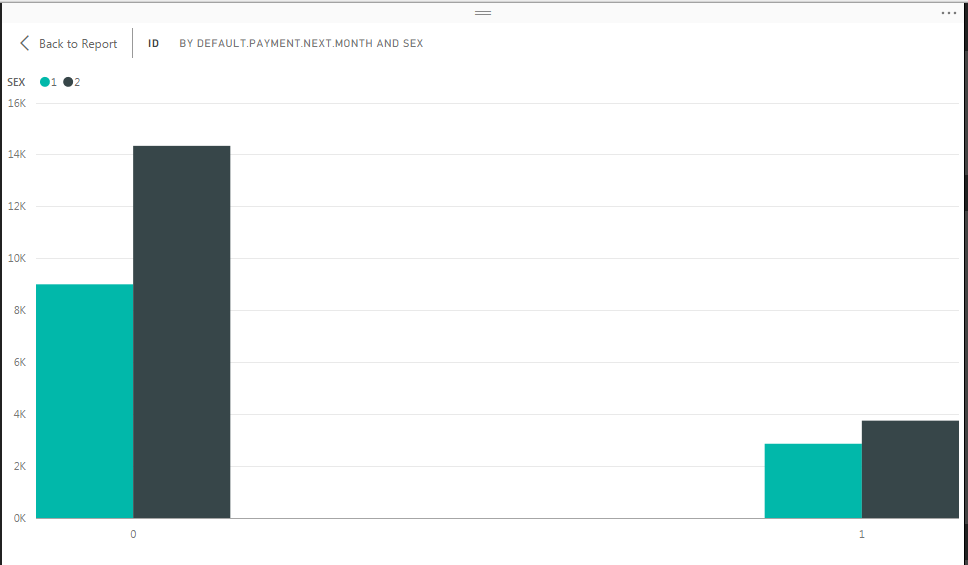
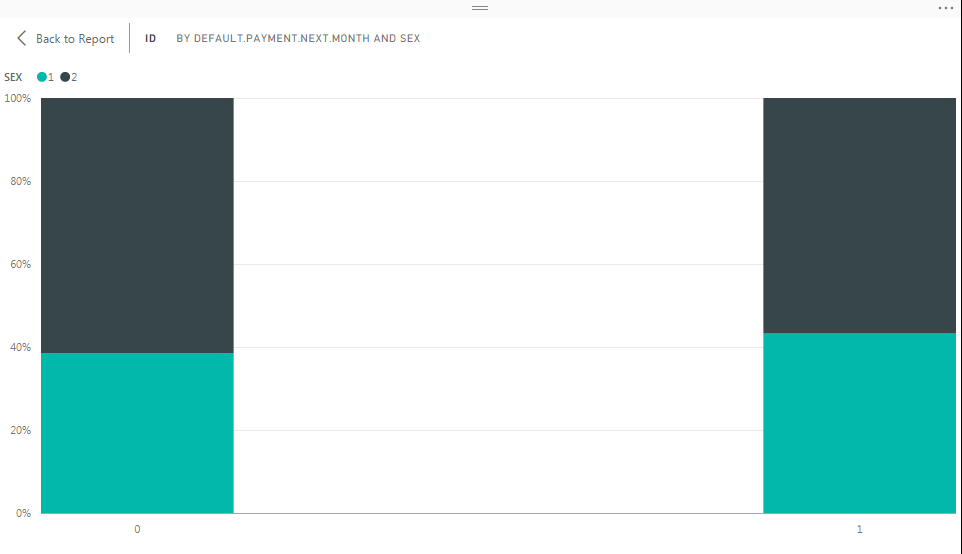
ADS Midterm Assignment

Team 8:

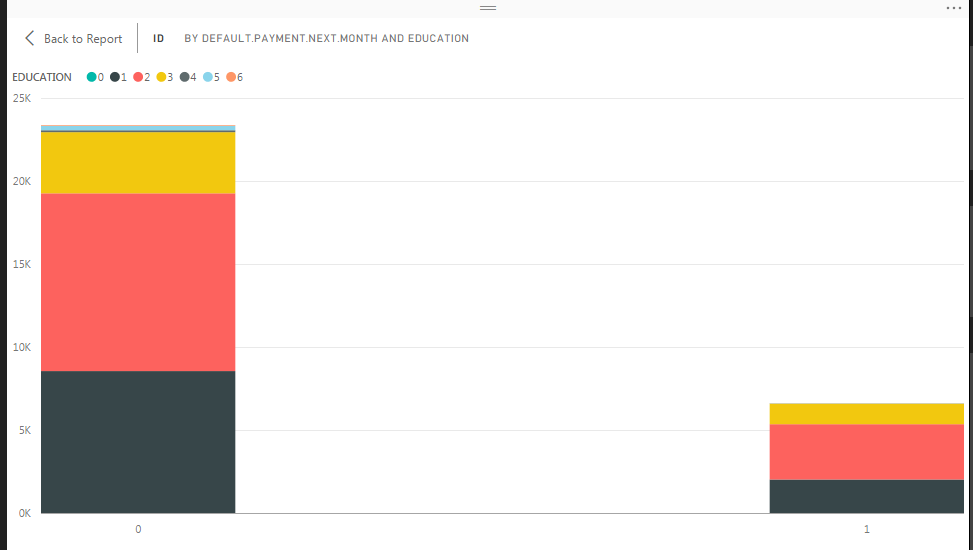
1. A) Use Power Bi to explore the data. Summarize your observation.
2. ID by Default.Payment.next.month and sex



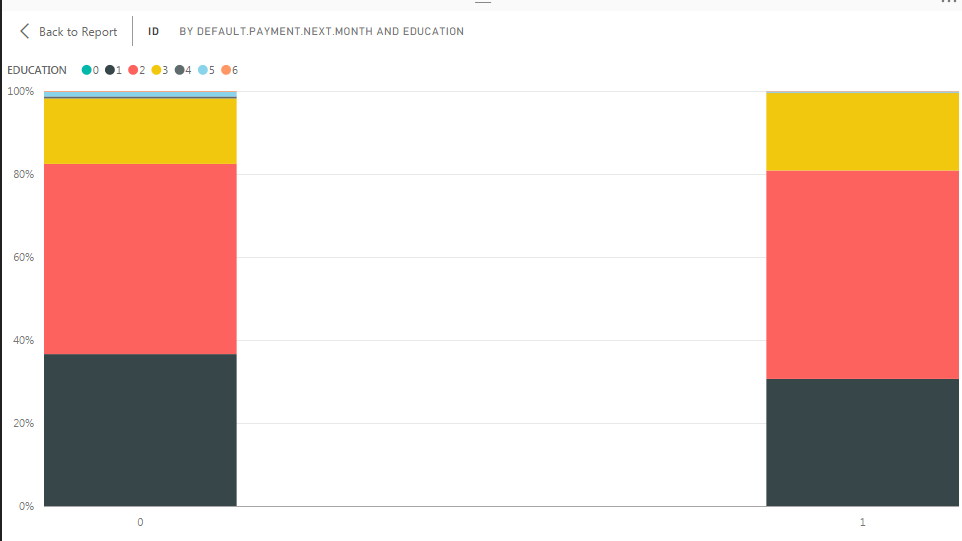
1. By taking into consideration Default.Payment.next.month and sex



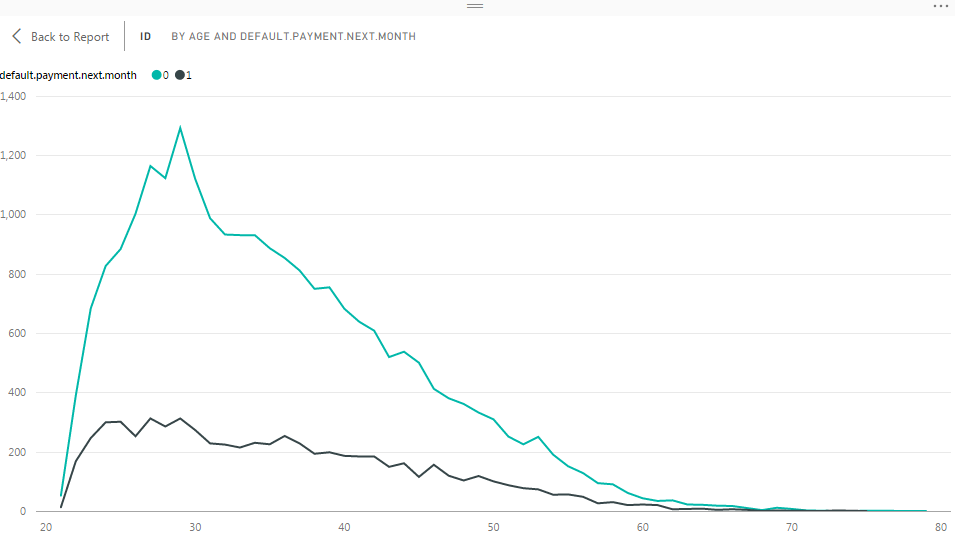
1. By Taking into consideration Default.Payment.next.month and education



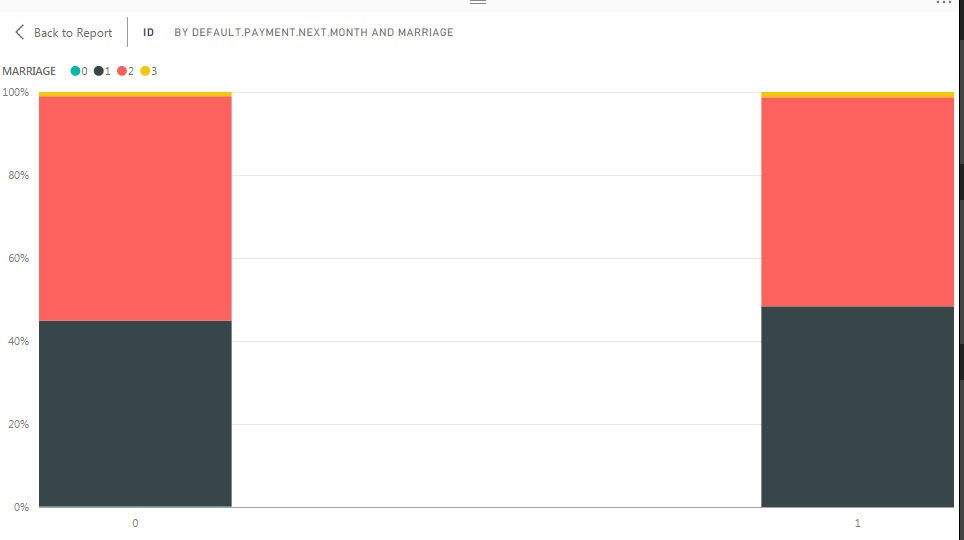
1. By taking into consideration Default.Payment.next.month and Education



1. By taking into consideration age and Default.Payment.next.month



1. By taking into consideration Default.payment.next.month and Marriage



1. Clean and preprocess the data if needed.
2. 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**

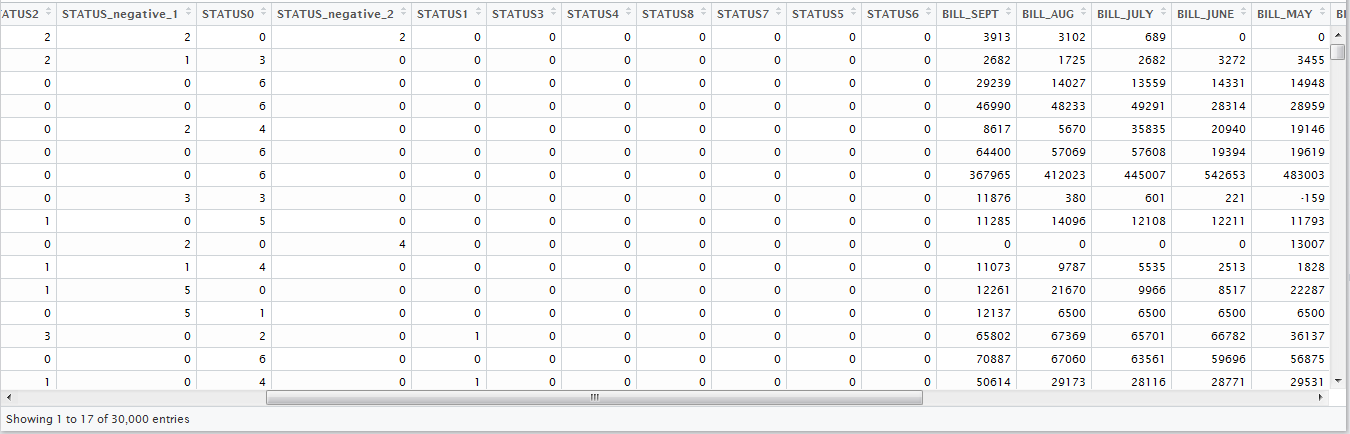
**#create a data frame with new values**

**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")**

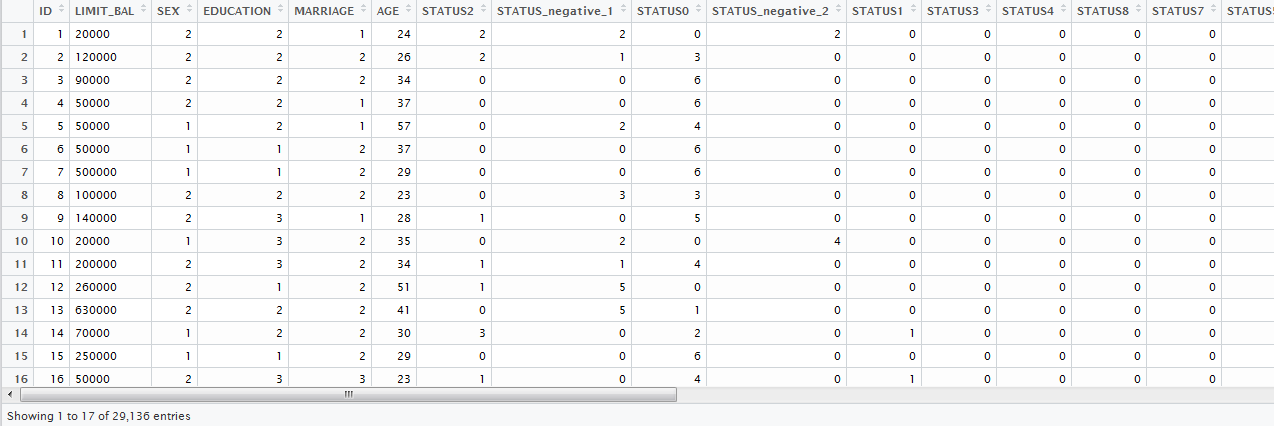
**View(df)**

****

1. 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)),]**

****

**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,]**

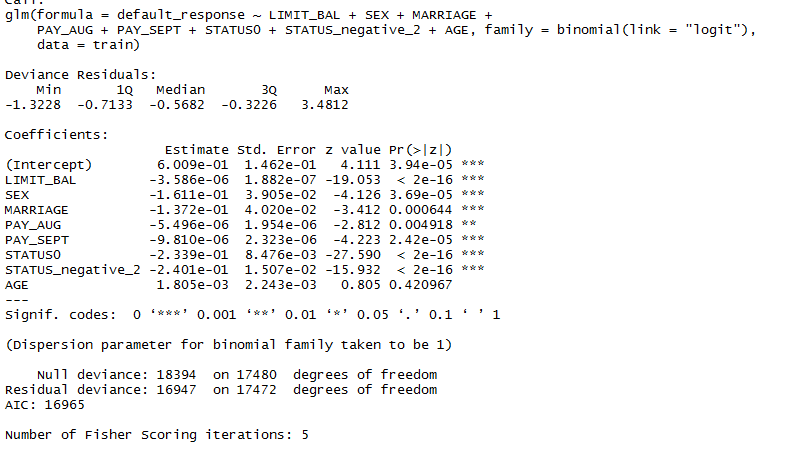
**# Rest 25% is test data**

**test <- df[-train\_ind,]**

1. Construct the logistic regression model

**fit<-glm(default\_response~.,data=train,family=binomial(link="logit"))**

**summary(fit)**

****

1. Predict the outcome using predict() function

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

1. Divide the predicted values based on probability values

**pred<- rep(1,length(test.probs))**

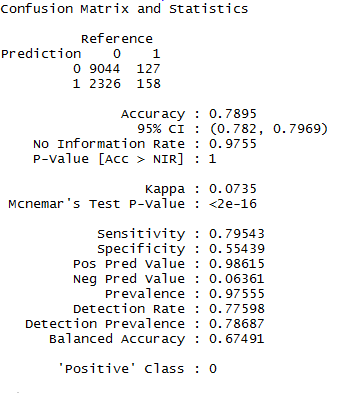
**pred[test.probs<=0.5]<-0**

1. Create the error table

**logisticregression\_table<-table(pred,test$default\_response)**

1. Create confusion matrix

**confusionMatrix(test$default\_response,pred)**

****

1. 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")**

**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]**

1. Construct the classification tree model

**tree.train<- tree(as.factor(default\_response)~.,df,subset=train)**

**summary(tree.train)**

1. Predict the outcome using predict() function

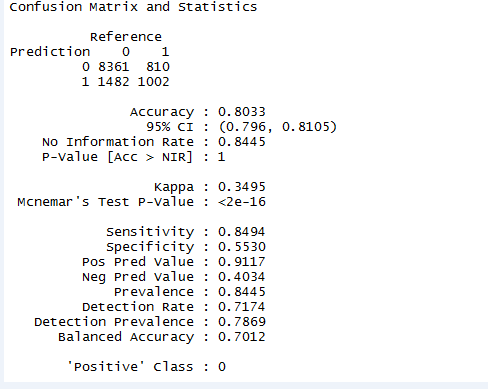
**tree.pred = predict(tree.train,df.test,type="class")**

1. Create the error table

**classification\_tree<-table(tree.pred,default\_response.test)**

1. Create confusion matrix

**confusionMatrix(default\_response.test,tree.pred)**

****

1. 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")**

**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,]**

1. 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)**

1. Predict the outcome using predict() function

**pr<-predict(seedsANN, test)**

1. Plot the neural network

**plotnet(seedsANN,alpha=0.5)**

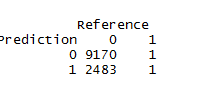
1. Divide the predicted status based on probability values

**pred<- rep(1,length(pr))**

**pred[pr<=0.25]<-0**

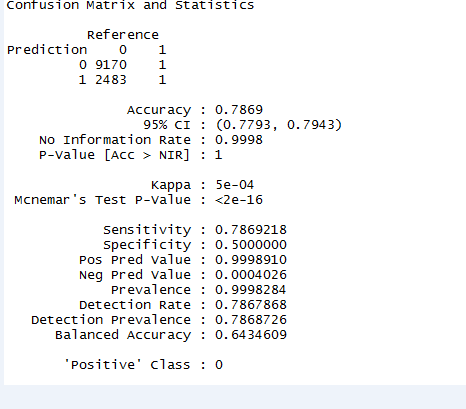
1. Create the error table

**neural\_network\_table<-table(pred,test$default\_response)**

****

1. Create confusion matrix

**confusionMatrix(test$default\_response,pred)**

****

1. 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")**

**We choose Neural Network over all other models**

**REASONS FOR CHOOSING NEURAL NETWORK?**

**Although the accuracy is low for neural network, but it gives better predictions in terms of value predictions.**

**Positive predictions for LR is 98.615**

**Positive predictions for CT is 91.17**

**Positive predictions for LR is 99.95**

**It has better ROC curve than other models.**

**4. a. Overall Error Percentage**

**Logistic Regression: 21.04%**

**Classification Tree:19.65%**

**Neural Network: 21.01%**

**Q.2**: ADVERTISEMENT ON INTERNET PAGES

**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.=="")]**

**#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)**

**LOGISTIC REGRESSION**

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,]**

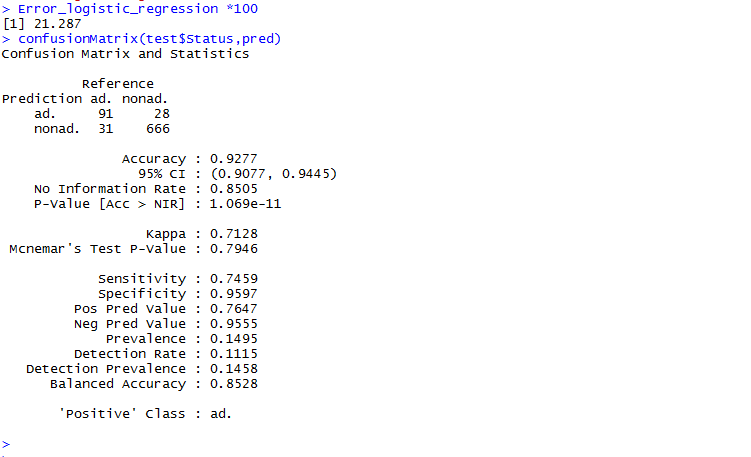
**# Rest 25% is test data**

**test <- df[-train\_ind,]**

1. Construct the logistic regression model

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

**summary(fit)**

****

1. Predict the outcome using predict() function

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

1. Divide the predicted values based on probability values

**pred<- rep("nonad.",length(test.probs))**

**pred[test.probs<=0.5]<-"ad."**

1. Create the error table and calculate error percentage

**logisticregression\_table<-table(pred,test$Status)**

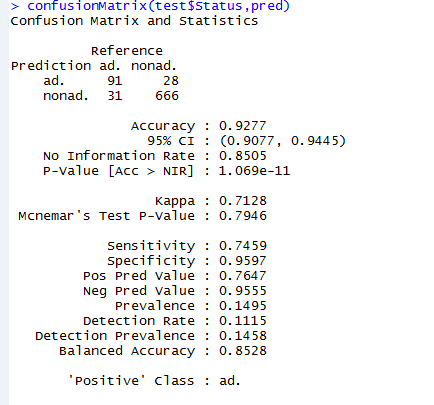
**Error\_logistic\_regression<- ((logisticregression\_table[1,2]) + (logisticregression\_table [2,1])) /(( logisticregression\_table 2,1]) + (logisticregression\_table [1,2]) + (logisticregression\_table [1,1])+(logisticregression\_table [2,2]))**

**Error\_logistic\_regression \*100**

**C:\Users\Sheetal\Desktop\conmatrixkk.PNG**

1. Create confusion matrix

**confusionMatrix(test$Status,pred)**

****

1. 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")**

**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)**

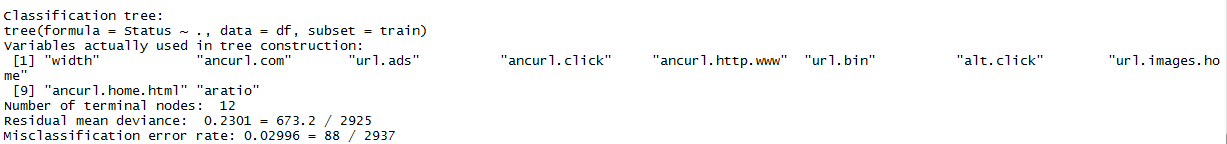
**df.test<-df[-train,]**

**Status.test <- df$Status[-train]**

1. Construct the classification tree model

**fit<- tree(Status~.,df,subset=train)**

**summary(tree)**

****

1. Predict the outcome using predict() function

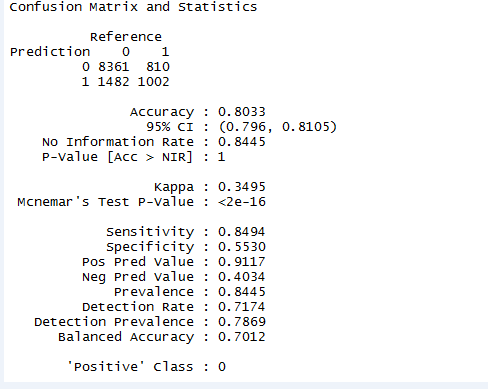
**tree.pred = predict(tree.train,df.test,type="class")**

1. Create the error table

**classification\_tree<-table(tree.pred,default\_response.test)**

1. Create confusion matrix

**confusionMatrix(default\_response.test,tree.pred)**

****

1. 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")**

**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,]**

1. 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)**

1. Predict the outcome using predict() function

**pr<-predict(seedsANN, test)**

1. Plot the neural network

**plotnet(seedsANN,alpha=0.5)**

1. Divide the predicted status based on probability values

**pred<- rep(1,length(pr))**

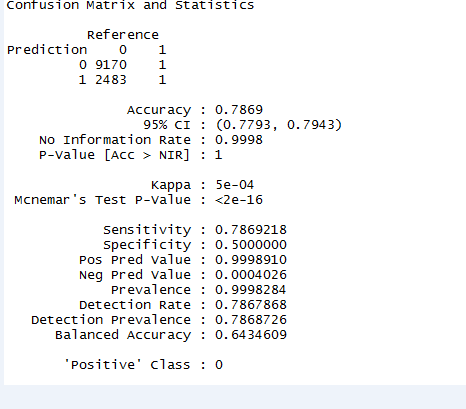
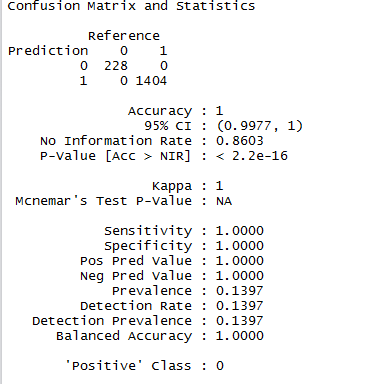
**pred[pr<=0.25]<-0**

1. Create the error table

**neural\_network\_table<-table(pred,test$default\_response)**

1. Create confusion matrix

**confusionMatrix(test$default\_response,pred)**

****

1. 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")**

**We choose Neural Network over all other models**

**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.**

Problem 3:

Predicting hourly power Generation up to 48 hours ahead at 7 wind farms

* Clean the Data using the following Code

#Importing all the packages used

install.packages("lubridate")

install.packages("zoo")

install.packages("tree")

install.packages("forecast")

#File1: windforecast\_wf1.csv, windforecast\_wf2.csv, windforecast\_wf3.csv, windforecast\_wf4.csv, windforecast\_wf5.csv, windforecast\_wf6.csv, windforecast\_wf7.csv

windforecast\_wf\_file<-read.table(file.choose(), header=T, fileEncoding="UTF-8", sep=",")

#Converting date into specific format

library(lubridate)

#Function to format "date" column to year, month, date and hour

formatDate\_wf<-ymd\_h(as.character(windforecast\_wf\_file$date))

#Adding hour to date so that date can be changed with hours, multiplying by 3600 to convert hour integer to seconds and adding in the time format

addHour\_wf<-formatDate\_wf + windforecast\_wf\_file$hors\*3600

#Filling the NA values with the na.locf method of filling missing values

library(zoo)

#Filling for column "u"

mergesdata\_u<- zoo(windforecast\_wf\_file$u)

X\_u<-na.locf(mergesdata\_u, na.rm=TRUE)

X\_u<-as.numeric(X\_u)

#Filling for column "v"

mergesdata\_v<- zoo(windforecast\_wf\_file$v)

X\_v<-na.locf(mergesdata\_v, na.rm=TRUE)

X\_v<-as.numeric(X\_v)

#Filling for column "ws"

mergesdata\_ws<- zoo(windforecast\_wf\_file$ws)

X\_ws<-na.locf(mergesdata\_ws, na.rm=TRUE)

X\_ws<-as.numeric(X\_ws)

#Filling for column "wd"

mergesdata\_wd<- zoo(windforecast\_wf\_file$wd)

X\_wd<-na.locf(mergesdata\_wd, na.rm=TRUE)

X\_wd<-as.numeric(X\_wd)

#Binding date to the windforecast\_wf1\_file1 data

dateChangedData\_wf<-data.frame(Date=addHour\_wf,u=X\_u,v=X\_v,ws=X\_ws,wd=X\_wd)

#Taking mean of the data values u, v, ws and wd for common date+hour

aggregated\_u<-aggregate(dateChangedData\_wf$u ~ dateChangedData\_wf$Date, dateChangedData\_wf, mean )

aggregated\_v<-aggregate(dateChangedData\_wf$v ~ dateChangedData\_wf$Date, dateChangedData\_wf, mean )

aggregated\_ws<-aggregate(dateChangedData\_wf$ws ~ dateChangedData\_wf$Date, dateChangedData\_wf, mean )

aggregated\_wd<-aggregate(dateChangedData\_wf$wd ~ dateChangedData\_wf$Date, dateChangedData\_wf, mean )

xx<-cbind(aggregated\_u,aggregated\_v,aggregated\_ws,aggregated\_wd)

rr<-data.frame(xx[1],xx[2],xx[4],xx[6],xx[8])

colnames(rr)<-c("Date","u","v","ws","wd")

#Separated Date column into year, month, day and hour for prediction variables

year\_wf<-year(rr$Date)

month\_wf<-month(rr$Date)

day\_wf<-day(rr$Date)

hour\_wf<-hour(rr$Date)

#Converting back to the date hour format like an integer number with class as date

dateHour\_wf<-format(rr$Date,"%Y%m%d%H")

#Final data with the unique rows with Date and corresponding u,v,ws,wd,Year,Month,Day and Hour

finaldf<-data.frame(Date=dateHour\_wf,rr[2:5],Year=year\_wf,Month=month\_wf,Day=day\_wf,Hour=hour\_wf)

# File2: train.csv

train\_file<-read.table(file.choose(), header=T, fileEncoding="UTF-8", sep=",")

#Adding only required columns for the windforecast files

#Run only when using windforecast\_wf1.csv as File 1

  date\_wp<-data.frame(Date=train\_file$date, wp1=train\_file$wp1)

#Run only when using windforecast\_wf2.csv as File 1

  #date\_wp<-data.frame(Date=train\_file$date, wp2=train\_file$wp2)

#Run only when using windforecast\_wf3.csv as File 1

  #date\_wp<-data.frame(Date=train\_file$date, wp3=train\_file$wp3)

#Run only when using windforecast\_wf4.csv as File 1

 #date\_wp<-data.frame(Date=train\_file$date, wp4=train\_file$wp4)

#Run only when using windforecast\_wf5.csv as File 1

 #date\_wp<-data.frame(Date=train\_file$date, wp5=train\_file$wp5)

#Run only when using windforecast\_wf6.csv as File 1

 #date\_wp<-data.frame(Date=train\_file$date, wp6=train\_file$wp6)

#Run only when using windforecast\_wf6.csv as File 1

 #date\_wp<-data.frame(Date=train\_file$date, wp7=train\_file$wp6)

#Merging the data by combining dateChangedData\_wf and the respective predicted wp column

merged.data <- merge(finaldf, date\_wp , by.x="Date", all.x = TRUE)

#26k

#The NA is replaced by 9999 so as to differentiate it at the time of na.locf

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

merged.data$wp1[[is.na](http://is.na/)(merged.data$wp1)]<-9999

View(merged.data)

#Replacing 0s with NA - wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

merged.data$wp1[merged.data$wp1 == 0] <- NA

#Handling NAs with na.locf - wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

mergeddata\_wp<- zoo(merged.data$wp1)

wp\_nonna<-na.locf(mergeddata\_wp, na.rm=TRUE)

wp\_nonna<-as.numeric(wp\_nonna)

#until here, zeroes replaced by na.locf, previous NAs replaced by 9999

#Changing 9999 back to the NA

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

merged.data$wp1[merged.data$wp1==9999]<-NA

#Omitting NA for the training model

na\_omit\_data<-na.omit(merged.data)

View(na\_omit\_data)

#Binding date to the windforecast\_wf1\_file1 data

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

mergeddata\_wp\_final<-data.frame(Date=finaldf$Date,u=merged.data$u, v=merged.data$v, ws=merged.data$ws, wd=merged.data$wd,

Year=merged.data$Year, Month=merged.data$Month, Day=merged.data$Day, Hour=merged.data$Hour, wp1=wp\_nonna)

View(mergeddata\_wp\_final)

#Regression, Neural Network and Regression trees to build prediction models

#-----1. Regression Model------

#Omitting the na values of predicted wind values so as to build the model

    #omit\_na\_wp\_rows<-na.omit(mergeddata\_wp\_final)

#Training Data

training\_Data<- na\_omit\_data

#Building the model with training\_Data

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

fit<-with(training\_Data,lm(wp1~u+v+ws+wd+Hour+Year+Month+Day))

summary(fit)

#Testing Data

testing\_Data<-mergeddata\_wp\_final

prediction<-predict.lm(fit, testing\_Data)

 #wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

dataframe<-data.frame(wp1=prediction)

View(dataframe)

library(forecast)

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

acc\_LR<-accuracy(prediction, training\_Data$wp6)

#-----2. Regression Tree------

training\_Data\_RT <- na\_omit\_data

library(tree)

library(MASS)

library(ISLR)

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

fit <- tree(wp1~u+v+ws+wd+Hour+Year+Month+Day, training\_Data\_RT)

summary(fit)

#Putting the file with NA values as the testing\_Data\_RT

testing\_Data\_RT<-mergeddata\_wp\_final

prediction<-predict(fit, testing\_Data\_RT)

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

dataframe<-data.frame(wp1=prediction)

library(forecast)

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

acc\_RT<-accuracy(prediction, training\_Data\_RT$wp1)

View(acc\_RT)

#-------3. Neural Network--------

library(MASS)

training\_Data\_NN <- na\_omit\_data

index <- sample(1:nrow(training\_Data\_NN),round(0.75\*nrow(training\_Data\_NN)))

training\_Data\_NN <- training\_Data\_NN[,-c(1)]

maxs <- apply(training\_Data\_NN, 2, max)

mins <- apply(training\_Data\_NN, 2, min)

scaled <- as.data.frame(scale(training\_Data\_NN, center = mins, scale =maxs - mins))

train\_ <- scaled[index,]

test\_ <- scaled[-index,]

#model for neural network

library(neuralnet)

n <- names(train\_)

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

f <- as.formula(paste("wp1 ~", paste(n[!n %in% "wp1"], collapse = " + ")))

nn <- neuralnet(f,data=train\_,hidden=5, threshold= 0.6,linear.output=T)

plot(nn)

#-------Predicting on the mergeddata\_wp\_final------

testing\_Data\_NN <- mergeddata\_wp\_final

testing\_Data\_NN <- testing\_Data\_NN[,-c(1)]

View(testing\_Data\_NN)

predictwp.nn <- compute(nn,testing\_Data\_NN[,1:8])

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

pr.nn\_nn <- predictwp.nn$net.result\*(max(training\_Data\_NN$wp1)-min(training\_Data\_NN$wp1))+min(training\_Data\_NN$wp1)

#wp1, wp2, wp3, wp4, wp5, wp6, wp7 depending on number of the csv file you took

test.r <- (testing\_Data\_NN$wp1)\*(max(training\_Data\_NN$wp1)-min(training\_Data\_NN$wp1))+min(training\_Data\_NN$wp1)

test.r\_nn<-data.frame(test.r)

colnames(test.r\_nn)<- c("wp1")

MSE.nn <- sum((test.r - pr.nn\_nn)^2)

#error

error<- test.r - pr.nn\_nn

#rmse

rmse <- function(error)

{

  sqrt(mean(error^2))

}

RMSE.nn<-rmse(error)

#mae

mae <- function(error)

{

  mean(abs(error))

}

MAE.nn<-mae(error)

#mape

mape<-function(error)

  mean(abs((error/test.r)))

MAPE.nn<-mape(error)

#----CSV for each windforecast file with the information of performance factors

error\_lr<-cbind("Linear Regresion",acc\_LR)

error\_rt<-cbind("Regression Tree",acc\_RT)

error\_nn<- cbind("Neural Network",RMSE.nn,MAE.nn,MAPE.nn)

final\_error<-cbind(error\_lr,error\_rt,error\_nn)

View(t(final\_error))

write.csv(t(final\_error),"PerformanceMetrics\_wf1.csv")

#-------The best model is the neural network model because the R-Squared and MAPE values are least for the neural network------

#Run only one at a time

#Run when taking windforecast\_wf1.csv

 wp1\_Values<-data.frame(test.r\_nn)

#Run when taking windforecast\_wf2.csv

 #wp2\_Values<- data.frame(test.r\_nn)

#Run when taking windforecast\_wf3.csv

 #wp3\_Values<- data.frame(test.r\_nn)

#Run when taking windforecast\_wf4.csv

 #wp4\_Values<- data.frame(test.r\_nn)

#Run when taking windforecast\_wf5.csv

 #wp5\_Values<- data.frame(test.r\_nn)

#Run when taking windforecast\_wf6.csv

 #wp6\_Values<- data.frame(test.r\_nn)

#Run when taking windforecast\_wf7.csv

 #wp7\_Values<- data.frame(test.r\_nn)

#The predicted values of wp1, wp2, wp3, wp4, wp5, wp6 and wp7

Benchmark<-cbind(wp1\_Values, wp2\_Values, wp3\_Values, wp4\_Values, wp5\_Values, wp6\_Values, wp7\_Values)

#Replacing the NA locations of file in

na\_locations <- which([is.na](http://is.na/" \t "_blank)(mergeddata\_wp\_final$wp1), arr.ind = TRUE)

mergeddata\_wp\_final$wp1[na\_locations] <- Benchmark$wp1[na\_locations]

wp1\_Values\_final<-mergeddata\_wp\_final$wp1

#Replaced wind values with the predicted values

BenchmarkFinal<-cbind(wp1\_Values\_final, wp2\_Values\_final, wp3\_Values\_final, wp4\_Values\_final, wp5\_Values\_final, wp6\_Values\_final, wp7=wp7\_Values\_final)

#The file with the unique identifier like in the benchmark.csv

ID<- [seq.int](http://seq.int/)(nrow(BenchmarkFinal))

FinalOutput<-cbind(ID, mergeddata\_wp\_final[1] ,BenchmarkFinal)

View(FinalOutput)

write.csv(FinalOutput, file="Benchmark.csv")

We choose neural network for this problem because

* RMSE value is low for this problem.
* R square value is also low