Midterm
Team 10: Suhas Pirankar, Zalak Shah
Course: Advances Data Science /Architecture

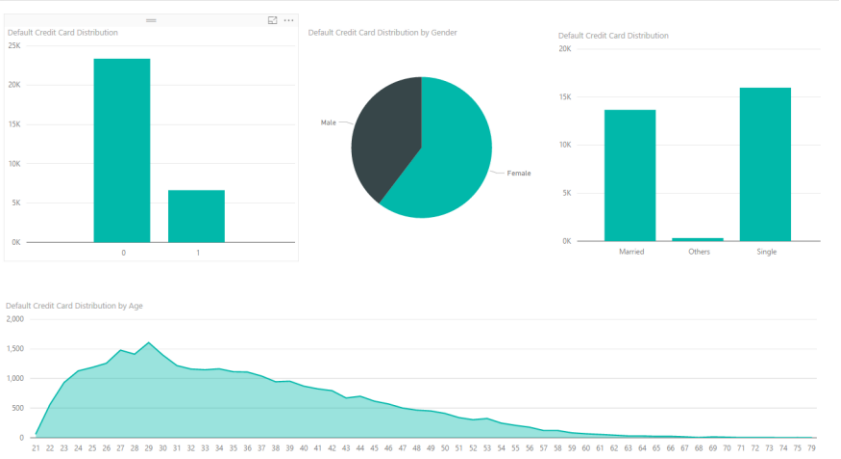
Problem 1: Credit Card Defaulter Data Set

Abstract: The dataset consists of the credit history for individual consumer and his/her family (supplementary) credit.

Propose Solution:
Built a predictive model using R in RStudio using the Logistic regression, Neural Networks and Classification Tree:

1.Data Exploration:

In order to explore the dataset we utilize the Power BI tool to visualize the data.



We visualize the data distribution based on Sex, Age and Marriage Status.

2. Data wrangling and cleansing:

In the following case, we used the feature engineering in order to in order the enhance the predictability of the model and derived the inference from the current fields.

- New Fields Created:
  - Total Bill Amount
  - Total Pay Amount
  - Count for the Pay Status (Frequency of the pay status for an induvial consumer)
- This are the derived fields used for the predictive analysis for the customer to do the Default or not

Logistic Regression:
We applied the logistic regression using the new derived fields and the avoiding the pay status of the 6 month credit history.
The accuracy for the model achieved by using the random sampling data to be 81 percent with sensitivity and specificity as 0.82 and 0.61 respectively.

We have considered the 14 derived and 5 original parameters for the creation of predictive model.

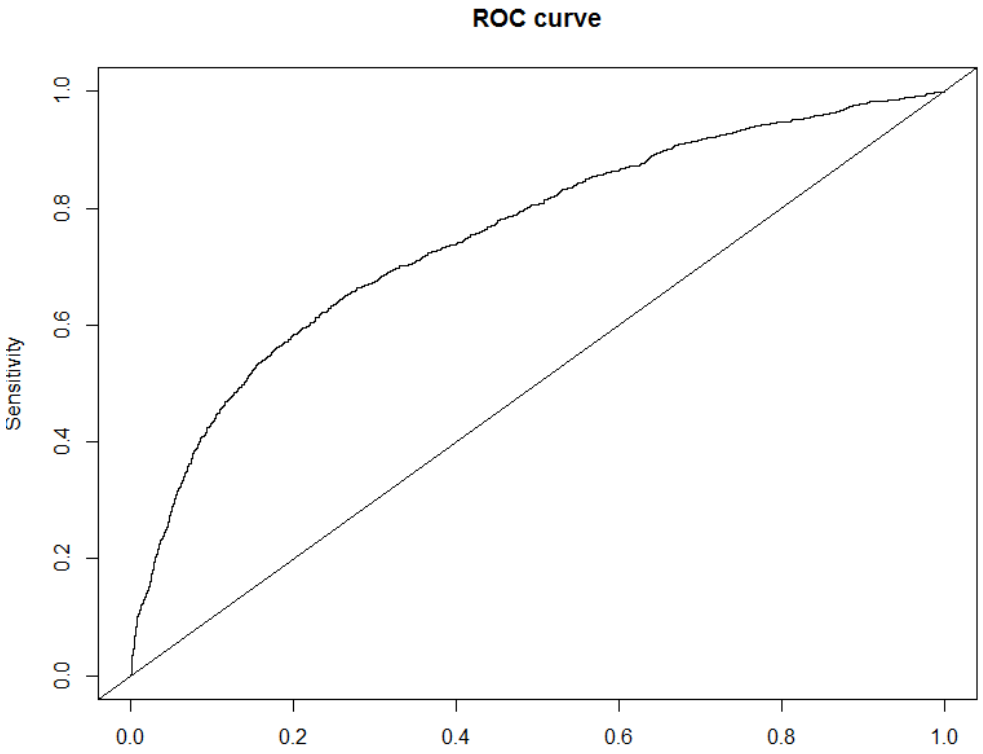
Confusion Matrix:

In here the Predictive model suggest that when the Prediction is Yes(i.e Default) then the accuracy is 1213 of (1213+416)

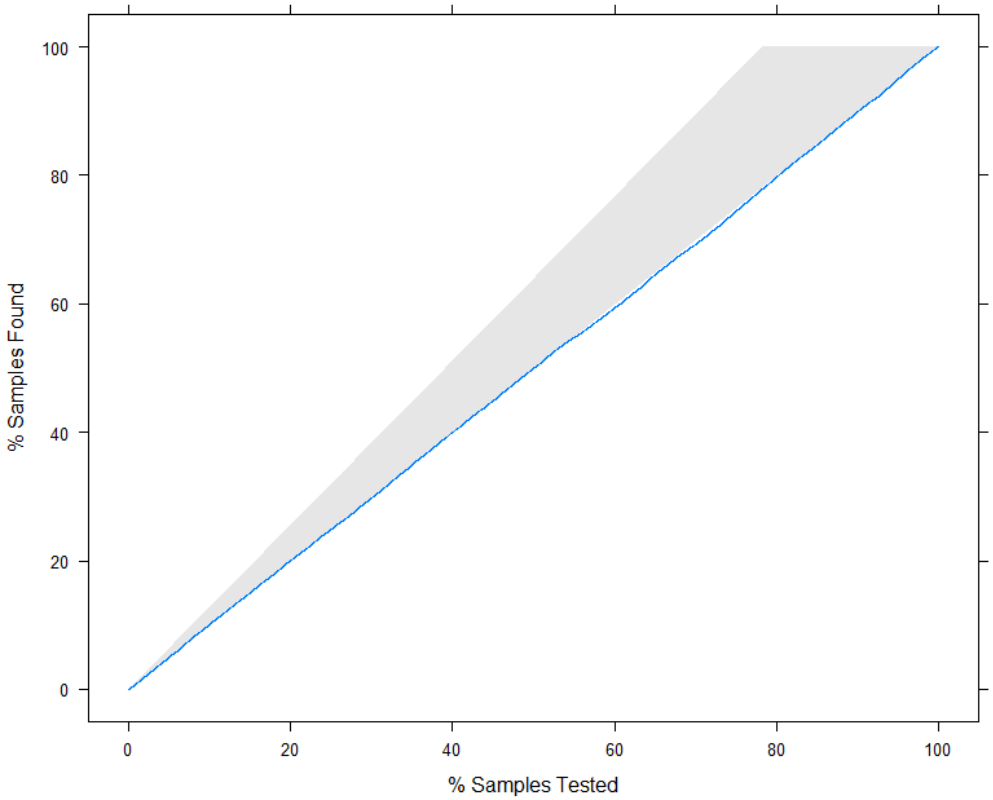
High Sensitivity measures the proportion of positives that are correctly identified and it is quantifying the false negatives.

```
          Reference
Prediction   No  Yes
       No  5607  264
       Yes 1213  416

               Accuracy : 0.8031
                 95% CI : (0.7939, 0.812)
    No Information Rate : 0.9093
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2665
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8221
            Specificity : 0.6118
         Pos Pred Value : 0.9550
         Neg Pred Value : 0.2554
             Prevalence : 0.9093
         Detection Rate : 0.7476
   Detection Prevalence : 0.7828
      Balanced Accuracy : 0.7170

       'Positive' Class : No
```

ROC Curve

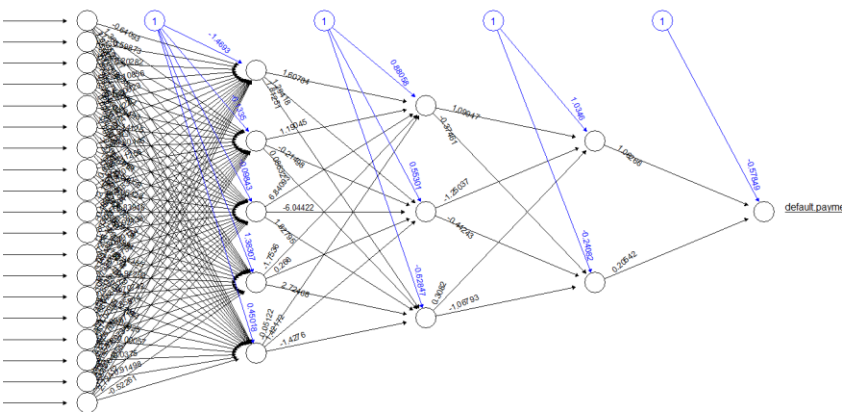


Lift Curve:



Neural Network

Neural Network Plot:



- We used the 3 layers with 5 3 and 2 weighing factor nodes

Classification Tree:

The classification tree is implemented and the following is the summary Snippet:

```
Regression tree:
tree(formula = default.payment.next.month ~ LIMIT_BAL + SEX +
    EDUCATION + MARRIAGE + AGE + count_2 + count_1 + count0 +
    count1 + count2 + count3 + count4 + count5 + count6 + count7 +
    count8 + count9 + TotalBillAmount + TotalPayAmount, data = train)
Variables actually used in tree construction:
[1] "count2"         "TotalPayAmount"
Number of terminal nodes:  4
Residual mean deviance:  0.1447323 = 3255.898 / 22496
Distribution of residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.6187793 -0.1085355 -0.1085355  0.0000000 -0.1085355  0.8914645
```

Flow of Implementation:

- Gathered relevant data
- Explore the data in power BI tool.
- Applied the feature engineering for the creation of new field.
- Build a logistic, neural regression and classification tree model
- Computed MAPE, RMS and MAE with the data set.
- Computed the Confusion matrix.
- Used regression Trees and Neural Networks techniques to build

Problem 2

We have a dataset which has details of the images such as height, width aspect ratio. We have other details like alt text, action URL etc. Below is more clear and in-depth metadata information about data.

Height: continuous. | possibly missing
Width: continuous. | possibly missing
Aspect Ratio: continuous. | possibly missing
local: 0,1.
| 457 features from url terms, each of the form "url*term1+term2...";

| for example:
url*images+buttons: 0,1.
...
| 495 features from origurl terms, in same form; for example:
origurl*labyrinth: 0,1.
...
| 472 features from ancurl terms, in same form; for example:
ancurl*search+direct: 0,1.
...
| 111 features from alt terms, in same form; for example:
alt*your: 0,1.
...
| 19 features from caption terms
caption*and: 0,1.
...

Problem Statement:

- Build a model to determine if a given image is Ad or Non Ad:
- Perform exploratory data analysis using POWER BI.
- Build prediction models to determine if image is and Ad using Logistic regression, Neural Network and Classification trees
- Discuss the model and evaluate the model's performance.

Steps in Data Cleaning

- ✓ Replace missing values with NA
- ✓ Replace NA with Mean of the column
- ✓ Check for outliers and replace them with mean.
- ✓ Round values to the nearby integer
- ✓ Convert columns into factor and prepare them for regression
- ✓ Manipulate the 'Ad' string column and make 1 for Ad and 0 for Non Ad
- ✓ Split data into Training and Test data set

Below is the data Cleaning code:

```r
setwd("/Users/suhaspirankar/Desktop/DataScience/Midterm/Problem2/ad-dataset");
getwd()

#Read content from the date file
mydata=read.table("ad.data",sep=',')

# make the column numeric
  mydata$V1=as.numeric(mydata$V1)
  mydata$V2=as.numeric(mydata$V2)
  mydata$V3=as.numeric(mydata$V3)

# Detecting, filtering outliers and replacing with mean of column
  for(j in 1:3){
    for (i in 1:nrow(mydata)){
      if ((mydata[i,j] < (mean(mydata[[j]])-(1.5)*sd(mydata[[j]]))|
           (mydata[i,j] > (mean(mydata[[j]])+1.5*sd(mydata[[j]])))))
      {mydata[i,j]<-mean(mydata[[j]])}
    }
  }

#convert  numeric to NA
  mydata$V1[mydata$V1==1]<-NA;
  mydata$V2[mydata$V2==1]<-NA;
  mydata$V3[mydata$V3==1]<-NA;

# convert direct missing characters to NA
  mydata[,4]=as.numeric(as.character(mydata[,4]));

```

```r
# Replace NA with Mean
mydata$V1[is.na(mydata$V1)] <- mean(mydata$V1, na.rm = TRUE)
mydata$V2[is.na(mydata$V2)] <- mean(mydata$V2, na.rm = TRUE)
mydata$V3[is.na(mydata$V3)] <- mean(mydata$V3, na.rm = TRUE)
mydata$V4[is.na(mydata$V4)] <- mean(mydata$V4, na.rm = TRUE)

# round the result and make it as interger
mydata$V1=as.integer(round(mydata$V1,digits=1))
mydata$V2=as.integer(round(mydata$V2,digits=1))

mydata$V3=as.integer(round(mydata$V3,digits=1))
mydata$V4=as.integer(round(mydata$V4,digits=1))

# factor each column as we need to build a regression
for (i in 4:ncol(mydata)) {
  mydata[,i]=factor(mydata[,i]);
}


# Create a new column and if it is ad enter 1 or enter 0
mydata$AdClass=ifelse(mydata$V1559=='ad.',1,0);
```

Split data into Test and Train

```
# Split the data into train and test

smp_size = floor(0.7 * nrow(mydata))
set.seed(123)
train_ind = sample(seq_len(nrow(mydata)), size = smp_size)
train = mydata[train_ind, ]
test = mydata[-train_ind, ]
```

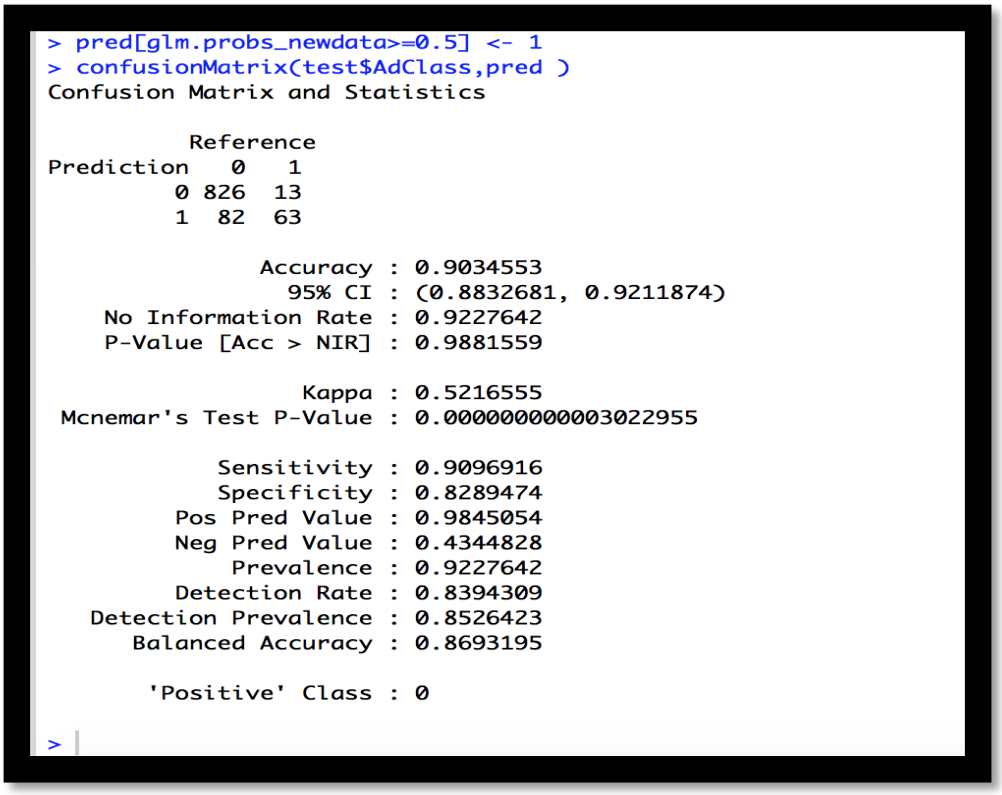
Logistic Regression:

```
Logistic_result=glm(formula=AdClass~V2+V3+V10+V32+V68,data = train,family =binomial)
summary(Logistic_result)

# Predict values so we get the probability
glm.probs=predict(Logistic_result ,type ="response");
summary(glm.probs)

# Generate the values using probability
glm_prediction=ifelse(glm.probs>0.1,"Ad","NoAd");
summary(glm_prediction)

glm_errorrate=sum(test$Predict!=glm_prediction)/length(test$AdClass);
summary(glm_errorrate)

#Predict using Model
glm.probs_newdata=predict(Logistic_result,newdata = test,type ="response");

glm_prediction_newdata=ifelse(glm.probs_newdata>0.1,"Ad.","No Ad.");
```

```
> summary(Logistic_result)

Call:
glm(formula = AdClass ~ V2 + V3 + V10 + V32 + V68, family = binomial,
    data = train)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.0497843  -0.4871518  -0.4668230  -0.3140013   3.1779910

Coefficients:
                Estimate    Std. Error   z value                  Pr(>|z|)
(Intercept) -2.6666293433  0.1988431924 -13.41071 < 0.0000000000000000222 ***
V2           0.0199447173  0.0022353265   8.92251 < 0.0000000000000000222 ***
V3          -0.0062917040  0.0007644385  -8.23049 < 0.0000000000000000222 ***
V10          3.1832841739  0.4853847816   6.55827       0.000000000054436 ***
V32          2.9340058453  0.8009782997   3.66303                 0.00024925 ***
V68          0.1300922456  0.4655926472   0.27941                 0.77992854
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1832.0851  on 2294  degrees of freedom
Residual deviance: 1614.7143  on 2289  degrees of freedom
AIC: 1626.7143

Number of Fisher Scoring iterations: 5
```

We have selected V2, V3, V10, V32, V68 as these have less p value and shows more stars.

Confusion Matrix for Logistic Regression Model.

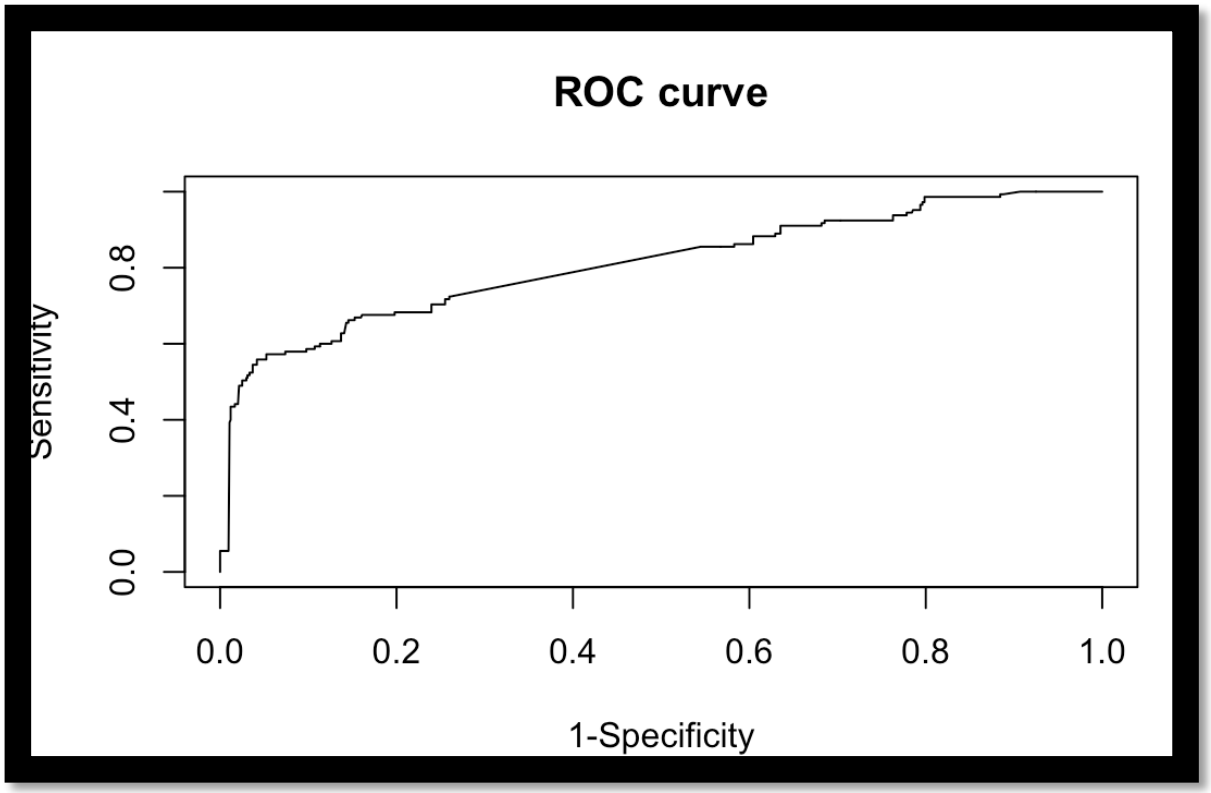


```
> pred[glm.probs_newdata>=0.5] <- 1
> confusionMatrix(test$AdClass,pred )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 826  13
         1  82  63

               Accuracy : 0.9034553
                 95% CI : (0.8832681, 0.9211874)
    No Information Rate : 0.9227642
    P-Value [Acc > NIR] : 0.9881559

                  Kappa : 0.5216555
 Mcnemar's Test P-Value : 0.000000000003022955

            Sensitivity : 0.9096916
            Specificity : 0.8289474
         Pos Pred Value : 0.9845054
         Neg Pred Value : 0.4344828
             Prevalence : 0.9227642
         Detection Rate : 0.8394309
   Detection Prevalence : 0.8526423
      Balanced Accuracy : 0.8693195

       'Positive' Class : 0

>
```

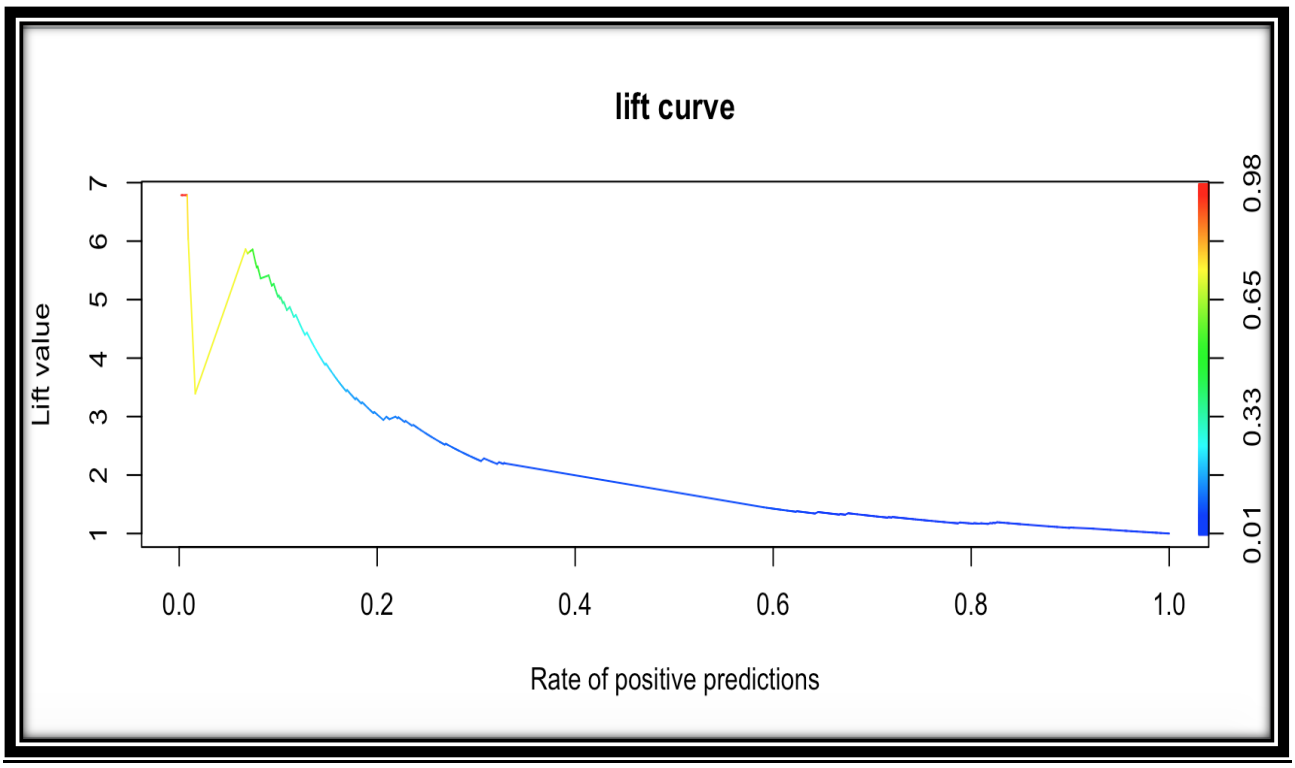
As we can see the accuracy of the model is 90.34 percent and the confusion matrix shows that it has predicted 826 values correct for Non ads and 13 of the non-ads were predicted as Ads which is just a small portion of the data. For the Ad values our confusion matrix shows that it has predicted 63 correct values and 83 images were predicted as non ads.

Below are some more details of our results.
Sensitivity of our model is 0.90
Specificity of our model is 0.82
Positive prediction value is 0.98
Negative prediction value is 0.43
We have an accuracy of 90% for our logistic regression model.

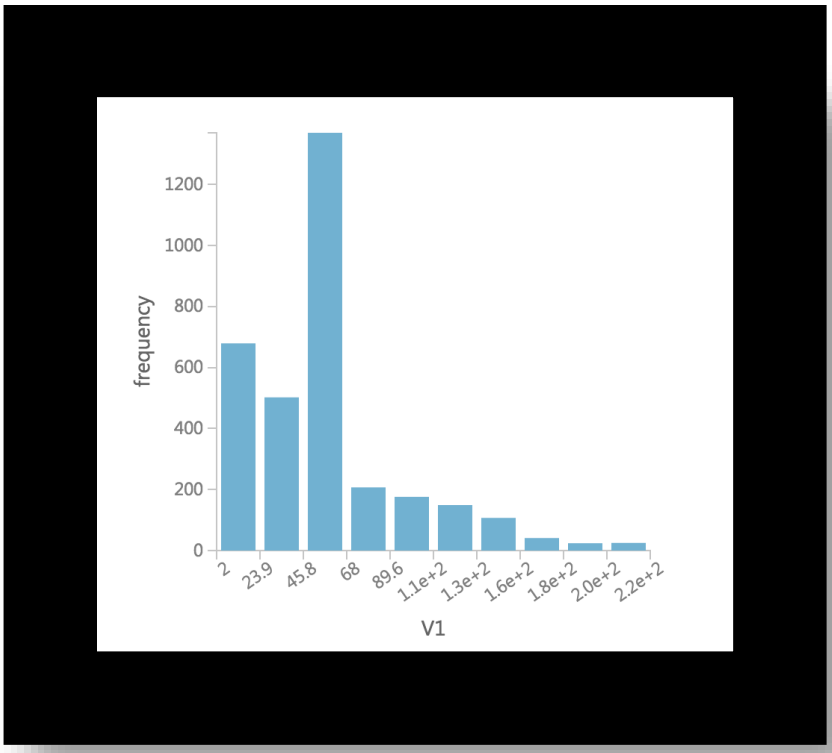
ROC curve is as below:

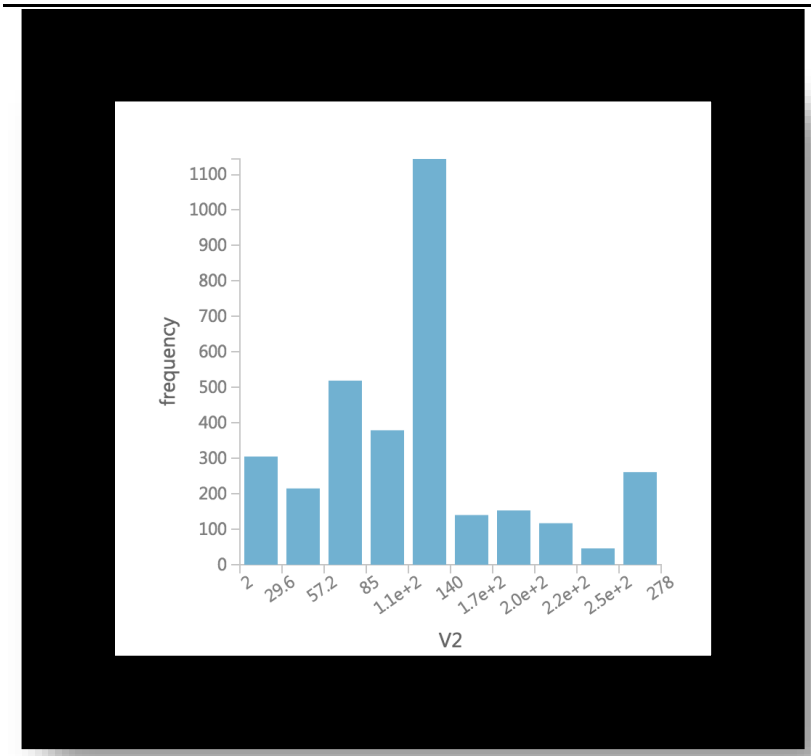


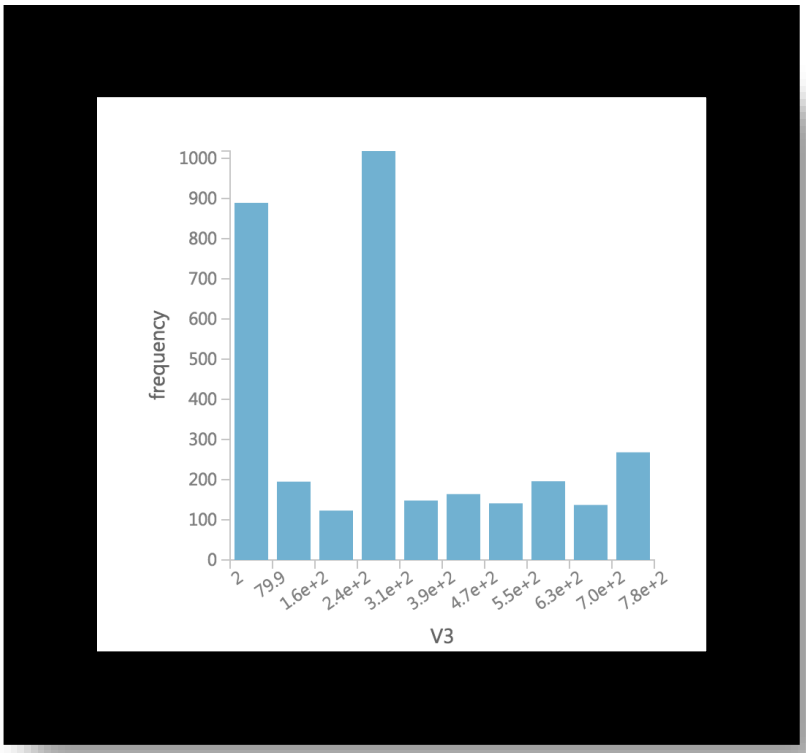
Lift chart:



Power BI Data Analysis: These below charts clearly shows that there are outliers in our data. So we worked on it and removed it in the data cleaning process.


frequency
1200
1000
800
600
400
200
0
2
23.9
45.8
68
89.6
1.1e+2
1.3e+2
1.6e+2
1.8e+2
2.0e+2
2.2e+2
V1

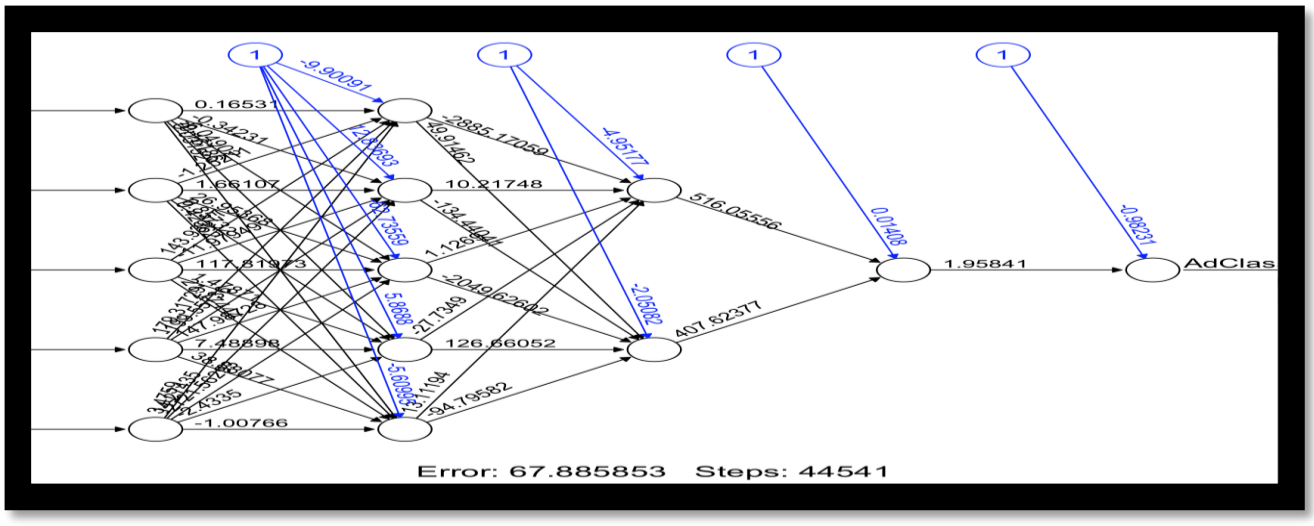

frequency
1100
1000
900
800
700
600
500
400
300
200
100
0
2
29.6
57.2
85
1.1e+2
140
1.7e+2
2.0e+2
2.2e+2
2.5e+2
278
V2



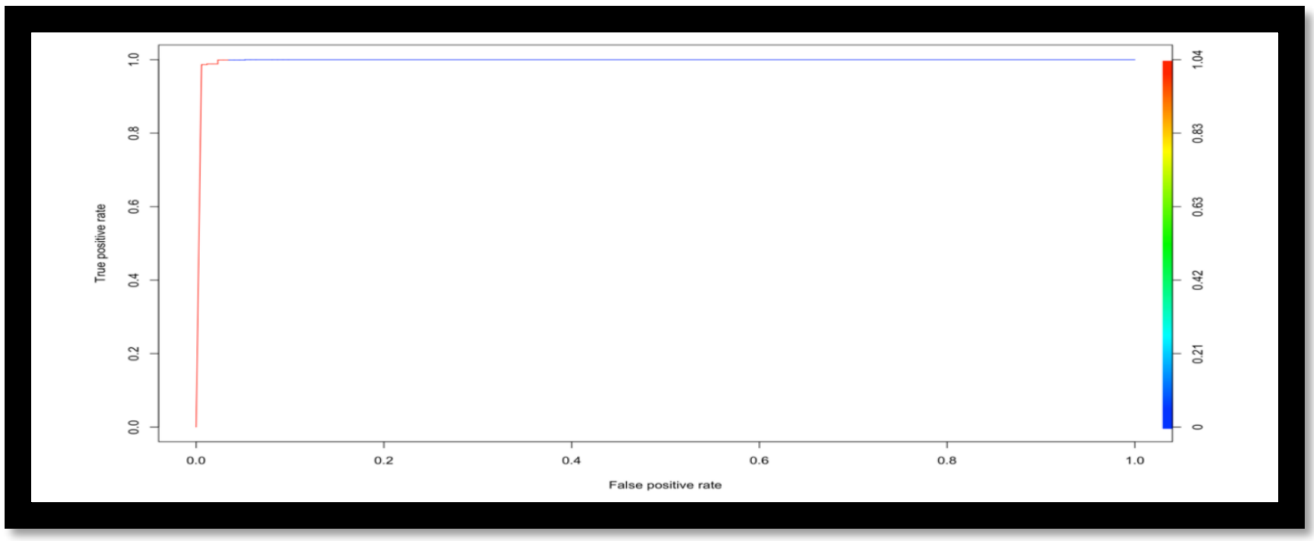
Neural Network:

```
114  # Beginning of Neural Network
115
116  install.packages("neuralnet")
117  library(MASS);
118  library(grid);
119  library(neuralnet);
120  head(train)
121  #Create a neural model
122  net_result=neuralnet(AdClass~V2+V3+V10+V32+V68,train,hidden = c(5,2,1),threshold = 0.5);
123
124  #Plot a neural Model
125  plot(net_result)
126
127  #Compute values using model
128  net_prediction=compute(net_result,test[,c(2,3,10,32,68)])
129
130  #Lift curve
131  perf = performance(prediction,"lift","rpp")
132  plot(perf, main="lift curve", colorize=T)
133
```

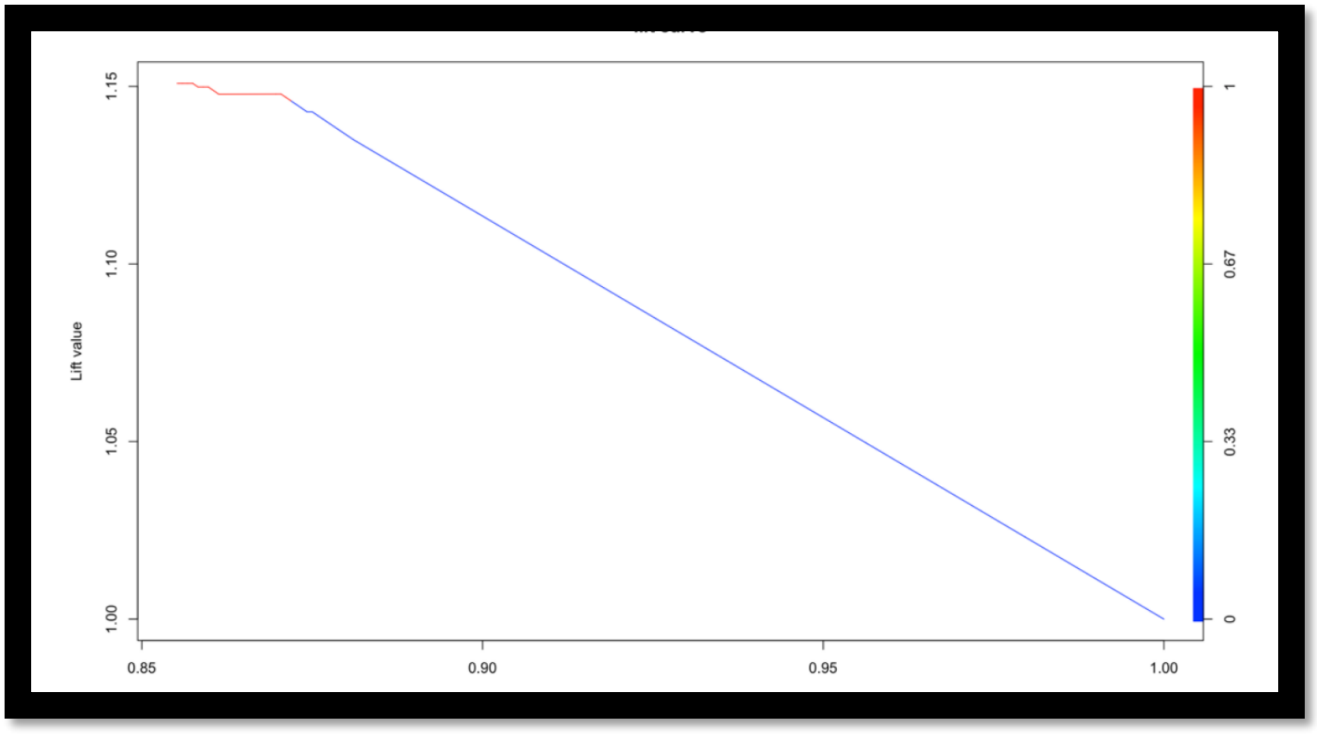
Below is our plot chart of neural network:



ROC Curve for Neural Network:



Lift Chart:



Accuracy of our neural network model 90.34 percent

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

          Reference
Prediction   0   1
         0 826  13
         1  82  63

               Accuracy : 0.9034553
                 95% CI : (0.8832681, 0.9211874)
    No Information Rate : 0.9227642
    P-Value [Acc > NIR] : 0.9881559

                  Kappa : 0.5216555
 Mcnemar's Test P-Value : 0.000000000003022955

            Sensitivity : 0.9096916
            Specificity : 0.8289474
         Pos Pred Value : 0.9845054
         Neg Pred Value : 0.4344828
             Prevalence : 0.9227642
         Detection Rate : 0.8394309
   Detection Prevalence : 0.8526423
      Balanced Accuracy : 0.8693195

       'Positive' Class : 0

>
```

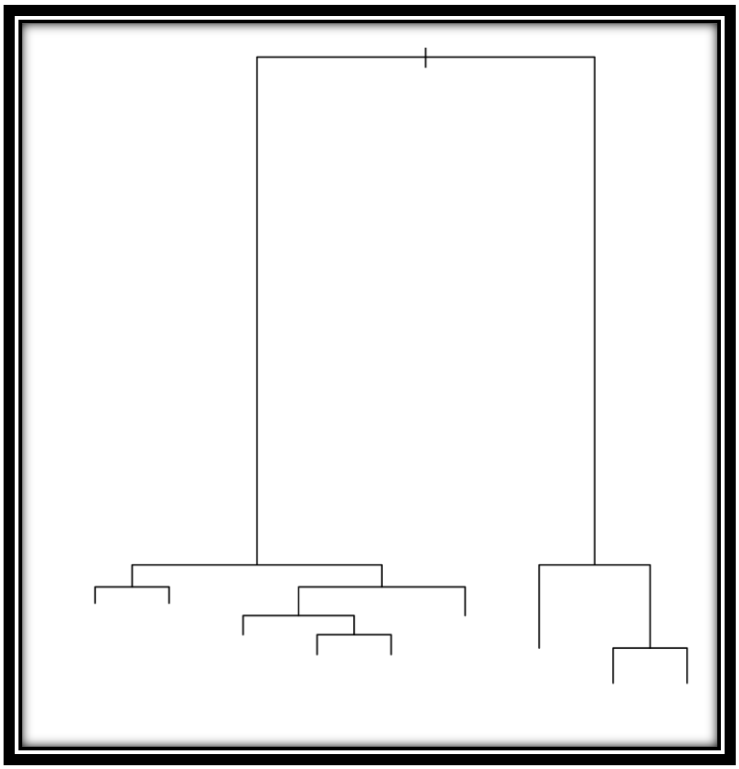
Sensitivity: 0.9096
Specificity: 0.8289
Accuracy: 0.8693
Positive prediction value is 0.98
Negative prediction value is 0.43

We have an accuracy of 90.34% for our neural network model.

Classification Tree:

```
#Beginning Tree
install.packages("tree");
library(tree);

#Create a Model
tree_result=tree(AdClass~V2+V3+V10+V32+V68,data = train);

#Plot the model
plot(tree_result)

#check the summary
summary(tree_result)
tree_prediction=predict(tree_result,newdata = test);
tree_MAPE=mean(abs(tree_prediction-test$AdClass)/test$AdClass);
tree_RMS=sqrt(mean((tree_prediction-test$AdClass)^2));
tree_MAE=mean(abs(tree_prediction-test$AdClass));
summary(tree_MAE)
summary(tree_RMS)
summary(tree_MAPE)

# Generate the values using probability
tree_predict=ifelse(tree_prediction>0.1,1,0);

#Predict using Model
tree.probs_newdata=predict(tree_result,test);
tree_prediction_newdata=ifelse(tree_prediction>0.1,1,0);
```

Below is the classification tree model that we built.



Confusion Matrix :

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 817  22
         1  72  73

               Accuracy : 0.9044715
                 95% CI : (0.8843678, 0.9221136)
    No Information Rate : 0.9034553
    P-Value [Acc > NIR] : 0.4842616

                  Kappa : 0.556608
 Mcnemar's Test P-Value : 0.000000432733

            Sensitivity : 0.9190101
            Specificity : 0.7684211
         Pos Pred Value : 0.9737783
         Neg Pred Value : 0.5034483
             Prevalence : 0.9034553
         Detection Rate : 0.8302846
   Detection Prevalence : 0.8526423
      Balanced Accuracy : 0.8437156

       'Positive' Class : 0
```

Sensitivity: 0.9190
Specificity: 0.7684
Accuracy: 0.8693
Positive prediction value is 0.97
Negative prediction value is 0.83
We have an accuracy of 90.44% for our Classification Tree model.

<u>Best Model | Conclusion:</u>

- ✓ Looking at overall ROC curves and Lift Charts
- ✓ We select the best model as Classification tree.
- ✓ Neural network model has average performance