# Analyzing the Performance of Stroke Prediction using ML Classification Algorithms

Vishakha Verma

**Introduction**

Stroke is the fifth cause of death in the United States, according to the Heart Disease and Stroke Statistics report. Those who suffer from stroke, if luckily survived, may also suffer from expensive medical bills and even disability. Foreseeing the underlying risk factors of stroke is highly valuable to stroke screening and prevention. In this project, the National Health and Nutrition Examination Survey (NHANES) data from the National Center for Health Statistics (NCHS) is used to develop machine learning models. The NHANES dataset holds an abundance of variables, ranging from demographics, medical history, physical examinations, biochemistry to dietary and lifestyle questionnaires.

**Data Collection**
The prediction target in the dataset is **MCQ160F**, a questionnaire question "Has a doctor or other health professional ever told you that you had a stroke?"
**Variable Selection**
Some Known features contributing to stroke are chosen from NHANES dataset for correlation evaluation and model development to predict **MCQ160F** as follows :
**DLQ040** - Because of a physical, mental, or emotional condition, {do you/does he/does she} have serious difficulty concentrating, remembering, or making decisions?
**CDQ001** - {Have you/Has SP} ever had any pain or discomfort in {your/her/his} chest?
**BPQ020** - {Have you/Has SP} ever been told by a doctor or other health professional that {you/s/he} had hypertension, also called high blood pressure?
**MCQ010** - The following questions are about different medical conditions. Has a doctor or other health professional ever told {you/SP} that {you have/s/he/SP has} asthma (az-ma)?
**MCQ035** - {Do you/Does SP} still have asthma (az-ma)?
**MCQ160b** - Has a doctor or other health professional ever told {you/SP} that {you/s/he} . . .had congestive heart failure?
**MCQ160e** - Has a doctor or other health professional ever told {you/SP} that {you/s/he} . . .had a heart attack (also called myocardial infarction (my-o-car-dee-al in-fark-shun))?
**MCQ160f** - Has a doctor or other health professional ever told {you/SP} that {you/s/he} . . .had a stroke?
**MCQ220** - {Have you/Has SP} ever been told by a doctor or other health professional that {you/s/he} had cancer or a malignancy (ma-lig-nan-see) of any kind?
**PAD660** - How much time {do you/does SP} spend doing vigorous-intensity sports, fitness or recreational activities on a typical day?
**DIQ010** - The next questions are about specific medical conditions. {Other than during pregnancy, {have you/has SP}/{Have you/Has SP}} ever been told by a doctor or health professional that {you have/{he/she/SP} has} diabetes or sugar diabetes?
**SMQ050Q** - How long has it been since {you/SP} quit smoking cigarettes?
**DUQ240** - Have you ever used cocaine, crack cocaine, heroin, or methamphetamine?
**WHQ030M** - Do you consider yourself now to be fat enough.
**RIDAGEYR**-Recode Best age in years of the sample person at time of HH screening. Individuals 85 and

over are topcoded at 85 years of age.
**RIAGENDR** - Gender Gender of the sample person
**PEASCST1** - Blood Pressure Status
**BPXSY3** - Systolic: Blood pres (3rd rdg) mm Hg
**BPXDI3** - Diastolic: Blood pres (3rd rdg) mm Hg}

### Loading Packages

```
suppressMessages(suppressWarnings(library(tidyr)))
suppressMessages(suppressWarnings(require(dplyr)))
suppressMessages(suppressWarnings(require(plyr)))
suppressMessages(suppressWarnings(library("plyr")))
suppressMessages(suppressWarnings(library(caret))) # for general data preparation and model fitting
suppressMessages(suppressWarnings(library(tidyverse)))
suppressWarnings(library(caTools)) # For Logistic regression
suppressWarnings(library(ROCR)) # For ROC curve to evaluate model
suppressMessages(suppressWarnings(library(MASS))) # For LDA and QDA
suppressWarnings(library(class)) # For KNN
```

### Data Loading and Cleaning

We first drop all the variables with all NAN values and create a dataframe with variables as columns and finally removing rows with NAN values . So our final data set has the variables
"DLQ040","CDQ001","BPQ020","MCQ010",
"MCQ160B","MCQ160E","MCQ160F","MCQ220",
"DIQ010","SMQ040","DUQ240","PEASCST1",
"BPXSY3","BPXDI3","RIAGENDR","RIDAGEYR".
Now we changed the target variable MCQ160F coding from 1, 2 to 1 (Negative), 0 (Positive)

```
a=read.csv("C:/Users/user/Downloads/diet.csv")
b=read.csv("C:/Users/user/Downloads/questionnaire.csv")
c=read.csv("C:/Users/user/Downloads/medications.csv")
d=read.csv("C:/Users/user/Downloads/labs.csv")
e=read.csv("C:/Users/user/Downloads/examination.csv")
f=read.csv("C:/Users/user/Downloads/demographic.csv")
df=join_all(list(a,b,c,d,e,f), by = 'SEQN', type = 'full')
df1<-subset(df,select=c("DLQ040","CDQ001","BPQ020","MCQ010","MCQ160B",
"MCQ160E","MCQ160F","MCQ220","DIQ010","SMQ040","DUQ240"
,"PEASCST1","BPXSY3","BPXDI3","RIAGENDR","RIDAGEYR"))
df2<-df1 %>% drop_na(MCQ160F)
df3 <- df2 %>% mutate(MCQ160F = ifelse( MCQ160F== 2,0,1))
data<-na.omit(df3)
head(data)
```

```
##   DLQ040 CDQ001 BPQ020 MCQ010 MCQ160B MCQ160E MCQ160F MCQ220 DIQ010 SMQ040
## 1      2      2      1      2       2       2       1      2      1      3
## 2      2      2      1      2       2       2       1      2      1      3
## 3      2      1      1      1       2       2       0      2      1      2
## 4      2      1      1      1       2       2       0      2      1      2
## 5      2      1      1      1       2       2       0      2      1      2
## 6      2      1      1      1       2       2       0      2      1      2
```

```
##   DUQ240 PEASCST1 BPXSY3 BPXDI3 RIAGENDR RIDAGEYR
## 1      7        1    102     74        1       69
## 2      7        1    102     74        1       69
## 3      1        1    156     42        1       54
## 4      1        1    156     42        1       54
## 5      1        1    156     42        1       54
## 6      1        1    156     42        1       54
```

### Splitting dataset

After completing data preprocessing and handling the imbalanced dataset, the next step is building the model. The undersampled data is split into training and testing data for better accuracy and efficiency for this task keeping the ratio as 80% training data and 20% testing data.

```
set.seed(10)
split <- sample.split(data, SplitRatio = 0.8)
train_reg <- subset(data, split == "TRUE")
test_reg <- subset(data, split == "FALSE")
```

### Fitting Logistic Regression Model

Fitting the Logistic Regression Model on the training data set .

```
k<-glm(MCQ160F~CDQ001+DLQ040+BPQ020+MCQ010+MCQ160B+MCQ160E+MCQ220+DIQ010+
         RIDAGEYR+RIAGENDR+BPXDI3+BPXSY3+PEASCST1+DUQ240+SMQ040,data=train_reg,family="binomial")
summary(k)
```

```
##
## Call:
## glm(formula = MCQ160F ~ CDQ001 + DLQ040 + BPQ020 + MCQ010 + MCQ160B +
##     MCQ160E + MCQ220 + DIQ010 + RIDAGEYR + RIAGENDR + BPXDI3 +
##     BPXSY3 + PEASCST1 + DUQ240 + SMQ040, family = "binomial",
##     data = train_reg)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1218  -0.4292  -0.3215  -0.2098   2.9749
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -13.757302 403.656789  -0.034   0.9728
## CDQ001       -0.150213   0.164587  -0.913   0.3614
## DLQ040       -0.733251   0.161134  -4.551 5.35e-06 ***
## BPQ020       -1.042900   0.213816  -4.878 1.07e-06 ***
## MCQ010       -0.766546   0.159124  -4.817 1.46e-06 ***
## MCQ160B      -0.414491   0.194948  -2.126   0.0335 *
## MCQ160E      -0.786696   0.195381  -4.026 5.66e-05 ***
## MCQ220       -0.178960   0.188171  -0.951   0.3416
## DIQ010       -0.277400   0.127943  -2.168   0.0301 *
## RIDAGEYR      0.032662   0.012174   2.683   0.0073 **
```

```
## RIAGENDR      0.207229   0.151374   1.369   0.1710
## BPXDI3        0.002120   0.004848   0.437   0.6619
## BPXSY3        0.001451   0.004082   0.355   0.7222
## PEASCST1     15.924717 403.654761   0.039   0.9685
## DUQ240       -0.013395   0.128501  -0.104   0.9170
## SMQ040       -0.015574   0.082794  -0.188   0.8508
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1665.4  on 2820  degrees of freedom
## Residual deviance: 1406.4  on 2805  degrees of freedom
## AIC: 1438.4
##
## Number of Fisher Scoring iterations: 14
```

**Conclusion :**
From the summary we can see that the p-values for the test of significance of the CDQ001,MCQ220,RIAGENDR,BPXDI3,BPX variables are greater than 0.05 i.e they are insignificant for prediction .So, we can drop them .
AIC(Alkaline Information criteria) value is 1438.4 i.e the lesser the better for the model.

**confusion matrix**
Now we will validate the model with testing data set by evaluating model accuracy using confusion matrix

```
predict_reg <- predict(k, test_reg, type = "response")

# Changing probabilities

predict_reg <- ifelse(predict_reg >0.5, 1, 0)
table(test_reg$MCQ160F, predict_reg)
```

```
##    predict_reg
##       0   1
##   0 861   0
##   1  75   4
```

```
missing_classerr <- mean(predict_reg != test_reg$MCQ160F)
print(paste('Accuracy =', 1 - missing_classerr))
```

```
## [1] "Accuracy = 0.920212765957447"
```

**Conclusion:**
Accuracy comes out to be 0.920212765957447 i.e 92% , which is quiet good for the model .

Now we will try fitting logistic regression after dropping variables that are insignificant i.e CDQ001,MCQ220,RIAGENDR,BPX

**Modified Data**
Getting the modified data by dropping the insignificant variables and then spilting the data into traing and testing data sets .

```
data<-subset(data,select=c("DLQ040","BPQ020","MCQ010","MCQ160B",
                           "MCQ160E","MCQ160F","DIQ010","RIDAGEYR"))
head(data)
```

```
##   DLQ040 BPQ020 MCQ010 MCQ160B MCQ160E MCQ160F DIQ010 RIDAGEYR
## 1      2      1      2       2       2       1      1       69
## 2      2      1      2       2       2       1      1       69
## 3      2      1      1       2       2       0      1       54
## 4      2      1      1       2       2       0      1       54
## 5      2      1      1       2       2       0      1       54
## 6      2      1      1       2       2       0      1       54
```

```
set.seed(6)
split <- sample.split(data, SplitRatio = 0.8)
train_reg <- subset(data, split == "TRUE")
test_reg <- subset(data, split == "FALSE")
```

**Fitting Logistic Regresssion**

Fitting the Logistic Regression Model on the new training data set .

```
l<-glm(MCQ160F~DLQ040+BPQ020+MCQ010+MCQ160B+MCQ160E+RIDAGEYR,data=train_reg,family="binomial")
summary(l)
```

```
##
## Call:
## glm(formula = MCQ160F ~ DLQ040 + BPQ020 + MCQ010 + MCQ160B +
##     MCQ160E + RIDAGEYR, family = "binomial", data = train_reg)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2745  -0.4101  -0.3219  -0.1974   2.9638
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.11060    0.84061   3.700 0.000215 ***
## DLQ040      -0.75443    0.15601  -4.836 1.33e-06 ***
## BPQ020      -1.16355    0.21511  -5.409 6.33e-08 ***
## MCQ010      -1.02607    0.14681  -6.989 2.77e-12 ***
## MCQ160B     -0.41514    0.18515  -2.242 0.024949 *
## MCQ160E     -1.04833    0.17078  -6.139 8.33e-10 ***
## RIDAGEYR     0.02819    0.01069   2.638 0.008343 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1660.7  on 2820  degrees of freedom
## Residual deviance: 1420.4  on 2814  degrees of freedom
## AIC: 1434.4
##
## Number of Fisher Scoring iterations: 6
```

**Conclusion :**

AIC value 1434.4 , get decrease little bit which is good for the model .

**Confusion Matrix**
Validating the model with testing data set .

```
predict_reg1 <- predict(l,test_reg, type = "response")

#Changing probabilities

predict_reg1 <- ifelse(predict_reg1 >0.5, 1, 0)
table(test_reg$MCQ160F, predict_reg1)
```

```
##    predict_reg1
##       0   1
##   0 855   5
##   1  76   4
```

```
missing_classerr1 <- mean(predict_reg1 != test_reg$MCQ160F)
print(paste('Accuracy =', 1 - missing_classerr1))
```

```
## [1] "Accuracy = 0.913829787234043"
```

```
print(paste('Precision=',855/(855+5)))
```

```
## [1] "Precision= 0.994186046511628"
```

```
print(paste('Recall=',855/(855+76)))
```

```
## [1] "Recall= 0.918367346938776"
```

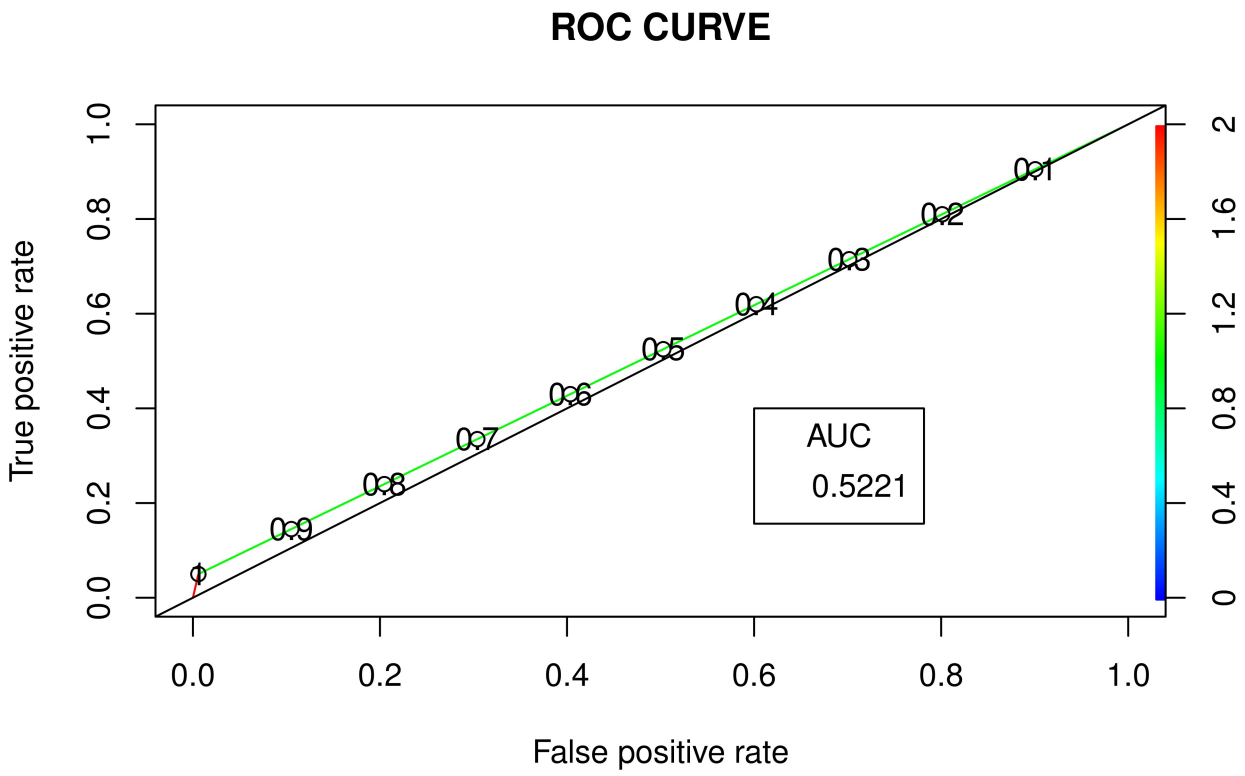
**Conclusion :**
Recall of the model is 0.918367346938776 i.e 92% approx .Precision of the model is 0.994186046511628 ie 100% approx . Accuracy of the model is 0.913829787234043 i.e 92% approx which is as good as the previous model .In the confusion matrix, we should not always look for accuracy but also for sensitivity and specificity. ROC (Receiver operating characteristics) and AUC(Area under the curve) curve is plotted.

**ROC-AUC Curve**

```
ROCPred <- prediction(predict_reg1, test_reg$MCQ160F)
ROCPer <- performance(ROCPred, measure = "tpr",
                      x.measure = "fpr")

auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
# Plotting curve
plot(ROCPer, colorize = TRUE,
```

```
     print.cutoffs.at = seq(0.1, by = 0.1),
     main = "ROC CURVE")
abline(a = 0, b = 1)
auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```




**Conclusion :** In ROC curve, the more the area under the curve, the better the model.AUC is 0.5221, so the more AUC is, the better the model performs.

Now we will fit few other models on the modified data to predict the MCQ160F and then compare them on the basis of Accuracy ,Precision ,Recall and ROC-AUC Curve .

**Linear Discriminant Analysis (LDA)** Now we use LDA on modified data for predicting the class of MCQ160F .

```
lda<-lda(MCQ160F~DLQ040+BPQ020+MCQ010+MCQ160B+MCQ160E+DIQ010+
          RIDAGEYR , data =train_reg)
predictions <- lda %>% predict(test_reg)
table(test_reg$MCQ160F, predictions$class)
```

```
##
##       0   1
##   0 843  17
##   1  62  18
```

```r
print(paste('Accuracy =', mean(predictions$class==test_reg$MCQ160F)))
```

```
## [1] "Accuracy = 0.915957446808511"
```

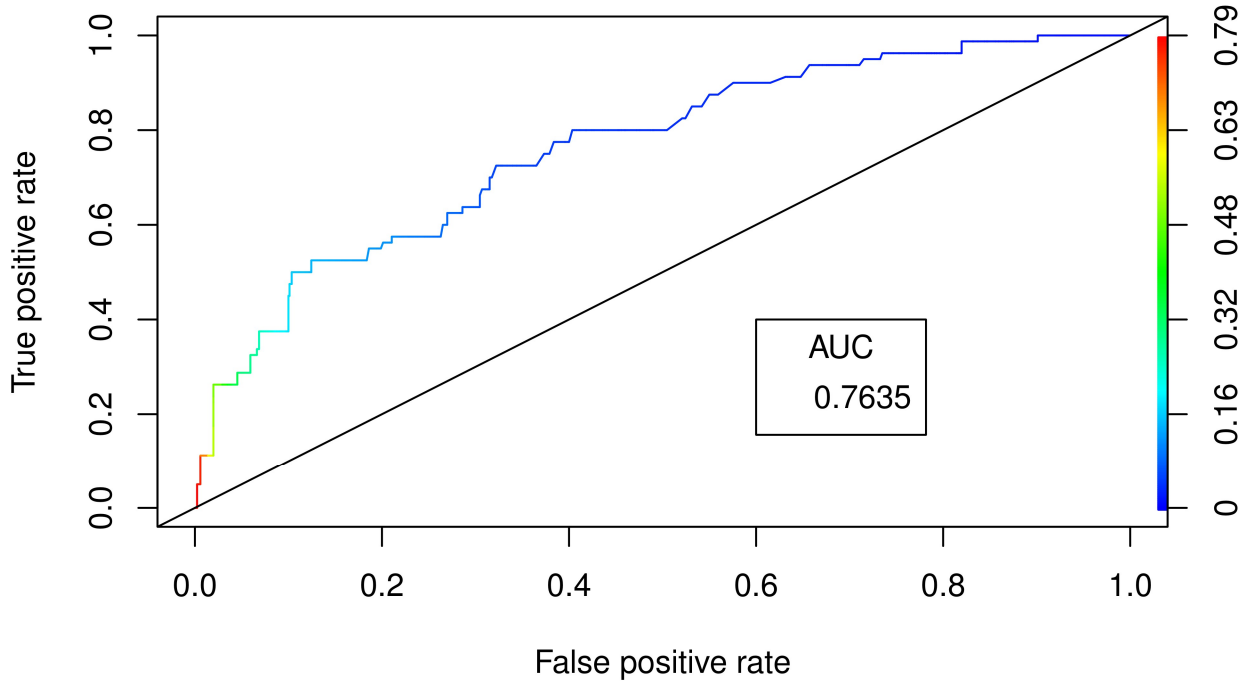
```r
print(paste('Precision=',843/(843+17)))
```

```
## [1] "Precision= 0.980232558139535"
```

```r
print(paste('Recall=',843/(843+62)))
```

```
## [1] "Recall= 0.931491712707182"
```

```r
#ROC-AUC Curve
lda.pred=predict(lda,test_reg)
pred <- prediction(lda.pred$posterior[,2],test_reg$MCQ160F )
perf <- performance(pred,"tpr","fpr")
auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
plot(perf,colorize=TRUE)
abline(a = 0, b = 1)
auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```



**Conclusion :**
So the Recall of the model is 0.931491712707182 i.e 93% approx which is more than logistic Regression model, Precision of the model is 0.980232558139535 i.e 98% approx which is less than Logistic Regression model .Accuracy of the model is 0.915957446808511 i.e 92% approx and AUC is 0.7635 which is more than Logistic Regression model . Hence we can say that LDA model fits better than Logistic Regression model here .

**Quadratic Discriminant Analys (QDA)**
Now we use QDA on modified data for predicting the class of MCQ160F .

```
qda<-qda(MCQ160F~DLQ040+BPQ020+MCQ010+MCQ160B+MCQ160E+DIQ010+
           RIDAGEYR , data =train_reg)
predictions <- qda %>% predict(test_reg)
table(test_reg$MCQ160F, predictions$class)
```

```
##
##       0   1
##   0 819  41
##   1  52  28
```

```
print(paste('Accuracy =', mean(predictions$class==test_reg$MCQ160F)))
```

```
## [1] "Accuracy = 0.901063829787234"
```

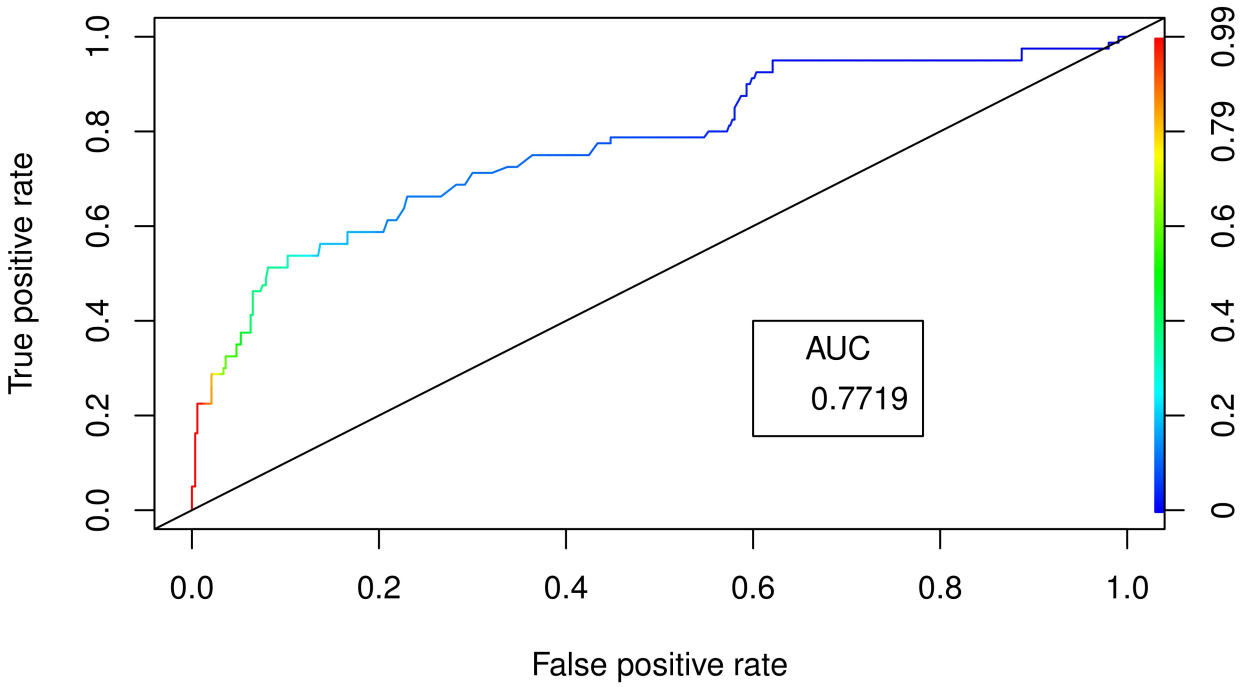
```
print(paste('precision=',819/(819+41)))
```

```
## [1] "precision= 0.952325581395349"
```

```
print(paste('Recall=',819/(819+52)))
```

```
## [1] "Recall= 0.940298507462687"
```

```
qda.pred=predict(qda,test_reg)
pred <- prediction(qda.pred$posterior[,2],test_reg$MCQ160F )
perf <- performance(pred,"tpr","fpr")
auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
plot(perf,colorize=TRUE)
abline(a = 0, b = 1)
auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```




**Conclusion :**
So the Recall of the model is 0.940298507462687 i.e 94% approx more than both Logistic Regression and LDA model , Precision is 0.952325581395349 i.e 95% approx which is less than Logistic Regression and LDA model ,accuracy of the model is 0.901063829787234 i.e 90% approx which is less than Logistic Regression model and LDA but AUC is 0.7719 which is more than Logistic Regression model and LDA .

**K-Nearest Neighbor (KNN)**

Now we will use KNN with different values of K on the modified Data to predict the class of MCQ160F .

```
classifier_knn <- knn(train = train_reg,
                      test = test_reg,
                      cl = train_reg$MCQ160F,
                      k = 1)

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_reg$MCQ160F)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.997872340425532"
```

```
#K=3
classifier_knn <- knn(train = train_reg,
                      test = test_reg,
```

```
                      cl = train_reg$MCQ160F,
                      k = 3)
# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_reg$MCQ160F)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.993617021276596"
```

```
# K=5
classifier_knn <- knn(train = train_reg,
                      test = test_reg,
                      cl = train_reg$MCQ160F,
                      k = 5)
# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_reg$MCQ160F)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.981914893617021"
```

Increasing value of k the accuracy is decreasing so K=1 will be better .

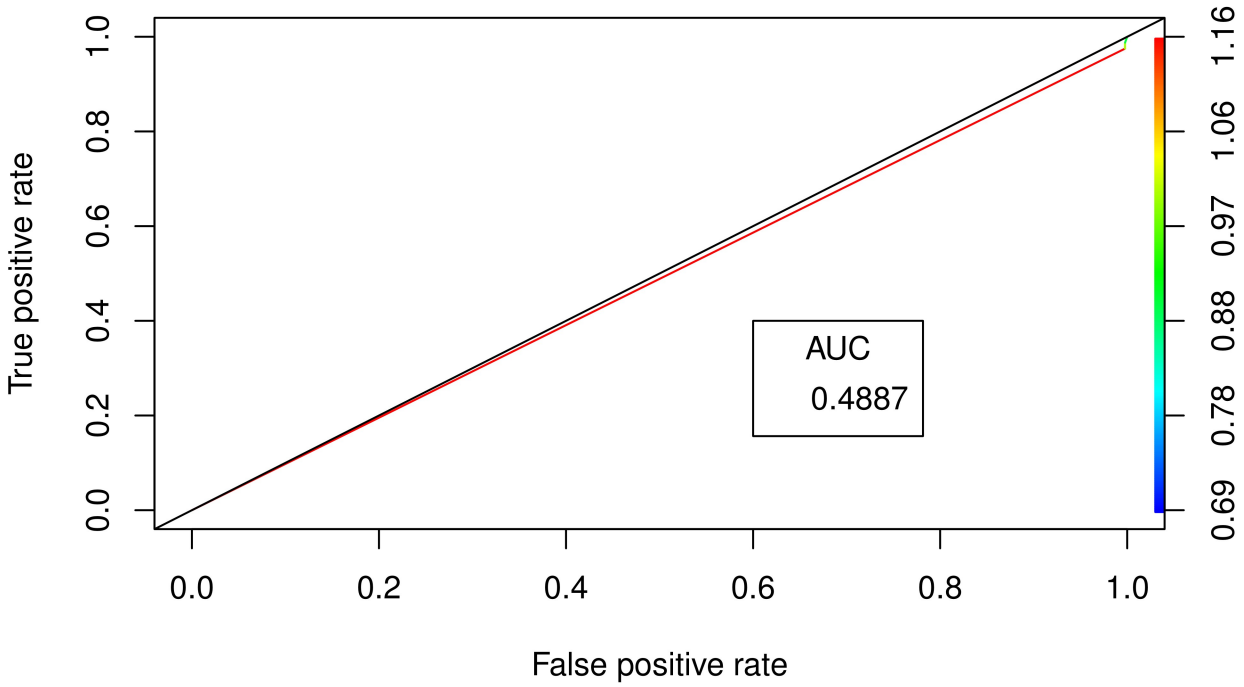
### Confusion Matrix and ROC-AUC Curve

For k=1

```
knn1 <- knn(train = train_reg,test = test_reg,cl = train_reg$MCQ160F,k = 1,prob=TRUE)
# Confusiin Matrix
cm <- table(test_reg$MCQ160F, knn1)
cm
```

```
##    knn1
##       0   1
##   0 860   0
##   1   2  78
```

```
prob <- attr(knn1, "prob")
pred <- prediction(prob,test_reg$MCQ160F )
perf <- performance(pred,"tpr","fpr")
auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
plot(perf,colorize=TRUE)
abline(a = 0, b = 1)
auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```

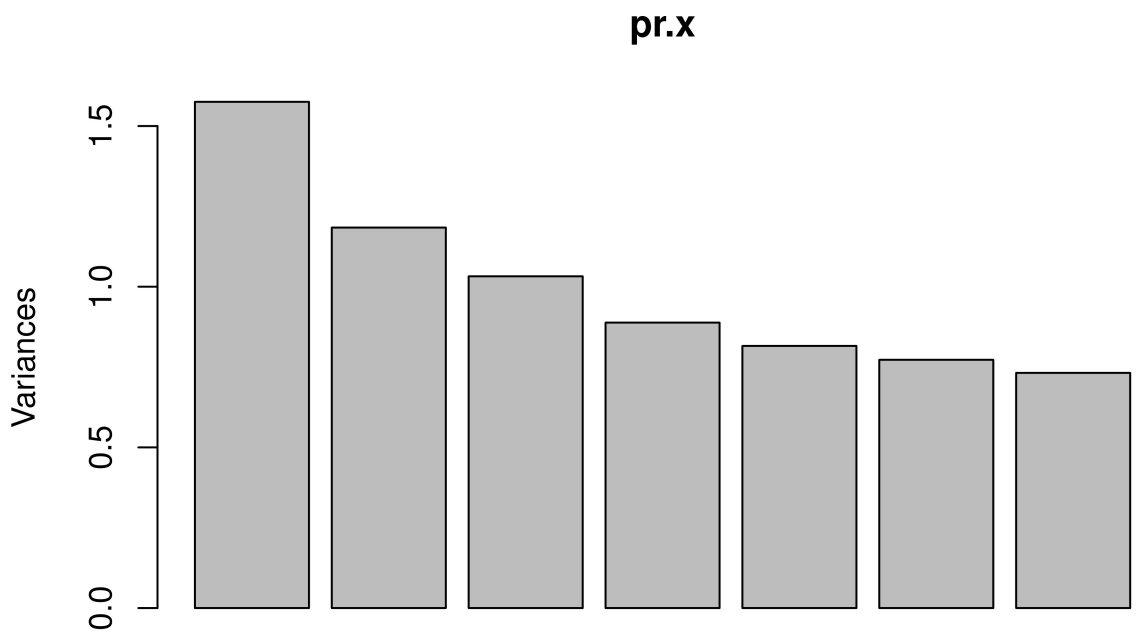



**Conclusion :**
For K=1 ,KNN has Accuracy =0.997872340425532,Recall=0.9976798143851,Precision=1 and AUC is 0.4887 .So,except AUC all the other evaluation metrics are better for this model .So, KNN with K=1 will be the best model in this case .

**Principal Component Analysis (PCA)**

PCA is one of the most popular linear dimension reduction algorithms. It is a projection based method that transforms the data by projecting it onto a set of orthogonal (perpendicular) axes.
"PCA works on a condition that while the data in a higher-dimensional space is mapped to data in a lower dimension space, the variance or spread of the data in the lower dimensional space should be maximum."
PCA will give more importance to the feature with high variance. So, after applying PCA it's interesting to see how much variance each principal component captures for that we have made a screeplot .

```
pr.x <- prcomp(data[,-c(7)], center = TRUE, scale. = TRUE)
screeplot(pr.x)
```




**Conclusion :**

The first 5 Principal Components are capturing around 80% of the variance so we can replace the 7 original features with the new 5 features having 80% of the information. So, we have reduced the 7 dimensions to only 5 dimensions while retaining most of the information.Now we will fit the model using these features as predictors .

**Fitting Logistict Regression**

```
p<-data.frame(pr.x$x)
data1<-data.frame(data$MCQ160F,p$PC1,p$PC2,p$PC3,p$PC4,p$PC5)
set.seed(6)
split <- sample.split(data1, SplitRatio = 0.8)
trdata <- subset(data1, split == "TRUE")
tsdata <- subset(data1, split == "FALSE")
l<-glm(data.MCQ160F~.,data=trdata,family="binomial")
summary(l)
```

```
##
## Call:
## glm(formula = data.MCQ160F ~ ., family = "binomial", data = trdata)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
```

```
## -1.89440  -0.07048  -0.03952  -0.01606   2.74507
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.9628     0.3717 -16.043  < 2e-16 ***
## p.PC1         2.6318     0.1892  13.912  < 2e-16 ***
## p.PC2         0.1472     0.1455   1.012   0.3115
## p.PC3         0.4949     0.1065   4.649 3.34e-06 ***
## p.PC4         0.2364     0.1218   1.942   0.0522 .
## p.PC5        -1.2969     0.1533  -8.462  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1443.6  on 2507  degrees of freedom
## Residual deviance:  390.7  on 2502  degrees of freedom
## AIC: 402.7
##
## Number of Fisher Scoring iterations: 8
```

```
predict_reg1 <- predict(l,tsdata, type = "response")

# Changing probabilities
predict_reg1 <- ifelse(predict_reg1 >0.5, 1, 0)
# Evaluating model accuracy
# using confusion matrix
table(tsdata$data.MCQ160F, predict_reg1)
```

```
##    predict_reg1
##        0    1
##   0 1123   16
##   1   42   72
```

```
missing_classerr1 <- mean(predict_reg1 != tsdata$data.MCQ160F)
print(paste('Accuracy =', 1 - missing_classerr1))
```

```
## [1] "Accuracy = 0.953711093375898"
```

**Conclusion**
AIC for the model get reduce to 402.7 which is a quiet small number than the AIC value for the model without dimension reduction .
Here it classifies all as 0 and the accuracy, Precision and Recall are 95%,99%,96% approx respectively which is quiet good .So, here dimension reduction works well .

Now we will evaluate the other models LDA,QDA and KNN (K=1) after applying PCA .

**LDA**

```
lda<-lda(data.MCQ160F~. , data =trdata)
predictions <- lda %>% predict(tsdata)
table(tsdata$data.MCQ160F, predictions$class)
```

```
##
##        0    1
##   0 1112   27
##   1   40   74
```

```
print(paste('Accuracy =', mean(predictions$class==tsdata$data.MCQ160F)))
```

```
## [1] "Accuracy = 0.946528332003192"
```

**Conclusion :**
Accuracy,Precision and Recall are 95%,98% and 97% approx respectively which is again better than the model before dimension reduction .

### QDA

```
qda<-qda(data.MCQ160F~. , data =trdata)
predictions <- qda %>% predict(tsdata)
table(tsdata$data.MCQ160F, predictions$class)
```

```
##
##        0    1
##   0 1115   24
##   1   32   82
```

```
print(paste('Accuracy =', mean(predictions$class==tsdata$data.MCQ160F)))
```

```
## [1] "Accuracy = 0.955307262569832"
```

**Conclusion :**
Accuracy,Precision and Recall are 96%,98% and 97% approx respectively which is again better than the model before dimension reduction .

### KNN (K=1)

```
knn1 <- knn(train = trdata,test = tsdata,cl = trdata$data.MCQ160F,k = 1,prob=TRUE)
cm <- table(tsdata$data.MCQ160F, knn1)
cm
```

```
##    knn1
##        0    1
##   0 1139    0
##   1    1  113
```

```r
misClassError <- mean(knn1 != tsdata$data.MCQ160F)
print(paste('Accuracy =', 1-misClassError))
```

```
## [1] "Accuracy = 0.999201915403033"
```

**Conclusion :**
Accuracy,Precision and Recall are 100%,100% and 100% approx respectively which is again better than the model before dimension reduction .

**Final Comment :**
So, comparing all the models based on Accuracy,Precision and Recall we found **PCA-KNN(K=1)** model performs best and hence we will go with this model .