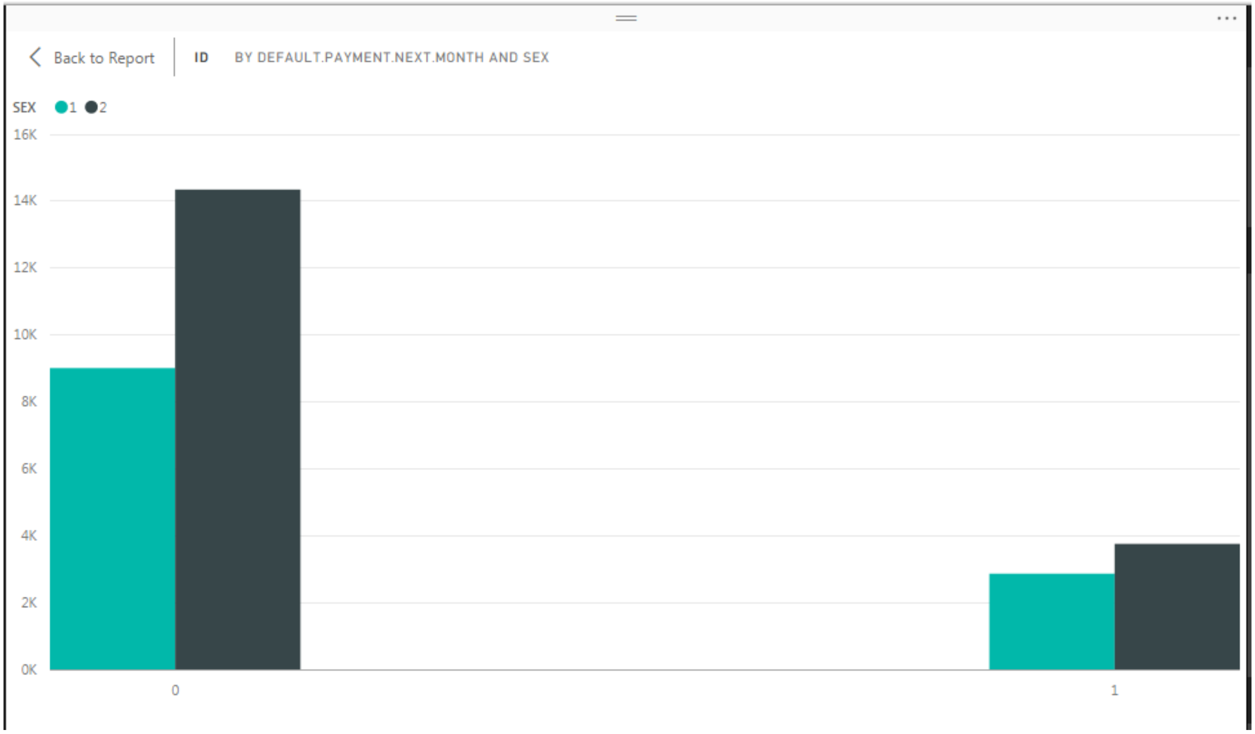
ADS Midterm Assignment

Team 8:

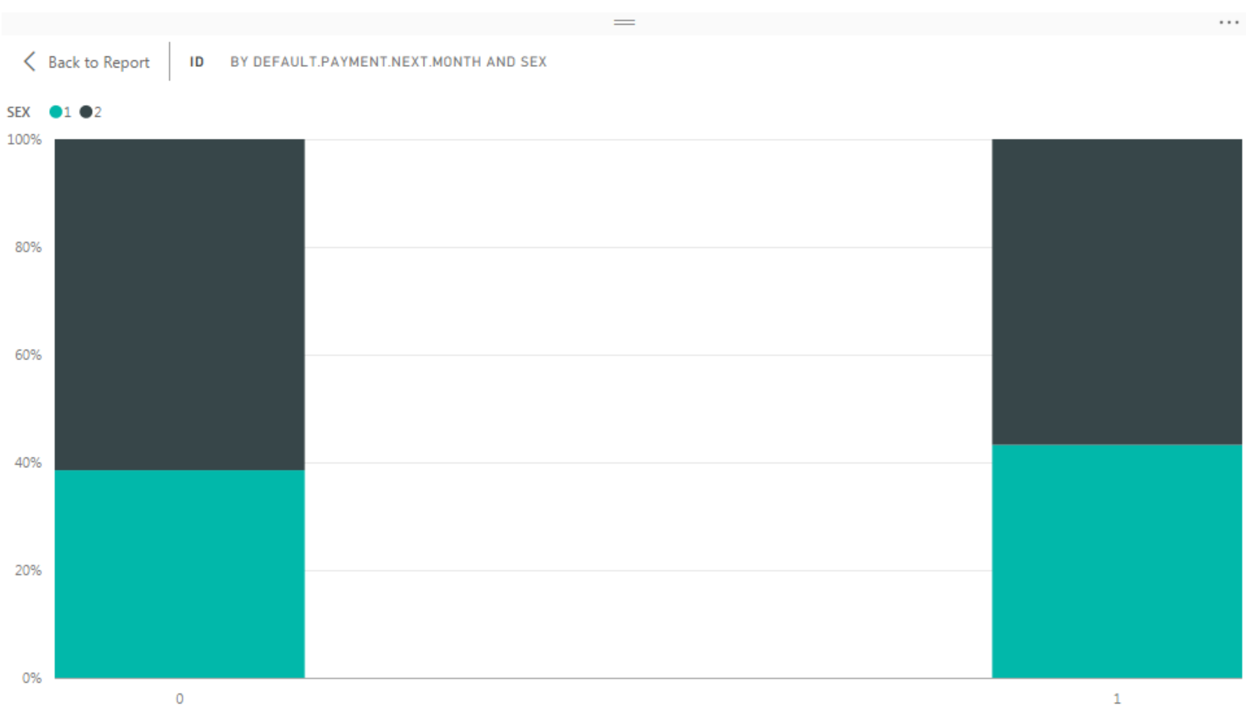
1) Credit Card Defaults

A) Use Power Bi to explore the data. Summarize your observation.

i) ID by Default.Payment.next.month and sex

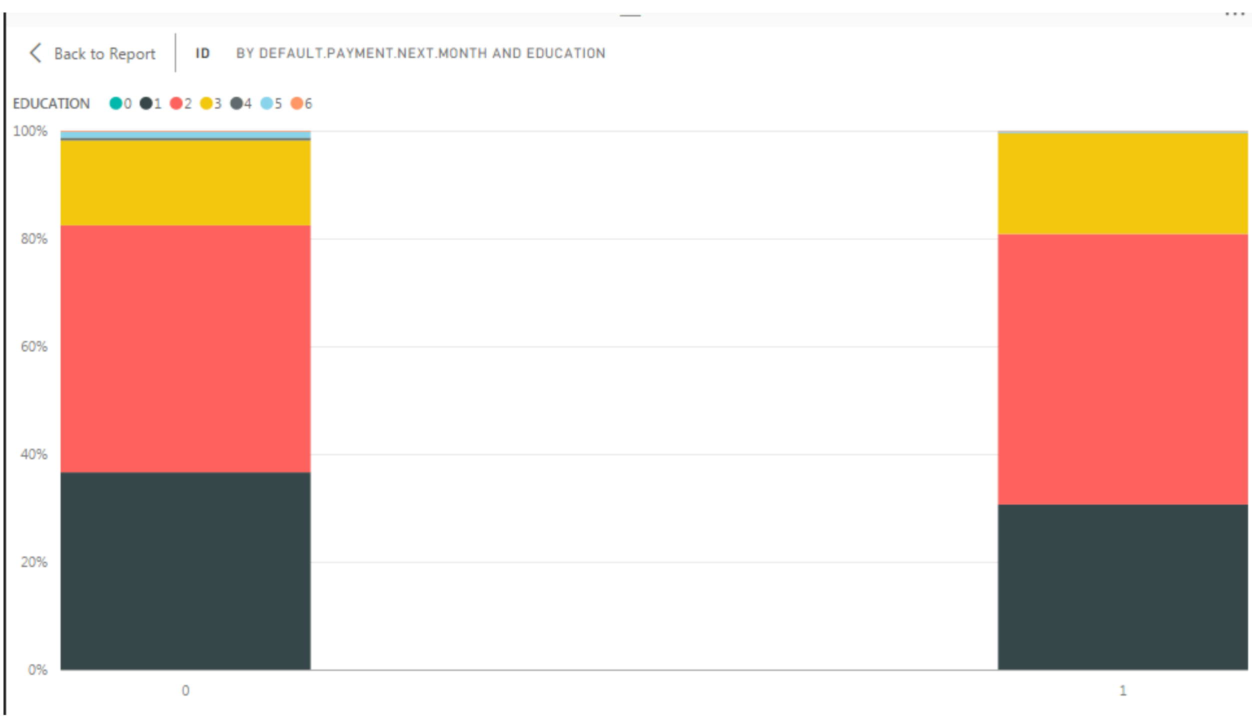


ii) By taking into consideration Default.Payment.next.month and sex

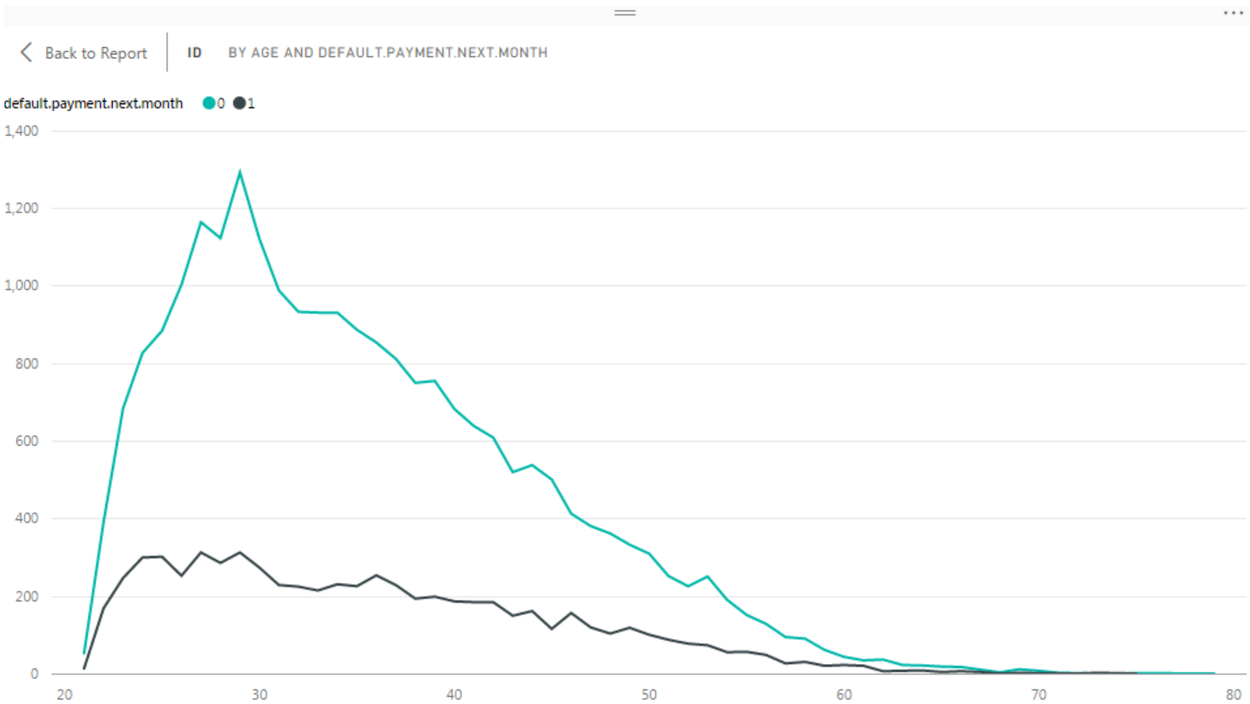


iii) By Taking into consideration Default.Payment.next.month and education

iv) By taking into consideration Default.Payment.next.month and Education

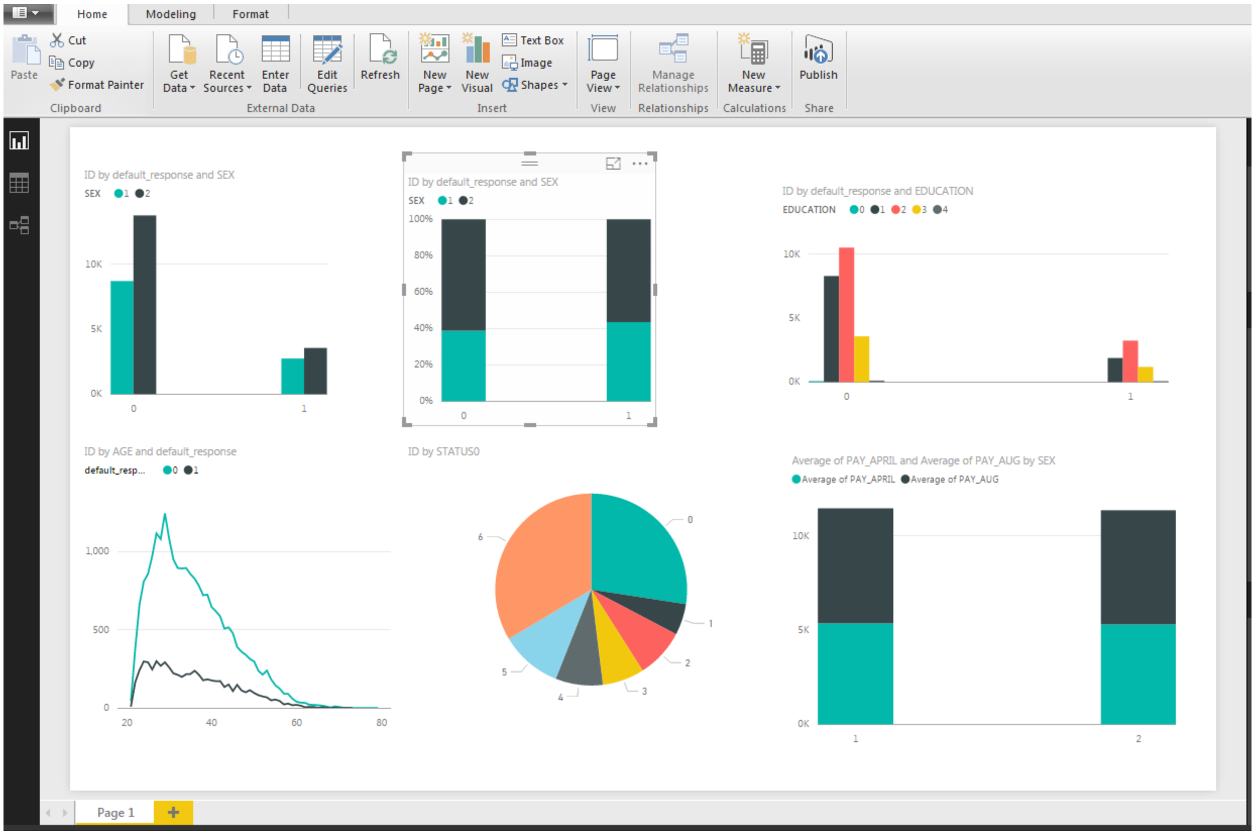


v) By taking into consideration age and Default.Payment.next.month



vi) By taking into consideration Default.payment.next.month and Marriage

VISUALISATION OF DATA AFTER CLEANING



2) Clean and preprocess the data if needed.

i. To clean the data first we added new columns with the count of occurrence of particular payment status for ever customer.

```
#count the occurence of specific payment status for every customer
levels=unique(do.call(c,df[,c(7:12)]))
out<-sapply(levels,function(x)rowSums((df[,c(7:12)])==x))

#Assign level names i.e Payment Status(-2,-1....) as column name
colnames(out)<-levels

#create a data frame with new values
df<-data.frame(df[1:6],out,df[13:25])

#assign names to columns of the data frame
colnames(df)<-
c("ID","LIMIT_BAL","SEX","EDUCATION","MARRIAGE","AGE","STATUS2","STATUS_negative_1","STATUS0","STATUS_negative_2","STATUS1","STATUS3","STATUS4","STATUS8","STATUS7","STATUS5","STATUS6","BILL_SEPT","BILL_AUG","BILL_JULY","BILL_JUNE","BILL_MAY","BILL_APRIL","PAY_SEPT","PAY_AUG","PAY_JULY","PAY_JUNE","PAY_MAY","PAY_APRIL","default_response")
View(df)
```

ATUS2	STATUS_negative_1	STATUS0	STATUS_negative_2	STATUS1	STATUS3	STATUS4	STATUS8	STATUS7	STATUS5	STATUS6	BILL_SEPT	BILL_AUG	BILL_JULY	BILL_JUNE	BILL_MAY	BI
2	2	0	2	0	0	0	0	0	0	0	3913	3102	689	0	0	
2	1	3	0	0	0	0	0	0	0	0	2682	1725	2682	3272	3455	
0	0	6	0	0	0	0	0	0	0	0	29239	14027	13559	14331	14948	
0	0	6	0	0	0	0	0	0	0	0	46990	48233	49291	28314	28959	
0	2	4	0	0	0	0	0	0	0	0	8617	5670	35835	20940	19146	
0	0	6	0	0	0	0	0	0	0	0	64400	57069	57608	19394	19619	
0	0	6	0	0	0	0	0	0	0	0	367965	412023	445007	542653	483003	
0	3	3	0	0	0	0	0	0	0	0	11876	380	601	221	-159	
1	0	5	0	0	0	0	0	0	0	0	11285	14096	12108	12211	11793	
0	2	0	4	0	0	0	0	0	0	0	0	0	0	0	13007	
1	1	4	0	0	0	0	0	0	0	0	11073	9787	5535	2513	1828	
1	5	0	0	0	0	0	0	0	0	0	12261	21670	9966	8517	22287	
0	5	1	0	0	0	0	0	0	0	0	12137	6500	6500	6500	6500	
3	0	2	0	1	0	0	0	0	0	0	65802	67369	65701	66782	36137	
0	0	6	0	0	0	0	0	0	0	0	70887	67060	63561	59696	56875	
1	0	4	0	1	0	0	0	0	0	0	50614	29173	28116	28771	29531	

Showing 1 to 17 of 30,000 entries

ii. Exclude the rows with all the bill and payment with zero values.

#exclude rows where all bill amount and payment amount is zero
df<-df[apply(df[,13:24],1,function(x) !all(x==0)),]

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	STATUS2	STATUS_negative_1	STATUS0	STATUS_negative_2	STATUS1	STATUS3	STATUS4	STATUS8	STATUS7	STATUS
1	1	20000	2	2	1	24	2	2	0	2	0	0	0	0	0	
2	2	120000	2	2	2	26	2	1	3	0	0	0	0	0	0	
3	3	90000	2	2	2	34	0	0	6	0	0	0	0	0	0	
4	4	50000	2	2	1	37	0	0	6	0	0	0	0	0	0	
5	5	50000	1	2	1	57	0	2	4	0	0	0	0	0	0	
6	6	50000	1	1	2	37	0	0	6	0	0	0	0	0	0	
7	7	500000	1	1	2	29	0	0	6	0	0	0	0	0	0	
8	8	100000	2	2	2	23	0	3	3	0	0	0	0	0	0	
9	9	140000	2	3	1	28	1	0	5	0	0	0	0	0	0	
10	10	20000	1	3	2	35	0	2	0	4	0	0	0	0	0	
11	11	200000	2	3	2	34	1	1	4	0	0	0	0	0	0	
12	12	260000	2	1	2	51	1	5	0	0	0	0	0	0	0	
13	13	630000	2	2	2	41	0	5	1	0	0	0	0	0	0	
14	14	70000	1	2	2	30	3	0	2	0	1	0	0	0	0	
15	15	250000	1	1	2	29	0	0	6	0	0	0	0	0	0	
16	16	50000	2	3	3	23	1	0	4	0	1	0	0	0	0	

Showing 1 to 17 of 29,136 entries

LOGISTIC REGRESSION

1. Divide the data into test and train data
   # Take 75% of the data as the sample data
   smp_size<-floor(0.60*nrow(df))
   set.seed(123)
   #Divide the data into train and test data. Sample data is basically train data
   train_ind<-sample(seq_len(nrow(df)),size=smp_size)
   train <- df[train_ind,]
   # Rest 25% is test data
   test <- df[-train_ind,]

2. Construct the logistic regression model
   fit<-glm(default_response~.,data=train,family=binomial(link="logit"))
   summary(fit)

```
Call:
glm(formula = default_response ~ LIMIT_BAL + SEX + MARRIAGE +
    PAY_AUG + PAY_SEPT + STATUS0 + STATUS_negative_2 + AGE, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3228  -0.7133  -0.5682  -0.3226   3.4812

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        6.009e-01  1.462e-01   4.111 3.94e-05 ***
LIMIT_BAL         -3.586e-06  1.882e-07 -19.053  < 2e-16 ***
SEX               -1.611e-01  3.905e-02  -4.126 3.69e-05 ***
MARRIAGE          -1.372e-01  4.020e-02  -3.412 0.000644 ***
PAY_AUG           -5.496e-06  1.954e-06  -2.812 0.004918 **
PAY_SEPT          -9.810e-06  2.323e-06  -4.223 2.42e-05 ***
STATUS0           -2.339e-01  8.476e-03 -27.590  < 2e-16 ***
STATUS_negative_2 -2.401e-01  1.507e-02 -15.932  < 2e-16 ***
AGE                1.805e-03  2.243e-03   0.805 0.420967
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 18394  on 17480  degrees of freedom
Residual deviance: 16947  on 17472  degrees of freedom
AIC: 16965

Number of Fisher Scoring iterations: 5
```

3. Predict the outcome using predict() function
   test.probs<-predict(fit,test,type='response')

4. Divide the predicted values based on probability values
   pred<- rep(1,length(test.probs))
   pred[test.probs<=0.5]<-0

5. Create the error table
   logisticregression_table<-table(pred,test$default_response)

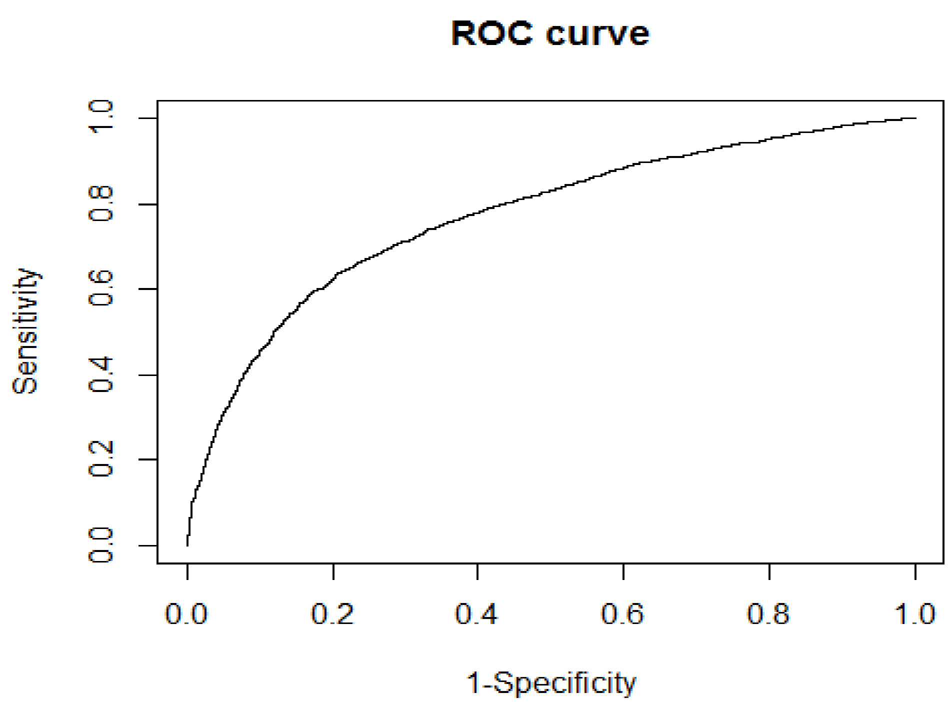
6. Create confusion matrix
   confusionMatrix(test$default_response,pred)

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 9044  127
         1 2326  158

               Accuracy : 0.7895
                 95% CI : (0.782, 0.7969)
    No Information Rate : 0.9755
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0735
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.79543
            Specificity : 0.55439
         Pos Pred Value : 0.98615
         Neg Pred Value : 0.06361
             Prevalence : 0.97555
         Detection Rate : 0.77598
   Detection Prevalence : 0.78687
      Balanced Accuracy : 0.67491

       'Positive' Class : 0
```

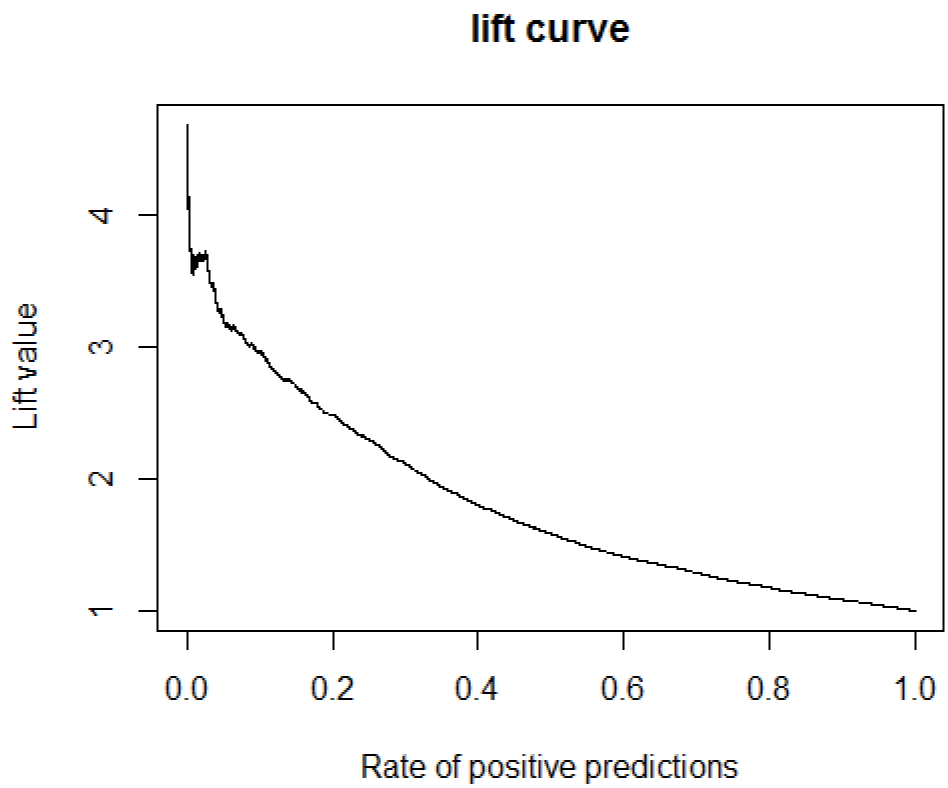
7. Create the ROC curve and Lift Chart

```
#create ROC curve
prediction <- prediction(test.probs, test$default_response)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

#Lift curve
perf <- performance(prediction,"lift","rpp")
plot(perf, main="lift curve")
```

<u>ROC Curve</u>



LIFT CURVE



CONFUSION MATRIX

```
> Error_logistic_regression *100
[1] 21.51682
>
> #create the confusion matrix
> confusionMatrix(test$default_response,pred)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4425   94
         1 1160  149

               Accuracy : 0.7848
                 95% CI : (0.7741, 0.7953)
    No Information Rate : 0.9583
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1309
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7923
            Specificity : 0.6132
         Pos Pred Value : 0.9792
         Neg Pred Value : 0.1138
             Prevalence : 0.9583
         Detection Rate : 0.7593
   Detection Prevalence : 0.7754
      Balanced Accuracy : 0.7027

       'Positive' Class : 0
```

CLASSIFICATION TREE

1. Divide the data into test and train data
   set.seed(2)
   smp_size<-floor(0.60*nrow(df))
   set.seed(123)
   train<-sample(seq_len(nrow(df)),size=smp_size)
   df.test<-df[-train,]
   default_response.test <- df$default_response[-train]

2. Construct the classification tree model
   tree.train<- tree(as.factor(default_response)~.,df,subset=train)
   summary(tree.train)

3. Predict the outcome using predict() function
   tree.pred = predict(tree.train,df.test,type="class")

4. Create the error table
   classification_tree<-table(tree.pred,default_response.test)

5. Create confusion matrix
   confusionMatrix(default_response.test,tree.pred)

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 8361  810
         1 1482 1002

               Accuracy : 0.8033
                 95% CI : (0.796, 0.8105)
    No Information Rate : 0.8445
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3495
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8494
            Specificity : 0.5530
         Pos Pred Value : 0.9117
         Neg Pred Value : 0.4034
             Prevalence : 0.8445
         Detection Rate : 0.7174
   Detection Prevalence : 0.7869
      Balanced Accuracy : 0.7012

       'Positive' Class : 0
```

6. Create the ROC curve and Lift Chart

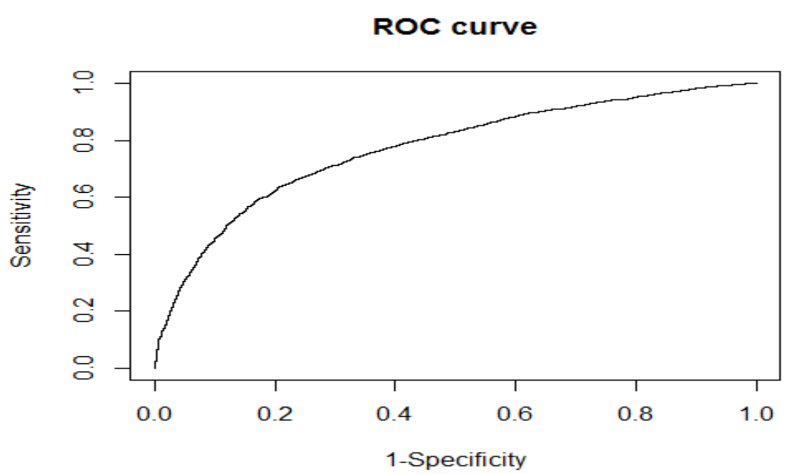
   #create ROC curve
   prediction <- prediction(tree.pred, default_response.test)
   performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
   plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

#Lift curve
perf <- performance(prediction,"lift","rpp")
plot(perf, main="lift curve")

CONFUSION MATRIX

```
  (classification_tree[1,1]+(classification_tree[2,2])))
>  Error*100
[1] 19.66538
> #classification_tree_fin <-cbind("Error",ErrorFin)
>
>
> #create confusion matrix
> confusionMatrix(default_response.test,tree.pred)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 8361  810
         1 1482 1002

               Accuracy : 0.8033
                 95% CI : (0.796, 0.8105)
    No Information Rate : 0.8445
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3495
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8494
            Specificity : 0.5530
         Pos Pred Value : 0.9117
         Neg Pred Value : 0.4034
             Prevalence : 0.8445
         Detection Rate : 0.7174
   Detection Prevalence : 0.7869
      Balanced Accuracy : 0.7012

       'Positive' Class : 0
```

ROC CURVE



NEURAL NETWORK

1. Divide the data into test and train data
   set.seed(2)
   smp_size<-floor(0.60*nrow(df))
   set.seed(123)
   train<-sample(seq_len(nrow(df)),size=smp_size)
   test<-df[-train,]

2. Construct the Neural Network model
   seedsANN = nnet(default_response~.,df[train,], hidden=3,size=3,rang = 0.1, decay = 5e-4, maxit = 200,MaxNWts = 1000)
3. Predict the outcome using predict() function

pr<-predict(seedsANN, test)

4. Plot the neural network
   plotnet(seedsANN,alpha=0.5)

5. Divide the predicted status based on probability values
   pred<- rep(1,length(pr))
   pred[pr<=0.25]<-0

6. Create the error table
   neural_network_table<-table(pred,test$default_response)

```
          Reference
Prediction    0    1
         0 9170    1
         1 2483    1
```

7. Create confusion matrix
   confusionMatrix(test$default_response,pred)

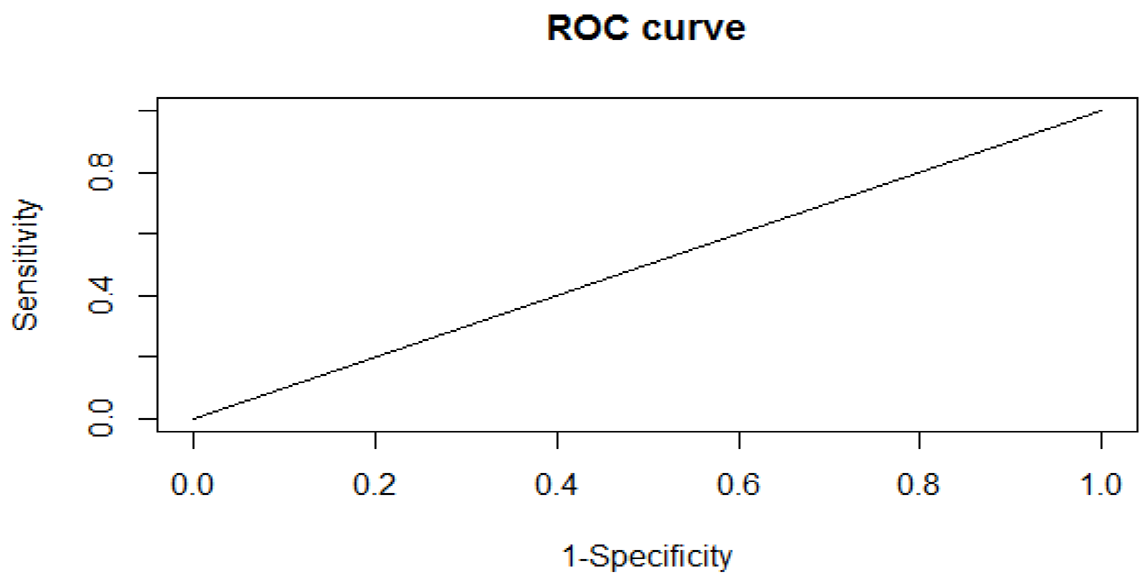
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 9170    1
         1 2483    1

               Accuracy : 0.7869
                 95% CI : (0.7793, 0.7943)
    No Information Rate : 0.9998
    P-Value [Acc > NIR] : 1

                  Kappa : 5e-04
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7869218
            Specificity : 0.5000000
         Pos Pred Value : 0.9998910
         Neg Pred Value : 0.0004026
             Prevalence : 0.9998284
         Detection Rate : 0.7867868
   Detection Prevalence : 0.7868726
      Balanced Accuracy : 0.6434609

       'Positive' Class : 0
```

8. Create the ROC curve and Lift Chart

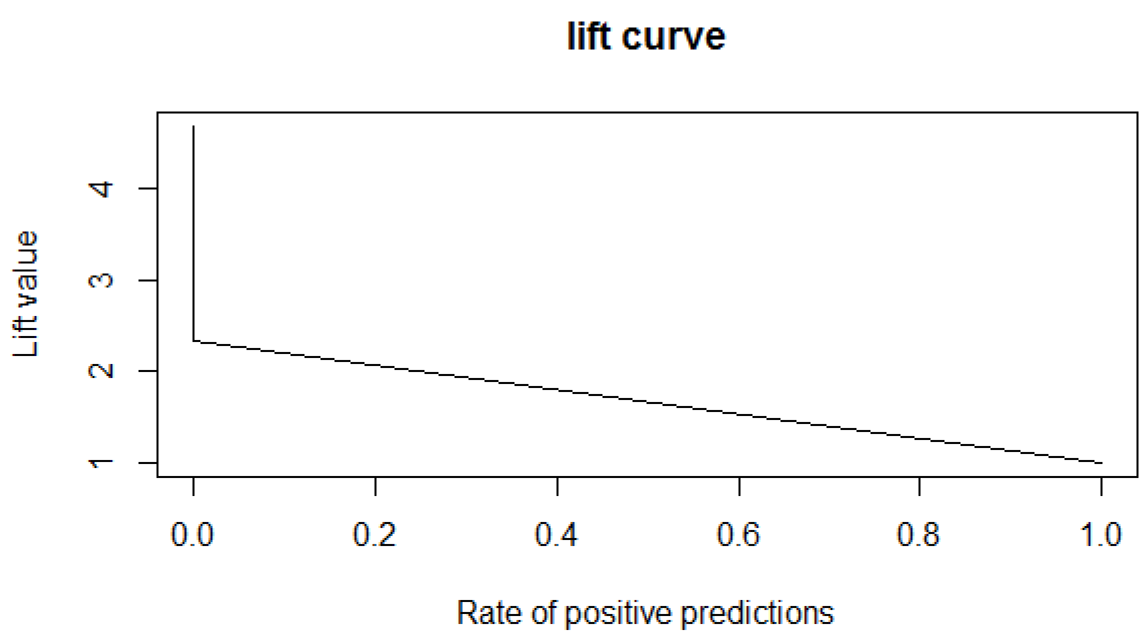
   #create ROC curve
   install.packages("ROCR")
   library(ROCR)
   prediction <- prediction(pr, test$default_response)
   performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
   plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

```
#create Lift curve
perf <- performance(prediction,"lift","rpp")
plot(perf, main="lift curve")
```

<u>ROC Curve</u>



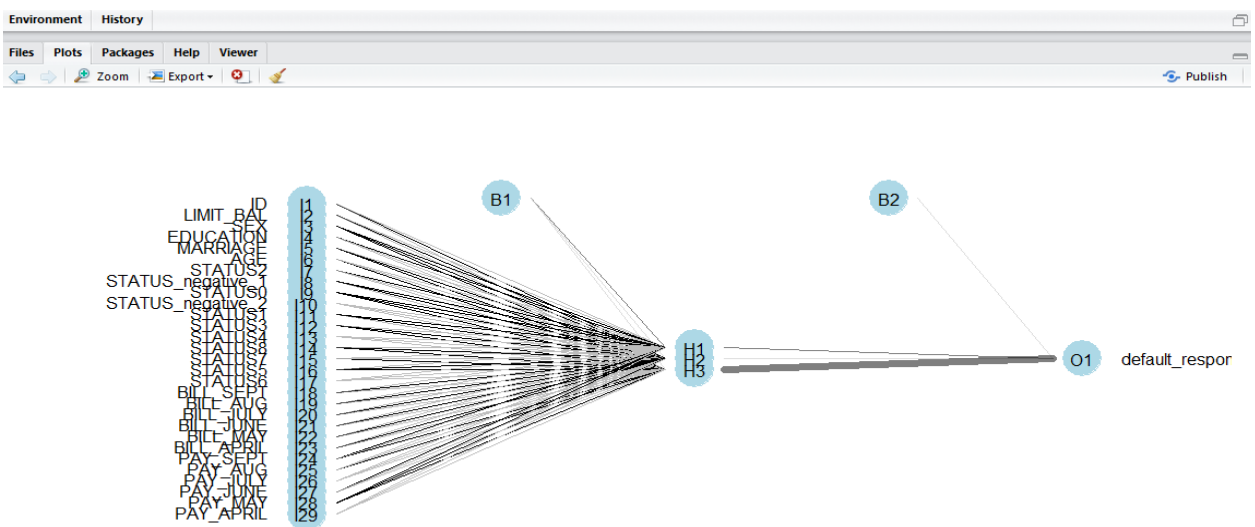
<u>Lift Curve</u>



Confusion Matrix

```
> pred[pr<=0.25]<-0
> neural_network_table<-table(pred,test$default_response)
> View(table)
>
> #create confusion matrix
> conMatrix<-confusionMatrix(test$default_response,pred)
> View(conMatrix)
Error in View : cannot coerce class ""confusionMatrix"" to a data.frame
> conMatrix
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 9170    1
         1 2483    1

               Accuracy : 0.7869
                 95% CI : (0.7793, 0.7943)
    No Information Rate : 0.9998
    P-Value [Acc > NIR] : 1

                  Kappa : 5e-04
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7869218
            Specificity : 0.5000000
         Pos Pred Value : 0.9998910
         Neg Pred Value : 0.0004026
             Prevalence : 0.9998284
         Detection Rate : 0.7867868
   Detection Prevalence : 0.7868726
      Balanced Accuracy : 0.6434609

       'Positive' Class : 0
```

NEURAL NETWORK PLOT



<u>**We choose Classification Tree over all other models**</u>

REASONS FOR CHOOSING CLASSIFICATION TREE?

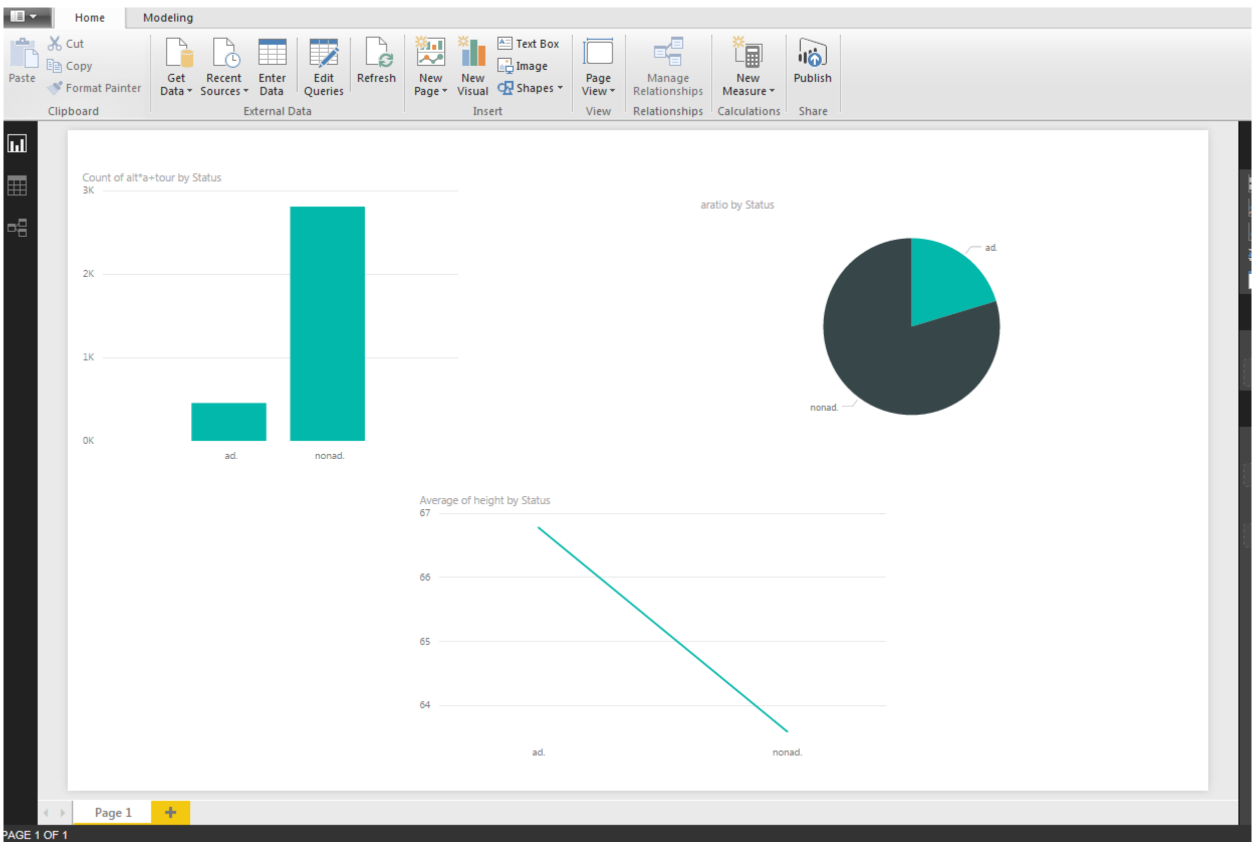
- **High Accuracy**
- **High Sensitivity**
- **Good positive prediction**
- **Low error percentage**

Overall Error Percentage

Logistic Regression: 21.04%
Classification Tree:19.65%
Neural Network: 21.01%

Q.2: ADVERTISEMENT ON INTERNET PAGES

-VISUALISATION



1. Perform cleaning on the dataset using the following code.

#Removing the first row ad [nonad|classes]
names_rm2 <- new[-c(1),]

#Removing the additional rows with no binary values (pipeline)
names_rm3<-
names_rm2[!(names_rm2$X..w..c4.5.alladA.names.file....automatically.generated.=="")]

```r
#binding the three rows in the beginning - height, width, ratio
names_rm4<-rbind(names_rm2[c(1:3),],names_rm3)

#Changing the name of the column by removing substrings - colons and zeroes in the end
clean_names_rm2<-sub(":.*","",names_rm4$rn)
removingcolon<-as.data.frame(clean_names_rm2)

#Transpose with column name as the first rows
transpose_names<-t(removingcolon)
colnames(transpose_names) <- transpose_names[1, ]

#Changing transpose_names matrix to dataframe
df1<-as.data.frame(transpose_names)

#Replacing "?" with the NA and changing the table.data to a data frame
df2<-as.data.frame(sapply(table.data,sub,pattern='\\?',replacement=NA))

#Removing the last column ad from df2 dataset
df2$V1559<-NULL

#Column names of transpose_names as Column names of table.data
colnames(df2) <- colnames(df1)


#Changing the 473rd, 534th and 956th column name because of multibyte error due to some absurd
characters
colnames(df2)[colnames(df2)=='origurl*target+\xfc\xbe\x99\x96\x84\xbcion']<-'origurl*target'
colnames(df2)[colnames(df2)=='origurl*\xfc\xbe\x99\x96\x84\xbcion+0']<-'origurl*534'
colnames(df2)[colnames(df2)=='origurl*\xfc\xbe\x99\x96\x84\xbcion']<-'origurl*956'

#Replacing the NA values in height with mean of the height column
height<-as.numeric(as.character((df2$height)))
install.packages("gtools")
library(gtools)
mean_height<-mean(height, na.rm=TRUE)
height<-na.replace(height, mean_height)
df2$height<-height
#View(df2$height)

#Replacing the NA values in width with mean of the width column
width<-as.numeric(as.character((df2$width)))
mean_width<-mean(width, na.rm=TRUE)
width<-na.replace(width, mean_width)
df2$width<-width

#Finding out the third column's NA values by dividing height by width
aratio<-as.numeric(as.character((df2$aratio)))
aratio_rep<-na.replace(aratio, 0)
df2$aratio<-aratio_rep
na_locations <- which(df2$aratio==0, arr.ind = TRUE)
df2$aratio[na_locations] <- df2$width[na_locations]/df2$height[na_locations]


#Adding the status column that would tell id the advertisement is ad or nonad
```

df3<-cbind(df2,Status=table.data$V1559)

#Remove the rows with NA values in "local" column
omitted_na<-na.omit(df3)

<u>LOGISTIC REGRESSION</u>

1. Divide the data into test and train data

   # Take 75% of the data as the sample data
   smp_size<-floor(0.75*nrow(df))
   set.seed(123)
   #Divide the data into train and test data. Sample data is basically train data
   train_ind<-sample(seq_len(nrow(df)),size=smp_size)
   train <- df[train_ind,]
   # Rest 25% is test data
   test <- df[-train_ind,]

2. Construct the logistic regression model

   fit<-glm(Status~.,data=train,family=binomial(link="logit"))
   summary(fit)

```
> Error_logistic_regression *100
[1] 21.287
> confusionMatrix(test$Status,pred)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.     91    28
    nonad.  31   666

               Accuracy : 0.9277
                 95% CI : (0.9077, 0.9445)
    No Information Rate : 0.8505
    P-Value [Acc > NIR] : 1.069e-11

                  Kappa : 0.7128
 Mcnemar's Test P-Value : 0.7946

            Sensitivity : 0.7459
            Specificity : 0.9597
         Pos Pred Value : 0.7647
         Neg Pred Value : 0.9555
             Prevalence : 0.1495
         Detection Rate : 0.1115
   Detection Prevalence : 0.1458
      Balanced Accuracy : 0.8528

       'Positive' Class : ad.

>
```

3. Predict the outcome using predict() function

   test.probs<-predict(fit,test,type='response')

4. Divide the predicted values based on probability values

   pred<- rep("nonad.",length(test.probs))
   pred[test.probs<=0.5]<-"ad."

5. Create the error table and calculate error percentage

logisticregression_table<-table(pred,test$Status)
Error_logistic_regression<- ((logisticregression_table[1,2]) + (logisticregression_table [2,1])) /((logisticregression_table 2,1]) + (logisticregression_table [1,2]) + (logisticregression_table [1,1])+(logisticregression_table [2,2]))
Error_logistic_regression *100

```
> Error_logistic_regression *100
[1] 21.287
```

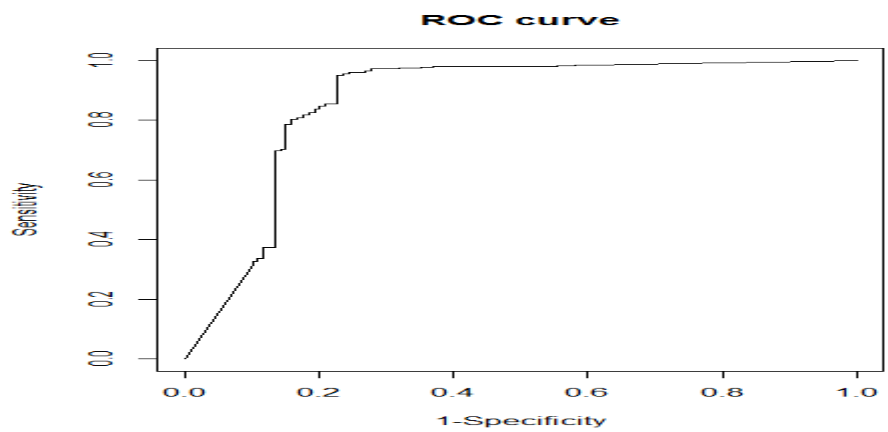
6. Create confusion matrix
   confusionMatrix(test$Status,pred)

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

          Reference
Prediction ad. nonad.
    ad.     91     28
    nonad.  31    666

               Accuracy : 0.9277
                 95% CI : (0.9077, 0.9445)
    No Information Rate : 0.8505
    P-Value [Acc > NIR] : 1.069e-11

                  Kappa : 0.7128
 Mcnemar's Test P-Value : 0.7946

            Sensitivity : 0.7459
            Specificity : 0.9597
         Pos Pred Value : 0.7647
         Neg Pred Value : 0.9555
             Prevalence : 0.1495
         Detection Rate : 0.1115
   Detection Prevalence : 0.1458
      Balanced Accuracy : 0.8528

       'Positive' Class : ad.
```

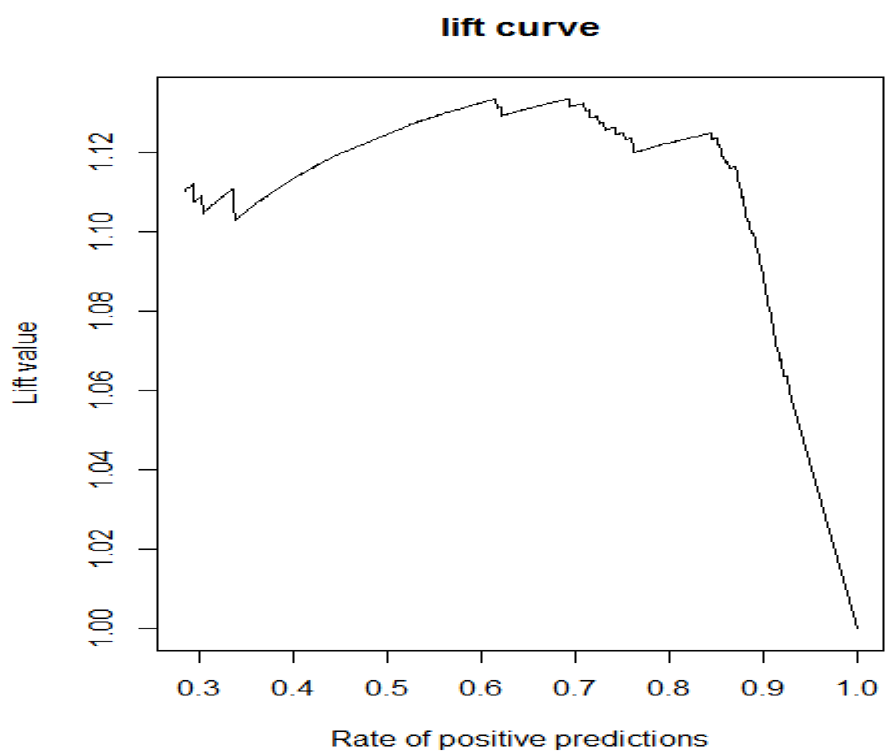
7. Create the ROC curve and Lift Chart
   #create ROC curve
   install.packages("ROCR")
   library(ROCR)
   prediction <- prediction(test.probs, test$Status)
   performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
   plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

   #create Lift curve
   perf <- performance(prediction,"lift","rpp")
   plot(perf, main="lift curve")

   <u>ROC Curve</u>


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

LIFT CURVE


lift curve
Lift value
Rate of positive predictions
1.00
1.02
1.04
1.06
1.08
1.10
1.12
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

CONFUSION MATRIX

```
> Error_logistic_regression *100
[1] 21.287
> confusionMatrix(test$Status,pred)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.     91     28
    nonad.  31    666

               Accuracy : 0.9277
                 95% CI : (0.9077, 0.9445)
    No Information Rate : 0.8505
    P-Value [Acc > NIR] : 1.069e-11

                  Kappa : 0.7128
 Mcnemar's Test P-Value : 0.7946

            Sensitivity : 0.7459
            Specificity : 0.9597
         Pos Pred Value : 0.7647
         Neg Pred Value : 0.9555
             Prevalence : 0.1495
         Detection Rate : 0.1115
   Detection Prevalence : 0.1458
      Balanced Accuracy : 0.8528

       'Positive' Class : ad.

>
```

CLASSIFICATION TREE

1. Divide the data into test and train data
   set.seed(2)
   smp_size<-floor(0.90*nrow(df))
   set.seed(123)
   train<-sample(seq_len(nrow(df)),size=smp_size)
   df.test<-df[-train,]
   Status.test <- df$Status[-train]

2. Construct the classification tree model
   fit<- tree(Status~.,df,subset=train)
   summary(tree)

```
Classification tree:
tree(formula = Status ~ ., data = df, subset = train)
Variables actually used in tree construction:
 [1] "width"          "ancurl.com"      "url.ads"        "ancurl.click"     "ancurl.http.www"  "url.bin"         "alt.click"       "url.images.ho
me"
 [9] "ancurl.home.html" "aratio"
Number of terminal nodes:  12
Residual mean deviance:  0.2301 = 673.2 / 2925
Misclassification error rate: 0.02996 = 88 / 2937
```

3. Predict the outcome using predict() function
   tree.pred = predict(fit,df.test,type="class")

4. Create the error table
   classification_tree<-table(tree.pred,Status.test)
   View(classification_tree)

```
> classification_tree
          Status.test
tree.pred ad. nonad.
   ad.     39      3
   nonad.   4    281
>
```

5. Create confusion matrix
   confusionMatrix(Status.test,tree.pred)

```
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.     39      4
    nonad.   3    281

               Accuracy : 0.9786
                 95% CI : (0.9564, 0.9914)
    No Information Rate : 0.8716
    P-Value [Acc > NIR] : 3.936e-12

                  Kappa : 0.9053
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9286
            Specificity : 0.9860
         Pos Pred Value : 0.9070
         Neg Pred Value : 0.9894
             Prevalence : 0.1284
         Detection Rate : 0.1193
   Detection Prevalence : 0.1315
      Balanced Accuracy : 0.9573

       'Positive' Class : ad.
```

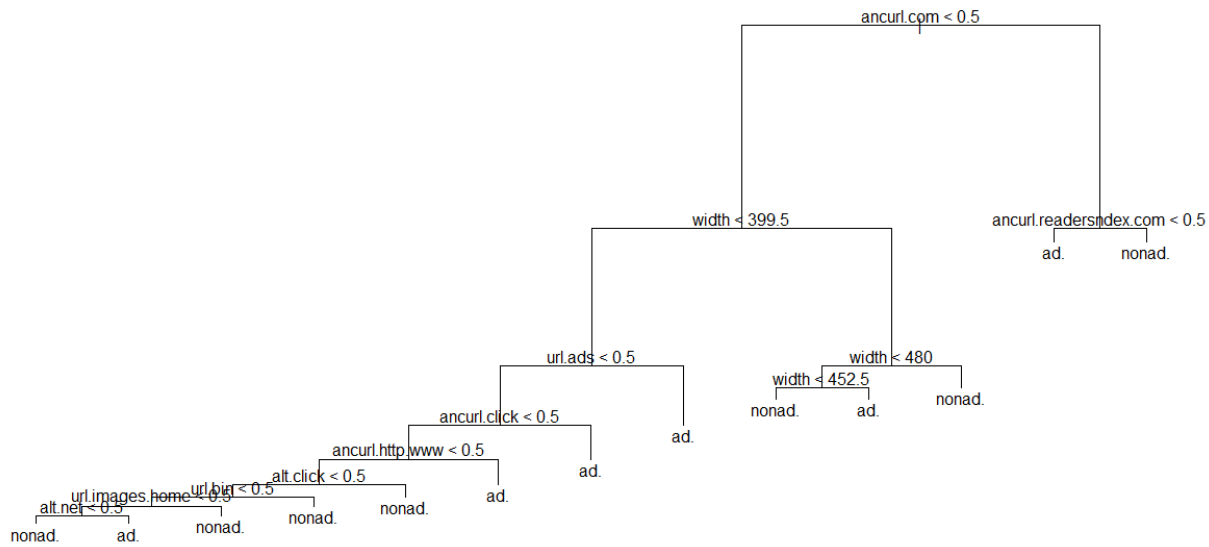
6. Create the ROC curve and Lift Chart
   install.packages("ROCR")
   library(ROCR)
   str(tree.pred)
   prediction <- prediction(tree.pred, Status.test$Status)

```
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

#create Lift curve
perf <- performance(prediction,"lift","rpp")
plot(perf, main="lift curve")
```

<u>**CONFUSION MATRIX**</u>

```
> confusionMatrix(Status.test,tree.pred)
Confusion Matrix and Statistics

          Reference
Prediction ad. nonad.
    ad.     39      4
    nonad.   3    281

               Accuracy : 0.9786
                 95% CI : (0.9564, 0.9914)
    No Information Rate : 0.8716
    P-Value [Acc > NIR] : 3.936e-12

                  Kappa : 0.9053
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9286
            Specificity : 0.9860
         Pos Pred Value : 0.9070
         Neg Pred Value : 0.9894
             Prevalence : 0.1284
         Detection Rate : 0.1193
   Detection Prevalence : 0.1315
      Balanced Accuracy : 0.9573

       'Positive' Class : ad.

>
```

CLASSIFICATION TREE



NEURAL NETWORK

1. Divide the data into test and train data
   set.seed(2)
   smp_size<-floor(0.50*nrow(df))
   set.seed(123)
   train<-sample(seq_len(nrow(df)),size=smp_size)
   test<-df[-train,]

2. Create new column i.e. Status_updated with '0' and '1' values.
   df["status_updated"]<-NA
   df$status_updated[df$Status == "ad."]<-0
   df$status_updated[df$Status == "nonad."]<-1

3. Construct the Neural Network model
   seedsANN = nnet(status_updated~.,df[train,], size=3,rang = 0.1,decay = 5e-4, maxit = 350,MaxNWts = 5000)
4. Predict the outcome using predict() function
   pr<-predict(seedsANN, test)

5. Plot the neural network
   plotnet(seedsANN,alpha=0.5)

6. Divide the predicted status based on probability values
   pred<- rep(1,length(pr))
   pred[pr<=0.5]<-0

7. Create the error table
   neural_network_table<-table(pred,test$status_updated)

8. Create confusion matrix
   confusionMatrix(test$status_updated,pred)

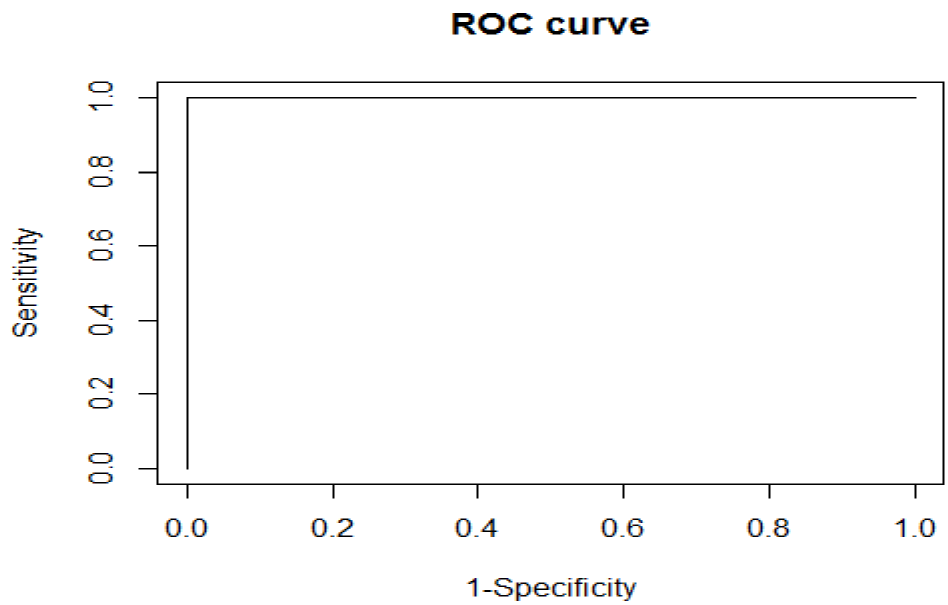
```
> confusionMatrix(test$status_updated,pred)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  228    0
         1    1 1403

               Accuracy : 0.9994
                 95% CI : (0.9966, 1)
    No Information Rate : 0.8597
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9975
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9956
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9993
             Prevalence : 0.1403
         Detection Rate : 0.1397
   Detection Prevalence : 0.1397
      Balanced Accuracy : 0.9978

       'Positive' Class : 0
```

9. Create the ROC curve and Lift Chart

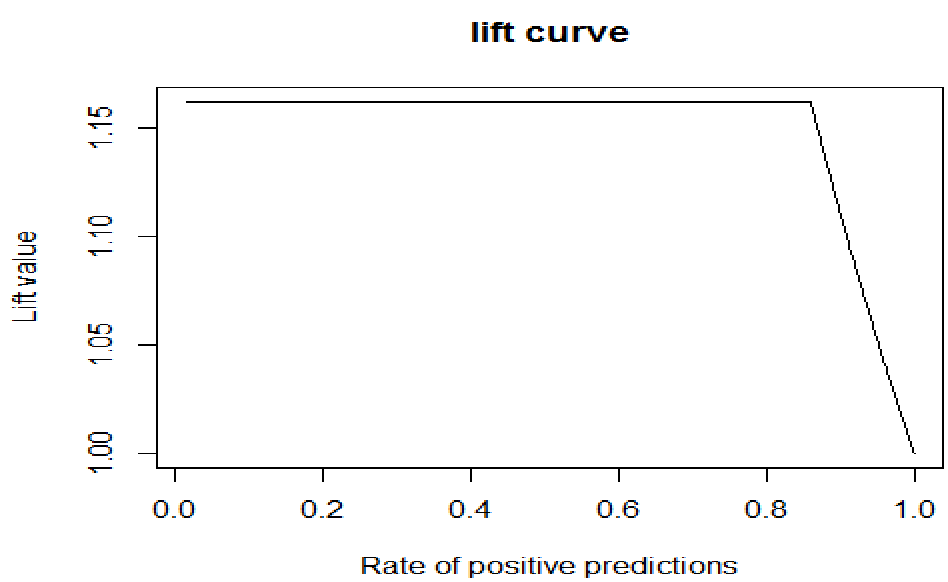
   #create ROC curve
   install.packages("ROCR")
   library(ROCR)
   prediction <- prediction(pr, test$default_response)
   performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
   plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

   #create Lift curve
   perf <- performance(prediction,"lift","rpp")
   plot(perf, main="lift curve")

ROC Curve



Lift Curve



<u>CONFUSION MATRIX</u>

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

          Reference
Prediction    0    1
         0  228    0
         1    1 1403

               Accuracy : 0.9994
                 95% CI : (0.9966, 1)
    No Information Rate : 0.8597
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9975
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9956
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9993
             Prevalence : 0.1403
         Detection Rate : 0.1397
   Detection Prevalence : 0.1397
      Balanced Accuracy : 0.9978

       'Positive' Class : 0
```

<u>We choose Neural Network over all other models</u>

REASONS FOR CHOOSING NEURAL NETWORK?

Neural Network have the best accuracy, sensitivity and specificity for this problem.

It has better ROC curve than other models.

Q. Error Percentage for Advertisement on Internet Pages.

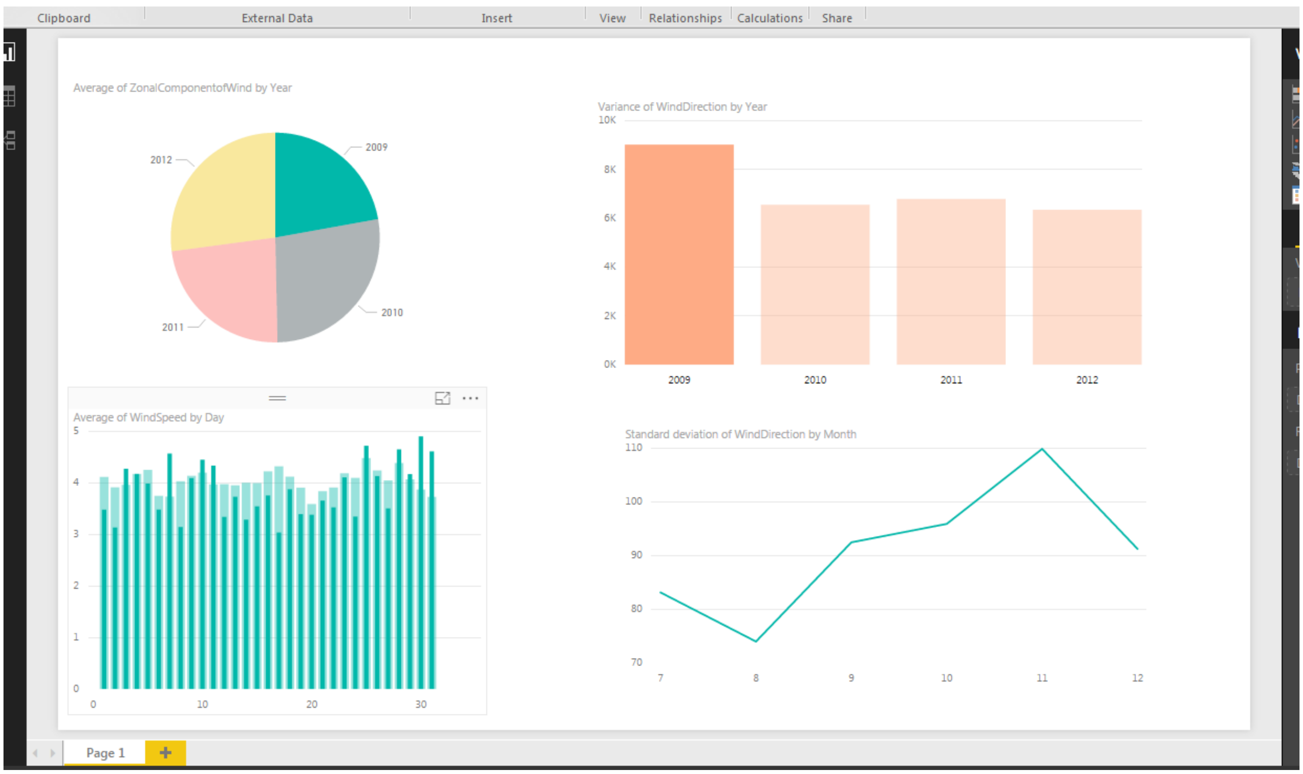
Logistic Regression: 7.23%

Classification Tree: 2.14%

Neural Network: 0.67%

Problem – 3 Wind Forecast

The problem describes the wind forecasting in an hour for the next 48 hours in the interval of 12 hours. The following observation was made while visualizing data in Power BI tool.



The following is the steps in cleansing the data and outputting the values

1) The windforecast_wp1.csv, windforecast_wp2.csv, windforecast_wp3.csv, windforecast_wp4.csv, windforecast_wp5.csv, windforecast_wp6.csv and windforecast_wp7.csv files are the files with u(Zonal wind component), v(Meridional wind component), ws(wind speed) and wd(wind direction)

2) Took input files as above mentioned CSVs one after the other. The “lubridate” library is used to format the dates and segregating the year, month, day and hour.

3) Used aggregate function to cumulate the data values u, v, ws and wd for common date with hour

4) The date was displayed as it was in the original file after adding hours to the date.

5) The merged data is obtained by using the training file of the corresponding windforecast file.

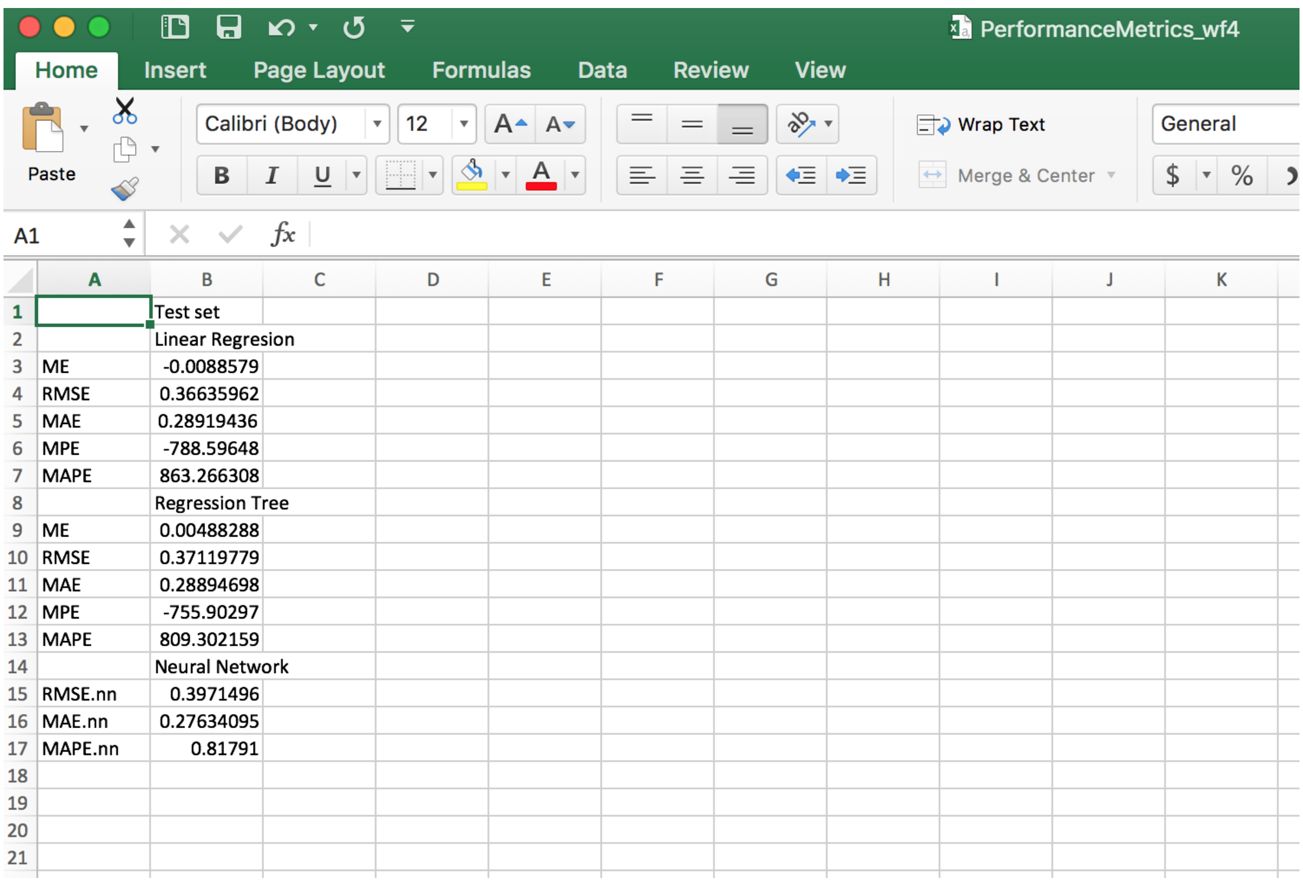
6) All the NAs are replaced by the zoo package's na.locf function. This provided the best accuracy for building the model. The training data is from the 2009/07/01 to 2010/12/31 and the testing data is from the dates 2011/01/01 to 2012/06/28.

7) The regression models are built for this dataset. Three models – Linear Regression, Regression Tree and the Neural Networks.

8) The RMSE, MAPE and MAE Values are predicted for each of the 7 wind forecast csv files.

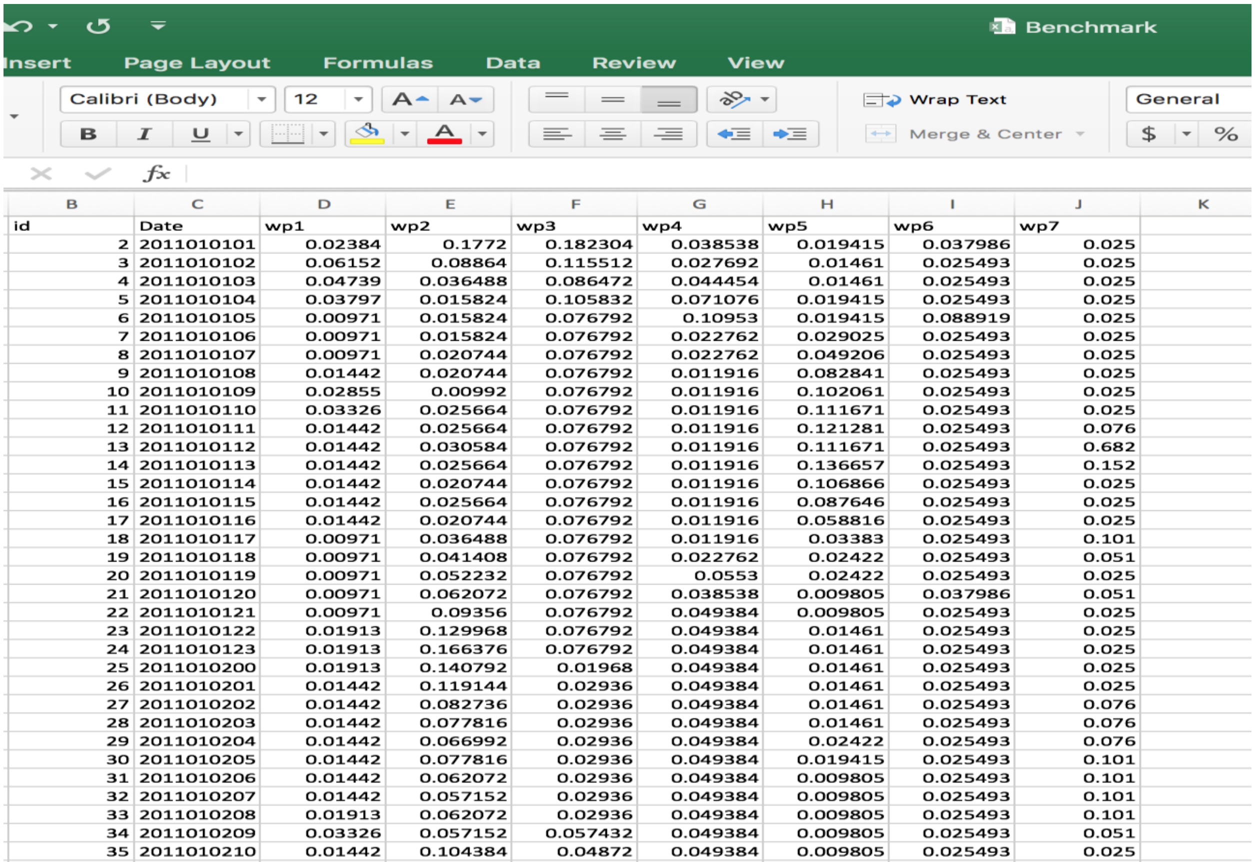
9) On comparing the Performance Metrics of each model, it was concluded that the best model is the Neural Network as it has the least MAPE error as compared to other 2 models.

10) Finally all the predicted values are merged in the following file.



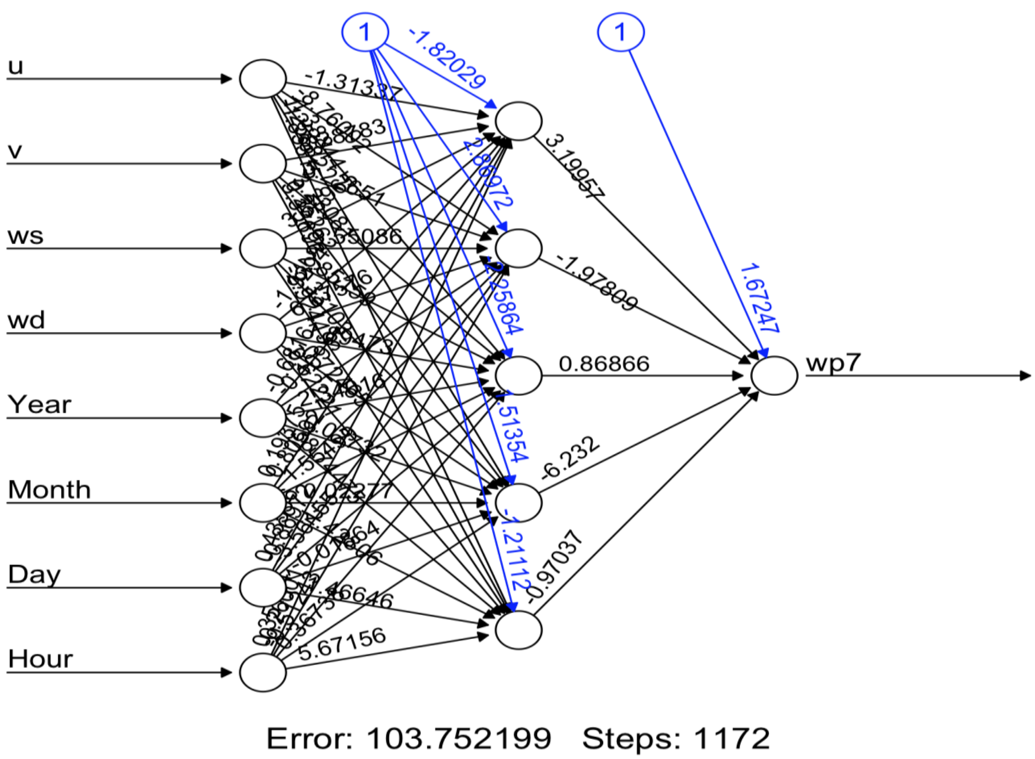
	A	B
1		Test set
2		Linear Regresion
3	ME	-0.0088579
4	RMSE	0.36635962
5	MAE	0.28919436
6	MPE	-788.59648
7	MAPE	863.266308
8		Regression Tree
9	ME	0.00488288
10	RMSE	0.37119779
11	MAE	0.28894698
12	MPE	-755.90297
13	MAPE	809.302159
14		Neural Network
15	RMSE.nn	0.3971496
16	MAE.nn	0.27634095
17	MAPE.nn	0.81791

Performance Metrics for wind forecast file



B	C	D	E	F	G	H	I	J	K
id	Date	wp1	wp2	wp3	wp4	wp5	wp6	wp7	
2	2011010101	0.02384	0.1772	0.182304	0.038538	0.019415	0.037986	0.025	
3	2011010102	0.06152	0.08864	0.115512	0.027692	0.01461	0.025493	0.025	
4	2011010103	0.04739	0.036488	0.086472	0.044454	0.01461	0.025493	0.025	
5	2011010104	0.03797	0.015824	0.105832	0.071076	0.019415	0.025493	0.025	
6	2011010105	0.00971	0.015824	0.076792	0.10953	0.019415	0.088919	0.025	
7	2011010106	0.00971	0.015824	0.076792	0.022762	0.029025	0.025493	0.025	
8	2011010107	0.00971	0.020744	0.076792	0.022762	0.049206	0.025493	0.025	
9	2011010108	0.01442	0.020744	0.076792	0.011916	0.082841	0.025493	0.025	
10	2011010109	0.02855	0.00992	0.076792	0.011916	0.102061	0.025493	0.025	
11	2011010110	0.03326	0.025664	0.076792	0.011916	0.111671	0.025493	0.025	
12	2011010111	0.01442	0.025664	0.076792	0.011916	0.121281	0.025493	0.076	
13	2011010112	0.01442	0.030584	0.076792	0.011916	0.111671	0.025493	0.682	
14	2011010113	0.01442	0.025664	0.076792	0.011916	0.136657	0.025493	0.152	
15	2011010114	0.01442	0.020744	0.076792	0.011916	0.106866	0.025493	0.025	
16	2011010115	0.01442	0.025664	0.076792	0.011916	0.087646	0.025493	0.025	
17	2011010116	0.01442	0.020744	0.076792	0.011916	0.058816	0.025493	0.025	
18	2011010117	0.00971	0.036488	0.076792	0.011916	0.03383	0.025493	0.101	
19	2011010118	0.00971	0.041408	0.076792	0.022762	0.02422	0.025493	0.051	
20	2011010119	0.00971	0.052232	0.076792	0.0553	0.02422	0.025493	0.025	
21	2011010120	0.00971	0.062072	0.076792	0.038538	0.009805	0.037986	0.051	
22	2011010121	0.00971	0.09356	0.076792	0.049384	0.009805	0.025493	0.025	
23	2011010122	0.01913	0.129968	0.076792	0.049384	0.01461	0.025493	0.025	
24	2011010123	0.01913	0.166376	0.076792	0.049384	0.01461	0.025493	0.025	
25	2011010200	0.01913	0.140792	0.01968	0.049384	0.01461	0.025493	0.025	
26	2011010201	0.01442	0.119144	0.02936	0.049384	0.01461	0.025493	0.025	
27	2011010202	0.01442	0.082736	0.02936	0.049384	0.01461	0.025493	0.076	
28	2011010203	0.01442	0.077816	0.02936	0.049384	0.01461	0.025493	0.076	
29	2011010204	0.01442	0.066992	0.02936	0.049384	0.02422	0.025493	0.076	
30	2011010205	0.01442	0.077816	0.02936	0.049384	0.019415	0.025493	0.101	
31	2011010206	0.01442	0.062072	0.02936	0.049384	0.009805	0.025493	0.101	
32	2011010207	0.01442	0.057152	0.02936	0.049384	0.009805	0.025493	0.101	
33	2011010208	0.01913	0.062072	0.02936	0.049384	0.009805	0.025493	0.101	
34	2011010209	0.03326	0.057152	0.057432	0.049384	0.009805	0.025493	0.051	
35	2011010210	0.01442	0.104384	0.04872	0.049384	0.009805	0.025493	0.025	

Benchmark accumulated predicted values file



Neural Network Model