Introduction

Decision Trees

A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute. Answers to this test may be “Yes” or “No” , “Less than” or “Greater than” etc. The branch which resulted from this test represents those outcomes, when no more branching is possible, we end up with leaf nodes which each of them represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. It is easy to understand how deciding mechanism works.

Naïve Bayes

Naive Bayes is based on Bayes' Theorem. It relies on the assumption of independence among predictors.In other words, it assumes relation between predictors is not dependent. The Naive Bayes model is useful for very large data sets.

k-NN

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The case being assigned to the class is the most common among its K nearest neighbors measured by a distance function. If K value is so low algorithm is prone to noisy data. If K value is too large decision will be biased to class having more samples.

Dataset

Dataset in this assignment contains comprehensive information about bike buyers. It involves data regarding person's marital status, gender, yearly income, number of children. Moreover, information related to their children which are at home, their English education level, whether they own a house or not, number of cars they posses, their commute distance, which region they are at ,their age and finally whether they are bike buyers or not is available.

Using str() function, we are able to see structure of the dataset, training_set and test_set have share the same structure, only difference between them is number of observations whereas 13.863 in the training_set and 4.621 in the test_set.

```
> str(training_set)
'data.frame':	13863 obs. of  12 variables:
 $ MaritalStatus       : Factor w/ 5 levels "1","2","3","4",..: 5 5 5 5 5 5 5 5 5 5 ...
 $ Gender              : Factor w/ 2 levels "1","2": 1 1 1 2 2 1 1 2 1 1 ...
 $ YearlyIncome        : int  90000 60000 60000 70000 80000 70000 60000 60000 70000 60000 ...
 $ TotalChildren       : int  2 3 3 0 5 0 3 4 0 4 ...
 $ NumberChildrenAtHome: int  0 3 3 0 5 0 3 4 0 4 ...
 $ EnglishEducation    : Factor w/ 5 levels "1","2","3","4",..: 5 5 5 5 5 5 5 5 5 5 ...
 $ HouseOwnerFlag      : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 2 1 2 ...
 $ NumberCarsOwned     : int  0 1 1 1 4 1 2 3 1 4 ...
 $ CommuteDistance     : int  2 1 5 10 2 10 1 20 10 20 ...
 $ Region              : Factor w/ 3 levels "1","2","3": 2 2 2 2 2 2 2 2 2 2 ...
 $ Age                 : int  50 51 51 49 48 51 52 52 52 53 ...
 $ BikeBuyer           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
> str(test_set)
'data.frame':	4621 obs. of  12 variables:
 $ MaritalStatus       : Factor w/ 5 levels "1","2","3","4",..: 5 5 5 1 1 3 3 2 3 3 ...
 $ Gender              : Factor w/ 2 levels "1","2": 2 2 1 2 1 1 2 2 1 2 ...
 $ YearlyIncome        : int  70000 70000 100000 20000 40000 60000 30000 40000 30000 60000 ...
 $ TotalChildren       : int  0 0 2 4 0 0 2 0 0 0 ...
 $ NumberChildrenAtHome: int  0 0 0 0 0 0 0 0 0 0 ...
 $ EnglishEducation    : Factor w/ 5 levels "1","2","3","4",..: 5 5 5 1 1 3 3 2 3 3 ...
 $ HouseOwnerFlag      : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 2 1 2 2 ...
 $ NumberCarsOwned     : int  1 1 3 2 2 2 2 2 2 2 ...
 $ CommuteDistance     : int  10 10 1 10 2 10 2 10 2 2 ...
 $ Region              : Factor w/ 3 levels "1","2","3": 2 2 1 2 1 1 2 1 1 1 ...
 $ Age                 : int  51 53 48 72 38 38 70 39 39 39 ...
 $ BikeBuyer           : Factor w/ 2 levels "0","1": 2 2 1 2 2 1 2 2 2 2 ...
```

Moreover, from the above figure it is possible to observe value range and density of values for each feature.

```
> summary(training_set)
 MaritalStatus Gender     YearlyIncome     TotalChildren   NumberChildrenAtHome EnglishEducation
 1:2451        1:7002   Min.   : 10000   Min.   :0.000   Min.   :0.000        1:2451          
 2:1201        2:6861   1st Qu.: 30000   1st Qu.:0.000   1st Qu.:0.000        2:1201          
 3:3802                 Median : 60000   Median :2.000   Median :0.000        3:3802          
 4:2376                 Mean   : 57494   Mean   :1.852   Mean   :1.014        4:2376          
 5:4033                 3rd Qu.: 70000   3rd Qu.:3.000   3rd Qu.:2.000        5:4033          
                        Max.   :170000   Max.   :5.000   Max.   :5.000                        
 HouseOwnerFlag NumberCarsOwned CommuteDistance  Region       Age          BikeBuyer
 0:4452         Min.   :0.000   Min.   : 1.000   1:7024   Min.   : 36.00   0:7014   
 1:9411         1st Qu.:1.000   1st Qu.: 1.000   2:2706   1st Qu.: 46.00   1:6849   
                Median :2.000   Median : 2.000   3:4133   Median : 53.00            
                Mean   :1.508   Mean   : 6.039            Mean   : 54.68            
                3rd Qu.:2.000   3rd Qu.:10.000            3rd Qu.: 62.00            
                Max.   :4.000   Max.   :20.000            Max.   :106.00            
```

Below line, represents the importance of attributes when they are used to predict BikeBuyer feature.

```
infoGain = information.gain(BikeBuyer~.,training_set)
```

```
> infoGain
                     attr_importance
MaritalStatus           0.0107673653
Gender                  0.0001670052
YearlyIncome            0.0026831351
TotalChildren           0.0100003816
NumberChildrenAtHome    0.0086893900
EnglishEducation        0.0107673653
HouseOwnerFlag          0.0000335436
NumberCarsOwned         0.0206438823
CommuteDistance         0.0106020872
Region                  0.0055910482
Age                     0.0267364247
```

Results

1) Decision Trees

```
decisiontreeModel = rpart(BikeBuyer ~ ., data = training_set,method = "class")
decisiontreePrediction = predict(decisiontreeModel, newdata=test_set, type="class")
decisiontreeResult = table(test_set$BikeBuyer,decisiontreePrediction)
```

Running the above code, I have developed a decision tree which estimates BikeBuyer attribute using rest of the attributes.

Confusion Matrix

```
 decisiontreePrediction
     0    1
0 1686  652
1  952 1331
```

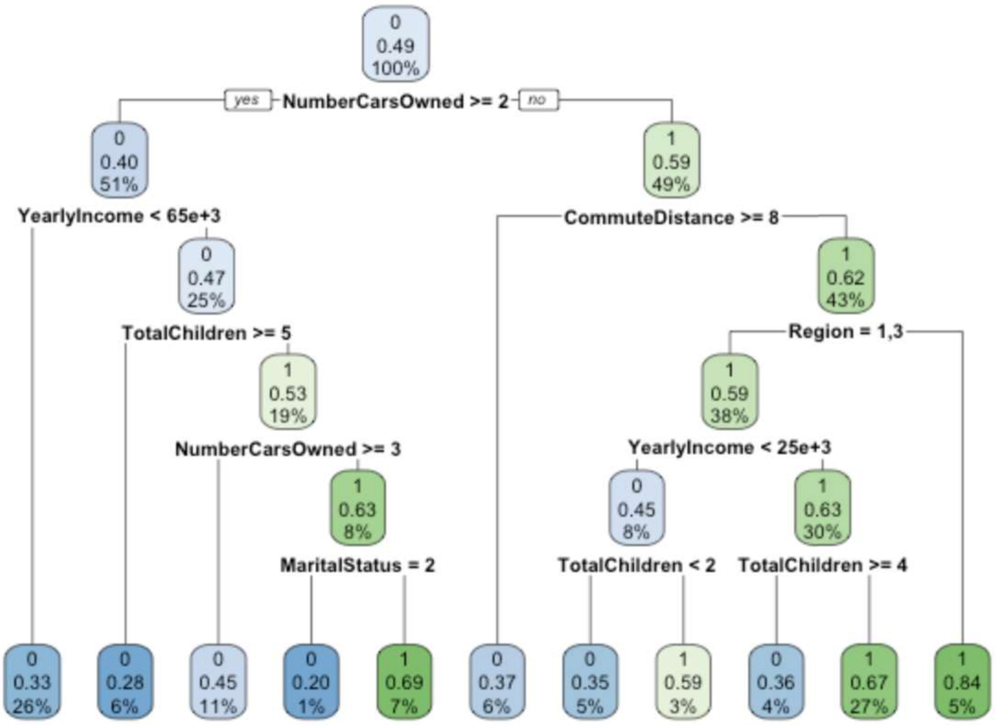
```
               Accuracy : 0.6529
                 95% CI : (0.639, 0.6666)
    No Information Rate : 0.5709
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3046

 Mcnemar's Test P-Value : 8.287e-14

            Sensitivity : 0.6391
            Specificity : 0.6712
         Pos Pred Value : 0.7211
         Neg Pred Value : 0.5830
             Prevalence : 0.5709
         Detection Rate : 0.3649
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.6552
```

Accuracy of the model is 65.3%.

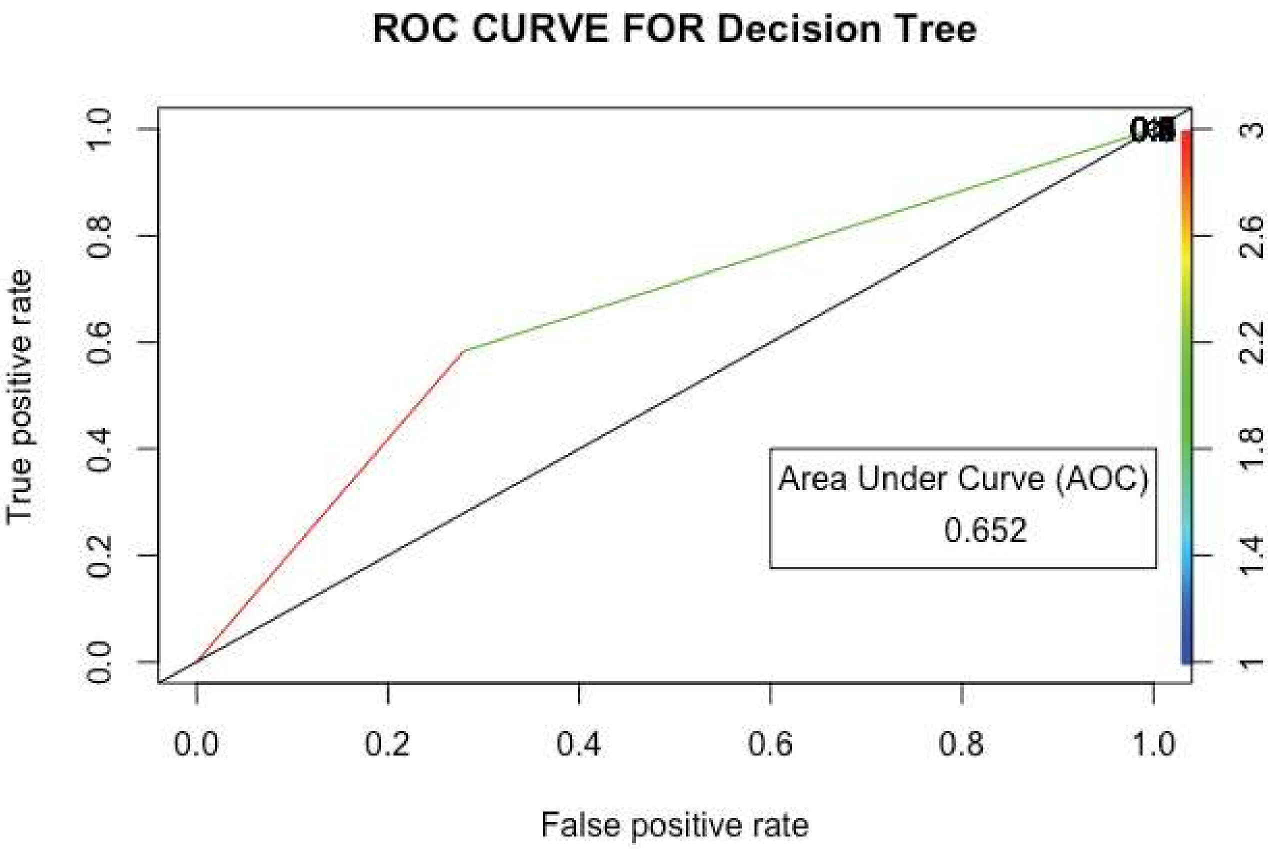
P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 63.91%.

Specificity (False Positive Rate) of the model is 67.12%.



Also, from the below figure it is possible to observe ROC curve for the model.



Next, I have applied normalization and discretization to requested attributes.

```
requestedNormalization = c(3:5) # 3:YearlyIncome 4:TotalChildren 5:NumberChildrenAtHome

training_set_processed = training_set
test_set_processed = test_set

age_cat1 = training_set %>% filter(Age<51) %>% mutate(Age=1)
age_cat2 = training_set %>% filter(Age>50 & Age<66) %>% mutate(Age=2)
age_cat3 = training_set %>% filter(Age>65) %>% mutate(Age=3)
training_set_processed$Age = c(age_cat1[,11] , age_cat2[,11] ,age_cat3[,11])

age_cat1 = test_set %>% filter(Age<51) %>% mutate(Age=1)
age_cat2 = test_set %>% filter(Age>50 & Age<66) %>% mutate(Age=2)
age_cat3 = test_set %>% filter(Age>65) %>% mutate(Age=3)
test_set_processed$Age = c(age_cat1[,11] , age_cat2[,11] ,age_cat3[,11])

training_set_processed[,requestedNormalization] = scale(training_set[,requestedNormalization])
test_set_processed[,requestedNormalization] = scale(test_set[,requestedNormalization])
```

```
decisiontreeProcessedModel = rpart(BikeBuyer ~ ., data = training_set_processed,method = "class")
decisiontreeProcessedPrediction = predict(decisiontreeProcessedModel, newdata=test_set_processed, type="class")
decisiontreeProcessedResult = table(test_set$BikeBuyer,decisiontreeProcessedPrediction)
```

I have build the model and performed prediction.

Confusion Matrix

```
  decisiontreeProcessedPrediction
       0    1
0 1456  882
1  819 1464
```

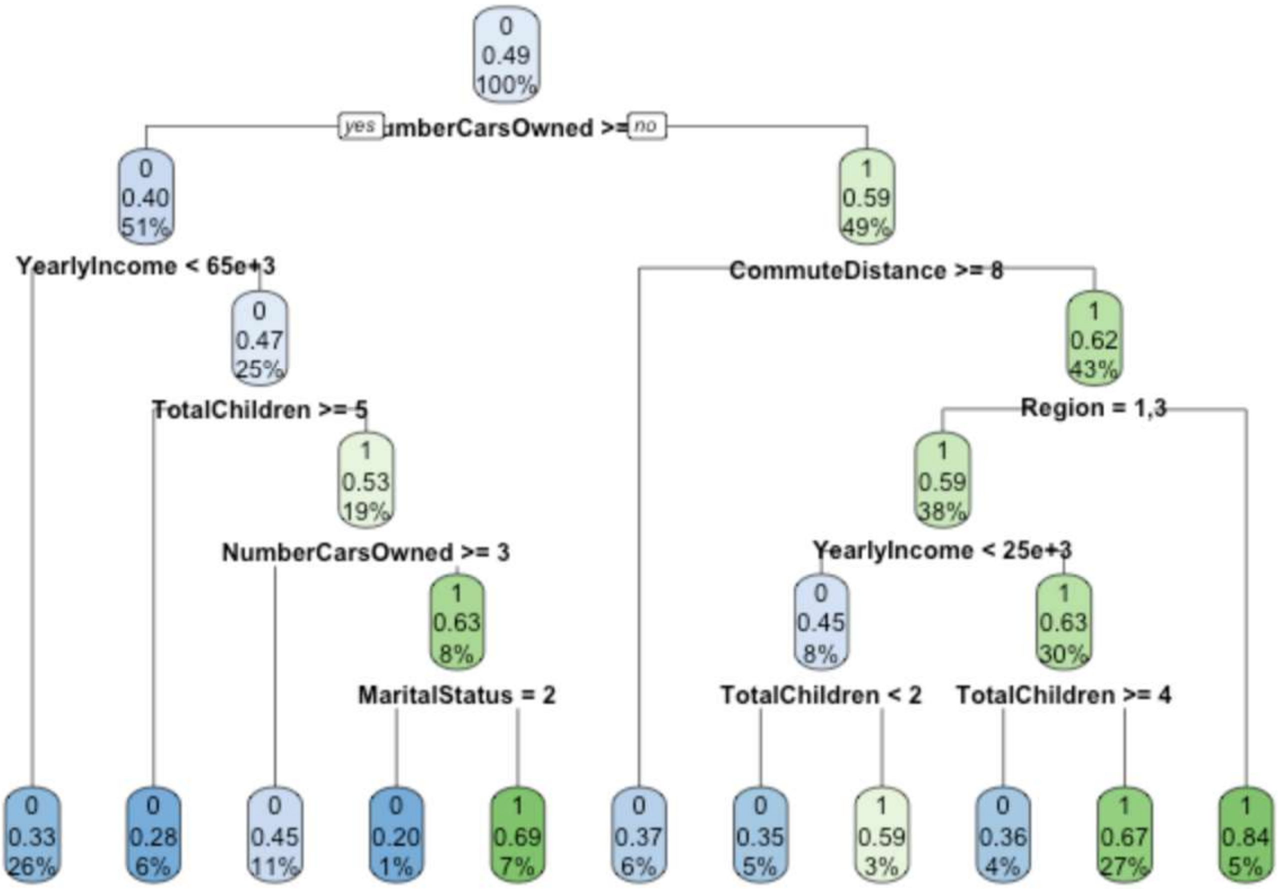
```
               Accuracy : 0.6319
                 95% CI : (0.6178, 0.6458)
    No Information Rate : 0.5077
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.2639

 Mcnemar's Test P-Value : 0.1328

            Sensitivity : 0.6400
            Specificity : 0.6240
         Pos Pred Value : 0.6228
         Neg Pred Value : 0.6413
             Prevalence : 0.4923
         Detection Rate : 0.3151
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.6320
```

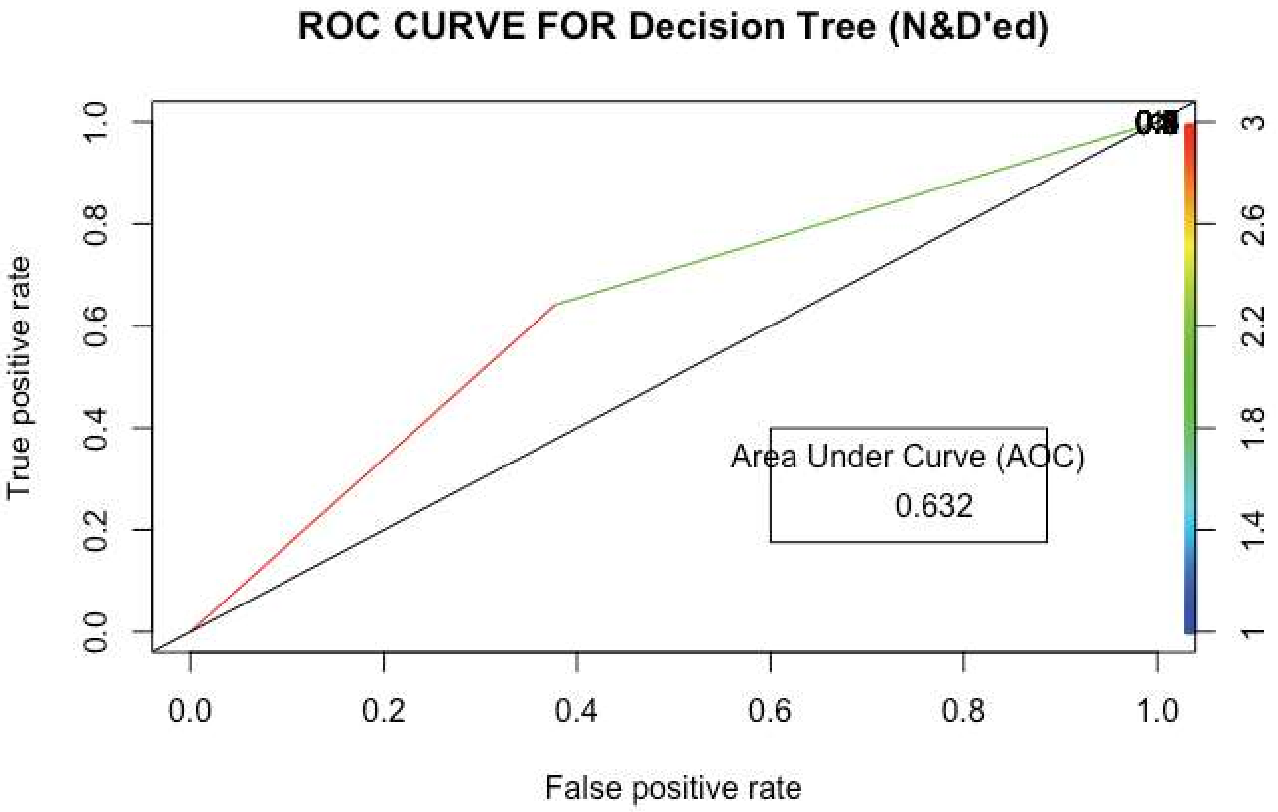
Accuracy of the model is 63.2%.

P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 64%.

Specificity (False Positive Rate) of the model is 62.4%.





In comparison to initial model I have not seen any significant difference in accuracy, however when we compare decision trees, latter one have a lower branching factor which I believe reduces the complexity.

2) Naive Bayes

```
naiveBayesModel = naiveBayes(BikeBuyer~.,data=training_set)
naiveBayesPrediction = predict(naiveBayesModel,newdata=test_set)
naiveBayesResult = table(test_set$BikeBuyer,naiveBayesPrediction)
```

I have constructed the model using the same formula as the decision tree model.

Confusion Matrix

```
 naiveBayesPrediction
      0    1
0 1351  987
1  710 1573
```

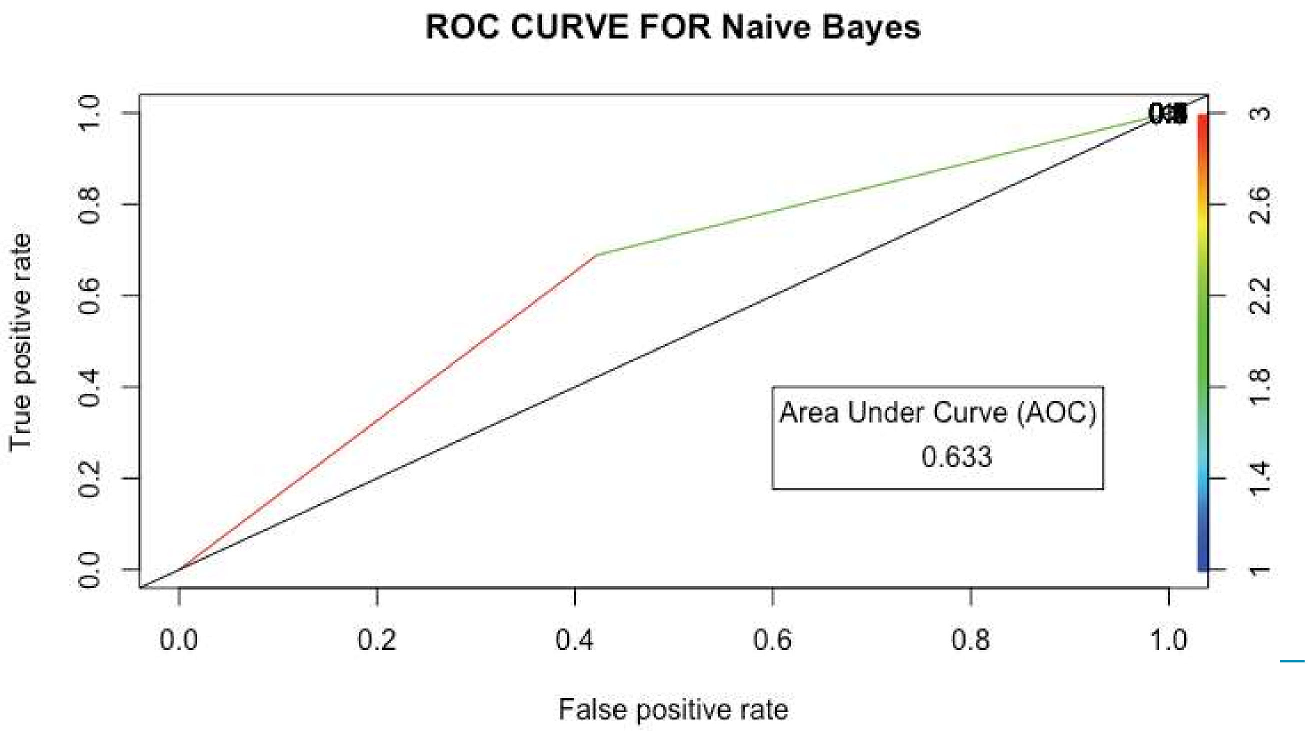
```
               Accuracy : 0.6328
                 95% CI : (0.6187, 0.6467)
    No Information Rate : 0.554
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2665

 Mcnemar's Test P-Value : 2.086e-11

            Sensitivity : 0.6555
            Specificity : 0.6145
         Pos Pred Value : 0.5778
         Neg Pred Value : 0.6890
             Prevalence : 0.4460
         Detection Rate : 0.2924
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.6350
```

Accuracy of the model is 63.3%.

P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 65.5%.

Specificity (False Positive Rate) of the model is 61.45%.



Then, I have applied normalization and discretization to the specified features.

```
age_cat1 = training_set %>% filter(Age<51) %>% mutate(Age=1)
age_cat2 = training_set %>% filter(Age>50 & Age<66) %>% mutate(Age=2)
age_cat3 = training_set %>% filter(Age>65) %>% mutate(Age=3)
training_set_processed$Age = c(age_cat1[,11] , age_cat2[,11] ,age_cat3[,11])

age_cat1 = test_set %>% filter(Age<51) %>% mutate(Age=1)
age_cat2 = test_set %>% filter(Age>50 & Age<66) %>% mutate(Age=2)
age_cat3 = test_set %>% filter(Age>65) %>% mutate(Age=3)
test_set_processed$Age = c(age_cat1[,11] , age_cat2[,11] ,age_cat3[,11])

training_set_processed[,requestedNormalization] = scale(training_set[,requestedNormalization])
test_set_processed[,requestedNormalization] = scale(test_set[,requestedNormalization])
```

Using processed data set , I have trained the model and did a prediction.

```
naiveBayesProcessedModel = naiveBayes(BikeBuyer~.,data=training_set_processed)
naiveBayesProcessedPrediction = predict(naiveBayesProcessedModel,newdata=test_set_processed)
naiveBayesProcessedResult = table(test_set$BikeBuyer,naiveBayesProcessedPrediction)
```

Confusion Matrix

```
 naiveBayesProcessedPrediction
     0    1
0 1302 1036
1  726 1557
```

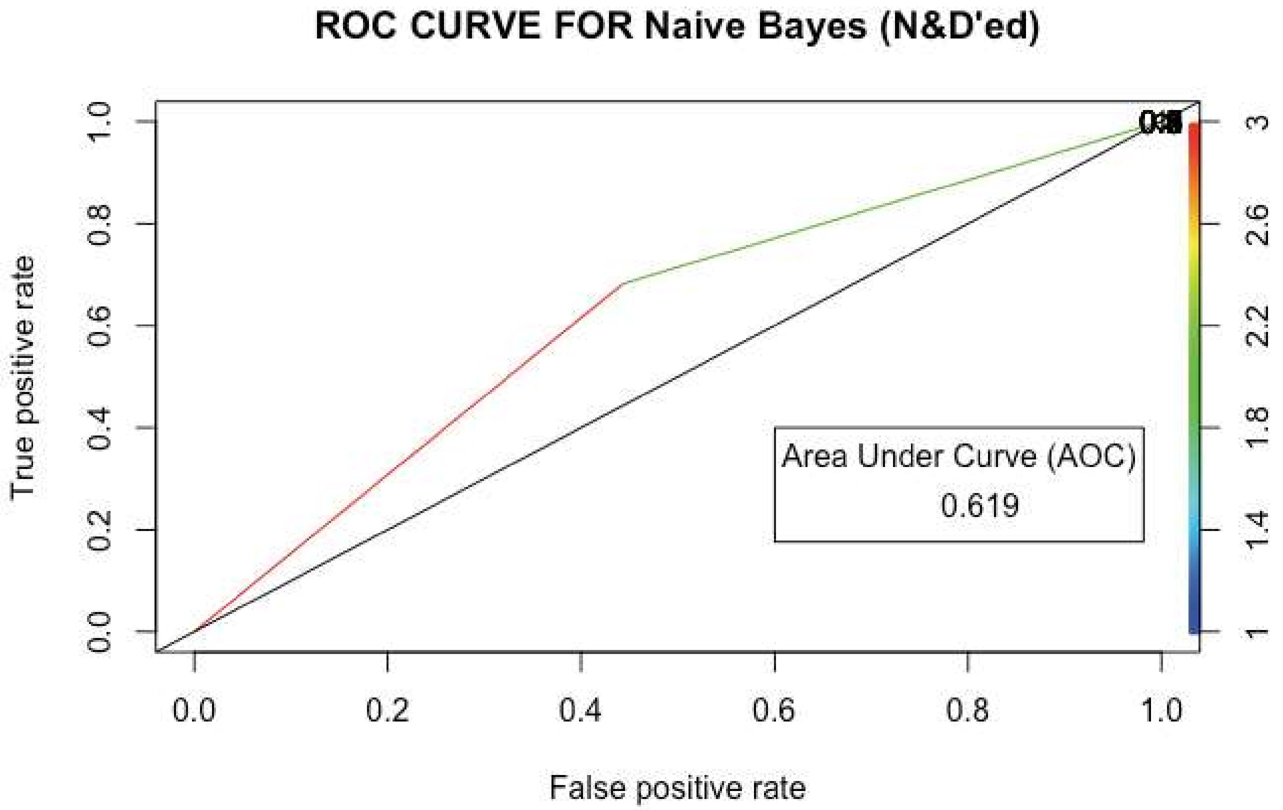
```
               Accuracy : 0.6187
                 95% CI : (0.6045, 0.6327)
    No Information Rate : 0.5611
    P-Value [Acc > NIR] : 1.220e-15

                  Kappa : 0.2385

Mcnemar's Test P-Value : 1.821e-13

            Sensitivity : 0.6420
            Specificity : 0.6005
         Pos Pred Value : 0.5569
         Neg Pred Value : 0.6820
             Prevalence : 0.4389
         Detection Rate : 0.2818
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.6212
```

Accuracy of the model is around 62%.

P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 64.2%.

Specificity (False Positive Rate) of the model is 60%.



3) K-NN

As K-NN works for numerical data only, first I have selected the features which have numerical data.

```
numericalAttributes = c(3:5,8,9,11)
```

```
$ YearlyIncome
$ TotalChildren
$ NumberChildrenAtHome
$ NumberCarsOwned
$ CommuteDistance
$ Age
```

From the left figure, you can check the name of these features.

Later on, I have performed PCA on the training data set having only numerical features to determine their importance.

```
> summary(training.pca)
Importance of components:
                         PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.586 1.0879 0.9644 0.7878 0.69613 0.51701
Proportion of Variance 0.419 0.1972 0.1550 0.1034 0.08077 0.04455
Cumulative Proportion  0.419 0.6162 0.7713 0.8747 0.95545 1.00000
```

Since distance is being used in KNN, it is crucial to scale features so that no bias occurs respect to their importance. Below figure demonstrates scale operation for the numerical features in the dataset.

```
training_set_scaled = scale(training_set[,numericalAttributes])
test_set_scaled = scale(test_set[,numericalAttributes])
```

Afterwards, I have trained the model and performed a prediction.

```
knnPrediction = knn(train=training_set_scaled,
                    test=test_set_scaled,
                    cl=training_set$BikeBuyer,k=kValue,prob=TRUE)
knnResult = table(test_set$BikeBuyer,knnPrediction)

confusionMatrix(knnResult)
plotROC(predictedSet = knnPrediction,testSet = test_set$BikeBuyer,
        title=paste("KNN , K=",kValue))
```

For k=3 results are as follows,

Confusion Matrix

```
  knnPrediction
       0    1
0 1779  559
1  510 1773
```

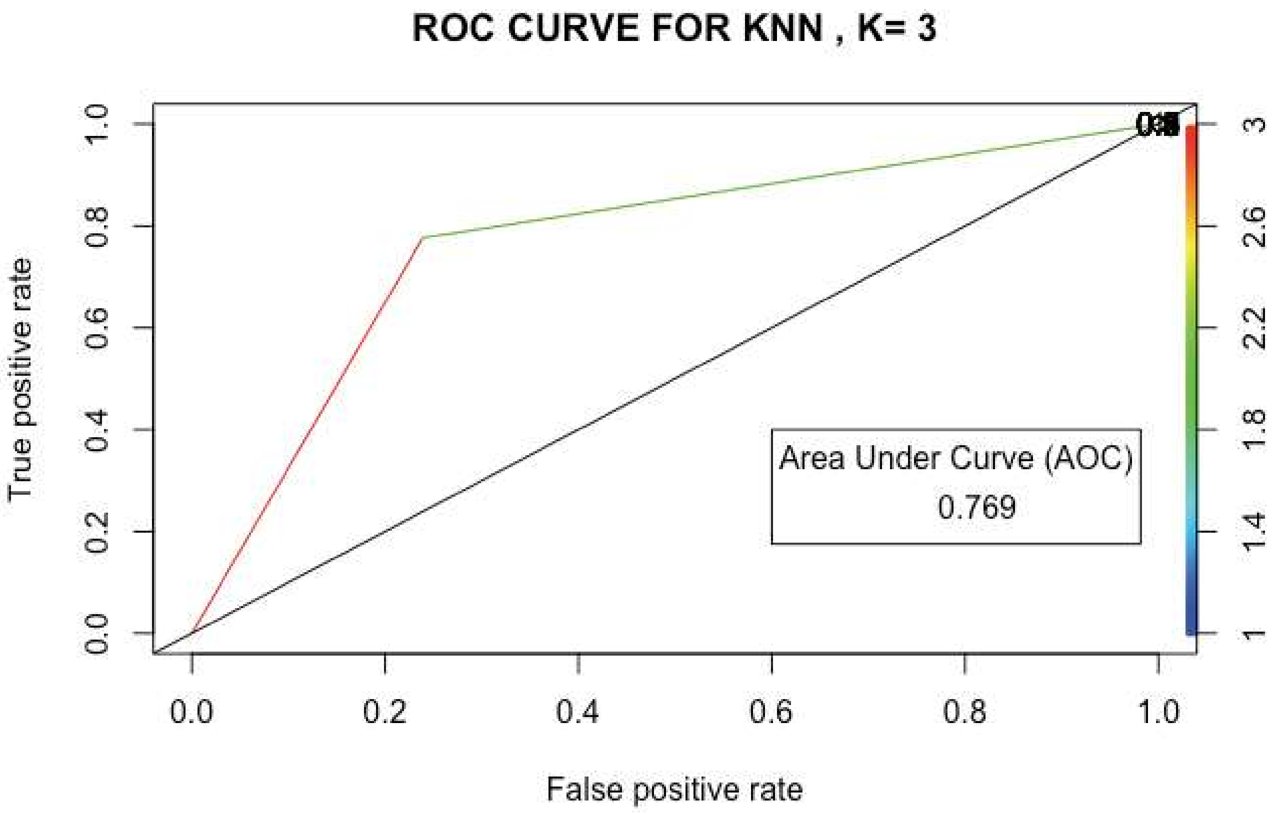
```
               Accuracy : 0.7687
                 95% CI : (0.7562, 0.7808)
    No Information Rate : 0.5047
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5374

 Mcnemar's Test P-Value : 0.1421

            Sensitivity : 0.7772
            Specificity : 0.7603
         Pos Pred Value : 0.7609
         Neg Pred Value : 0.7766
             Prevalence : 0.4953
         Detection Rate : 0.3850
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.7687
```

Accuracy of the model is 76.87%.

P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 77.72%.

Specificity (False Positive Rate) of the model is 76.03%.



For k=5 I have obtained the results shown below,

Confusion Matrix

```
 knnPrediction
      0    1
0 1779  559
1  512 1771
```

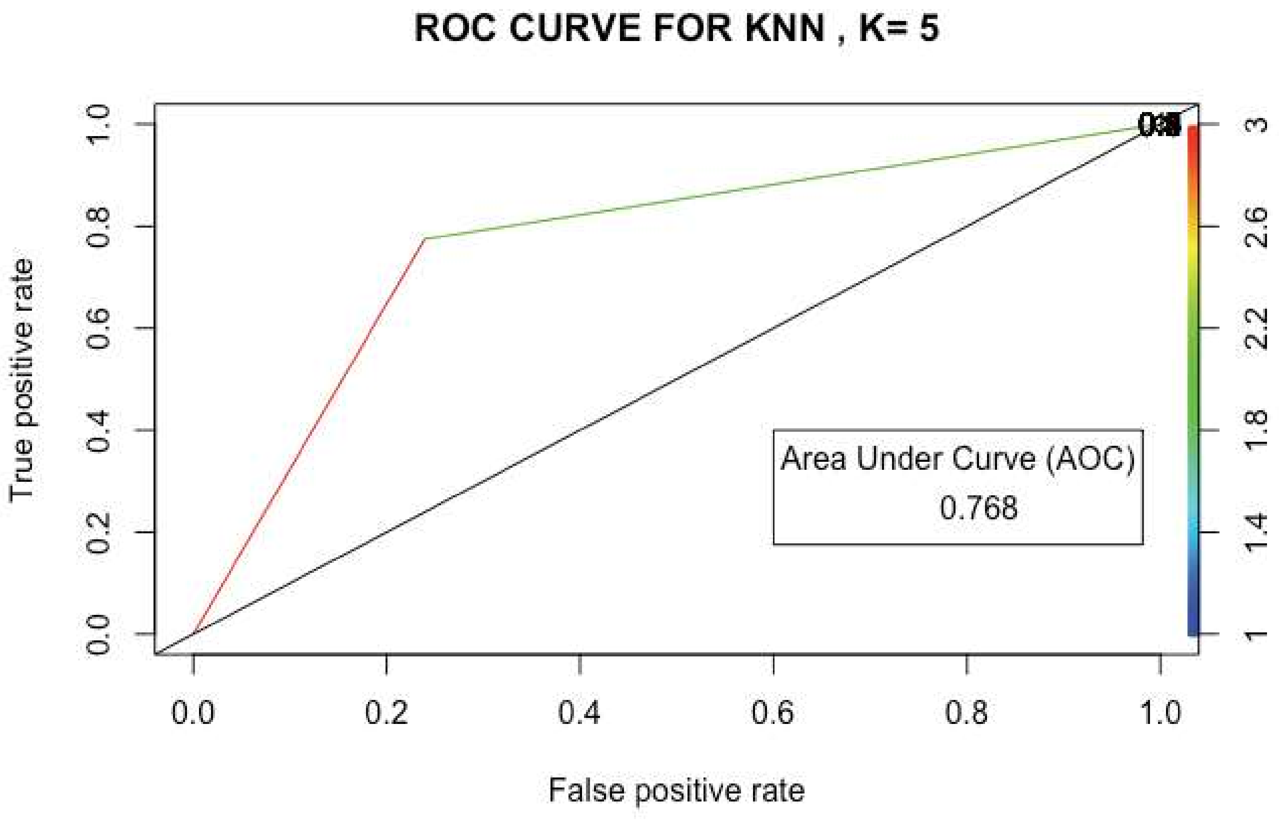
```
                  Accuracy : 0.7682
                    95% CI : (0.7558, 0.7803)
       No Information Rate : 0.5042
       P-Value [Acc > NIR] : <2e-16

                     Kappa : 0.5365

Mcnemar's Test P-Value : 0.1598

               Sensitivity : 0.7765
               Specificity : 0.7601
            Pos Pred Value : 0.7609
            Neg Pred Value : 0.7757
                Prevalence : 0.4958
            Detection Rate : 0.3850
      Detection Prevalence : 0.5060
         Balanced Accuracy : 0.7683
```

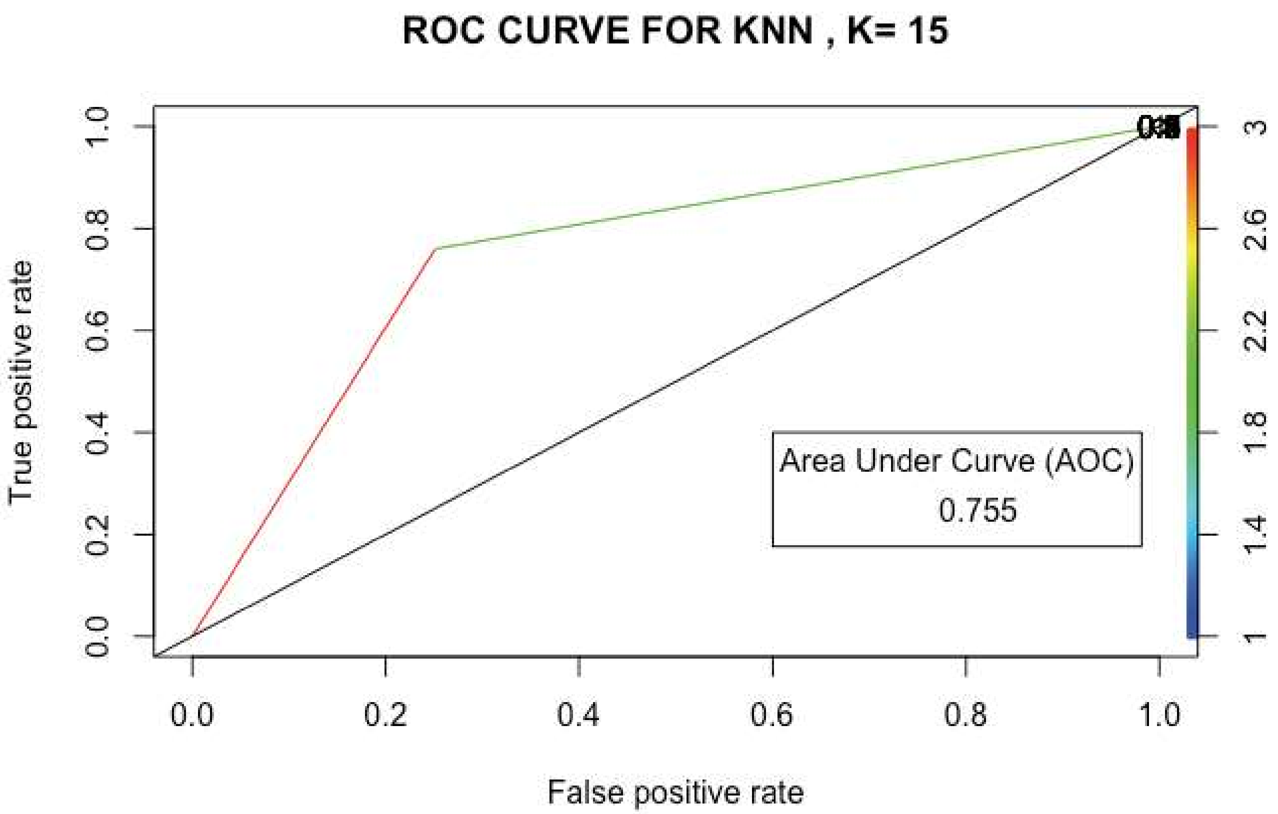
Accuracy of the model is 76.82%.

P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 77.65%.

Specificity (False Positive Rate) of the model is 76%.





I have realized that when I increase the K, accuracy started to decrease which I believe it is because of under-fit (high bias and low variance) issue.

Finally, I have performed the same procedure with applying normalization and discretization to the requested features.

```
training_set_processed = scale(training_set[,c(numericalAttributes,requestedNormalization)])
test_set_processed = scale(test_set[,c(numericalAttributes,requestedNormalization)])
```

I have trained my model using the requested features and numerical ones which were intersecting anyway.

```
knnPrediction = knn(train=training_set_processed,test=test_set_processed,
                    cl=training_set$BikeBuyer,
                    k=kValue,prob=TRUE)
knnResult = table(test_set$BikeBuyer,knnPrediction)
```

For k=3 I have obtained the results shown below,

Confusion Matrix

```
  knnPrediction
       0    1
0 1778  560
1  490 1793
```

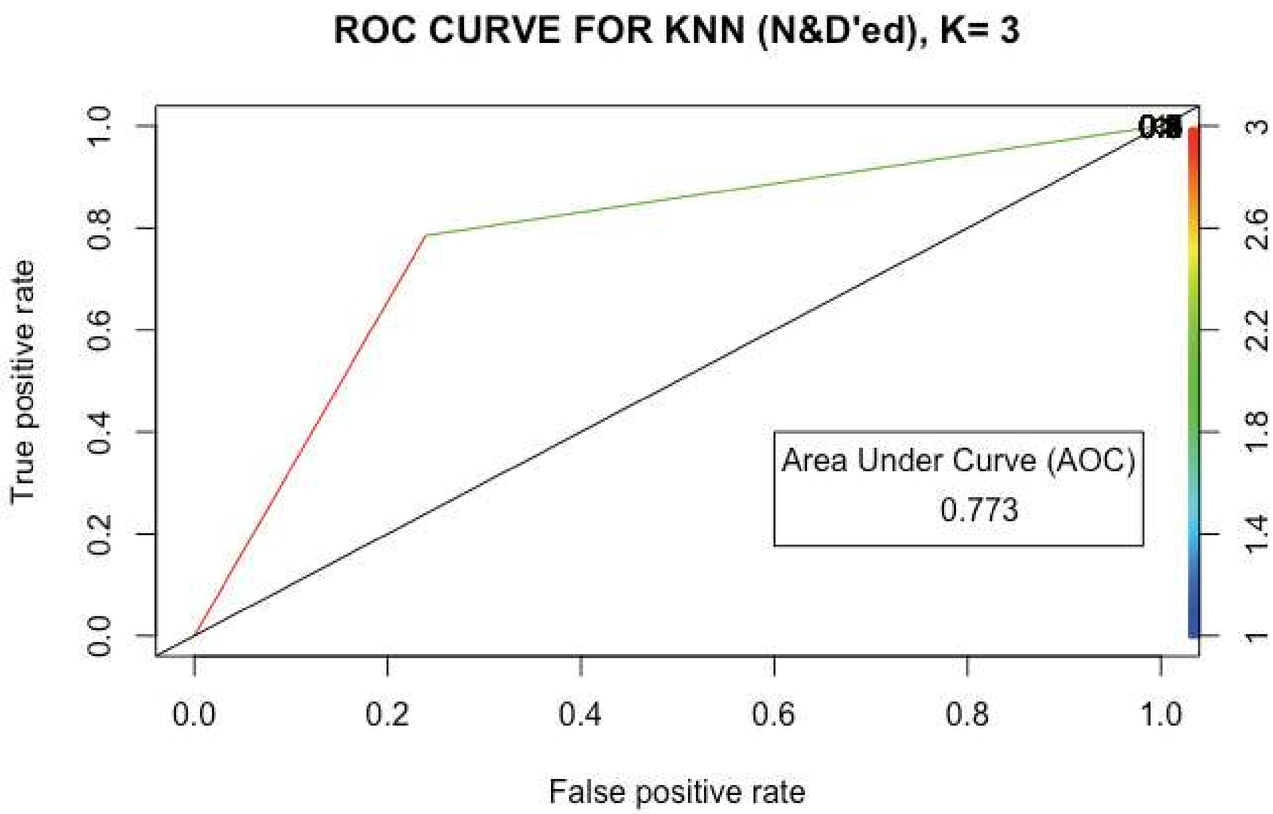
```
               Accuracy : 0.7728
                 95% CI : (0.7604, 0.7848)
    No Information Rate : 0.5092
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5457

Mcnemar's Test P-Value : 0.03322

            Sensitivity : 0.7840
            Specificity : 0.7620
         Pos Pred Value : 0.7605
         Neg Pred Value : 0.7854
             Prevalence : 0.4908
         Detection Rate : 0.3848
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.7730
```

Accuracy of the model is around 77.3%.

P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 78.40%.

Specificity (False Positive Rate) of the model is 76.2%.



For k=5,

Confusion Matrix

```
  knnPrediction
       0    1
0 1779  559
1  512 1771
```

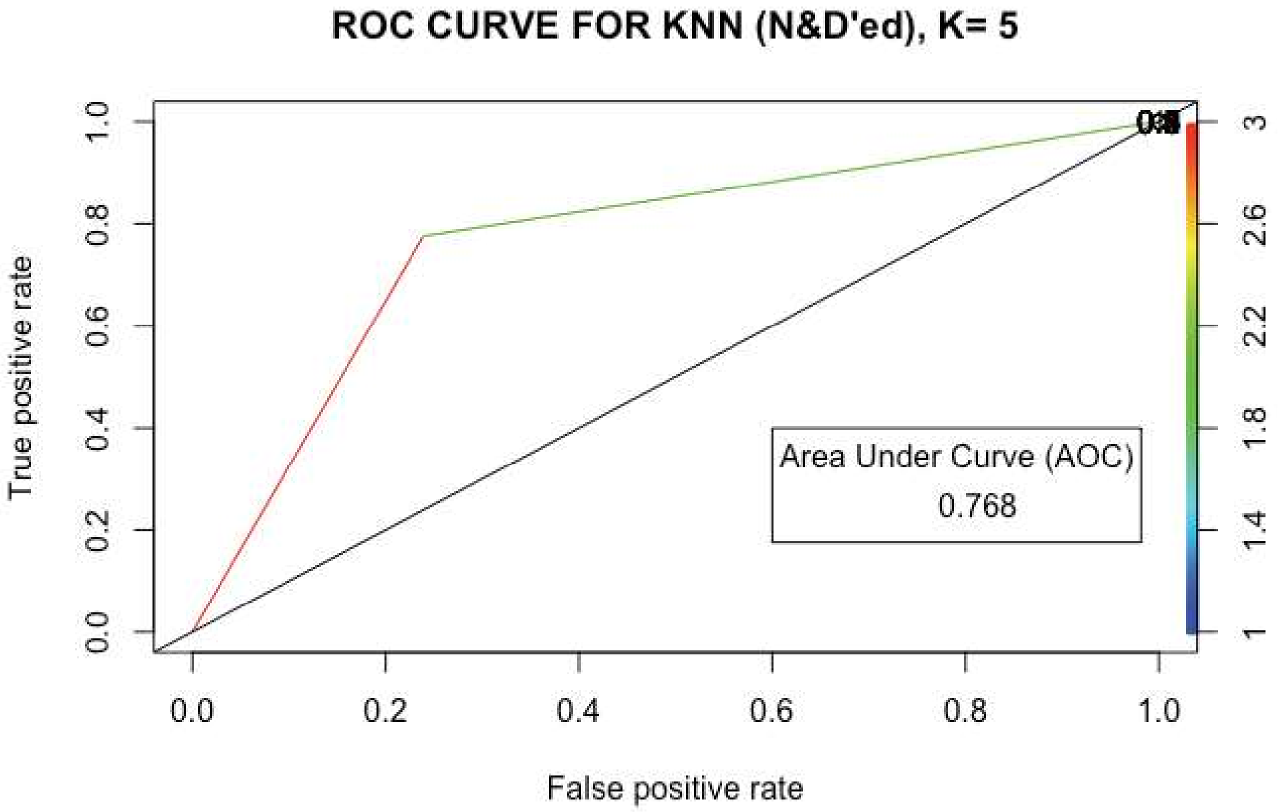
```
               Accuracy : 0.7682
                 95% CI : (0.7558, 0.7803)
    No Information Rate : 0.5042
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5365

Mcnemar's Test P-Value : 0.1598

            Sensitivity : 0.7765
            Specificity : 0.7601
         Pos Pred Value : 0.7609
         Neg Pred Value : 0.7757
             Prevalence : 0.4958
         Detection Rate : 0.3850
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.7683
```

Accuracy of the model is 76.82%.

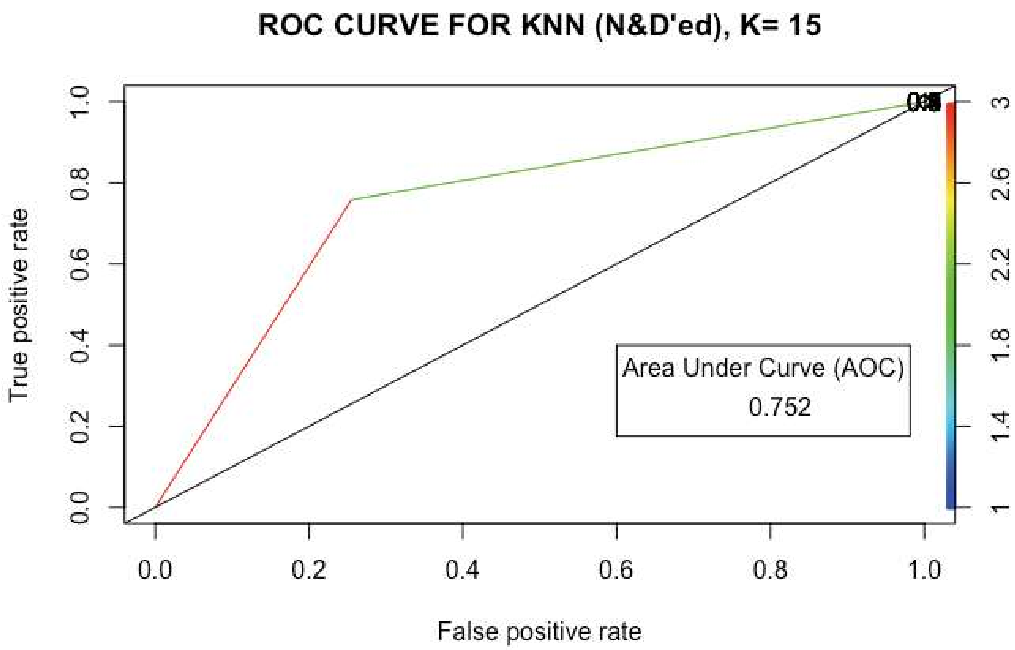
P-value is below 0.05.

Sensitivity (True Positive Rate) of the model is 77.65%.

Specificity (False Positive Rate) of the model is 76%.



For larger values of K, accuracy tend to drop again which may be again because of under-fit.



Conclusion

In this assignment, I have learnt fundamental workflow that is required to be followed when processing data. Doing principle component analysis to determine important features or using information gain metrics to understand dataset in depth. Also, now I'm aware of importance in splitting data to training and test, selecting proper formula for training which otherwise may cause over-fit or under-fit.

Moreover, I have gained insight on how to apply different classification methods to given data set and about those methods restrictions and assumptions. Such as, kNN works with numerical data only and it is utmost important to normalize features. Naive Bayes assumes features are independent, overfitting issue in decision trees.

Finally, I have learnt metrics that needs to be taken into account while evaluating classification model, such as accuracy, p-value, sensitivity, specificity.

In this dataset kNN outperformed other classification methods according to my findings.