Created and Submitted by Mr. Nilesh N. Kulkarni

Name of the Project: College Admission

Background and Objective: Every year thousands of applications are being submitted by international students for admission in colleges of the USA. It becomes an iterative task for the Education Department to know the total number of applications received and then compare that data with the total number of applications successfully accepted and visas processed. Hence to make the entire process easy, the education departments in the US analyze the factors that influence the admission of a student into colleges. The objective of this exercise is to analyze the same.

Domain: Education

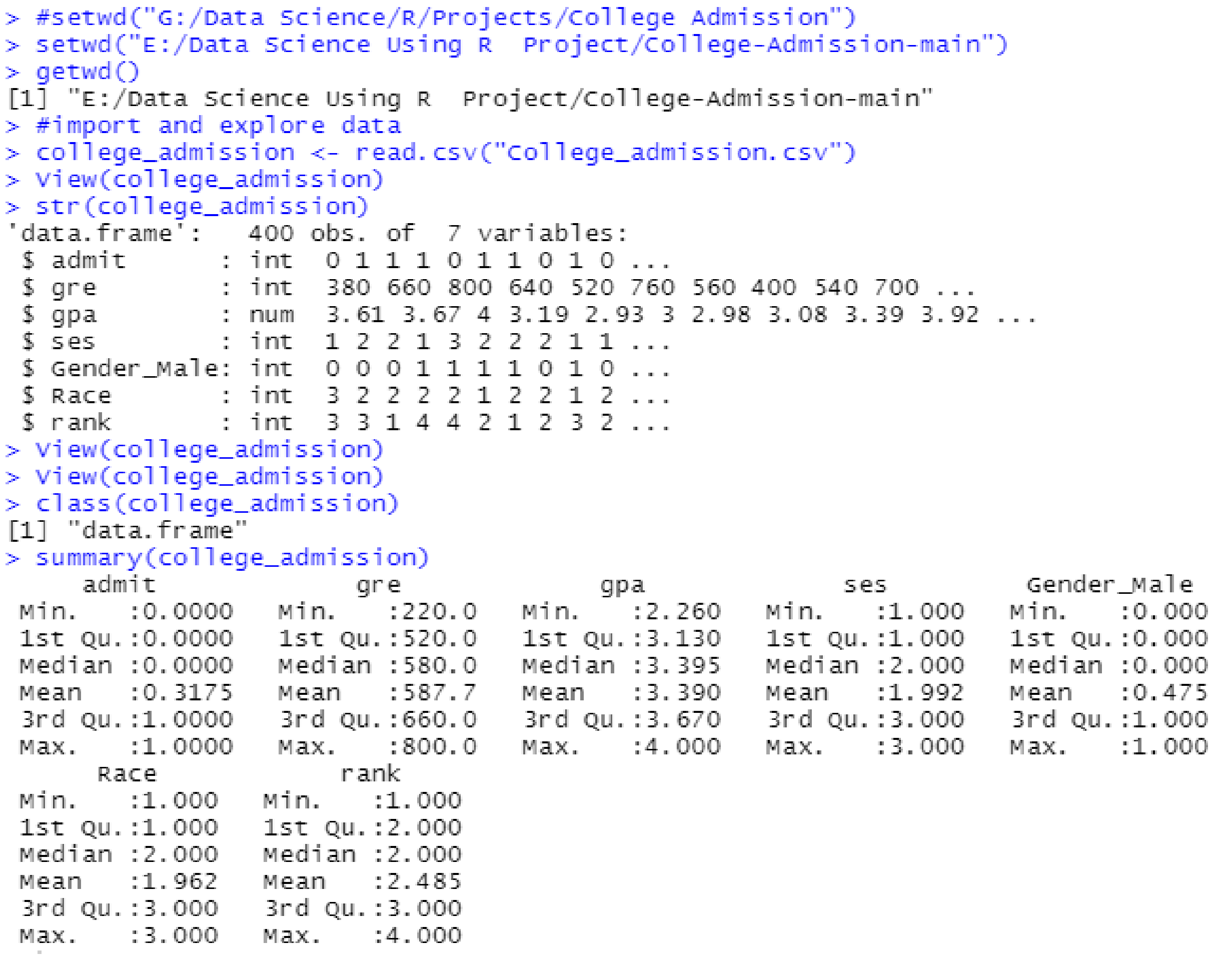
Dataset Description:

Attribute	**Description**
GRE	Graduate Record Exam Scores
GPA	Grade Point Average It refers to the prestige of the undergraduate institution.
Rank	The variable rank takes on the values 1 through 4. Institutions with a rank of 1 have the highest prestige, while those with a rank of 4 have the lowest.
Admit	It is a response variable; admit/don’t admit is a binary variable where 1 indicates that student is admitted and 0 indicates that student is not admitted.
SES	SES refers to socioeconomic status: 1 - low, 2 - medium, 3 - high.
Gender_male	Gender_male (0, 1) = 0 -> Female, 1 -> Male
Race	Race – 1, 2, and 3 represent Hispanic, Asian, and African-American

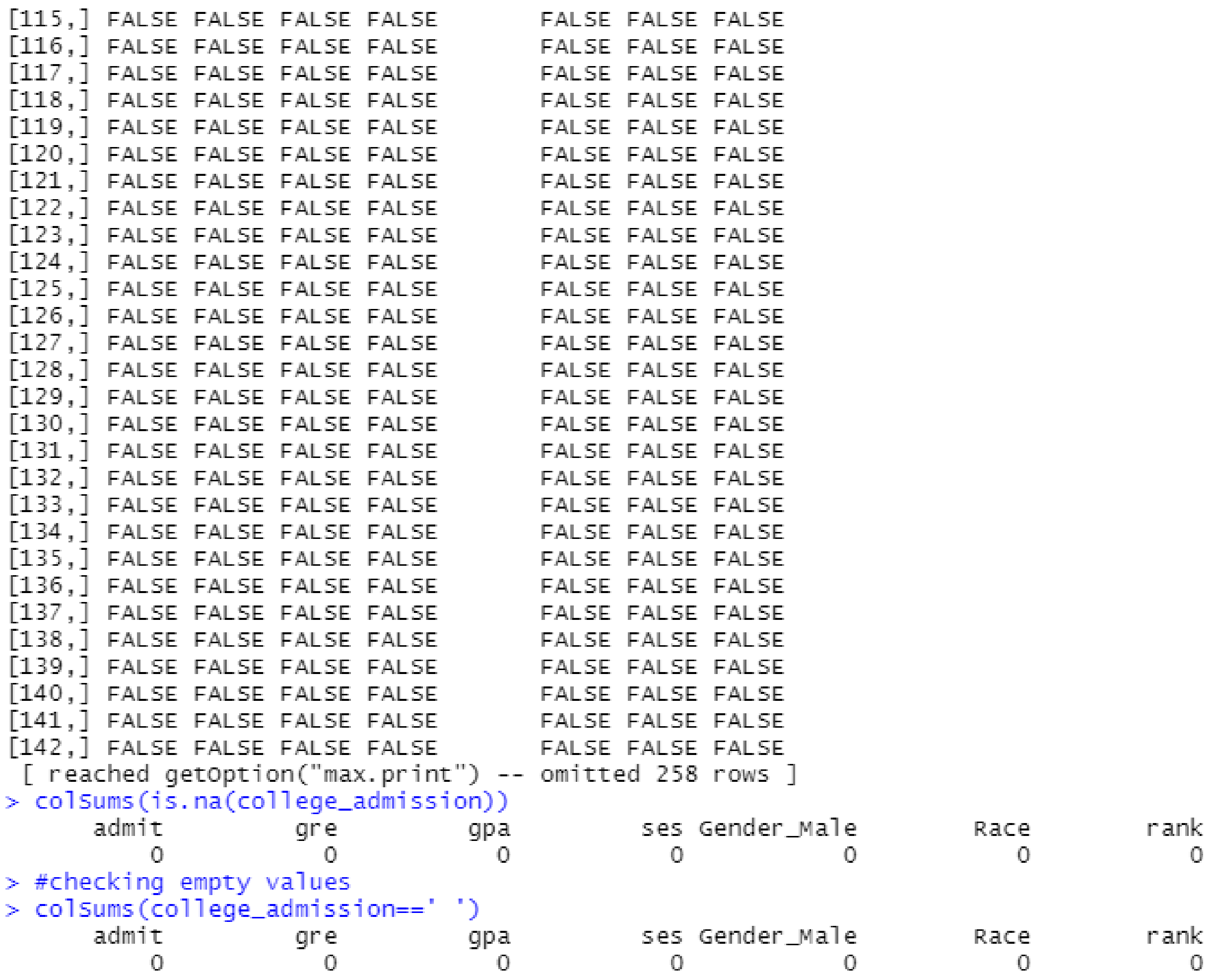
Analysis Tasks: Analyze the historical data and determine the key drivers for admission.

Predictive:

1) **Find the missing values. (if any, perform missing value treatment)**

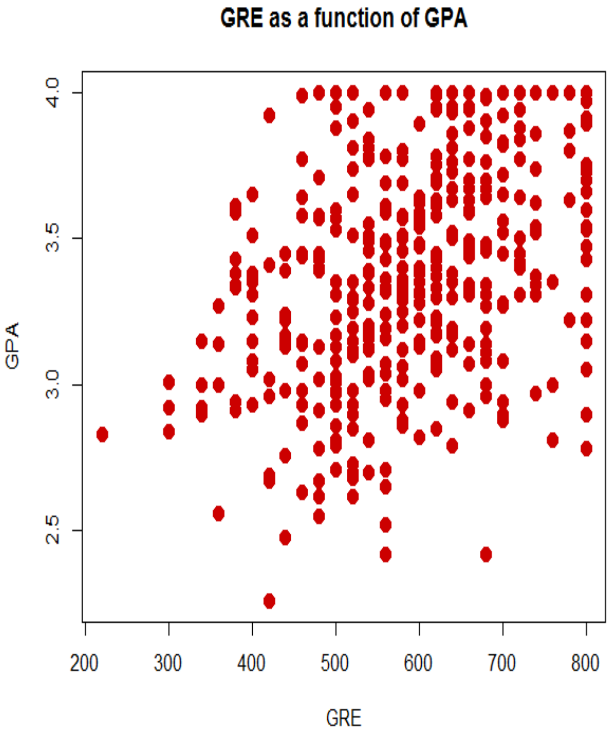


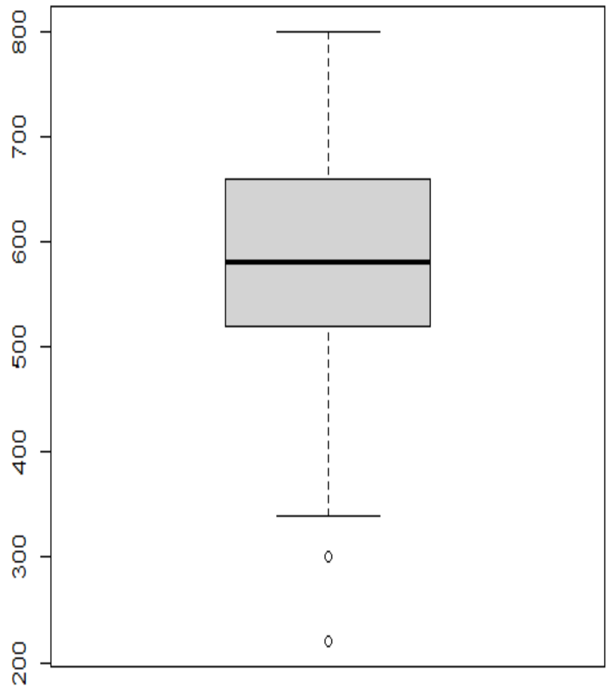
```
> #setwd("G:/Data Science/R/Projects/College Admission")
> setwd("E:/Data Science Using R  Project/College-Admission-main")
> getwd()
[1] "E:/Data Science Using R  Project/College-Admission-main"
> #import and explore data
> college_admission <- read.csv("College_admission.csv")
> View(college_admission)
> str(college_admission)
'data.frame':	400 obs. of  7 variables:
 $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
 $ Gender_Male: int  0 0 0 1 1 1 1 0 1 0 ...
 $ Race       : int  3 2 2 2 2 1 2 2 1 2 ...
 $ rank       : int  3 3 1 4 4 2 1 2 3 2 ...
> View(college_admission)
> View(college_admission)
> class(college_admission)
[1] "data.frame"
> summary(college_admission)
     admit             gre             gpa             ses         Gender_Male   
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000   Min.   :0.000  
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:1.000   1st Qu.:0.000  
 Median :0.0000   Median :580.0   Median :3.395   Median :2.000   Median :0.000  
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :1.992   Mean   :0.475  
 3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000   3rd Qu.:1.000  
 Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :3.000   Max.   :1.000  
      Race            rank      
 Min.   :1.000   Min.   :1.000  
 1st Qu.:1.000   1st Qu.:2.000  
 Median :2.000   Median :2.000  
 Mean   :1.962   Mean   :2.485  
 3rd Qu.:3.000   3rd Qu.:3.000  
 Max.   :3.000   Max.   :4.000  
```

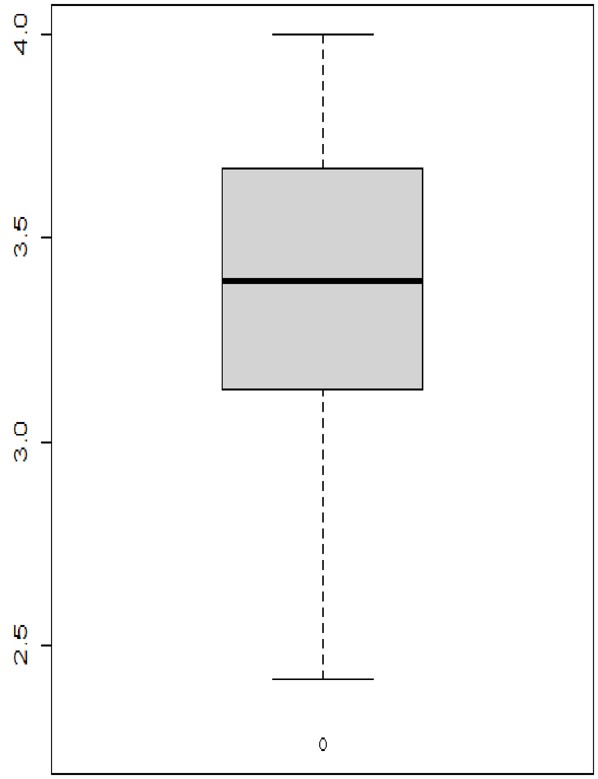


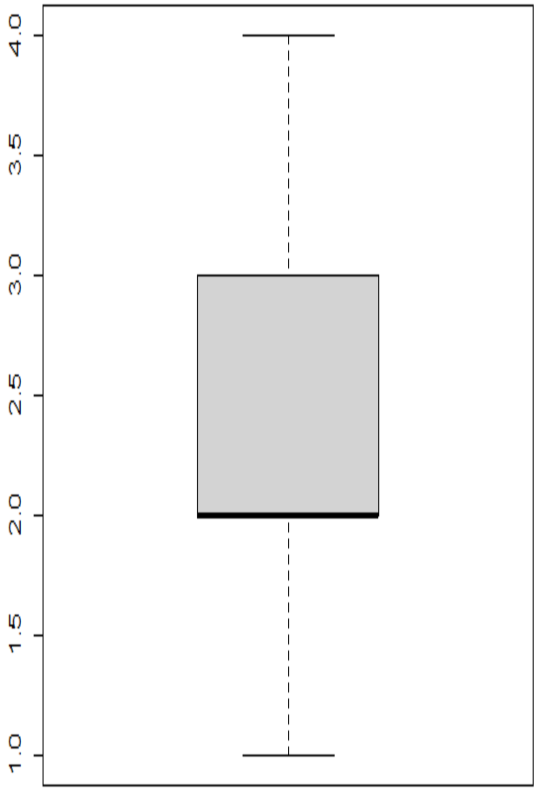
```
[115,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[116,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[117,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[118,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[119,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[120,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[121,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[122,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[123,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[124,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[125,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[126,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[127,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[128,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[129,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[130,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[131,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[132,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[133,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[134,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[135,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[136,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[137,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[138,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[139,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[140,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[141,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
[142,] FALSE FALSE FALSE FALSE       FALSE FALSE FALSE
 [ reached getOption("max.print") -- omitted 258 rows ]
> colSums(is.na(college_admission))
      admit         gre         gpa         ses Gender_Male        Race        rank 
          0           0           0           0           0           0           0 
> #checking empty values
> colSums(college_admission==' ')
      admit         gre         gpa         ses Gender_Male        Race        rank 
          0           0           0           0           0           0           0 
```

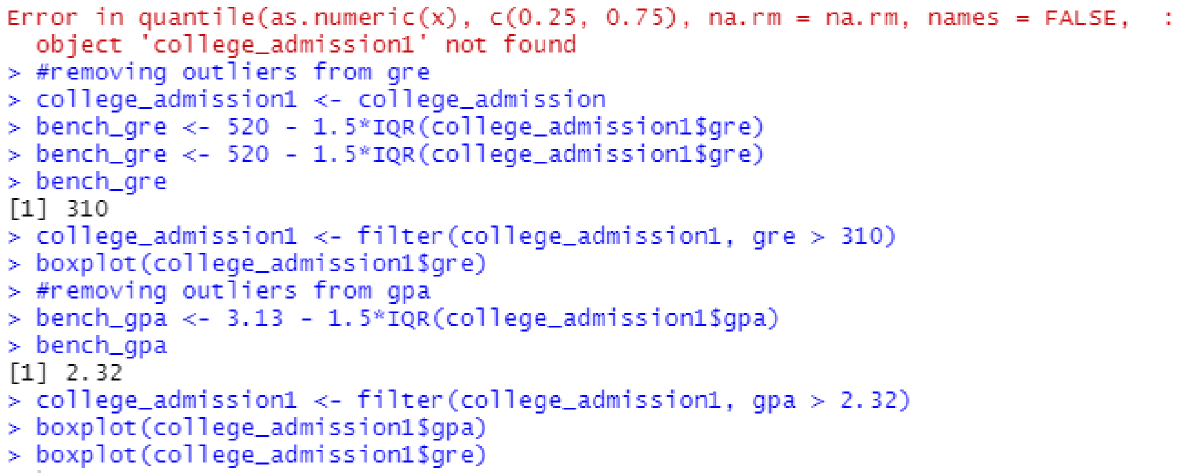
2) Find Outliers (if any, then perform outlier treatment)



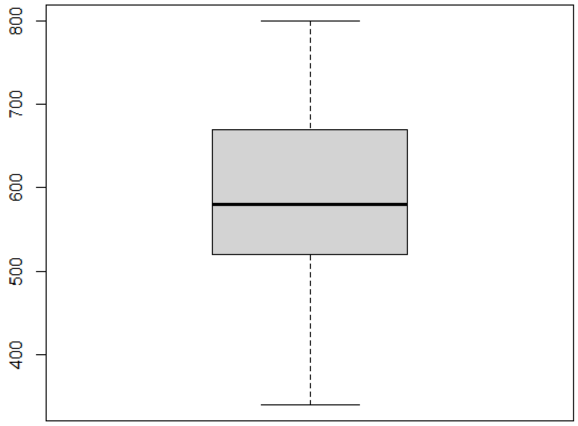


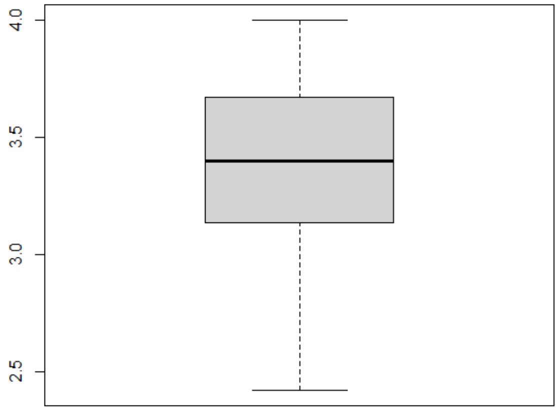


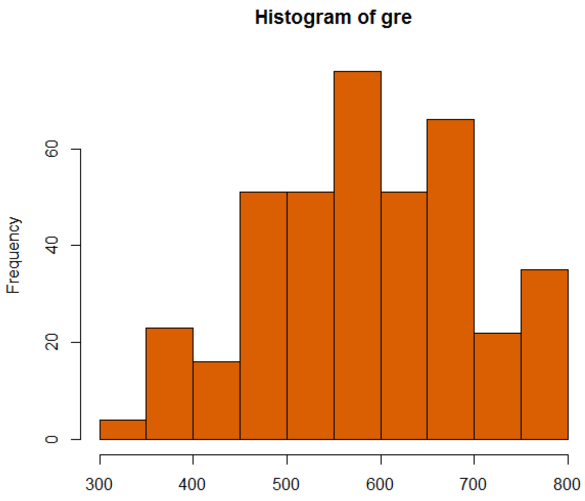


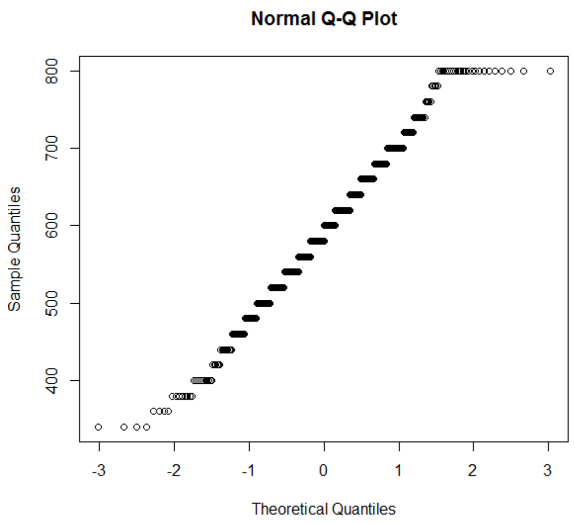


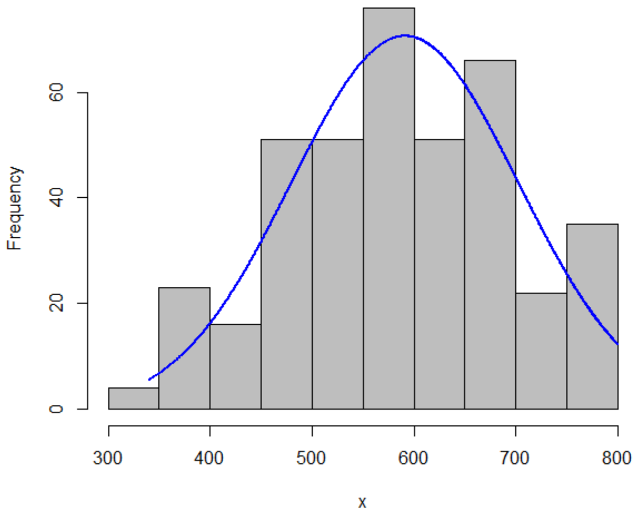
```
Error in quantile(as.numeric(x), c(0.25, 0.75), na.rm = na.rm, names = FALSE,  :
  object 'college_admission1' not found
> #removing outliers from gre
> college_admission1 <- college_admission
> bench_gre <- 520 - 1.5*IQR(college_admission1$gre)
> bench_gre <- 520 - 1.5*IQR(college_admission1$gre)
> bench_gre
[1] 310
> college_admission1 <- filter(college_admission1, gre > 310)
> boxplot(college_admission1$gre)
> #removing outliers from gpa
> bench_gpa <- 3.13 - 1.5*IQR(college_admission1$gpa)
> bench_gpa
[1] 2.32
> college_admission1 <- filter(college_admission1, gpa > 2.32)
> boxplot(college_admission1$gpa)
> boxplot(college_admission1$gre)
```





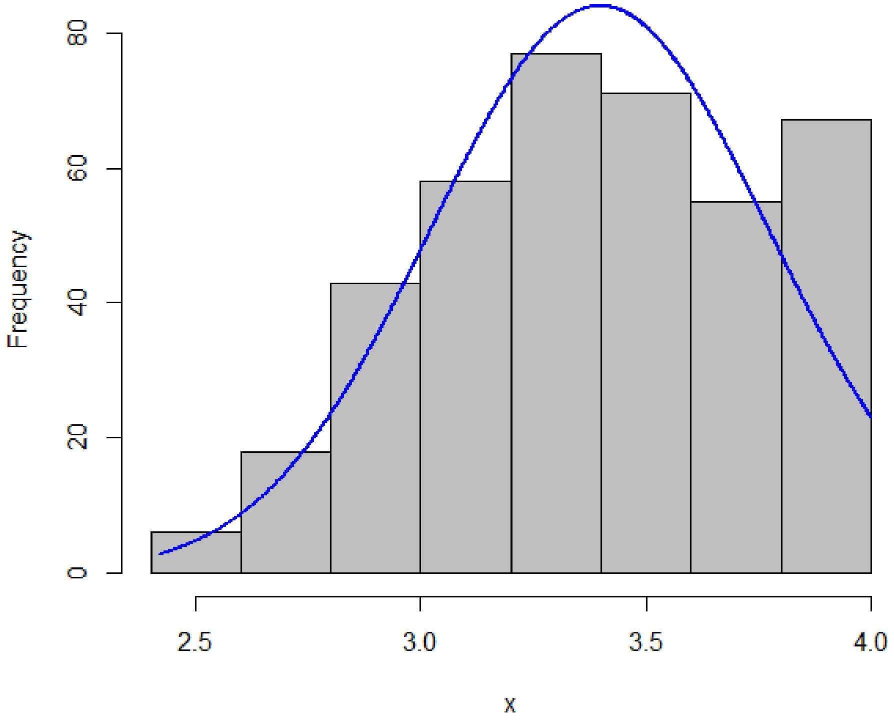


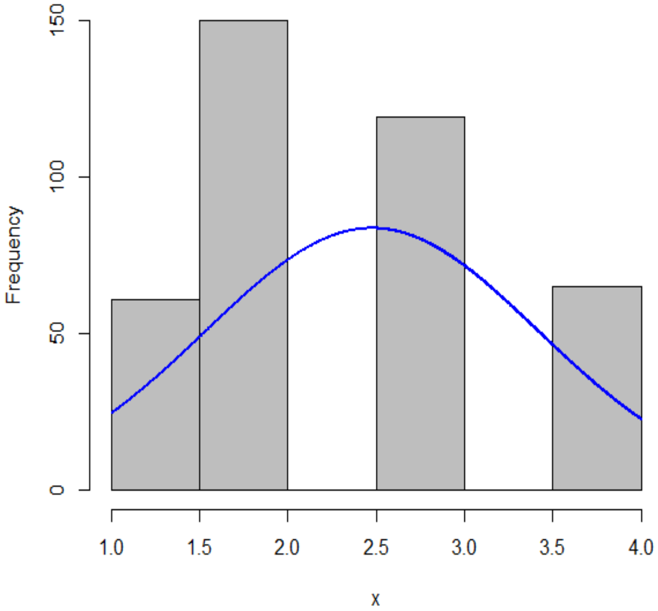




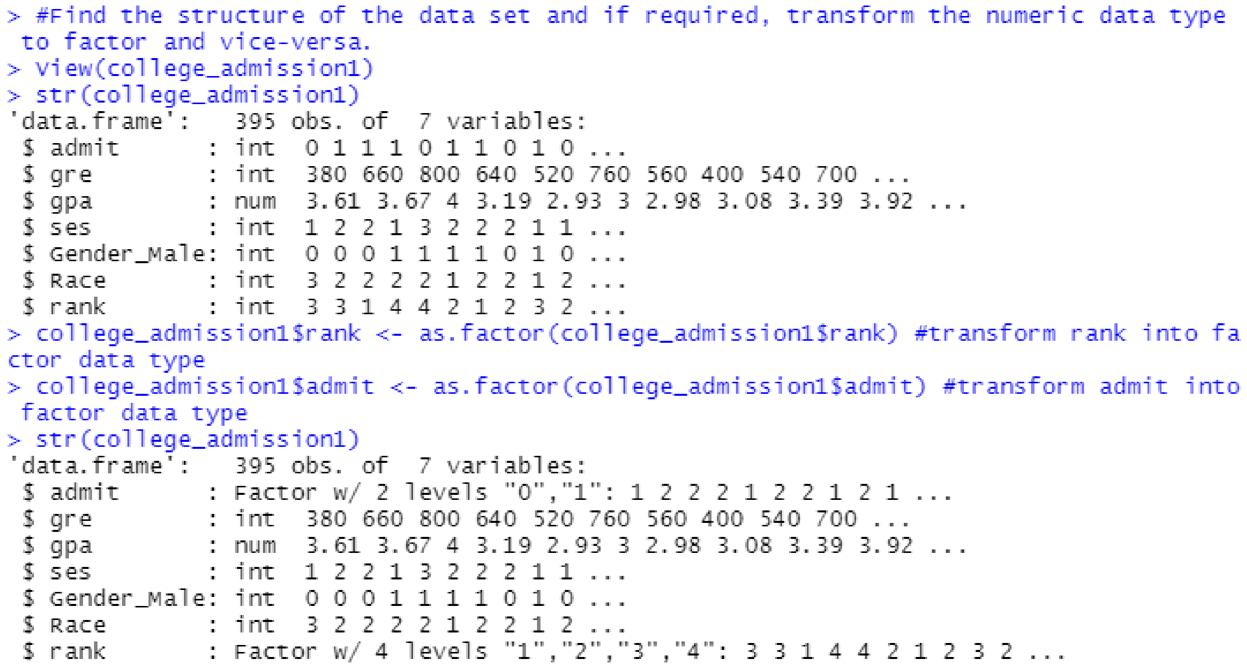
Comment: GRE data is normally distributed.

Commnent: GPA data is also normally distributed.





Comments: Rank is normally distributed

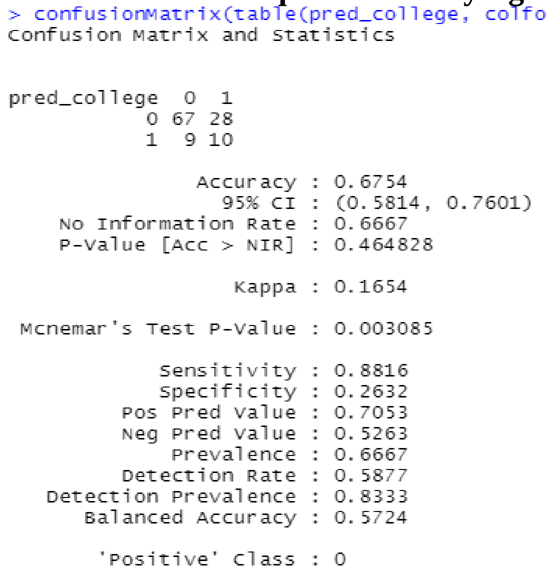


```
> #Find the structure of the data set and if required, transform the numeric data type
 to factor and vice-versa.
> View(college_admission1)
> str(college_admission1)
'data.frame':	395 obs. of  7 variables:
 $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
 $ Gender_Male: int  0 0 0 1 1 1 1 0 1 0 ...
 $ Race       : int  3 2 2 2 2 1 2 2 1 2 ...
 $ rank       : int  3 3 1 4 4 2 1 2 3 2 ...
> college_admission1$rank <- as.factor(college_admission1$rank) #transform rank into fa
ctor data type
> college_admission1$admit <- as.factor(college_admission1$admit) #transform admit into
 factor data type
> str(college_admission1)
'data.frame':	395 obs. of  7 variables:
 $ admit      : Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
 $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
 $ Gender_Male: int  0 0 0 1 1 1 1 0 1 0 ...
 $ Race       : int  3 2 2 2 2 1 2 2 1 2 ...
 $ rank       : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
```

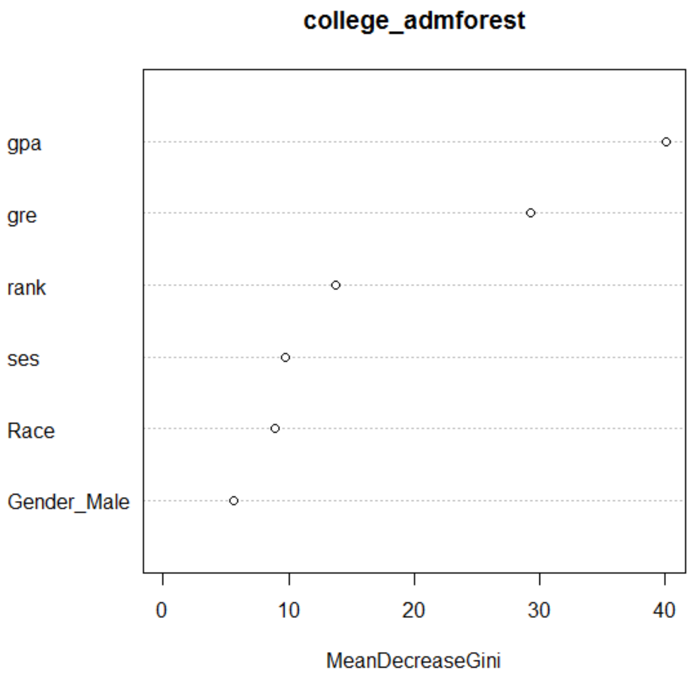
3) Normalize the data if not normally distributed.

All the data is normally distributed. The entire graph indicates the same.

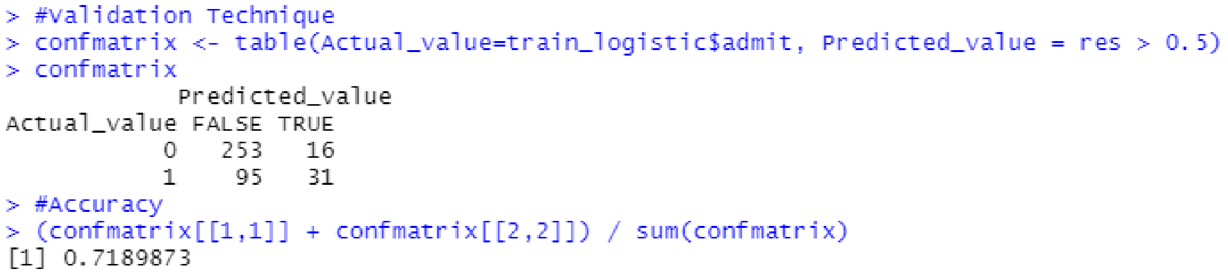
4) Use variable reduction techniques to identify significant variables.



```
> confusionMatrix(table(pred_college, colforest_test$admit))
Confusion Matrix and Statistics

            
pred_college  0  1
           0 67 28
           1  9 10
                                          
               Accuracy : 0.6754          
                 95% CI : (0.5814, 0.7601)
    No Information Rate : 0.6667          
    P-Value [Acc > NIR] : 0.464828        
                                          
                  Kappa : 0.1654          
                                          
 Mcnemar's Test P-Value : 0.003085        
                                          
            Sensitivity : 0.8816          
            Specificity : 0.2632          
         Pos Pred Value : 0.7053          
         Neg Pred Value : 0.5263          
             Prevalence : 0.6667          
         Detection Rate : 0.5877          
   Detection Prevalence : 0.8333          
      Balanced Accuracy : 0.5724          
                                          
       'Positive' Class : 0               
```

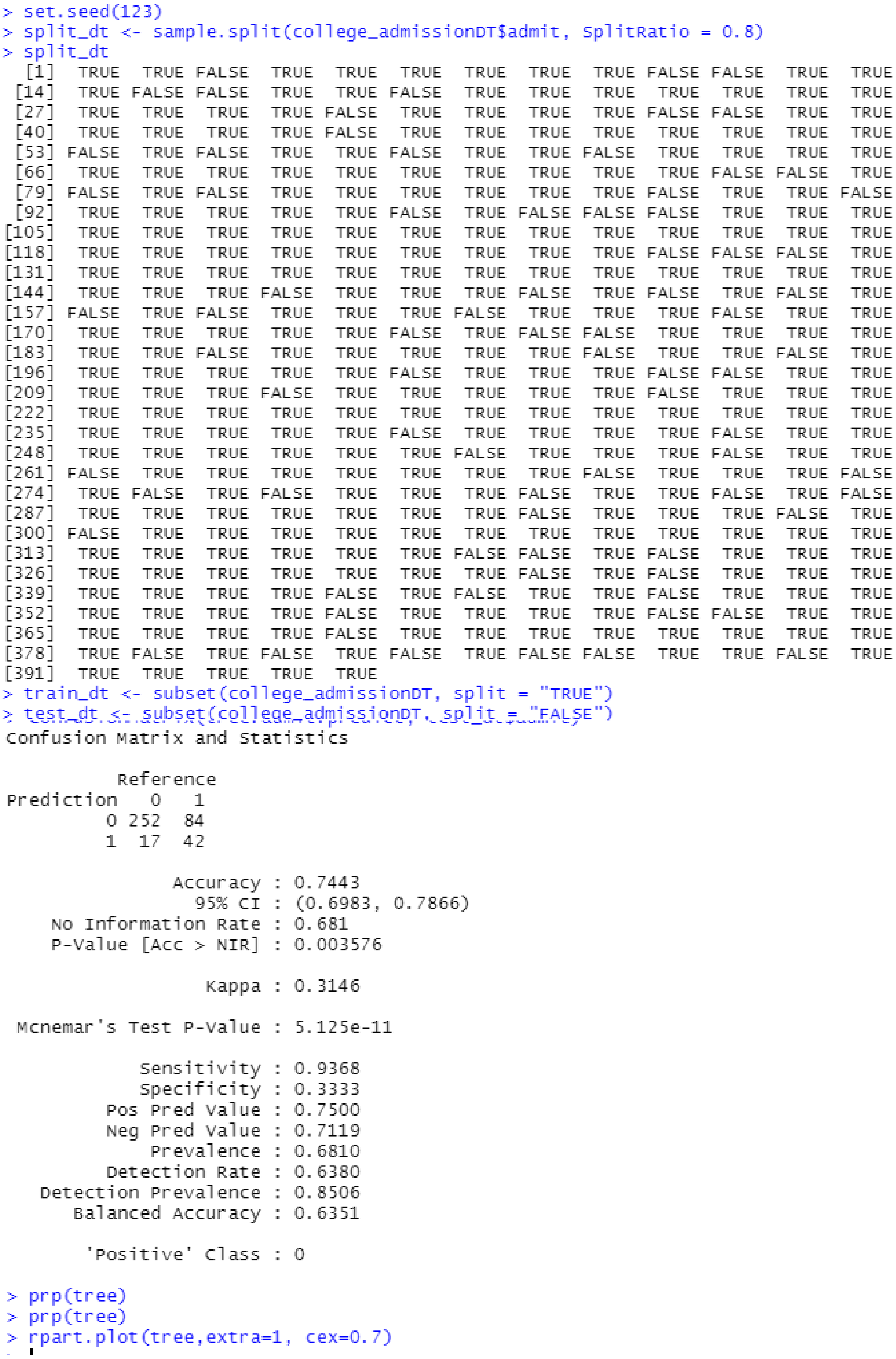


Comment: As per Random Forest method, GPA, GRE, and Rank are the significant variable with accuracy is 67.54%.

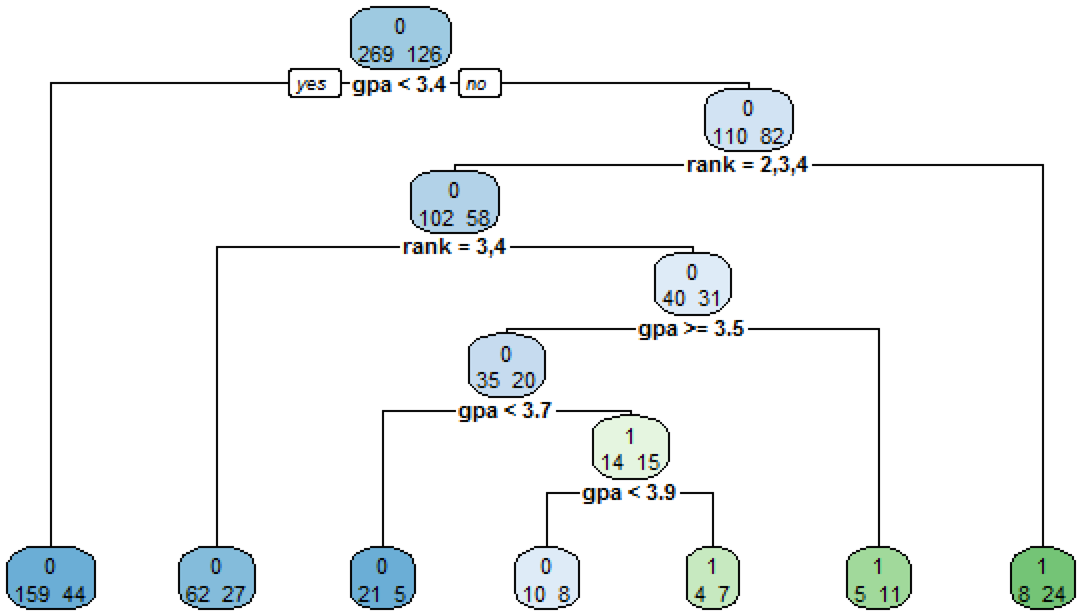


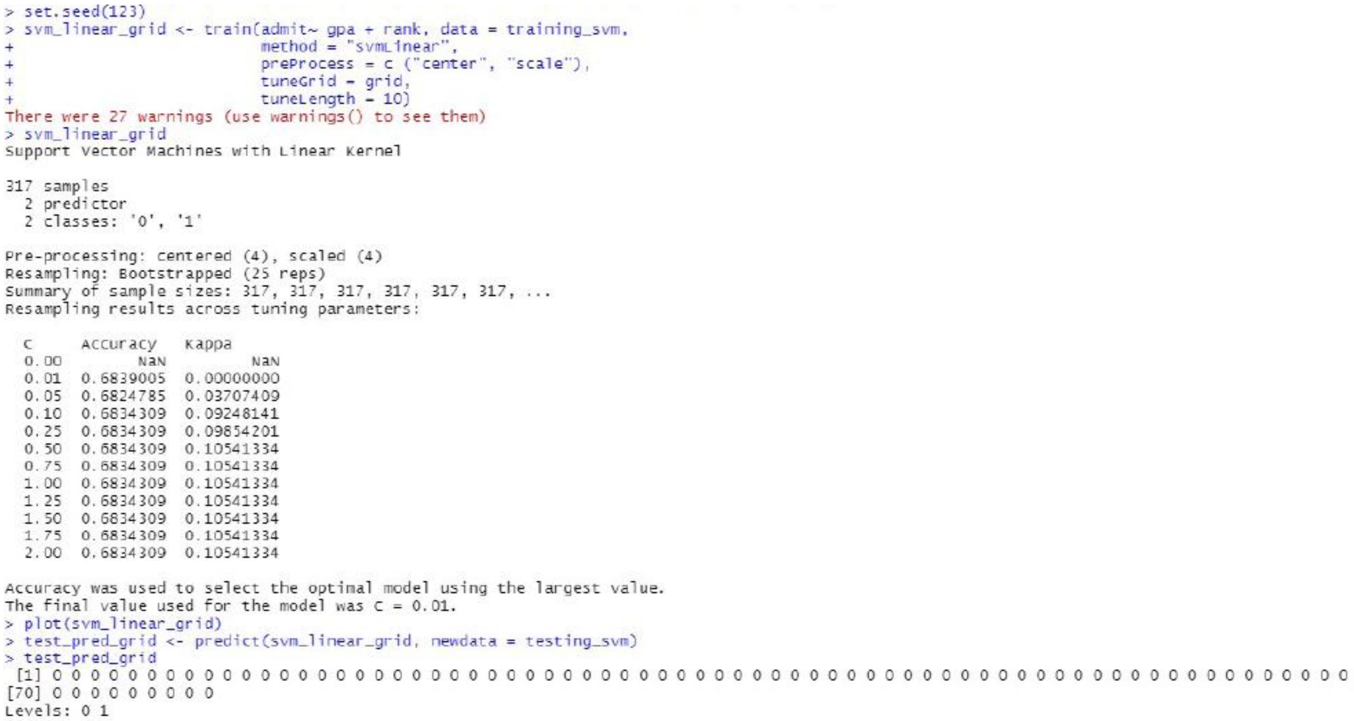
```
> #Validation Technique
> confmatrix <- table(Actual_value=train_logistic$admit, Predicted_value = res > 0.5)
> confmatrix
             Predicted_value
Actual_value FALSE TRUE
           0   253   16
           1    95   31
> #Accuracy
> (confmatrix[[1,1]] + confmatrix[[2,2]]) / sum(confmatrix)
[1] 0.7189873
```

Remark: From testing various models the best model with highest accuracy is college_model3 (gpa+rank) and it gives 71.89% accuracy**.**

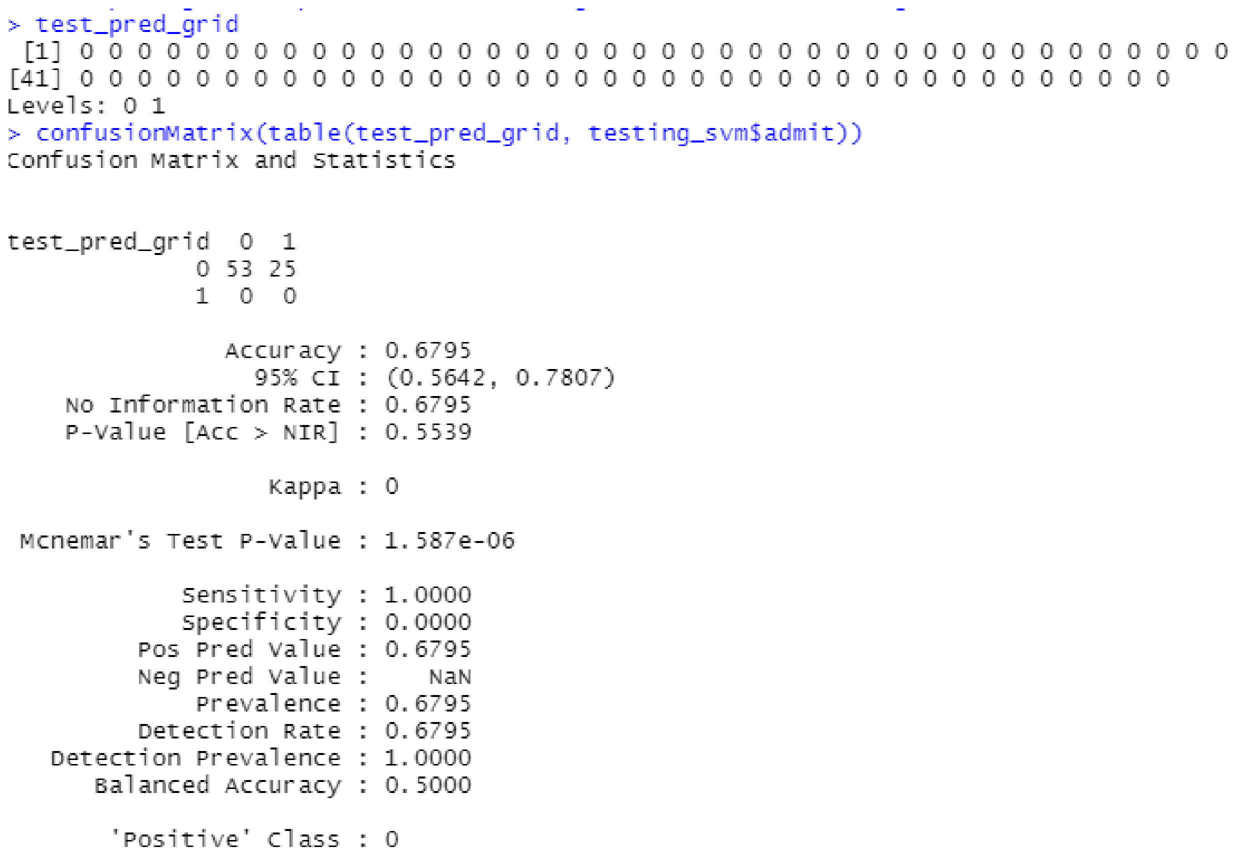


```
> set.seed(123)
> split_dt <- sample.split(college_admissionDT$admit, SplitRatio = 0.8)
> split_dt
  [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
 [14]  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [27]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
 [40]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [53] FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [66]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
 [79] FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
 [92]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE
[105]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[118]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
[131]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[144]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
[157] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
[170]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
[183]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
[196]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[209]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[222]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[235]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
[248]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
[261] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
[274]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE
[287]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
[300] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[313]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
[326]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
[339]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[352]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[365]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[378]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
[391]  TRUE  TRUE  TRUE  TRUE  TRUE
> train_dt <- subset(college_admissionDT, split = "TRUE")
> test_dt <- subset(college_admissionDT, split = "FALSE")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 252  84
         1  17  42

               Accuracy : 0.7443
                 95% CI : (0.6983, 0.7866)
    No Information Rate : 0.681
    P-Value [Acc > NIR] : 0.003576

                  Kappa : 0.3146

 Mcnemar's Test P-Value : 5.125e-11

            Sensitivity : 0.9368
            Specificity : 0.3333
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.7119
             Prevalence : 0.6810
         Detection Rate : 0.6380
   Detection Prevalence : 0.8506
      Balanced Accuracy : 0.6351

       'Positive' Class : 0

> prp(tree)
> prp(tree)
> rpart.plot(tree,extra=1, cex=0.7)
```





```
> set.seed(123)
> svm_linear_grid <- train(admit~ gpa + rank, data = training_svm,
+                           method = "svmLinear",
+                           preProcess = c ("center", "scale"),
+                           tuneGrid = grid,
+                           tuneLength = 10)
There were 27 warnings (use warnings() to see them)
> svm_linear_grid
Support Vector Machines with Linear Kernel

317 samples
  2 predictor
  2 classes: '0', '1'

Pre-processing: centered (4), scaled (4)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 317, 317, 317, 317, 317, 317, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.00        NaN         NaN
  0.01  0.6839005  0.00000000
  0.05  0.6824785  0.03707409
  0.10  0.6834309  0.09248141
  0.25  0.6834309  0.09854201
  0.50  0.6834309  0.10541334
  0.75  0.6834309  0.10541334
  1.00  0.6834309  0.10541334
  1.25  0.6834309  0.10541334
  1.50  0.6834309  0.10541334
  1.75  0.6834309  0.10541334
  2.00  0.6834309  0.10541334

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.01.
> plot(svm_linear_grid)
> test_pred_grid <- predict(svm_linear_grid, newdata = testing_svm)
> test_pred_grid
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[70] 0 0 0 0 0 0 0 0 0
Levels: 0 1
```



```
> test_pred_grid
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[41] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Levels: 0 1
> confusionMatrix(table(test_pred_grid, testing_svm$admit))
Confusion Matrix and Statistics


test_pred_grid  0  1
             0 53 25
             1  0  0

               Accuracy : 0.6795
                 95% CI : (0.5642, 0.7807)
    No Information Rate : 0.6795
    P-Value [Acc > NIR] : 0.5539

                  Kappa : 0

 Mcnemar's Test P-Value : 1.587e-06

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.6795
         Neg Pred Value :    NaN
             Prevalence : 0.6795
         Detection Rate : 0.6795
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

5) Select the best model.

Classifier	Accuracy
Decision Tree	74.44%
Random Forest	67.54%
Support Vector Machine	67.95%
Logistic Regression	71.9%

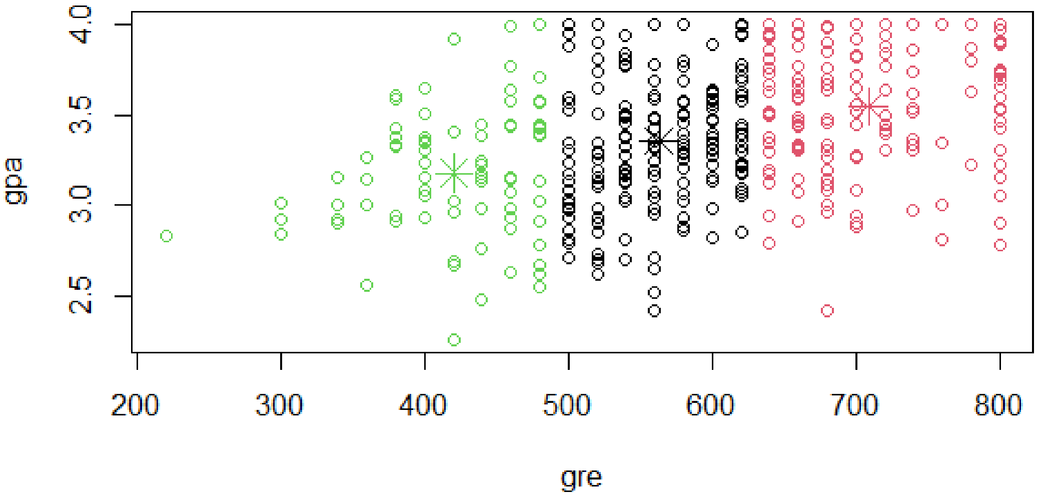
Remark: From above table, it can be seen that Decision Tree classifier gives good classification accuracy.

6) Identify other Machine learning or statistical techniques

Other supervised classifiers such as Naïve Bayes classifier, K nearest Neighbor (KNN) classifier can be used.

Descriptive Statistics

Categorize the average of grade point into High, Medium, and Low (with admission probability percentages) and plot it on a point chart. Cross grid for admission variables with GRE Categorization is shown below:


gpa
2.5
3.0
3.5
4.0
200
300
400
500
600
700
800
gre