College Admission Using Classification

Business Scenario:

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.

Expectation:

Hence to make the entire process mentioned above easy, the education department in the US analyse the factors that influence the admission of a student into colleges. The objective of this exercise is to analyse the same using different classification models like Logistic Regression, Decision Tree and SVM.

Coding:

Loading the csv file into a “college” and proceeding further.

```
#Data Loading and Manipulation

college<- read.csv(file.choose(),header=T)
head(college)
college
str(college)
any(is.na(college))  #checking for any NaNs in dataset

pairs(college, col=college$Direction)  #plotting pairplot to check correlation and dependability
summary(college)

#checking for outliers using box plot
```

```
boxplot(college,main="Checking outlier in the dataset")
```

#Visualising columns “GRE” and “GPA” for outliers

```
boxplot(college$gre,main="GRE")
boxplot(college$gpa,main="GPA")
```

#checking normalization of data using plot

```
library(rcompanion)
plotNormalHistogram(college)
```

#Normalization of the dataset

```
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }
college_normal<- as.data.frame(lapply(college[1:7], normalize))
summary(college_normal$admit)
summary(college$gre)
summary(college$gpa)
```

#checking whether normalization completed successfully

```
plotNormalHistogram(college_normal,main="Normalization Completed")
college_final[sapply(college_final, is.numeric)] <-
lapply(college_final[sapply(college_final, is.numeric)],as.factor)
str(college_final)
```

#Standardization of the data

```
college_clean<-scale(college_final)
any(is.na(college_clean))  #checking for NaNs
```

#Removing the Outliers

```
IQR_gre = 660.0-520.0
Upfen_gre=660.0-1.5*IQR_gre
Upfen_gre

IQR_gpa=3.670-3.130
Upfen_gpa=3.670-1.5*IQR_gpa
Upfen_gpa
college_clean<-subset(college_clean,college_clean["gre"]<= Upfen_gre&
college_clean["gre"]<= Upfen_gpa)

boxplot(college_clean) #checking for outliers
```

#PlotDistribution For Normalization Check

```
plotDistribution = function (x) {
N = length(x)
x <- na.omit(x)
hist(x,col = "light blue",
probability = TRUE)
lines(density(x), col = "red", lwd = 3)
rug(x)
print(N-length(x))
}
plotDistribution(college_clean)

library(rcompanion)
plotNormalHistogram(college_clean)
```

#shapiro test on the dataset

```
shapiro.test(college_clean)

#Checking for correlation amongst datapoints
library(corrplot)
correlations <- cor(college_clean)
corrplot(correlations, method="circle")

#Taking a subset for model creation
college_final<-college_clean[,c('admit','gre','gpa','rank')]
college_final

#Creating train and test dataset

library(caTools)
set.seed(2020)
split_data<- sample.split(college_final["admit"],SplitRatio=3/4)
split_data

train_data <- subset(college, split_data == T)
test_data <- subset(college, split_data == F)

#Creating Logistic Model
model_lr<-glm(admit~.,data=train_data,family='binomial')
summary(model_lr)

predictions <- predict(model_lr,type='response',test_data["admit"],na.action =
na.pass)
predictions
mean(predictions) #checking for prediction mean
```

```
#Confusion Matrix
confusion_matrix<-table(predict, test_data["admit"])
confusion_matrix

#Accuracy Score
accuracy<-sum(diag(confusion_matrix))/sum(confusion_matrix)
accuracy
```

Decision Tree

```
#Data Loading and Manipulation
college_final<-college_clean[,c('admit','gre','gpa','rank')]
college_final

#Splitting dataset
library(caTools)
set.seed(2020)
split_data<- sample.split(college_final["admit"],SplitRatio=3/4)
split_data

train_data_dt <- subset(college, split_data == T)
test_data_dt <- subset(college, split_data == F)

#Plotting decision tree
library('rpart')
library('rpart.plot')
```

```
ctrt_dt<-rpart.control(minsplit =90 ,minbucket = 32,xval=)
model<-rpart(admit~.,control=ctrt_dt,data=train_data_dt) #creating model
rpart.plot(model)
summary(model)
#Predictions
y_pred<-predict(model,test_data_dt,type='vector')

#Confusion Matrix
tab<-table(y_pred,test_data_dt$admit)
tab
#Accuracy Score
accuracy<-sum(diag(tab))/sum(tab)
accuracy
```

Support Vector Machine

```
#Data Manipulation
college_final<-college_clean[,c('admit','gre','gpa','rank')]
college_final

#Factorization of “admit” column
college_final["admit"] = factor(college_final["admit"], levels =c(0, 1))

#Splitting train and test data
library(caTools)
set.seed(2020)
split_data<- sample.split(college_final["admit"],SplitRatio=3/4)
split_data
```

```
train_data <- subset(college, split_data == T)
test_data <- subset(college, split_data == F)

train_data = scale(train_data[-1])
test_data = scale(test_data[-1])

df2<-train_data[complete.cases(train_data),] #getting rid of NaNs
str(df2)

install.packages('e1071')
library(e1071)
```

#Creating SVM Model:

```
classifier = svm(formula = df2 ~ .,
data = train_data,
type = 'C-classification',
kernel = 'linear')

dat = data.frame(train_data, y = as.factor(train_data["admit"]))
svmfit = svm(train_data["admit"] ~ ., data = dat, kernel = "linear", cost = 10,
scale=FALSE)
```

Predicting the Test set results

```
y_pred = predict(classifier, newdata = test_data[-1])

#Confusion Matrix
cm = table(test_set[, -1], y_pred)
cm
```

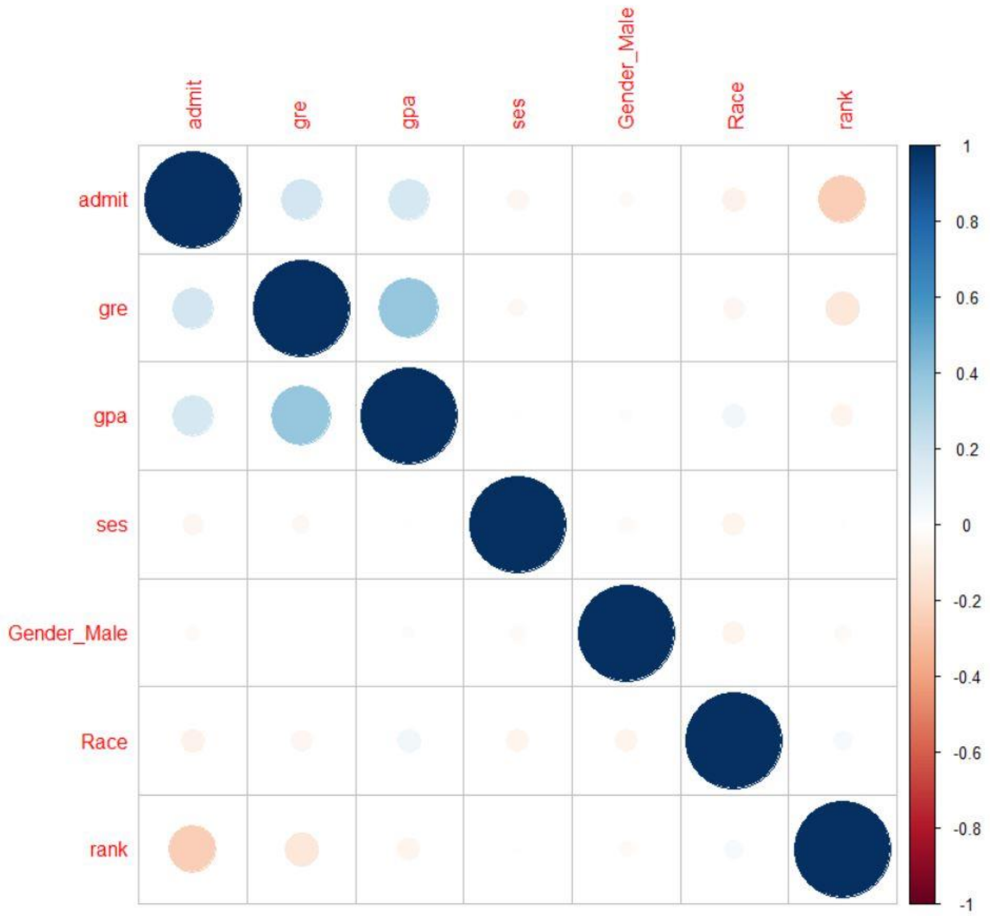
#Accuracy Score

acc = sum(diag(cm))/sum(cm)

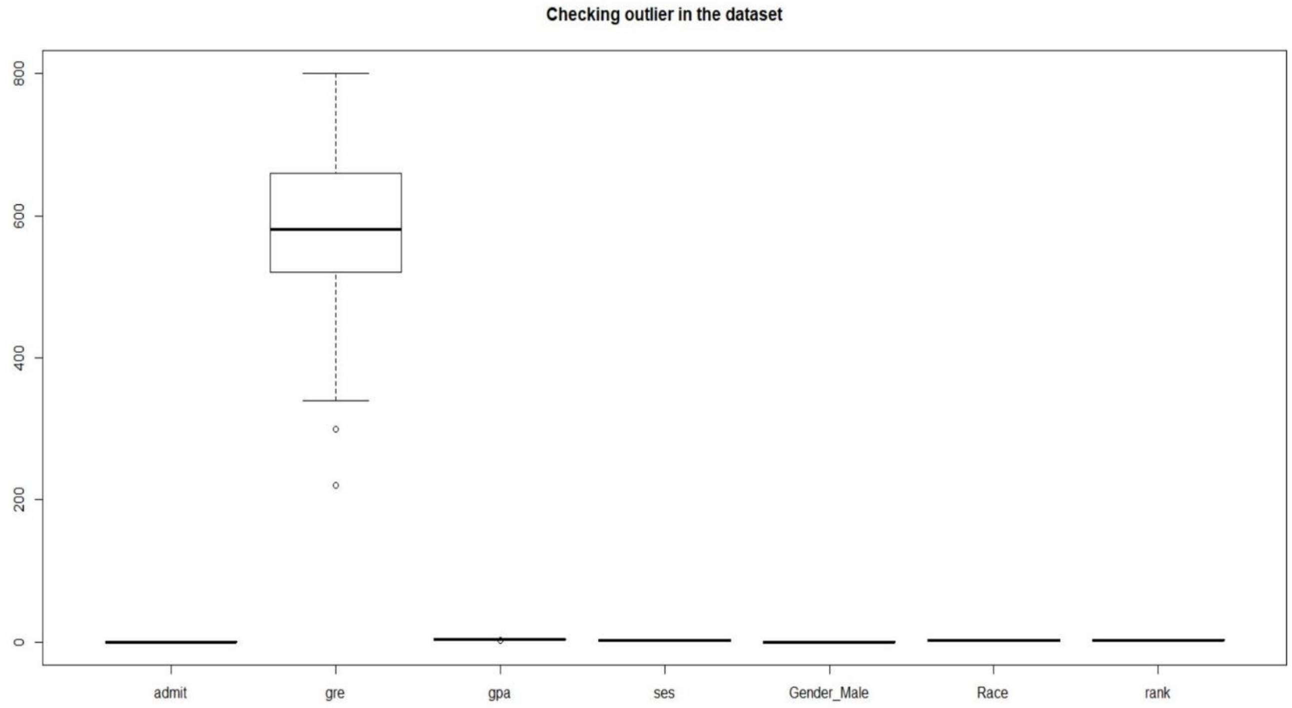
acc

Output:

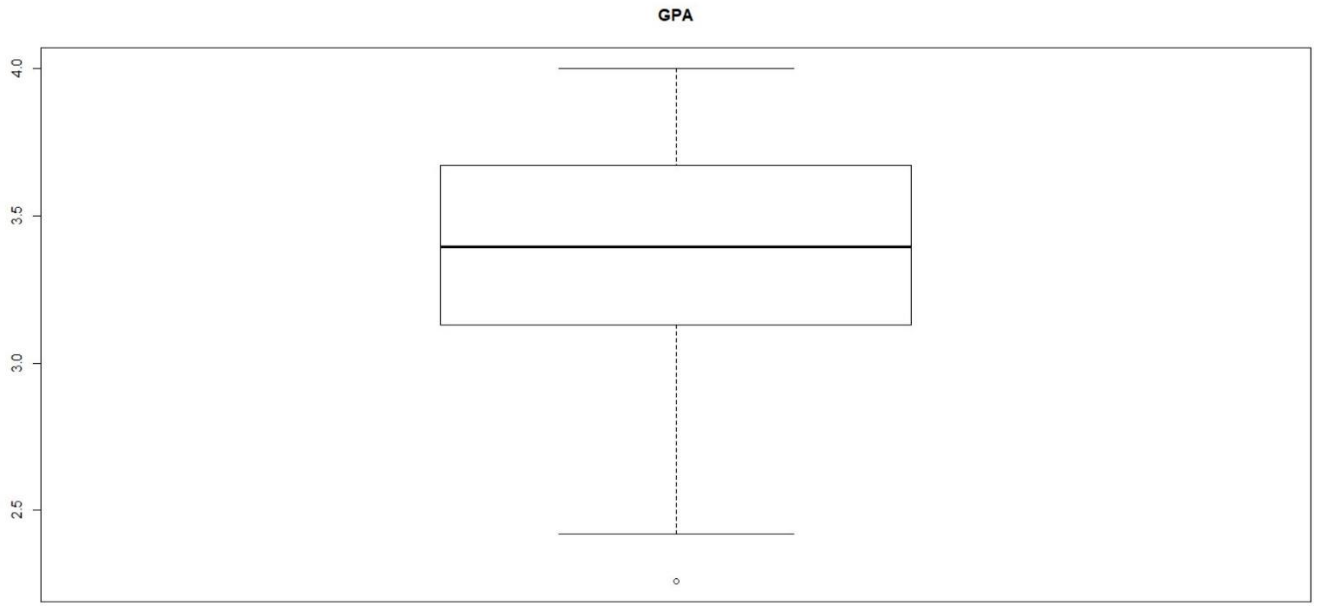
Plots:



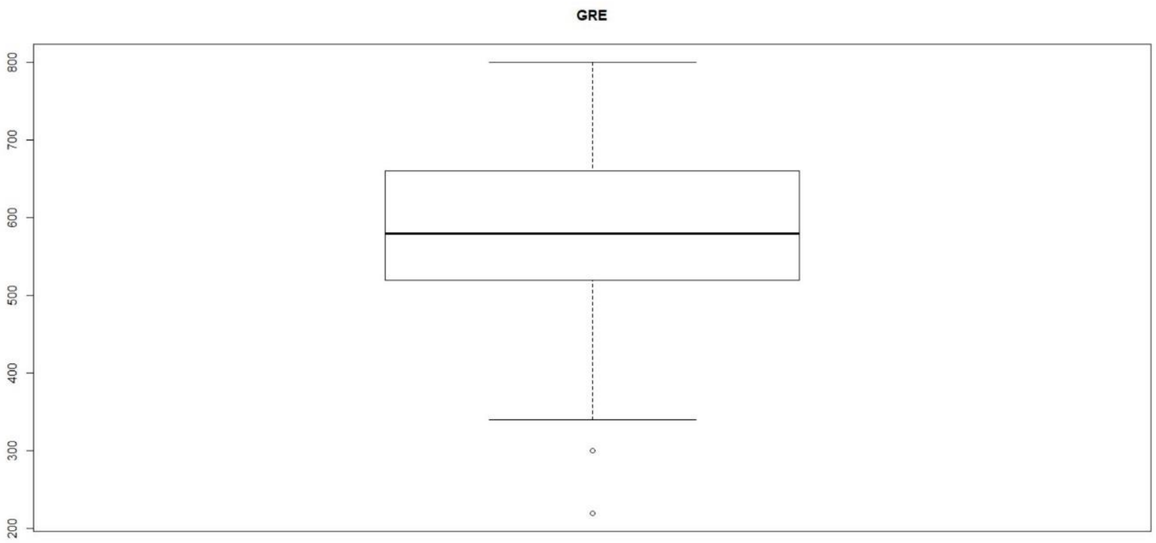
Correlation Chart



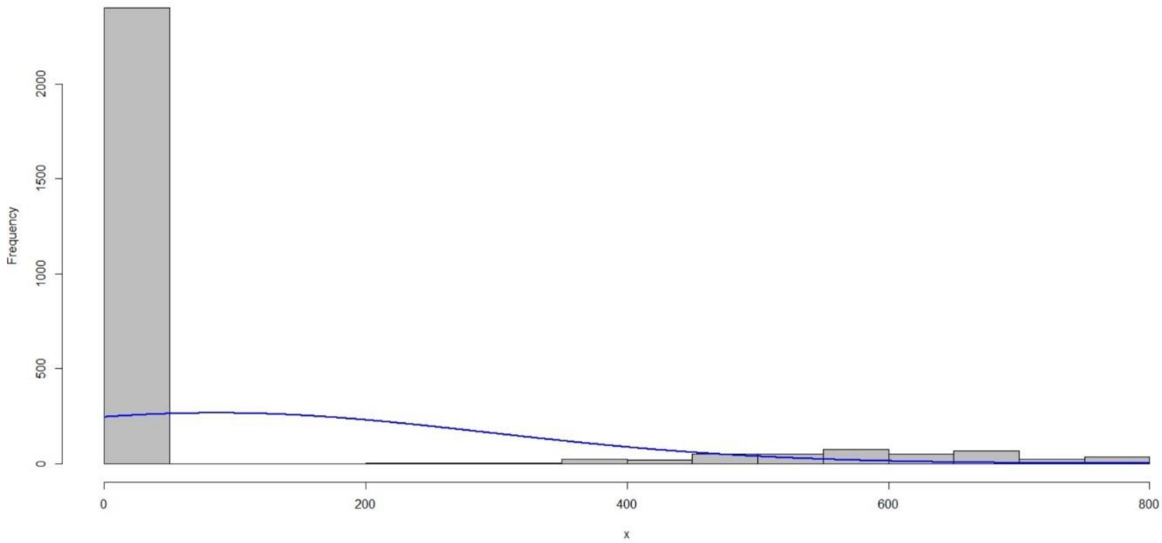
Outlier Check



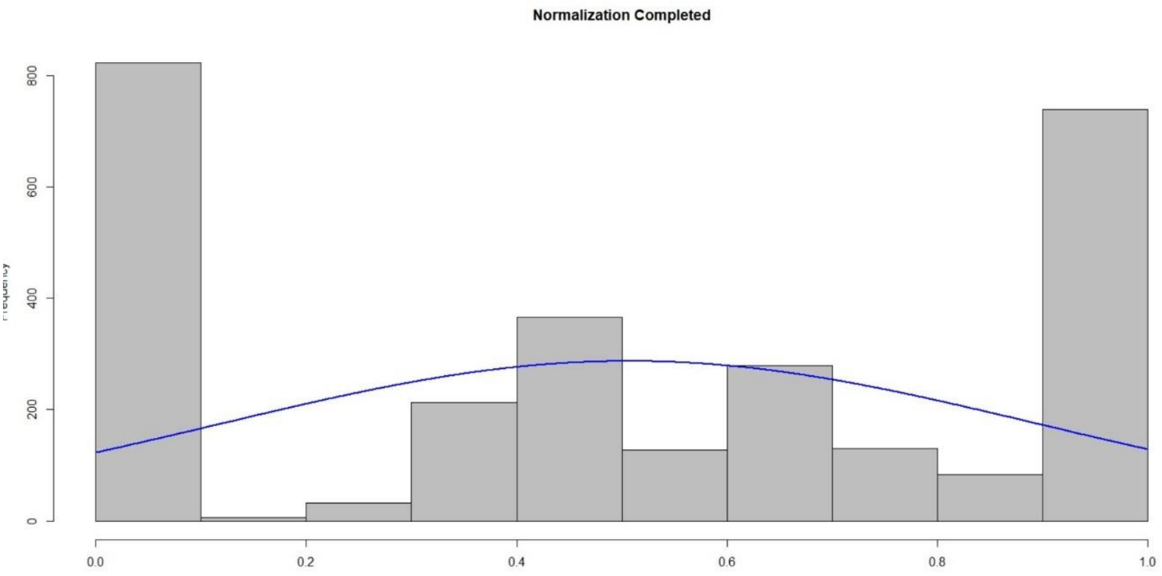
GPA Outliers



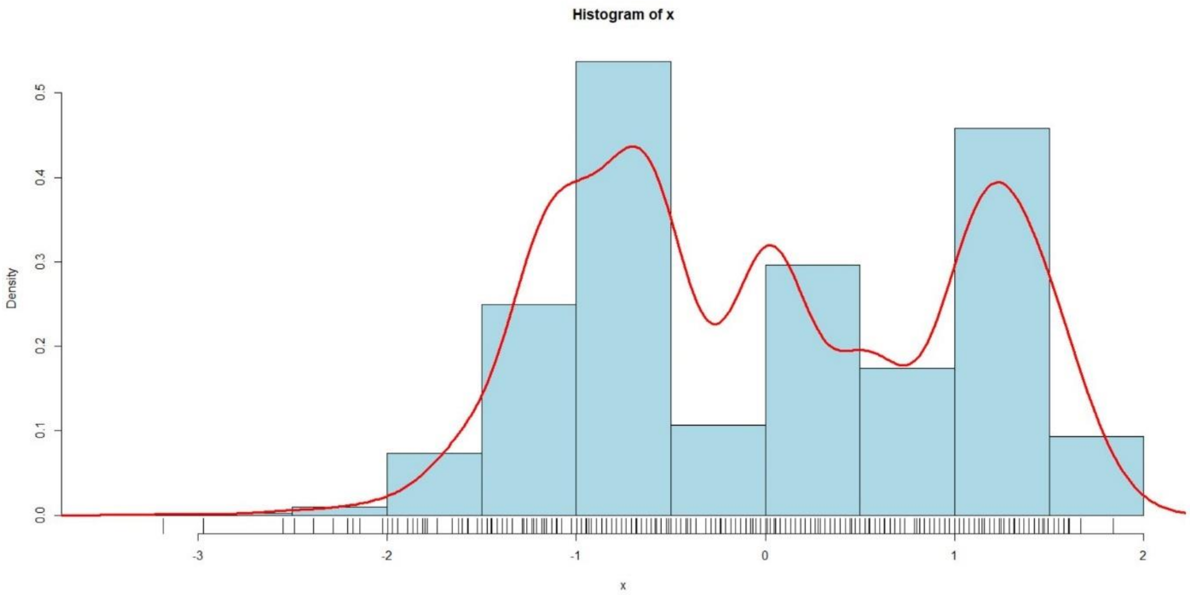
GRE Outliers



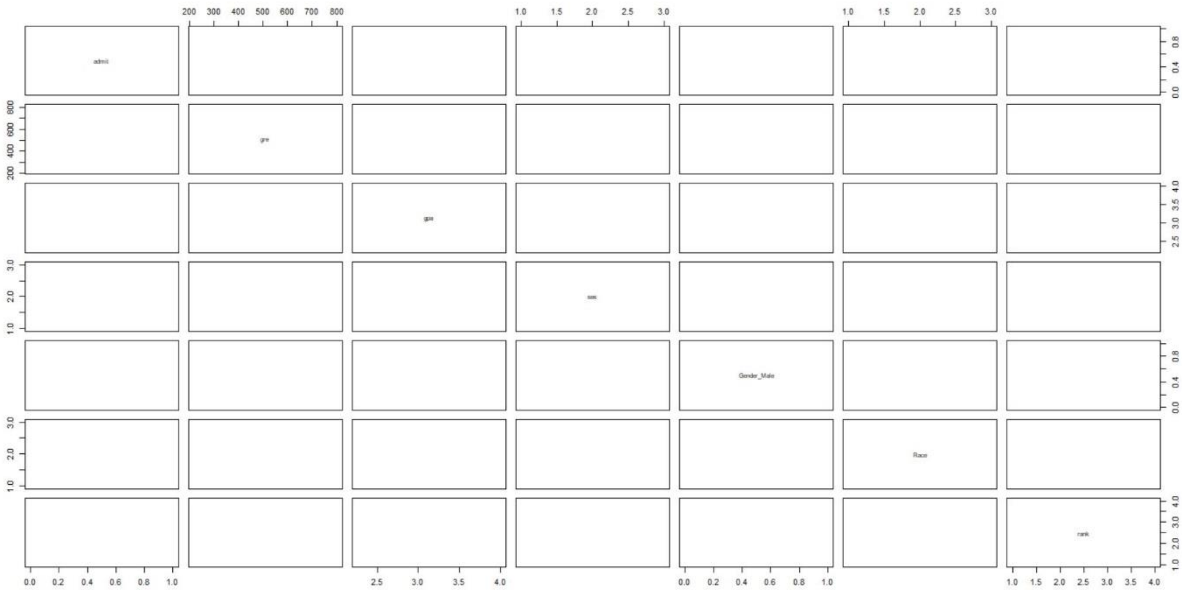
Skewness Chart(Right)



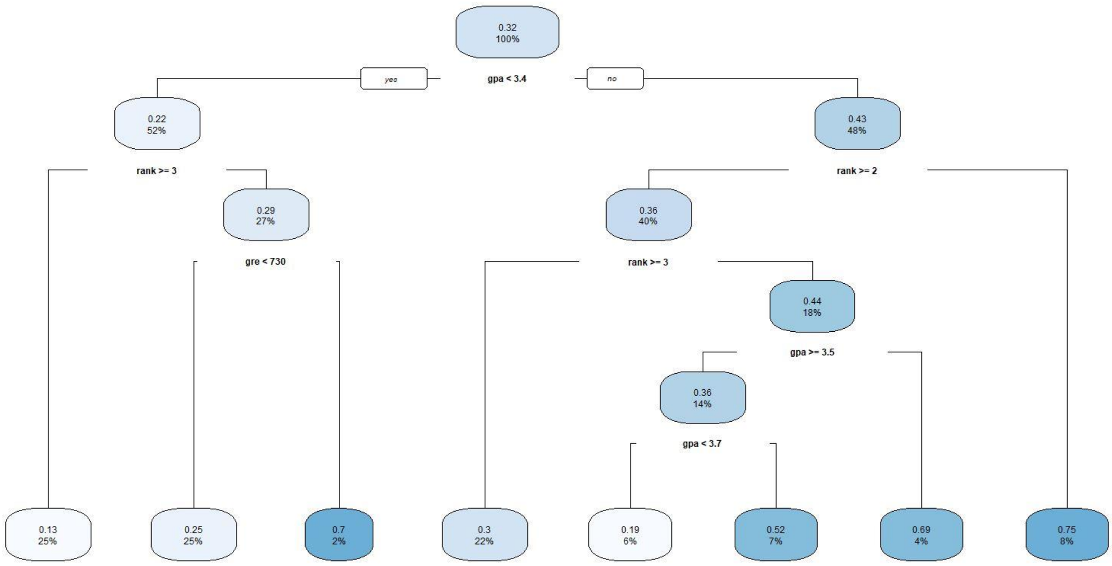
Normalized Dataset



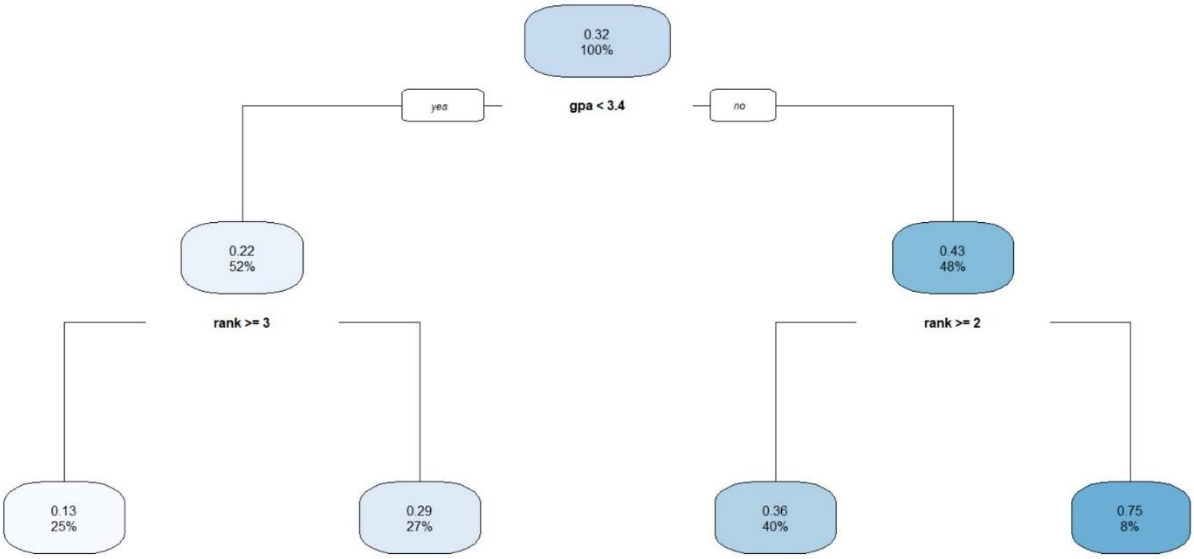
Density Plot



Pair Plot



Decision Tree



Final Tree for accuracy

<u>Output Snippets:</u>

```
> college<- read.csv(file.choose(),header=T)
> head(college)
  admit gre  gpa ses Gender_Male Race rank
1     0 380 3.61   1           0    3    3
2     1 660 3.67   2           0    2    3
3     1 800 4.00   2           0    2    1
4     1 640 3.19   1           1    2    4
5     0 520 2.93   3           1    2    4
6     1 760 3.00   2           1    1    2
> college
    admit gre  gpa ses Gender_Male Race rank
1       0 380 3.61   1           0    3    3
2       1 660 3.67   2           0    2    3
3       1 800 4.00   2           0    2    1
4       1 640 3.19   1           1    2    4
5       0 520 2.93   3           1    2    4
6       1 760 3.00   2           1    1    2
7       1 560 2.98   2           1    2    1
8       0 400 3.08   2           0    2    2
9       1 540 3.39   1           1    1    3
10      0 700 3.92   1           0    2    2
11      0 800 4.00   1           1    1    4
12      0 440 3.22   3           0    2    1
13      1 760 4.00   3           1    2    1
14      0 700 3.08   2           0    2    2
15      1 700 4.00   2           1    1    1
16      0 480 3.44   3           0    1    3
17      0 780 3.87   2           0    3    4
18      0 360 2.56   3           1    3    3
19      0 800 3.75   1           1    3    2
20      1 540 3.81   1           0    3    1
21      0 500 3.17   3           0    2    3
22      1 660 3.63   1           0    1    2
23      0 600 2.82   1           0    3    4
24      0 680 3.19   1           0    1    4
25      1 760 3.35   2           0    2    2
26      1 800 3.66   2           1    1    1
27      1 620 3.61   2           0    1    1
28      1 520 3.74   2           0    3    4
29      1 780 3.22   1           0    1    2
30      0 520 3.29   1           0    1    1
31      0 540 3.78   1           1    1    4
32      0 760 3.35   2           1    1    3
```

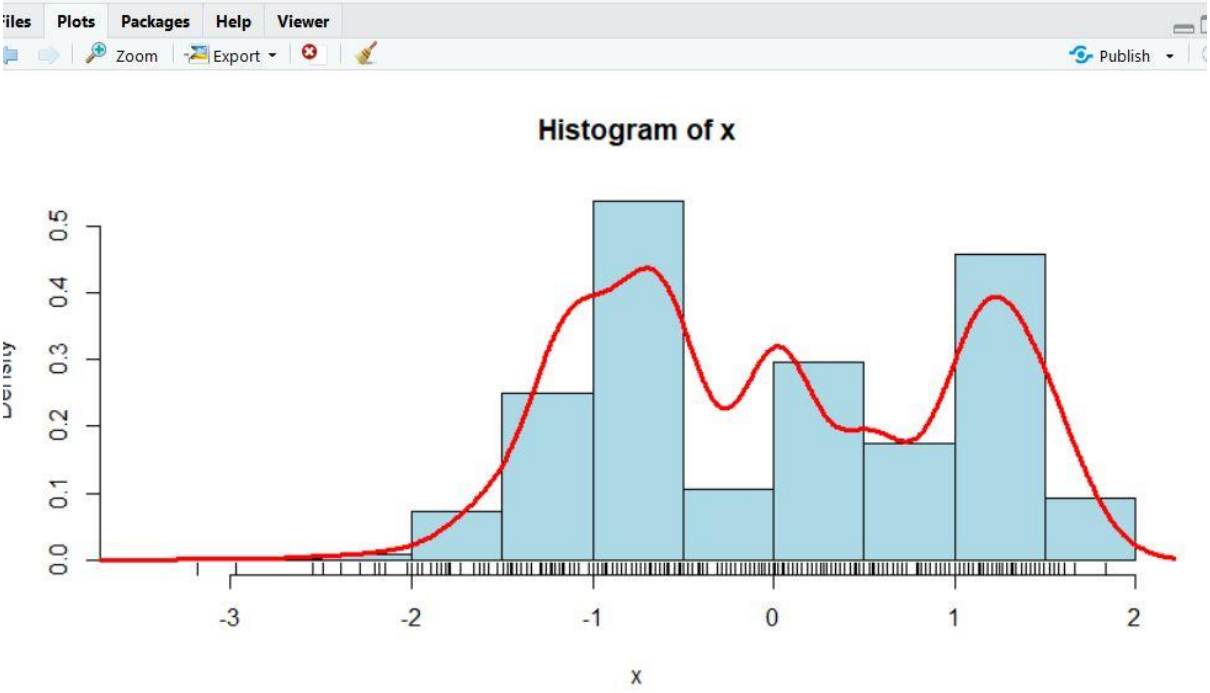
```
> str(college)
'data.frame':   400 obs. of  7 variables:
 $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
 $ Gender_Male: 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 ...
> any(is.na(college))
[1] FALSE
> pairs(college, col=college$Direction)
> summary(college)
     admit             gre             gpa             ses         Gender_Male         Race            rank      
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000   Min.   :0.000   Min.   :1.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:1.000   1st Qu.:0.000   1st Qu.:1.000   1st Qu.:2.000  
 Median :0.0000   Median :580.0   Median :3.395   Median :2.000   Median :0.000   Median :2.000   Median :2.000  
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :1.992   Mean   :0.475   Mean   :1.962   Mean   :2.485  
 3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000   3rd Qu.:1.000   3rd Qu.:3.000   3rd Qu.:3.000  
 Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :3.000   Max.   :1.000   Max.   :3.000   Max.   :4.000  
>
```

```
 Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :3.000   Max.   :1.000   Max.   :1.000   Max
  :3.000   Max.   :4.000  
> boxplot(college,main="Checking outlier in the dataset")
> boxplot(college$gre,main="GRE")
> boxplot(college$gpa,main="GPA")
> library(rcompanion)
Warning message:
package ‘rcompanion’ was built under R version 3.6.3 
> plotNormalHistogram(college)
> 
> normalize <- function(x) {
+   return ((x - min(x)) / (max(x) - min(x))) }
> college_normal<- as.data.frame(lapply(college[1:7], normalize))
> college_normal
    admit       gre        gpa ses Gender_Male Race
1       0 0.2758621 0.77586207 0.0           0  1.0
2       1 0.7586207 0.81034483 0.5           0  0.5
3       1 1.0000000 1.00000000 0.5           0  0.5
4       1 0.7241379 0.53448276 0.0           1  0.5
5       0 0.5172414 0.38505747 1.0           1  0.5
6       1 0.9310345 0.42528736 0.5           1  0.0
7       1 0.5862069 0.41379310 0.5           1  0.5
8       0 0.3103448 0.47126437 0.5           0  0.5
9       1 0.5517241 0.64942529 0.0           1  0.0
10      0 0.8275862 0.95402299 0.0           0  0.5
11      0 1.0000000 1.00000000 0.0           1  0.0
12      0 0.3793103 0.55172414 1.0           0  0.5
13      1 0.9310345 1.00000000 1.0           1  0.5
14      0 0.8275862 0.47126437 0.5           0  0.5
15      1 0.8275862 1.00000000 0.5           1  0.0
16      0 0.4482759 0.67816092 1.0           0  0.0
17      0 0.9655172 0.92528736 0.5           0  1.0
18      0 0.2413793 0.17241379 1.0           1  1.0
19      0 1.0000000 0.85632184 0.0           1  1.0
20      1 0.5517241 0.89080460 0.0           0  1.0
21      0 0.4827586 0.52298851 1.0           0  0.5
22      1 0.7586207 0.78735632 0.0           0  0.0
23      0 0.6551724 0.32183908 0.0           0  1.0
24      0 0.7931034 0.53448276 0.0           0  0.0
25      1 0.9310345 0.62643678 0.5           0  0.5
26      1 1.0000000 0.80459770 0.5           1  0.0
27      1 0.6896552 0.77586207 0.5           0  0.0
28      1 0.5172414 0.85057471 0.5           0  1.0
```

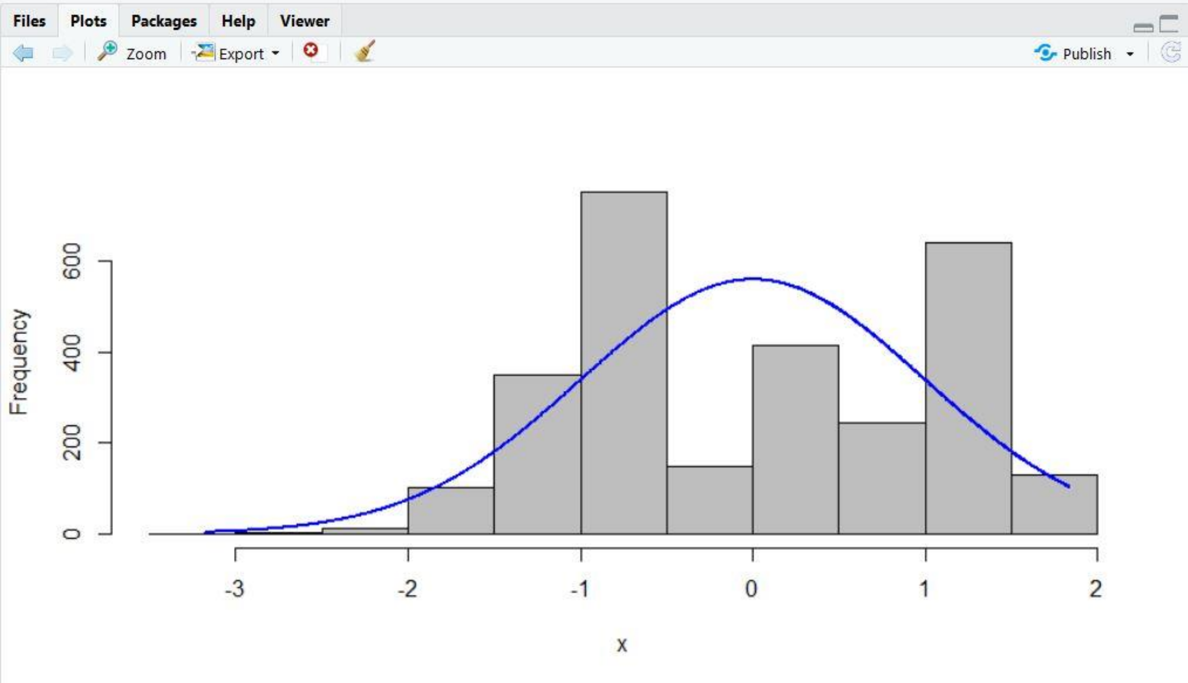
```
[ reached 'max' / getOption("max.print") -- omitted 258 rows ]
>
> summary(college$gre)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  220.0   520.0   580.0   587.7   660.0   800.0 
> summary(college$gpa)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.260   3.130   3.395   3.390   3.670   4.000 
>
> plotNormalHistogram(college_normal,main="Normalization Completed")
> 
```

```
> plotNormalHistogram(college_normal,main="Normalization Completed")
> IQR_gre = 660.0-520.0
> Upfen_gre=660.0-1.5*IQR_gre
> Upfen_gre
[1] 450
>
> IQR_gpa=3.670-3.130
> Upfen_gpa=3.670-1.5*IQR_gpa
> Upfen_gpa
[1] 2.86
> college_clean<-scale(college)
> college_clean
           admit         gre          gpa          ses Gender_Male        Race
 [1,] -0.6812037 -1.79801097  0.578347918 -1.227200236       -0.95  1.26020471
 [2,]  1.4643197  0.62588442  0.736007505  0.009273553       -0.95  0.04554957
 [3,]  1.4643197  1.83783211  1.603135233  0.009273553       -0.95  0.04554957
 [4,]  1.4643197  0.45274903 -0.525269190 -1.227200236        1.05  0.04554957
 [5,] -0.6812037 -0.58606328 -1.208460734  1.245747343        1.05  0.04554957
 [6,]  1.4643197  1.49156134 -1.024524549  0.009273553        1.05 -1.16910557
 [7,]  1.4643197 -0.23979251 -1.077077744  0.009273553        1.05  0.04554957
 [8,] -0.6812037 -1.62487559 -0.814311766  0.009273553       -0.95  0.04554957
 [9,]  1.4643197 -0.41292789  0.000262766 -1.227200236        1.05 -1.16910557
[10,] -0.6812037  0.97215519  1.392922450 -1.227200236       -0.95  0.04554957
[11,] -0.6812037  1.83783211  1.603135233 -1.227200236        1.05 -1.16910557
[12,] -0.6812037 -1.27860482 -0.446439397  1.245747343       -0.95  0.04554957
[13,]  1.4643197  1.49156134  1.603135233  1.245747343        1.05  0.04554957
[14,] -0.6812037  0.97215519 -0.814311766  0.009273553       -0.95  0.04554957
[15,]  1.4643197  0.97215519  1.603135233  0.009273553        1.05 -1.16910557
[16,] -0.6812037 -0.93233405  0.131645755  1.245747343       -0.95 -1.16910557
[17,] -0.6812037  1.66469673  1.261539461  0.009273553       -0.95  1.26020471
[18,] -0.6812037 -1.97114636 -2.180694853  1.245747343        1.05  1.26020471
[19,] -0.6812037  1.83783211  0.946220287 -1.227200236        1.05  1.26020471
[20,]  1.4643197 -0.41292789  1.103879874 -1.227200236       -0.95  1.26020471
[21,] -0.6812037 -0.75919866 -0.577822386  1.245747343       -0.95  0.04554957
[22,]  1.4643197  0.62588442  0.630901114 -1.227200236       -0.95 -1.16910557
[23,] -0.6812037  0.10647826 -1.497503309 -1.227200236       -0.95  1.26020471
[24,] -0.6812037  0.79901980 -0.525269190 -1.227200236       -0.95 -1.16910557
[25,]  1.4643197  1.49156134 -0.104843625  0.009273553       -0.95  0.04554957
[26,]  1.4643197  1.83783211  0.709730907  0.009273553        1.05 -1.16910557
[27,]  1.4643197  0.27961365  0.578347918  0.009273553       -0.95 -1.16910557
[28,]  1.4643197 -0.58606328  0.919943690  0.009273553       -0.95  1.26020471
[29,]  1.4643197  1.66469673 -0.446439397 -1.227200236       -0.95 -1.16910557
```

```
plotDistribution = function (x) {
  N = length(x)
  x <- na.omit(x)
  hist( x,col = "light blue",
        probability = TRUE)
  lines(density(x), col = "red", lwd = 3)
  rug(x)
  print(N-length(x))
}

plotDistribution(college_clean)
1] 0
```



```
> plotDistribution(college_clean)
[1] 0
> library(rcompanion)
> plotNormalHistogram(college_clean)
>
> shapiro.test(college_clean)

        Shapiro-Wilk normality test

data:  college_clean
W = 0.9423, p-value < 2.2e-16

>
```



Environment | History | Connections

Import Dataset

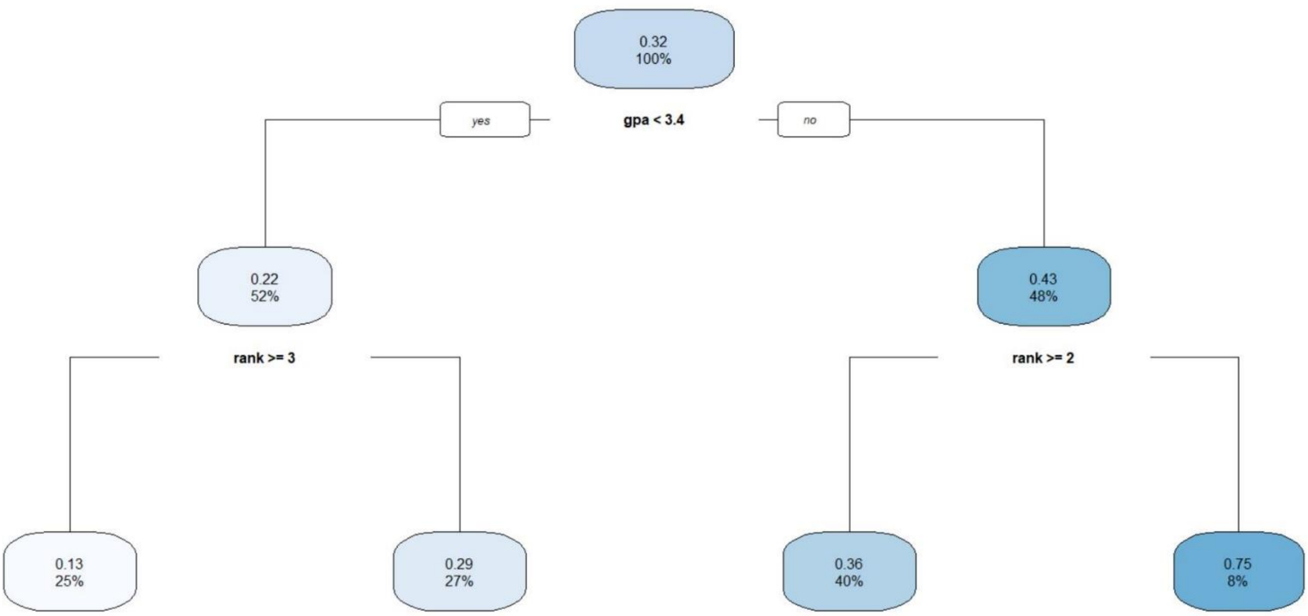
Global Environment

Data

```
college                    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 ...
```

```
> split_data<- sample.split(college_final["admit"],SplitRatio=3
/4)
> #split_data
>
> train_data <- subset(college, split_data == T)
> test_data <- subset(college, split_data == F)
> #test_data
>
> model_lr<-glm(admit~.,data=train_data,family='binomial')
> summary(model_lr)

Call:
glm(formula = admit ~ ., family = "binomial", data = train_data
)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6919  -0.8830  -0.6336   1.1443   2.1668

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.853135   1.188142  -2.401   0.0163 *
gre          0.002192   0.001097   1.999   0.0456 *
gpa          0.828926   0.331558   2.500   0.0124 *
ses         -0.149553   0.141519  -1.057   0.2906
Gender_Male -0.166132   0.228314  -0.728   0.4668
Race        -0.172074   0.138897  -1.239   0.2154
rank        -0.562333   0.127778  -4.401 1.08e-05 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 456.51  on 393  degrees of freedom
AIC: 470.51

Number of Fisher Scoring iterations: 4

> predictions <- predict(model_lr,type='response',test_data["ad
mit"],na.action = na.pass)
```

```
Console   Terminal ×
~/
> table(college$gre)

220 300 340 360 380 400 420 440 460 480 500 520 540 560 580
  1   3   4   4   8  11   7  10  14  16  21  24  27  24  29
600 620 640 660 680 700 720 740 760 780 800
 23  30  21  24  20  22  11  11   5   5  25
> table(college$gpa)

2.26 2.42 2.48 2.52 2.55 2.56 2.62 2.63 2.65 2.67 2.68 2.69
   1    2    1    1    1    1    2    1    1    2    1    1
 2.7 2.71 2.73 2.76 2.78 2.79 2.81 2.82 2.83 2.84 2.85 2.86
   2    2    1    1    2    2    3    2    1    1    2    2
2.87 2.88  2.9 2.91 2.92 2.93 2.94 2.95 2.96 2.97 2.98    3
   1    2    4    3    2    5    3    1    2    2    6    4
3.01 3.02 3.03 3.04 3.05 3.06 3.07 3.08 3.09  3.1 3.11 3.12
   2    4    1    2    3    1    4    4    1    1    1    4
3.13 3.14 3.15 3.16 3.17 3.18 3.19  3.2 3.21 3.22 3.23 3.24
   5    4    7    2    5    1    5    2    1    5    3    2
3.25 3.27 3.28 3.29  3.3 3.31 3.32 3.33 3.34 3.35 3.36 3.37
   2    3    4    2    4    8    4    5    5    7    4    3
3.38 3.39  3.4 3.41 3.42 3.43 3.44 3.45 3.46 3.47 3.48 3.49
   5    3    7    1    1    5    3    7    5    3    3    5
 3.5 3.51 3.52 3.53 3.54 3.55 3.56 3.57 3.58 3.59  3.6 3.61
   4    5    4    2    3    1    3    3    5    5    3    3
3.62 3.63 3.64 3.65 3.66 3.67 3.69  3.7 3.71 3.72 3.73 3.74
   2    6    5    4    1    4    3    3    2    1    2    4
3.75 3.76 3.77 3.78  3.8 3.81 3.82 3.83 3.84 3.85 3.86 3.87
   2    2    5    4    2    3    1    1    2    1    2    1
3.88 3.89  3.9 3.91 3.92 3.93 3.94 3.95 3.97 3.98 3.99    4
   3    3    3    1    2    1    5    5    1    1    3   28
> class(college$gre)
[1] "integer"
> class(college$gpa)
[1] "numeric"
>
> college_final<-college_clean[,c('admit','gre','gpa','rank')]
> #college_final
>
> library(caTools)
> set.seed(2020)
> split_data<- sample.split(college_final["admit"],SplitRatio=3/4)
> #split_data
>
```



```
> train_data_dt <- subset(college, split_data == T)
> test_data_dt <- subset(college, split_data == F)
>
>
> library('rpart')
> library('rpart.plot')
>
>
> ctrt_dt<-rpart.control(minsplit =90 ,minbucket = 32,xval=)
> model<-rpart(admit~.,control=ctrt_dt,data=train_data_dt)
> rpart.plot(model)
> summary(model)
Call:
rpart(formula = admit ~ ., data = train_data_dt, control = ctrt_dt)
  n= 400

          CP nsplit rel error    xerror       xstd
1 0.05115411      0 1.0000000 1.0050133 0.03942821
2 0.04619615      1 0.9488459 1.0301373 0.04407875
3 0.01575944      2 0.9026497 1.0371066 0.04735738
4 0.01000000      3 0.8868903 0.9948527 0.04776241

Variable importance
rank  gpa  gre Race  ses
  47   39   11    2    1

Node number 1: 400 observations,    complexity param=0.05115411
  mean=0.3175, MSE=0.2166938
  left son=2 (208 obs) right son=3 (192 obs)
  Primary splits:
      gpa  < 3.415 to the left,  improve=0.051154110, (0 missing)
      rank < 2.5   to the right, improve=0.044890180, (0 missing)
      gre  < 510   to the left,  improve=0.027942520, (0 missing)
      Race < 1.5   to the right, improve=0.007248128, (0 missing)
      ses  < 1.5   to the right, improve=0.002182185, (0 missing)
  Surrogate splits:
      gre  < 610   to the left,  agree=0.645, adj=0.260, (0 split)
      Race < 2.5   to the left,  agree=0.545, adj=0.052, (0 split)
      rank < 1.5   to the right, agree=0.528, adj=0.016, (0 split)

Node number 2: 208 observations,    complexity param=0.01575944
```

```
Node number 2: 208 observations,    complexity param=0.01575944
  mean=0.2163462, MSE=0.1695405
  left son=4 (99 obs) right son=5 (109 obs)
  Primary splits:
      rank < 2.5   to the right, improve=0.03873562, (0 missing)
      gre  < 570   to the left,  improve=0.03148175, (0 missing)
      ses  < 2.5   to the right, improve=0.02161037, (0 missing)
      Race < 1.5   to the right, improve=0.01971392, (0 missing)
      gpa  < 3.235 to the right, improve=0.01493426, (0 missing)
  Surrogate splits:
      gre         < 530   to the left,  agree=0.582, adj=0.121, (0 split)
      gpa         < 3.225 to the right, agree=0.553, adj=0.061, (0 split)
      ses         < 2.5   to the right, agree=0.548, adj=0.051, (0 split)
      Gender_Male < 0.5   to the left,  agree=0.534, adj=0.020, (0 split)

Node number 3: 192 observations,    complexity param=0.04619615
  mean=0.4270833, MSE=0.2446832
  left son=6 (160 obs) right son=7 (32 obs)
  Primary splits:
      rank        < 1.5   to the right, improve=0.085232820, (0 missing)
      ses         < 2.5   to the left,  improve=0.005877847, (0 missing)
      gre         < 650   to the left,  improve=0.005699774, (0 missing)
      gpa         < 3.945 to the left,  improve=0.005361822, (0 missing)
      Gender_Male < 0.5   to the right, improve=0.005260641, (0 missing)

Node number 4: 99 observations
  mean=0.1313131, MSE=0.11407

Node number 5: 109 observations
  mean=0.293578, MSE=0.20739

Node number 6: 160 observations
  mean=0.3625, MSE=0.2310938

Node number 7: 32 observations
  mean=0.75, MSE=0.1875
```

```r
library('rpart')
library('rpart.plot')


ctrt_dt<-rpart.control(minsplit =90 ,minbucket = 32,xval=)
model<-rpart(admit~.,control=ctrt_dt,data=train_data_dt)
rpart.plot(model)
summary(model)

y_pred<-predict(model,test_data_dt,type='vector')
y_pred

tab<-table(y_pred,test_data_dt$admit)
tab

sum(diag(tab))/sum(tab)
```

147:1 (Top Level)

Console Terminal

~/

```
> tab
   0  1
0 80 11
1  6  3
> sum(diag(tab))/sum(tab)
[1] 0.83
>
```

Console Terminal

~/

```
> college_final<-college_clean[,c('admit','gre','gpa','rank')]
> college_final
          admit         gre          gpa       rank
 [1,] -0.6812037 -1.79801097  0.578347918  0.5452850
 [2,]  1.4643197  0.62588442  0.736007505  0.5452850
 [3,]  1.4643197  1.83783211  1.603135233 -1.5723268
 [4,]  1.4643197  0.45274903 -0.525269190  1.6040909
 [5,] -0.6812037 -0.58606328 -1.208460734  1.6040909
 [6,]  1.4643197  1.49156134 -1.024524549 -0.5135209
 [7,]  1.4643197 -0.23979251 -1.077077744 -1.5723268
 [8,] -0.6812037 -1.62487559 -0.814311766 -0.5135209
 [9,]  1.4643197 -0.41292789  0.000262766  0.5452850
[10,] -0.6812037  0.97215519  1.392922450 -0.5135209
[11,] -0.6812037  1.83783211  1.603135233  1.6040909
[12,] -0.6812037 -1.27860482 -0.446439397 -1.5723268
[13,]  1.4643197  1.49156134  1.603135233 -1.5723268
[14,] -0.6812037  0.97215519 -0.814311766 -0.5135209
[15,]  1.4643197  0.97215519  1.603135233 -1.5723268
[16,] -0.6812037 -0.93233405  0.131645755  0.5452850
[17,] -0.6812037  1.66469673  1.261539461  1.6040909
[18,] -0.6812037 -1.97114636 -2.180694853  0.5452850
[19,] -0.6812037  1.83783211  0.946220287 -0.5135209
[20,]  1.4643197 -0.41292789  1.103879874 -1.5723268
[21,] -0.6812037 -0.75919866 -0.577822386  0.5452850
[22,]  1.4643197  0.62588442  0.630901114 -0.5135209
[23,] -0.6812037  0.10647826 -1.497503309  1.6040909
[24,] -0.6812037  0.79901980 -0.525269190  1.6040909
[25,]  1.4643197  1.49156134 -0.104843625 -0.5135209
[26,]  1.4643197  1.83783211  0.709730907 -1.5723268
[27,]  1.4643197  0.27961365  0.578347918 -1.5723268
[28,]  1.4643197 -0.58606328  0.919943690  1.6040909
[29,]  1.4643197  1.66469673 -0.446439397 -0.5135209
[30,] -0.6812037 -0.58606328 -0.262503212 -1.5723268
[31,] -0.6812037 -0.41292789  1.025050081  1.6040909
[32,] -0.6812037  1.49156134 -0.104843625  0.5452850
[33,] -0.6812037  0.10647826  0.026539364  0.5452850
[34,]  1.4643197  1.83783211  1.603135233  0.5452850
[35,] -0.6812037 -1.97114636 -0.656652179 -1.5723268
[36,] -0.6812037 -1.62487559 -0.893141560 -0.5135209
[37,] -0.6812037 -0.06665712 -0.367609603 -1.5723268
[38,] -0.6812037 -0.58606328 -1.287290527  0.5452850
[39,]  1.4643197 -0.75919866 -0.682928777 -0.5135209
[40,]  1.4643197 -0.58606328 -1.865375679  0.5452850
```

```
>
> college_final["admit"] = factor(college_final["admit"], levels = c(0, 1))
> college_final

-0.681203686  1.464319734  1.464319734  1.464319734 -0.681203686  1.464319734  1.464319734 -0.681203686

 1.464319734 -0.681203686 -0.681203686 -0.681203686  1.464319734 -0.681203686  1.464319734 -0.681203686

-0.681203686 -0.681203686 -0.681203686  1.464319734 -0.681203686  1.464319734 -0.681203686 -0.681203686

 1.464319734  1.464319734  1.464319734  1.464319734  1.464319734 -0.681203686 -0.681203686 -0.681203686

-0.681203686  1.464319734 -0.681203686 -0.681203686 -0.681203686 -0.681203686  1.464319734  1.464319734

-0.681203686  1.464319734  1.464319734 -0.681203686 -0.681203686  1.464319734  1.464319734 -0.681203686

-0.681203686 -0.681203686 -0.681203686 -0.681203686 -0.681203686  1.464319734 -0.681203686  1.464319734

-0.681203686 -0.681203686 -0.681203686 -0.681203686  1.464319734 -0.681203686 -0.681203686  1.464319734

-0.681203686 -0.681203686 -0.681203686 -0.681203686 -0.681203686 -0.681203686 -0.681203686 -0.681203686

-0.681203686 -0.681203686 -0.681203686 -0.681203686 -0.681203686  1.464319734 -0.681203686  1.464319734

-0.681203686 -0.681203686 -0.681203686 -0.681203686  1.464319734 -0.681203686 -0.681203686 -0.681203686
```

```
> train_data <- subset(college, split_data == T)
> test_data <- subset(college, split_data == F)
>
> train_data = scale(train_data[-1])
> test_data = scale(test_data[-1])
>
> df2<-train_data[complete.cases(train_data),]
>
> install.packages('e1071')
Error in install.packages : Updating loaded packages
> library(e1071)
>
> str(df2)
 num [1:400, 1:6] -1.798 0.626 1.838 0.453 -0.586 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:400] "1" "2" "3" "4" ...
  ..$ : chr [1:6] "gre" "gpa" "ses" "Gender_Male" ...
> library(caTools)
> set.seed(2020)
> split_data<- sample.split(college_final["admit"],SplitRatio=3/4)
> #split_data
>
> train_data <- subset(college, split_data == T)
> test_data <- subset(college, split_data == F)
>
> train_data = scale(train_data[-1])
> test_data = scale(test_data[-1])
>
> df2<-train_data[complete.cases(train_data),]
>
> #install.packages('e1071')
> library(e1071)
>
> str(df2)
 num [1:400, 1:6] -1.798 0.626 1.838 0.453 -0.586 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:400] "1" "2" "3" "4" ...
  ..$ : chr [1:6] "gre" "gpa" "ses" "Gender_Male" ...
>
```

```r
train_data = scale(train_data[-1])
test_data = scale(test_data[-1])

df2<-train_data[complete.cases(train_data),]

#install.packages('e1071')
library(e1071)

str(df2)

classifier = svm(formula = df2 ~ .,
                 data = train_data,
                 type = 'C-classification',
                 kernel = 'linear')

dat = data.frame(train_data, y = as.factor(train_data["admit"]))
svmfit = svm(train_data["admit"] ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
#print(svmfit)


# Predicting the Test set results
y_pred = predict(classifier, newdata = test_data[-1])

cm = table(test_data[, -1], y_pred)

cm
acc = sum(diag(cm))/sum(cm)
acc

```

193:4 (Top Level)

Console Terminal

~/

```
> cm
   0  1
0 73  5
1  7 15
> acc = sum(diag(cm))/sum(cm)
> acc
[1] 0.88
>
```

Accuracy Score Calculations/Analysis:

- Logistic Regression: 0.66 or 66%
- Decision Tree: 0.83 or 83%
- SVM: 0.88 or 88%

So, clearly SVM gives the best accuracy score of 88%. Hence, I would have used this model to deploy the business requirement mentioned in the business scenario.