



# College Admissions

*Business Analytic Foundation with R Tools- Question*

## Abstract

The education department in the US wants to analyze the factors that influence the admission of a student into colleges, to make the entire admissions process easy.

**Presented by: Ankita Agarwal**

## Problem Statement:

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 department 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

## Detailed description of the given dataset:

Attribute	Description
GRE	Graduate Record Exam Scores
GPA	Grade Point Average
Rank	It refers to the prestige of the undergraduate institution. 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

## To Analyze:

Analyze the historical data and determine the key drivers for admission.

Predictive:

- Find the missing values. (if any, perform missing value treatment)
- Find outliers (if any, then perform outlier treatment)
- Find the structure of the data set and if required, transform the numeric data type to factor and vice-versa.
- Find whether the data is normally distributed or not. Use the plot to determine the same.
- Normalize the data if not normally distributed.
- Use variable reduction techniques to identify significant variables.
- Run logistic model to determine the factors that influence the admission process of a student (Drop insignificant variables)
- Calculate the accuracy of the model and run validation techniques.
- Try other modelling techniques like decision tree and SVM and select a champion model
- Determine the accuracy rates for each kind of model
- Select the most accurate model
- Identify other Machine learning or statistical techniques

Descriptive:

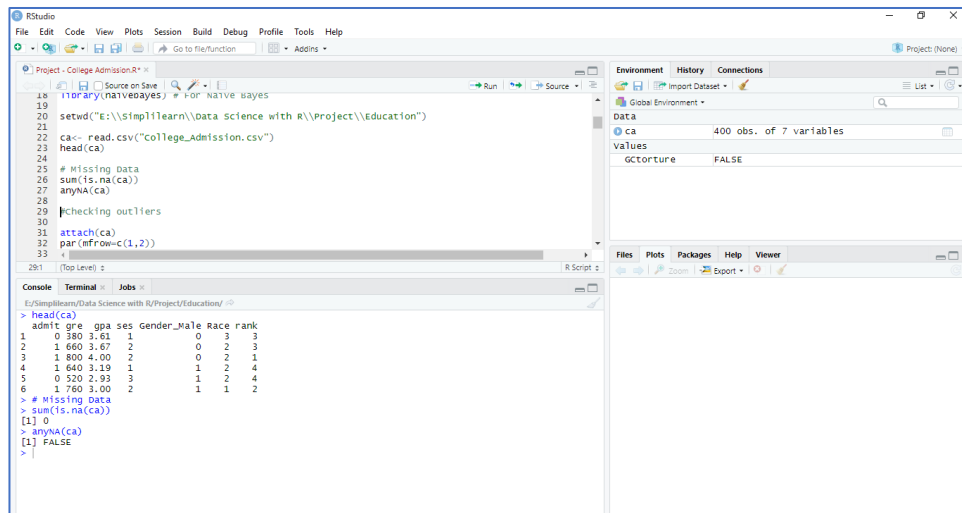
- Categorize the average of grade point into High, Medium, and Low (with admission probability percentages) and plot it on a point chart.
- Cross grid for admission variables with GRE Categorization is shown below:

<b>GRE</b>	<b>Categorized</b>
0-440	Low
440-580	Medium
580+	High

# Analysis and Interpretations:

## Predictive

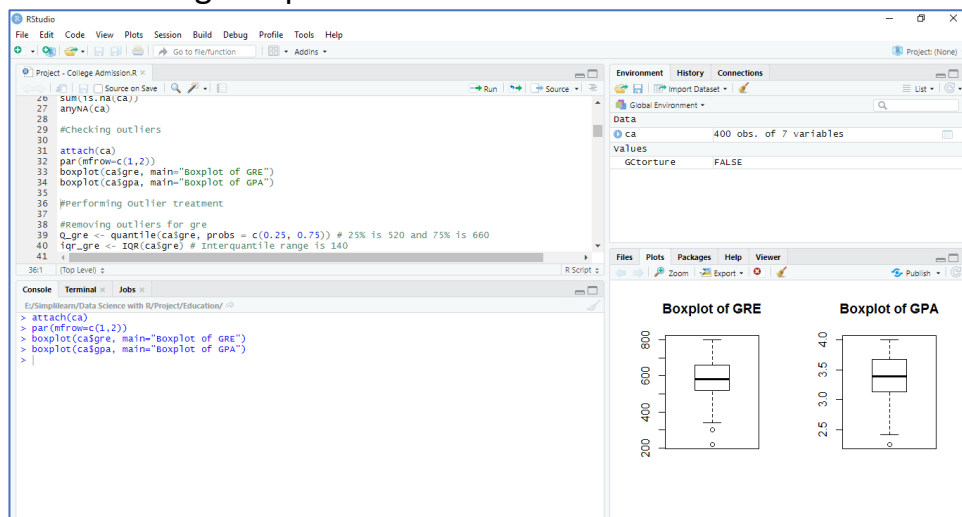
### 1. Find the missing values. (if any, perform missing value treatment)



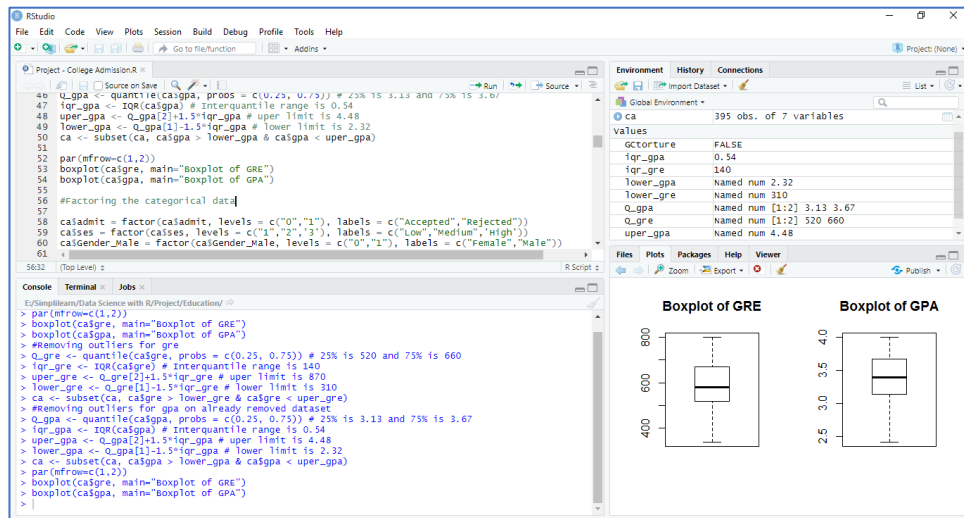
There are no missing values in the given dataset.

### 2. Find outliers (if any, then perform outlier treatment).

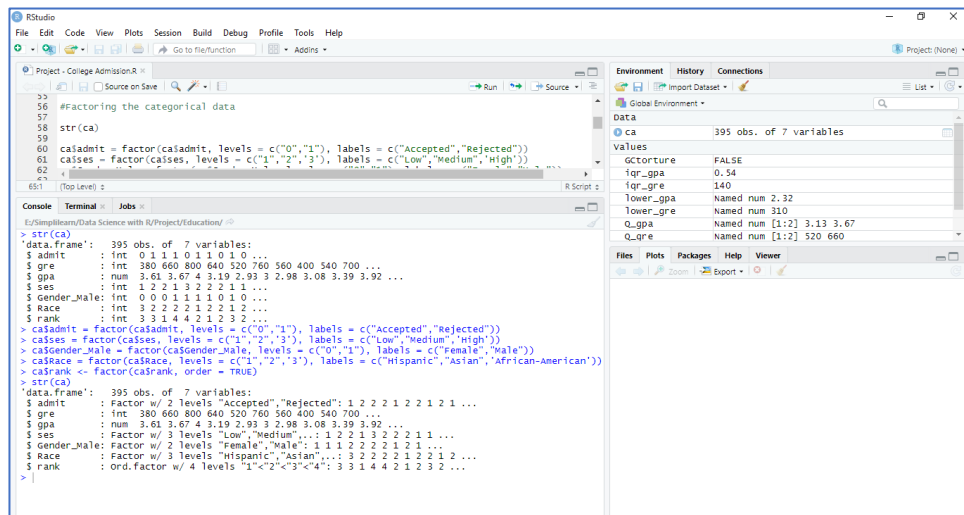
Checked outliers using box plots.



There are outliers in both GRE and GPA. To remove the outliers, we use the equation:  $(Q1 - 1.5 * IQR)$  to  $(Q2 + 1.5 * IQR)$

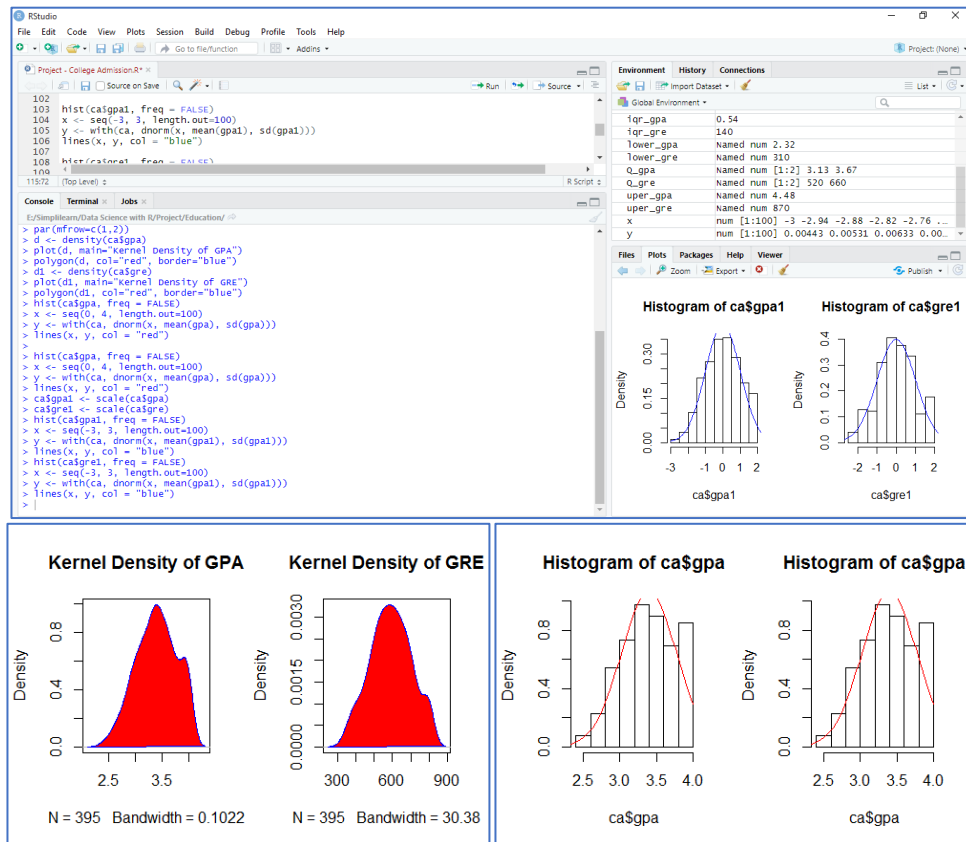


### 3. Find the structure of the data set and if required, transform the numeric data type to factor and vice-versa.



### 4. Find whether the data is normally distributed or not. Use the plot to determine the same.

We use density plots and histograms to check for normality of data.

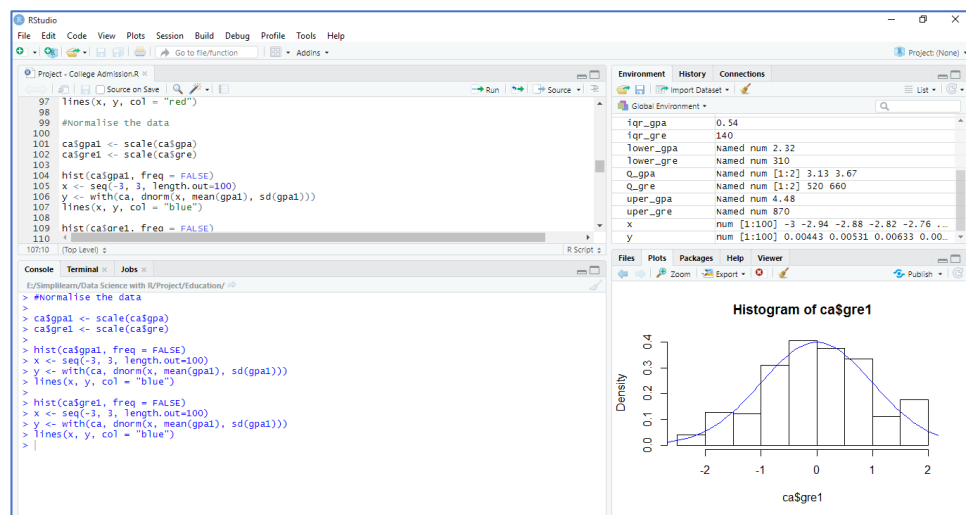


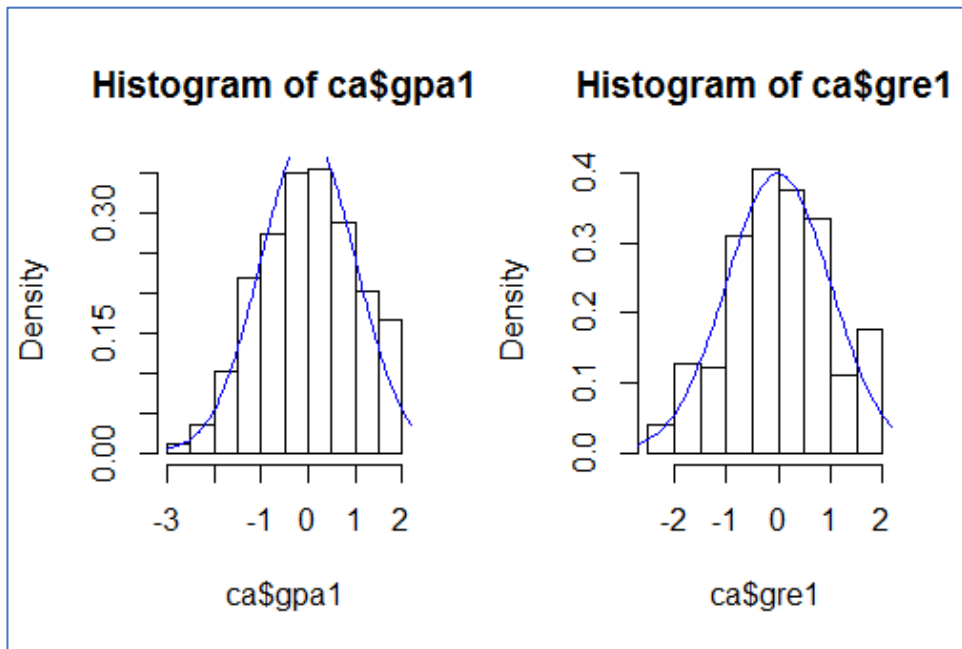
## Interpretation:

GRE data is not normally distributed: mean (591.2) > median (580.0), so right skewness, GPA also not normally distributed: median (3.4) > mean (3.398), so left skewness.

## 5. Normalize the data if not normally distributed.

Normalized using Scale function





## 6. Use variable reduction techniques to identify significant variables.

Using logistic regression to check significant variables. Also, checking the same using Anova.

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
> model <- glm(admit ~ ., family = binomial(link = 'logit'), data = ca)
> summary(model)

Call:
glm(formula = admit ~ ., family = binomial(link = "logit"), data = ca)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6898   -0.8728   -0.6207    1.0848    2.2082

Coefficients: (2 not defined because of singularities)
(Intercept)   -2.786e+00  1.792e+00  -1.555   0.1200
gre            2.950e-06  2.153e-03   0.001   0.9989
gpa            8.003e-01  3.385e-01  2.364   0.0181
sesmedium     -1.679e-01  2.796e-01  -0.601   0.5482
seshigh       -3.036e-01  2.882e-01  -1.053   0.2922
gender_male    -2.112e-01  2.325e-01  -0.908   0.3637
raceasian     -4.596e-01  2.843e-01  -1.617   0.1060
raceafrican-american -2.610e-01  2.786e-01  -0.566   0.5690
rank_l        -1.186e+00  2.918e-01  -4.065   4.81e-05 ***
rank_q         2.956e-01  2.578e-01  1.147   0.2515
rank_c         6.786e-02  2.151e-01   0.115   0.7534
gre_category_low -1.251e+00  7.956e-01  -1.574   0.1154
gre_category_medium -3.702e-01  4.106e-01  -0.902   0.3672
gpa1           NA         NA         NA      NA
gre1           NA         NA         NA      NA

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    null deviance: 494.62 on 394 degrees of freedom
Residual deviance: 446.68 on 382 degrees of freedom
AIC: 472.68

Number of Fisher scoring iterations: 4

```

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
#variable reduction techniques to identify significant variables
model <- glm(admit ~ ., family = binomial(link = 'logit'), data = ca)
summary(model)
anova(model, test = 'chisq')

# Logistic regression model with significance independent variable
splitindex <- createDataPartition(ca$admit, p = .70, list = FALSE, times = 1)
train <- ca[splitindex,]
test <- ca[-splitindex,]

R Console
Model: binomial, link: logit
Response: admit
Terms added sequentially (first to last)

Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL 13.7962 394 494.62
gre 1 2.7442 392 475.08 0.002037 ***
gpa 1 1.2922 390 473.79 0.5240769
ses 1 0.3503 389 473.44 0.5539623
gender_male 1 3.0213 387 470.42 0.2207621
race 1 21.0432 384 449.37 0.0001031 ***
gre_category 2 2.6953 382 446.68 0.2398513
gpa1 0 0.0000 382 446.68
gre1 0 0.0000 382 446.68

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

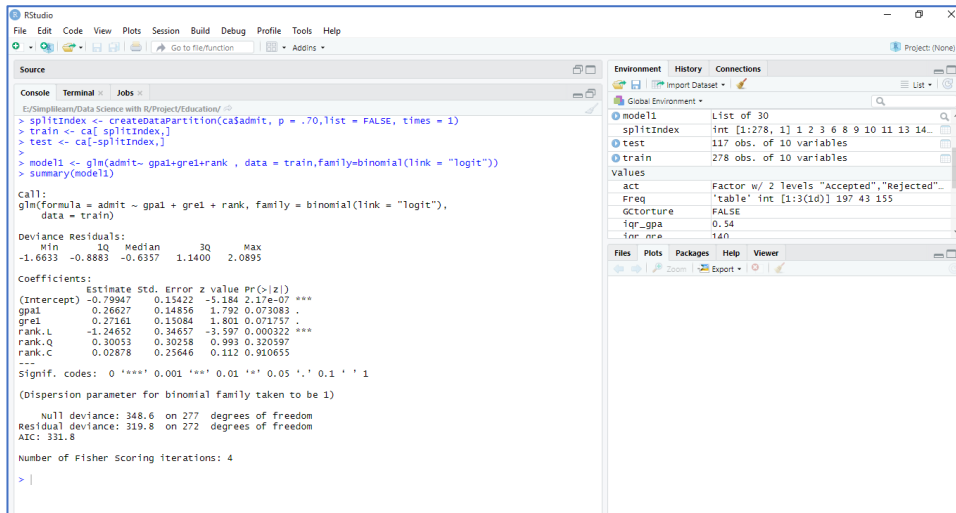
```

## Interpretation:

As can be seen clearly, using both logistic regression and Anova, that GRE, GPA and Rank are the only significant variables in the dataset.

## 7. Run logistic model to determine the factors that influence the admission process of a student (Drop insignificant variables)

As we already know that GRE, GPA and Rank are the only significant variables in the dataset. We would be using only these for our analysis. We will split the data set into train and test to predict the outcomes and check for accuracy



```
Source
File Edit Code View Plots Session Build Debug Profile Tools Help
> splitIndex <- createDataPartition(ca$admit, p = .70, list = FALSE, times = 1)
> train <- ca[splitIndex,]
> test <- ca[-splitIndex,]
> model1 <- glm(admit~ gpa+gre+rank , data = train,family=binomial(link = "logit"))
> summary(model1)

Call:
glm(formula = admit ~ gpa + gre + rank, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6633   -0.8883   -0.6357   1.1400   2.0895

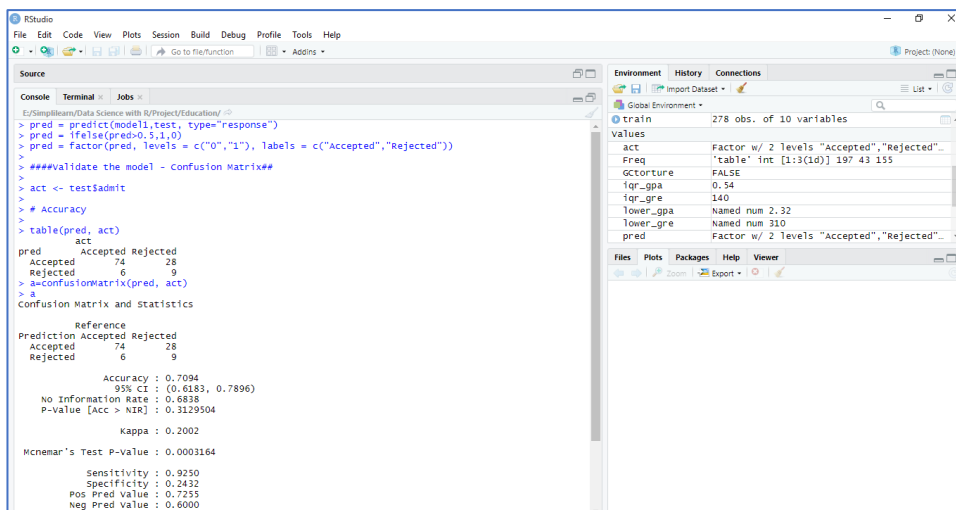
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.79947    0.15422  -5.184 2.17e-07 ***
gpa         0.26627    0.14856   1.792 0.073083 .
gre         0.27161    0.15084   1.801 0.071757 .
rank.L      1.24652    0.34657   3.597 0.000322 ***
rank.Q      0.30053    0.30258   0.993 0.320597
rank.C      0.02878    0.25646   0.112 0.910655
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 348.6 on 277 degrees of freedom
Residual deviance: 319.8 on 272 degrees of freedom
AIC: 331.8

Number of Fisher Scoring iterations: 4
>
```

## 8. Calculate the accuracy of the model and run validation techniques.



```
Source
File Edit Code View Plots Session Build Debug Profile Tools Help
> pred = predict(model1, test, type="response")
> pred = ifelse(pred>0.5,1,0)
> pred = factor(pred, levels = c("0","1"), labels = c("Accepted","Rejected"))
>
> ###validate the model - confusion matrix##
> act <- test$admit
>
> # Accuracy
> table(pred, act)
      act
pred   Accepted Rejected
Accepted  74      28
Rejected   6       9
> a=confusionMatrix(pred, act)
> a
Confusion Matrix and Statistics

              Reference
Prediction Accepted Rejected
Accepted      74      28
Rejected       6       9

Accuracy : 0.7094
95% CI : (0.6183, 0.7896)
No Information Rate : 0.6838
P-value [ACC > NIR] : 0.3129504
Kappa : 0.2002
McNemar's Test P-Value : 0.0003164

Sensitivity : 0.9250
Specificity : 0.2432
Pos Pred Value : 0.7255
Neg Pred Value : 0.6000
Prevalence : 0.6838
```

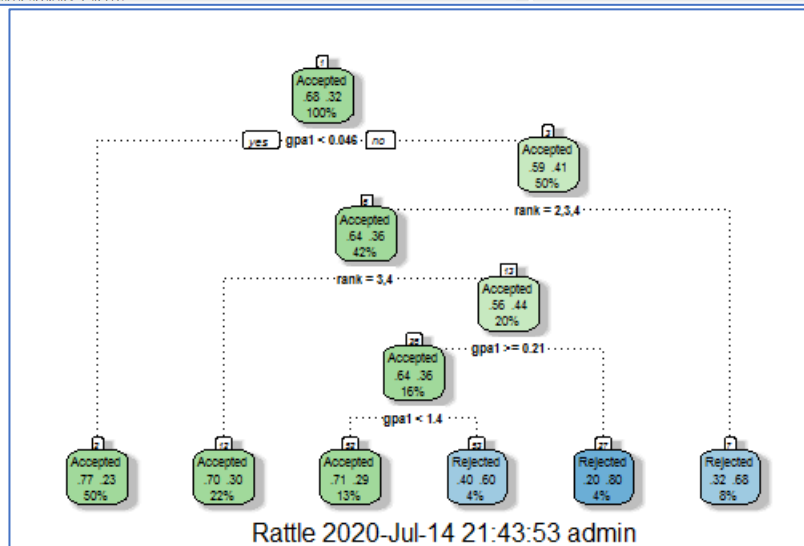
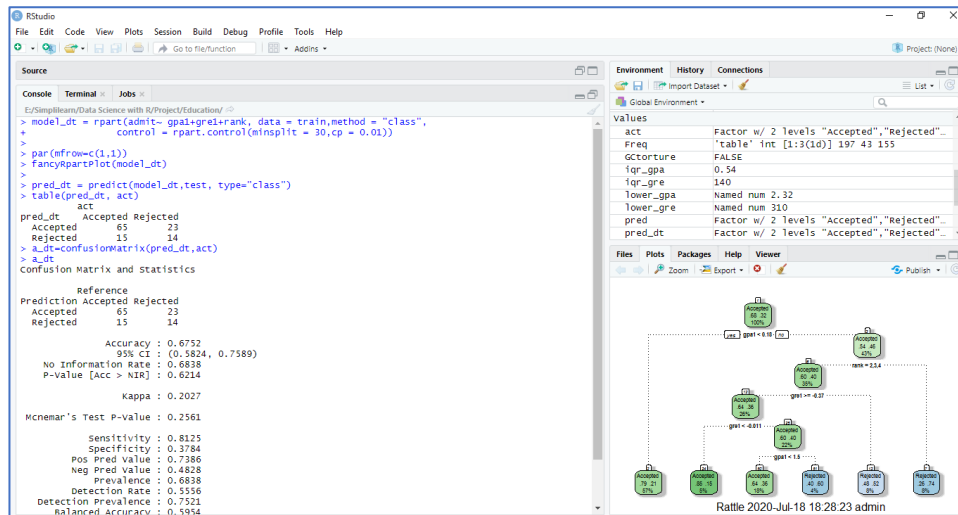
## Interpretation:

As noted, the accuracy of the model is 70.94%



## 9. Try other modelling techniques like decision tree and SVM and select a champion model. Identify other Machine learning or statistical techniques.

### Decision Tree: -



### Interpretation:

As noted, the accuracy of the model is 67.52%

### SVM: -

```

Source
E:/SimpleLearn/Data Science with R/Project/Education/ <R>
> svmfit.svm(admit~ gpa+gre+rank, data = train, kernel="linear",
+ scale = 7)
> pred_svm = predict(svmfit,test, type="response")
> table(pred_svm, act)
pred_svm      Accepted Rejected
Accepted      74         28
Rejected       6          9
> a_svm=confusionMatrix(pred_svm,act)
> a_svm
Confusion Matrix and Statistics

              Reference
Prediction Accepted Rejected
Accepted         74         28
Rejected          6          9

      Accuracy : 0.7094
      95% CI   : (0.6183, 0.7896)
    No Information Rate : 0.6838
    P-value [Acc > NIR] : 0.3129504

      Kappa : 0.2002

  Mcnemar's Test P-value : 0.0003164

      Sensitivity : 0.9250
      Specificity : 0.2432
    Pos Pred Value : 0.7255
    Neg Pred Value : 0.6000
      Prevalence : 0.6838
    Detection Rate : 0.6325
    Detection Prevalence : 0.8718
    Balanced Accuracy : 0.5841

'Positive' Class : Accepted

```

## Interpretation:

As noted, the accuracy of the model is 70.94%

## Random Forest: -

```

Source
E:/SimpleLearn/Data Science with R/Project/Education/ <R>
> fit_rf = randomforest(admit~ gpa+gre+rank, data = train, do.trace=F)
> pred_rf = predict(fit_rf,test)
> table(pred_rf, act)
pred_rf      Accepted Rejected
Accepted      76         23
Rejected       4         14
> a_rf=confusionMatrix(pred_rf,act)
> a_rf
Confusion Matrix and Statistics

              Reference
Prediction Accepted Rejected
Accepted         76         23
Rejected          4         14

      Accuracy : 0.7692
      95% CI   : (0.6823, 0.8421)
    No Information Rate : 0.6838
    P-value [Acc > NIR] : 0.027021

      Kappa : 0.381

  Mcnemar's Test P-value : 0.000532

      sensitivity : 0.9500
      Specificity : 0.3784
    Pos Pred Value : 0.7677
    Neg Pred Value : 0.7778
      Prevalence : 0.6838
    Detection Rate : 0.6496
    Detection Prevalence : 0.8462
    Balanced Accuracy : 0.6642

'Positive' Class : Accepted

```

## Interpretation:

As noted, the accuracy of the model is 76.92%

## Naïve Bayes: -

```

Source
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
E:/SimpleLearn/Data Science with R/Project/Education/ <
> fit_nb = naive_bayes(admit~ gpa+gre+rank, data = train)
> pred_nb = predict(fit_nb, test)
> table(pred_nb, act)
pred_nb      act
Accepted      76      23
Rejected       4      14
> a_nb=confusionMatrix(pred_nb,act)
Confusion Matrix and Statistics

              Reference
Prediction Accepted Rejected
Accepted          76         23
Rejected           4         14

    Accuracy : 0.7692
    95% CI   : (0.6823, 0.8421)
  No Information Rate : 0.6838
    P-value [acc > NIR] : 0.027021

    Kappa : 0.381

  McNemar's Test P-Value : 0.000532

    Sensitivity : 0.9500
    Specificity : 0.3784
    Pos Pred Value : 0.7677
    Neg Pred Value : 0.7778
    Prevalence : 0.6838
    Detection Rate : 0.6496
    Detection Prevalence : 0.8462
    Balanced Accuracy : 0.6642

    'Positive' class : Accepted
  > |
  
```

## Interpretation:

As noted, the accuracy of the model is 76.92%

## 10. Determine the accuracy rates for each kind of model. Select the most accurate model.

Model	Accuracy
Logistic Regression	70.94%
Decision Tree	67.52%
SVM	70.94%
Random Forest	76.92%
Naïve Bayes	76.92%

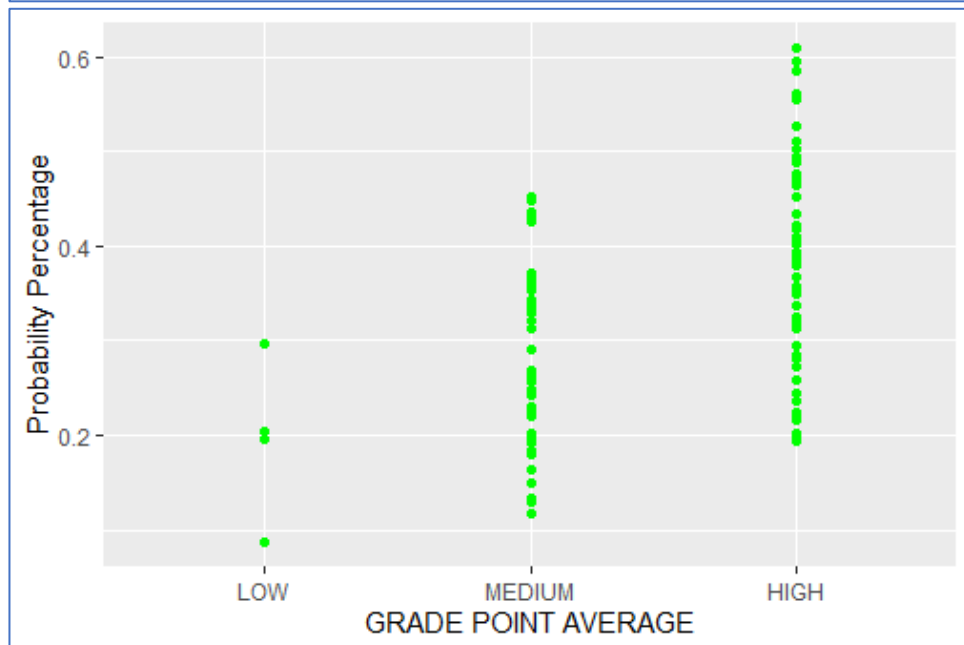
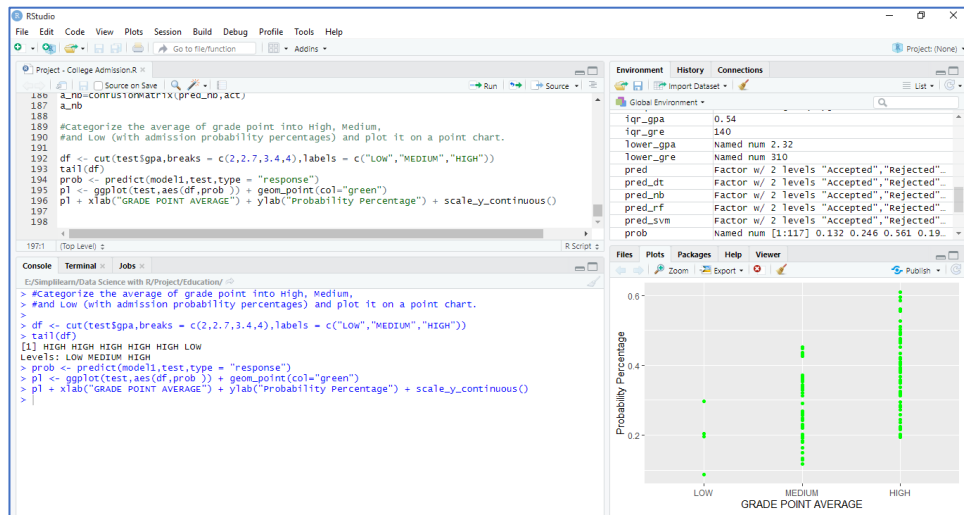
## Interpretation:

Accordingly, we can say that Random Forest or Naïve Bayes is the most accurate Model.

## Descriptive

## 11. 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 given above.

GPA data are categorized using K-means clustering using K=3. The clusters and admission probability of them are summarized below.



# Programming Codes:

## #Reading Comcast Data and loading libraries

```
rm(list=ls())

library(dplyr)
library(ggplot2)
library(lubridate)
library(plyr)
library(tidyverse)
library(caret)
library(rattle)
library(party) # For decision tree
library(rpart) # for Rpart
library(rpart.plot) #for Rpart plot
library(lattice) # Used for Data Visualization
library(randomForest)# FOr Random Forest
library(pROC)
library(e1071) # For SVM
library(naivebayes) # For Naive Bayes

setwd("E:\\Simplilearn\\Data Science with R\\Project\\Education")

ca<- read.csv("College_Admission.csv")
head(ca)
```

## # Missing Data

```
sum(is.na(ca))
anyNA(ca)
```

## #Checking outliers

```
attach(ca)
par(mfrow=c(1,2))
boxplot(ca$gre, main="Boxplot of GRE")
boxplot(ca$gpa, main="Boxplot of GPA")
```

## #Performing Outlier treatment

## #Removing outliers for gre

```
Q_gre <- quantile(ca$gre, probs = c(0.25, 0.75)) # 25% is 520 and 75% is 660
iqr_gre <- IQR(ca$gre) # Interquantile range is 140
```

```

uper_gre <- Q_gre[2]+1.5*iqr_gre # upper limit is 870
lower_gre <- Q_gre[1]-1.5*iqr_gre # lower limit is 310
ca <- subset(ca, ca$gre > lower_gre & ca$gre < uper_gre)

```

#### #Removing outliers for gpa on already removed dataset

```

Q_gpa <- quantile(ca$gpa, probs = c(0.25, 0.75)) # 25% is 3.13 and 75% is 3.67
iqr_gpa <- IQR(ca$gpa) # Interquantile range is 0.54
uper_gpa <- Q_gpa[2]+1.5*iqr_gpa # upper limit is 4.48
lower_gpa <- Q_gpa[1]-1.5*iqr_gpa # lower limit is 2.32
ca <- subset(ca, ca$gpa > lower_gpa & ca$gpa < uper_gpa)

```

```

par(mfrow=c(1,2))
boxplot(ca$gre, main="Boxplot of GRE")
boxplot(ca$gpa, main="Boxplot of GPA")

```

#### #Factoring the categorical data

```

str(ca)

ca$admit = factor(ca$admit, levels = c("0","1"), labels = c("Accepted","Rejected"))
ca$ses = factor(ca$ses, levels = c("1","2","3"), labels = c("Low","Medium","High"))
ca$Gender_Male = factor(ca$Gender_Male, levels = c("0","1"), labels = c("Female","Male"))
ca$Race = factor(ca$Race, levels = c("1","2","3"), labels = c("Hispanic","Asian","African-American"))
ca$rank <- factor(ca$rank, order = TRUE)

```

#### #Categorising GRE Marks to Category

```

ca = mutate(ca, GRE_category = ifelse(gre <= 440, "Low",
                                     ifelse(gre <= 580, "Medium", "High")))
Freq = table(ca$GRE_category)
Freq

```

#### #Checking if normally distributed

```
summary(ca)
```

#### # Density plot

```

par(mfrow=c(1,2))
d <- density(ca$gpa)
plot(d, main="Kernel Density of GPA")
polygon(d, col="red", border="blue")

```

```
d1 <- density(ca$gre)
plot(d1, main="Kernel Density of GRE")
polygon(d1, col="red", border="blue")
```

```
hist(ca$gpa, freq = FALSE)
x <- seq(0, 4, length.out=100)
y <- with(ca, dnorm(x, mean(gpa), sd(gpa)))
lines(x, y, col = "red")
```

```
hist(ca$gpa, freq = FALSE)
x <- seq(0, 4, length.out=100)
y <- with(ca, dnorm(x, mean(gpa), sd(gpa)))
lines(x, y, col = "red")
```

#### #Normalise the data

```
ca$gpa1 <- scale(ca$gpa)
ca$gre1 <- scale(ca$gre)
```

```
hist(ca$gpa1, freq = FALSE)
x <- seq(-3, 3, length.out=100)
y <- with(ca, dnorm(x, mean(gpa1), sd(gpa1)))
lines(x, y, col = "blue")
```

```
hist(ca$gre1, freq = FALSE)
x <- seq(-3, 3, length.out=100)
y <- with(ca, dnorm(x, mean(gpa1), sd(gpa1)))
lines(x, y, col = "blue")
```

#### #variable reduction techniques to identify significant variables

```
model <- glm(admit~ ., family = binomial(link = 'logit'), data = ca)
summary(model)
```

```
anova(model, test = 'Chisq')
```

#### # Logistic regression model with significance independent variable

```
set.seed(123)
splitIndex <- createDataPartition(ca$admit, p = .70, list = FALSE, times = 1)
train <- ca[ splitIndex,]
test <- ca[-splitIndex,]
```

```
model1 <- glm(admit~ gpa1+gre1+rank , data = train,family=binomial(link = "logit"))
summary(model1)
```

#accuracy of the model and run validation techniques

#Predict on Test through Model

```
pred = predict(model1,test, type="response")
pred = ifelse(pred>0.5,1,0)
pred = factor(pred, levels = c("0","1"), labels = c("Accepted","Rejected"))
```

####Validate the model - Confusion Matrix##

```
act <- test$admit
```

# Accuracy

```
table(pred, act)
a=confusionMatrix(pred, act)
a
```

#Model generation using other ML techniques

#1. Decision tree

```
model_dt = rpart(admit~ gpa1+gre1+rank, data = train,method = "class",
  control = rpart.control(minsplit = 30,cp = 0.01))
```

```
par(mfrow=c(1,1))
fancyRpartPlot(model_dt)
```

```
pred_dt = predict(model_dt,test, type="class")
table(pred_dt, act)
a_dt=confusionMatrix(pred_dt,act)
a_dt
```

#2. SVM

```
svmfit =svm(admit~ gpa1+gre1+rank, data = train, kernel="linear",
  scale = T)
pred_svm = predict(svmfit,test, type="response")
table(pred_svm, act)
a_svm=confusionMatrix(pred_svm,act)
a_svm
```



### #3. Random Forest

```
fit_rf = randomForest(admit~ gpa1+gre1+rank, data = train, do.trace=F)
pred_rf = predict(fit_rf,test)
table(pred_rf, act)
a_rf=confusionMatrix(pred_rf,act)
a_rf
```

### #4. Naive Bayes

```
fit_nb = naive_bayes(admit~ gpa1+gre1+rank, data = train)
pred_nb = predict(fit_nb,test)
table(pred_nb, act)
a_nb=confusionMatrix(pred_nb,act)
a_nb
```

#Categorize the average of grade point into High, Medium,  
#and Low (with admission probability percentages) and plot it on a point chart.

```
df <- cut(test$gpa,breaks = c(2,2.7,3.4,4),labels = c("LOW","MEDIUM","HIGH"))
tail(df)
prob <- predict(model1,test,type = "response")
pl <- ggplot(test,aes(df,prob )) + geom_point(col="green")
pl + xlab("GRADE POINT AVERAGE") + ylab("Probability Percentage") + scale_y_continuous()
```

-----The End-----