COLLEGE ADMISSION

Project Report

Aparna Nair

aparna.nair.vijay@gmail.com

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. 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 analyse the same.

- **Domain:** Education

Objectives

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)
2. Find outliers (if any, then perform outlier treatment)
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.
5. Normalize the data if not normally distributed.
6. Use variable reduction techniques to identify significant variables.
7. Run logistic model to determine the factors that influence the admission process of a student (Drop insignificant variables)
8. Calculate the accuracy of the model and run validation techniques.
9. Try other modelling techniques like decision tree and SVM and select a champion model
10. Determine the accuracy rates for each kind of model
11. Select the most accurate model
12. 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:

GRE	Categorized
0-440	Low
440-580	Medium
580+	High

DATASET DESCRIPTION

❖ Dataset has 400 observations of 7 variables

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

Statistical algorithm execution – Rcode and outputs

❖ Importing the Data Set and analysing the data

```r
#College Admission

#Importing the dataset and analyzing the data
library(dplyr)
setwd("C:/Users/CON_AVIJAY02/Documents/R/WD")
College_Admission= read.csv("College_admission.csv")

View(College_Admission)
str(College_Admission)

```

Predictive

❖ Objective 1-Find the missing values. (if any, perform missing value treatment)

```r
#•  Find the missing values. (if any, perform missing value treatment)
summary(College_Admission)

#No missing value found

```

❖ Output

```
> summary(College_Admission)
     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  
> 
```

No missing values found in the Dataset.

❖ Objective 2 - Find outliers (if any, then perform outlier treatment)

```r
#Find outliers (if any, then perform outlier treatment)


LT=mean(College_Admission$gre)-2*sd(College_Admission$gre)
UT=mean(College_Admission$gre)+2*sd(College_Admission$gre)

LT
UT
#threshold

College_Admission$gre=ifelse(College_Admission$gre<LT,LT,College_Admission$gre)
```

❖ Output

```
> LT=mean(College_Admission$gre)-2*sd(College_Admission$gre)
> UT=mean(College_Admission$gre)+2*sd(College_Admission$gre)
> LT
[1] 356.6669
> UT
[1] 818.7331
>
```

```
> sum(College_Admission$gre<LT)
[1] 8
>
```

❖ Outliers present below lower threshold value for GRE attribute

```
#No Outlier present above the upper threshold
sum(College_Admission$gre>UT)
```

❖ No outliers present above the upper threshold value for GRE attribute

```
> sum(College_Admission$gre>UT)
[1] 0
>
```

```
# Outlier present below the lower threshold
sum(College_Admission$gre<LT)
```

```
#Replacing the outlier with Lower Threshold value
College_Admission$gre=ifelse(College_Admission$gre<LT,LT,College_Admission$gre)
```

- Objective 4 - Find the structure of the data set and if required, transform the numeric data type to factor and vice-versa

```
#•  Find the structure of the data set and if required, transform the numeric data type to factor and vice-versa.
str(College_Admission)
```

- Output

```
> str(College_Admission)
'data.frame':   400 obs. of  7 variables:
 $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre        : num  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 ...
>
```

- Numeric variables in GRE and GPA transformed in to Factors using cut function

```
#Transforming numeric variable gre in to Factors
College_Admission$gregroup=cut(College_Admission$gre,breaks=c(200,400,600,800),labels = c("200-400","401-600","601-800"))


#Transforming numeric variable gpa in to Factors
College_Admission$gpagroup=cut(College_Admission$gpa,breaks=c(2.0,2.5,3,3.5,4),labels = c("2.0-2.5","2.6-3.0","3.1-3.5","3.6-4.0"))
```

	admit	gre	gpa	ses	Gender_Male	Race	rank	gregroup	gpagroup
1	0	380.0000	3.61	1	0	3	3	200-400	3.6-4.0
2	1	660.0000	3.67	2	0	2	3	601-800	3.6-4.0
3	1	800.0000	4.00	2	0	2	1	601-800	3.6-4.0
4	1	640.0000	3.19	1	1	2	4	601-800	3.1-3.5
5	0	520.0000	2.93	3	1	2	4	401-600	2.6-3.0
6	1	760.0000	3.00	2	1	1	2	601-800	2.6-3.0
7	1	560.0000	2.98	2	1	2	1	401-600	2.6-3.0
8	0	400.0000	3.08	2	0	2	2	200-400	3.1-3.5
9	1	540.0000	3.39	1	1	1	3	401-600	3.1-3.5
10	0	700.0000	3.92	1	0	2	2	601-800	3.6-4.0
11	0	800.0000	4.00	1	1	1	4	601-800	3.6-4.0
12	0	440.0000	3.22	3	0	2	1	401-600	3.1-3.5
13	1	760.0000	4.00	3	1	2	1	601-800	3.6-4.0
14	0	700.0000	3.08	2	0	2	2	601-800	3.1-3.5
15	1	700.0000	4.00	2	1	1	1	601-800	3.6-4.0
16	0	480.0000	3.44	3	0	1	3	401-600	3.1-3.5
17	0	780.0000	3.87	2	0	3	4	601-800	3.6-4.0
18	0	360.0000	2.56	3	1	3	3	200-400	2.6-3.0
19	0	800.0000	3.75	1	1	3	2	601-800	3.6-4.0
20	1	540.0000	3.81	1	0	3	1	401-600	3.6-4.0

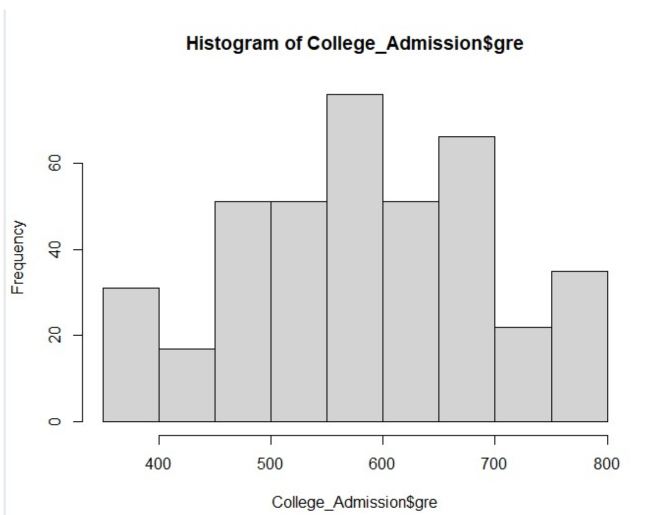
```r

#Transforming numeric variable gpa in to Factors
College_Admission$gpagroup=cut(College_Admission$gpa,breaks=c(2.0,2.5,3,3.5,4),labels = c("2.0-2.5","2.6-3.0","3.1-3.5","3.6-4.0"))

#Removing gre and gpa column
Admisson= College_Admission[,-c(2,3)]
View(Admisson)
```

- ❖ Removing GRE and GPA column and storing in a new dataframe

- ❖ Objective 5 - Find whether the data is normally distributed or not. Use the plot to determine the same.
- ❖ Objective 6 - Normalize the data if not normally distributed.

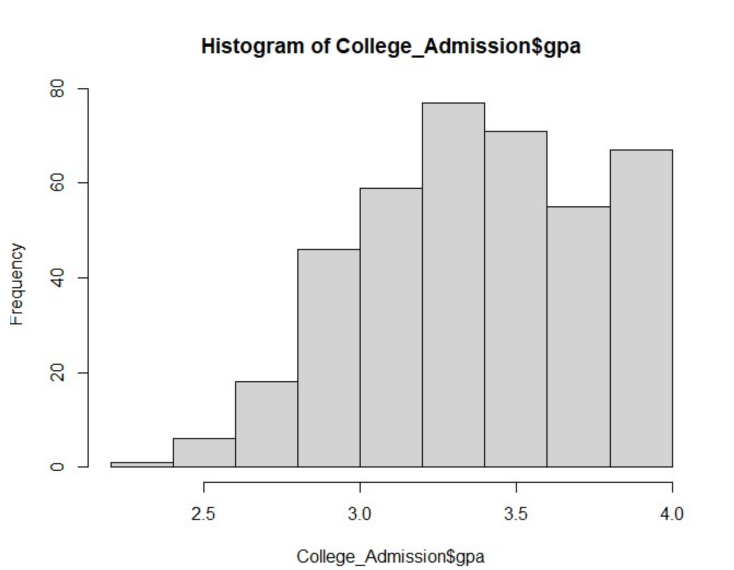
Plotted a histogram and density plot for GRE and GPA

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

library(ggplot2)

hist(College_Admission$gre)
```



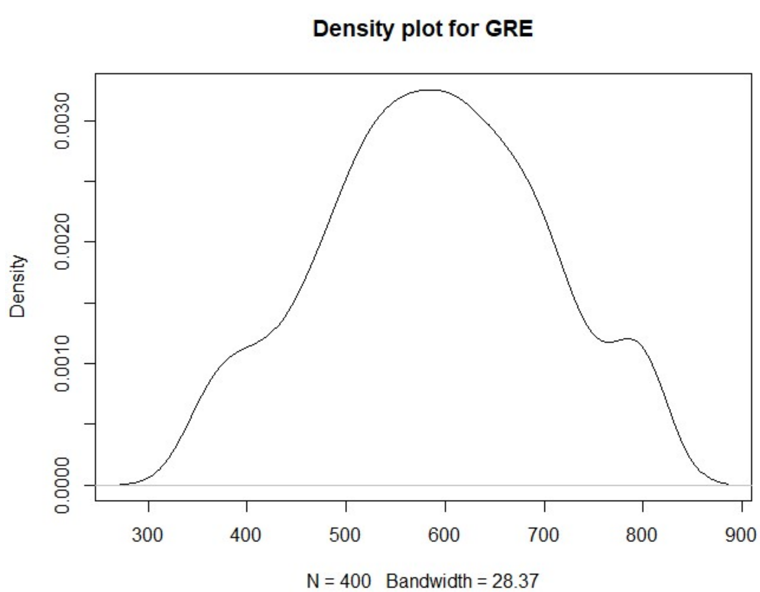
- ❖ GRE attribute has a nearly normal distribution

```r
hist(College_Admission$gpa)
```



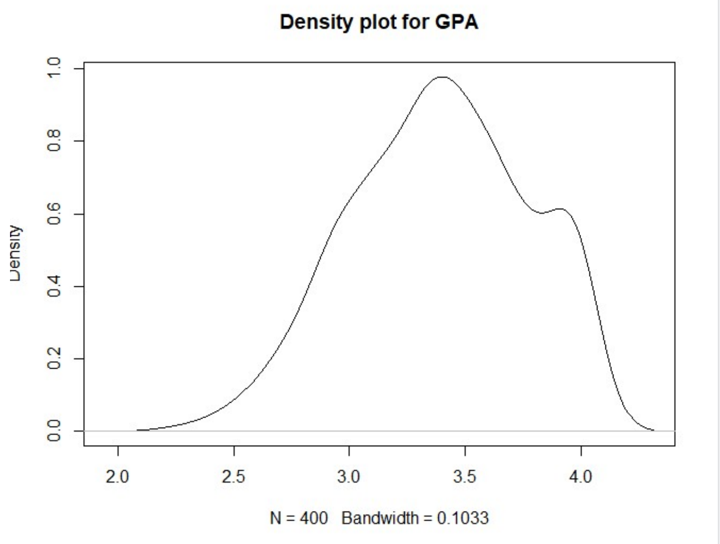
❖ GPA attribute has a skewed distribution
Density plot for GRE and GPA attribute

```
hist(College_Admission$gpa)

#density plot
plot(density(College_Admission$gre),main="Density plot for GRE")

```



Almost normal distribution for GRE

```
plot(density(College_Admission$gpa),main="Density plot for GPA")
```



Skewed distribution (negative – skewed) for GPA attribute

Normalize data log function was used

```
log(College_Admission)
```

- ❖ Objective 7 - Use variable reduction techniques to identify significant variables.
- ❖ Objective 8- Run logistic model to determine the factors that influence the admission process of a student (Drop insignificant variables)

```
sum(Admisson$admit)
#127 admitted
```

```
> sum(Admisson$admit)
[1] 127
>
```

Out of 400 applicants ,only 127 applicants got admission

Splitting the data set in to 70:30 ratio for Training and Test dataset

```r
# splitting data set  in 70:30 Training and Test data set

set.seed(2.2)
inTrain=createDataPartition(College_Admission$admit, p=0.7,list=FALSE)

Training=College_Admission[inTrain,]
Testing=College_Admission[-inTrain,]

sum(Training$admit)
400*0.7
sum(Testing$admit)
400*0.3
fit0=glm(admit~ ., data=Training,family=binomial(link="logit"))

summary(fit0)
```

Fit0 – logistic regression model for the dependent variable Admit

Output

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7266  -0.8921  -0.6317   1.1867   2.2596

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.722552   1.391079  -1.957   0.0503 .
gre          0.002364   0.001328   1.780   0.0751 .
gpa          0.871306   0.386949   2.252   0.0243 *
ses         -0.254487   0.168048  -1.514   0.1299
Gender_Male -0.156946   0.273692  -0.573   0.5663
Race        -0.131961   0.170614  -0.773   0.4393
rank        -0.639996   0.157302  -4.069 4.73e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 354.57  on 279  degrees of freedom
Residual deviance: 321.28  on 273  degrees of freedom
AIC: 335.28

Number of Fisher Scoring iterations: 4
```

Removing the variables with p value less than 0.1 using step AIC function

```r

# aim is to eliminate variables with  p value >0.1 to get more or equal to 90% confidence level
library(MASS)
fitA=stepAIC(fit0, direction = "both")
fitA
summary(fitA)

# p value for ses >0.1, removing ses. Race and gender removed by step AIC method
```

```
Start:  AIC=335.28
admit ~ gre + gpa + ses + Gender_Male + Race + rank

              Df Deviance    AIC
- Gender_Male  1   321.61 333.61
- Race         1   321.88 333.88
<none>             321.28 335.28
- ses          1   323.60 335.60
- gre          1   324.50 336.50
- gpa          1   326.48 338.48
- rank         1   339.44 351.44

Step:  AIC=333.61
admit ~ gre + gpa + ses + Race + rank

              Df Deviance    AIC
- Race         1   322.13 332.13
<none>             321.61 333.61
- ses          1   323.89 333.89
- gre          1   324.80 334.80
+ Gender_Male  1   321.28 335.28
- gpa          1   326.76 336.76
- rank         1   339.64 349.64

Step:  AIC=332.13
admit ~ gre + gpa + ses + rank

              Df Deviance    AIC
<none>             322.13 332.13
- ses          1   324.36 332.36
- gre          1   325.43 333.43
+ Race         1   321.61 333.61
+ Gender_Male  1   321.88 333.88
- gpa          1   327.07 335.07
- rank         1   340.40 348.40
> step

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

Coefficients:
(Intercept)          gre          gpa          ses         rank
  -2.981857     0.002386     0.844963    -0.248817    -0.640158

Degrees of Freedom: 279 Total (i.e. Null);  275 Residual
Null Deviance:      354.6
Residual Deviance: 322.1        AIC: 332.1
```

Race and Gender_Male attribute were removed by stepAIC function(model with lowest AIC value selected)

Summary of FitA model

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7050  -0.8915  -0.6285   1.1680   2.2674

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.983378   1.361189  -2.192   0.0284 *
gre          0.002370   0.001332   1.780   0.0751 .
gpa          0.848257   0.384251   2.208   0.0273 *
ses         -0.248506   0.167768  -1.481   0.1385
rank        -0.640632   0.156788  -4.086 4.39e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 354.57  on 279  degrees of freedom
Residual deviance: 322.21  on 275  degrees of freedom
AIC: 332.21

Number of Fisher Scoring iterations: 3
```

Since the p value for ses >0.1 in fitA, removed ses attribute in fitB model

```
# p value for ses >0.1, removing ses. Race and gender removed by step AIC method


fitB=update(fitA, .~. -ses, data=Training)
summary(fitB)
```

```
> fitB=update(fitA, .~. -ses, data=Training)
> summary(fitB)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5733  -0.8899  -0.6430   1.1552   2.1942

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.265404   1.347502  -2.423   0.0154 *
gre          0.002396   0.001331   1.800   0.0719 .
gpa          0.768260   0.378792   2.028   0.0425 *
rank        -0.621786   0.155408  -4.001 6.31e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 354.57  on 279  degrees of freedom
Residual deviance: 324.43  on 276  degrees of freedom
AIC: 332.43

Number of Fisher Scoring iterations: 3
```

Significant variables -gre,gpa,rank

Insignificant variable -gender,race,ses

Admission of a student is dependent on GRE,GPA and Rank

- Objective 8- Calculate the accuracy of the model and run validation techniques.
- Confusion matrix and ROC curve used to check the accuracy of the model
  Validating the accuracy of fitB model(significant variables-gre,gpa and rank)

```
#predict the accuracy of model

fitB$coefficients
names(fitB$coefficients)

names(fitB)
```

```
> fitB$coefficients
 (Intercept)          gre          gpa         rank
-3.263901163  0.002410174  0.765118700 -0.621339294
> names(fitB$coefficients)
[1] "(Intercept)" "gre"         "gpa"         "rank"
> names(fitB)
 [1] "coefficients"      "residuals"         "fitted.values"     "effects"           "R"                 "rank"              "qr"
 [8] "family"            "linear.predictors" "deviance"          "aic"               "null.deviance"     "iter"              "weights"
[15] "prior.weights"     "df.residual"       "df.null"           "y"                 "converged"         "boundary"          "model"
[22] "call"              "formula"           "terms"             "data"              "offset"            "control"           "method"
[29] "contrasts"         "xlevels"
>
```

```
# probability values output for logistic regression
fitB$fitted.values

```

Predicting the probability values for fitB model by the predict function using Training and Test data set

```
# probability values output for logistic regression
fitB$fitted.values

Pred=predict(fitB, newdata=Training[,-1], type="response")
Pred

Pred_T=predict(fitB, newdata=Testing[,-1], type="response")
Pred_T # predictions for test data
```

Confusion matrix for fit B model Training data set

```
# classifying probability values
Pred1=ifelse(Pred<0.5,0,1)
View(Pred1)

# crosstab confusion matrix

library(e1071)
TrainingResult=table(Training$admit,Pred1,dnn=list('actual','predicted'))
TrainingResult
```

```

caret::confusionMatrix(TrainingResult)
```

```
> caret::confusionMatrix(TrainingResult,positive='1')
Confusion Matrix and Statistics

      predicted
actual   0   1
     0 172  16
     1  67  25

               Accuracy : 0.7036
                 95% CI : (0.6463, 0.7564)
    No Information Rate : 0.8536
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2174

 Mcnemar's Test P-Value : 4.06e-08

            Sensitivity : 0.60976
            Specificity : 0.71967
         Pos Pred Value : 0.27174
         Neg Pred Value : 0.91489
             Prevalence : 0.14643
         Detection Rate : 0.08929
   Detection Prevalence : 0.32857
      Balanced Accuracy : 0.66471

       'Positive' Class : 1

>
```

Accuracy- 70.36 %,Sensitivity – 60.97% and Specificity – 71.967%

Confusion matrix for fitB Test data set

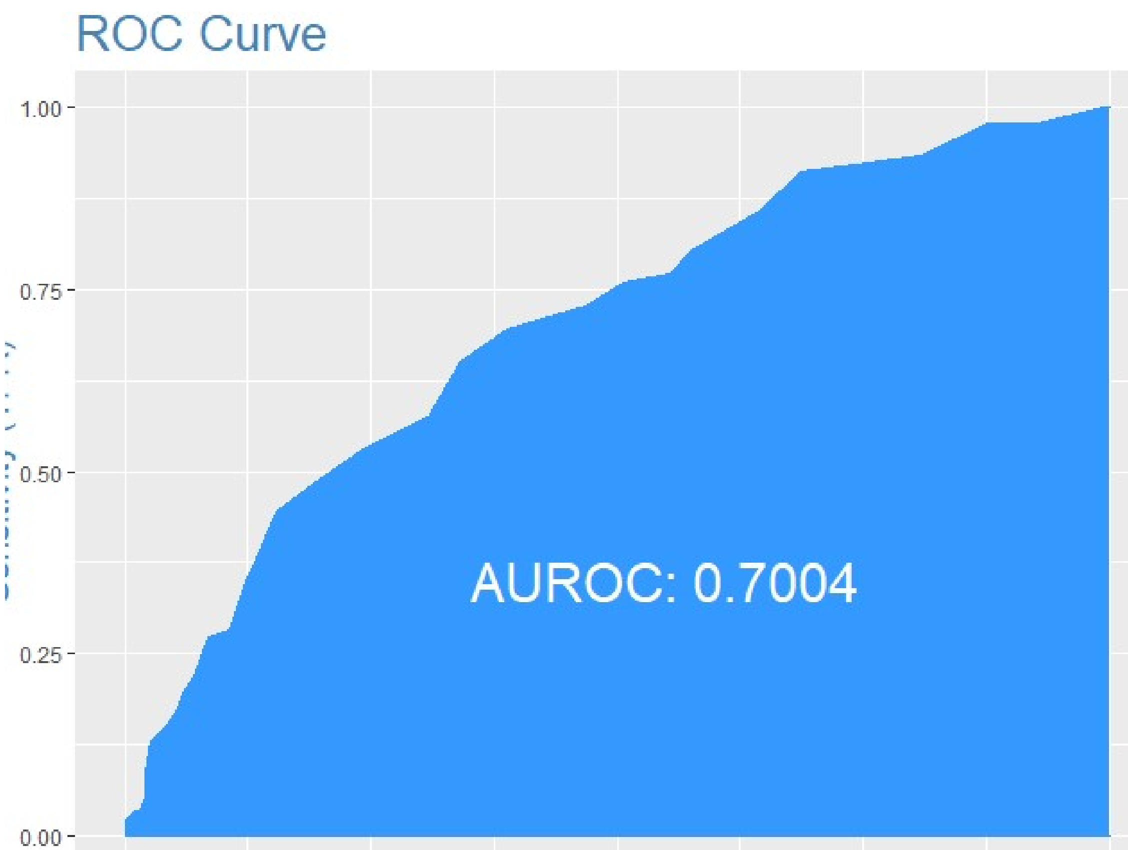
```
# classifying probability values
Pred2=ifelse(Pred_T<0.5,0,1)
View(Pred2)

# crosstab confusion matrix for Test data set

library(e1071)
TestingResult=table(Testing$admit,Pred2,dnn=list('actual','predicted'))
TestingResult

caret::confusionMatrix(TestingResult,positive='1')
```

```
> caret::confusionMatrix(TestingResult,positive='1')
Confusion Matrix and Statistics

      predicted
actual  0  1
     0 76  9
     1 22 13

               Accuracy : 0.7417
                 95% CI : (0.6538, 0.8172)
    No Information Rate : 0.8167
    P-Value [Acc > NIR] : 0.98458

                  Kappa : 0.2981

 Mcnemar's Test P-Value : 0.03114

            Sensitivity : 0.5909
            Specificity : 0.7755
         Pos Pred Value : 0.3714
         Neg Pred Value : 0.8941
             Prevalence : 0.1833
         Detection Rate : 0.1083
   Detection Prevalence : 0.2917
      Balanced Accuracy : 0.6832

       'Positive' Class : 1

>
```

Accuracy-74.17%,Sensitivity – 59.09%,Specificity-77.55%

ROC curve for fitB

```
#Area under curve (AOC)/ROC Curve - more the area better model ROC>0.5
library(survey)
library(survival)
library(InformationValue)
#dev.off()
?plotROC
plotROC(actuals=Training$admit,predictedScores=as.numeric(fitted(fitB)))
```



ROC-70.04%

Accuracy of Fit A model (significant variables – gre,gpa,rank and ses)

Confusion matrix for Training dataset for fitA model

```
# Confusion matrix and ROC curve values for FitA

Pred_fitA_Training=predict(fitA,Training[,-1],type="response")
Pred_fitATrain= ifelse(Pred_fitA_Training<0.5,0,1)

Pred_fitA_Test=predict(fitA,Testing[,-1],type="response")
Pred_fitAtest= ifelse(Pred_fitA_Test<0.5,0,1)

#crosstab confusion matrix for Testing data set for fitA
TrainingResult=table(Training$admit,Pred_fitATrain,dnn=list('actual','predicted'))
caret::confusionMatrix(TrainingResult,positive='1')
```

```
Confusion Matrix and Statistics

      predicted
actual   0   1
     0 170  18
     1  69  23

               Accuracy : 0.6893
                 95% CI : (0.6315, 0.743)
    No Information Rate : 0.8536
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1797

 Mcnemar's Test P-Value : 8.296e-08

            Sensitivity : 0.56098
            Specificity : 0.71130
         Pos Pred Value : 0.25000
         Neg Pred Value : 0.90426
             Prevalence : 0.14643
         Detection Rate : 0.08214
   Detection Prevalence : 0.32857
      Balanced Accuracy : 0.63614

       'Positive' Class : 1

>
```

Accuracy -68.93% ,Sensitivity-56.09% and Specificity – 71.13%

Confusion matrix for fitA model using Test data set

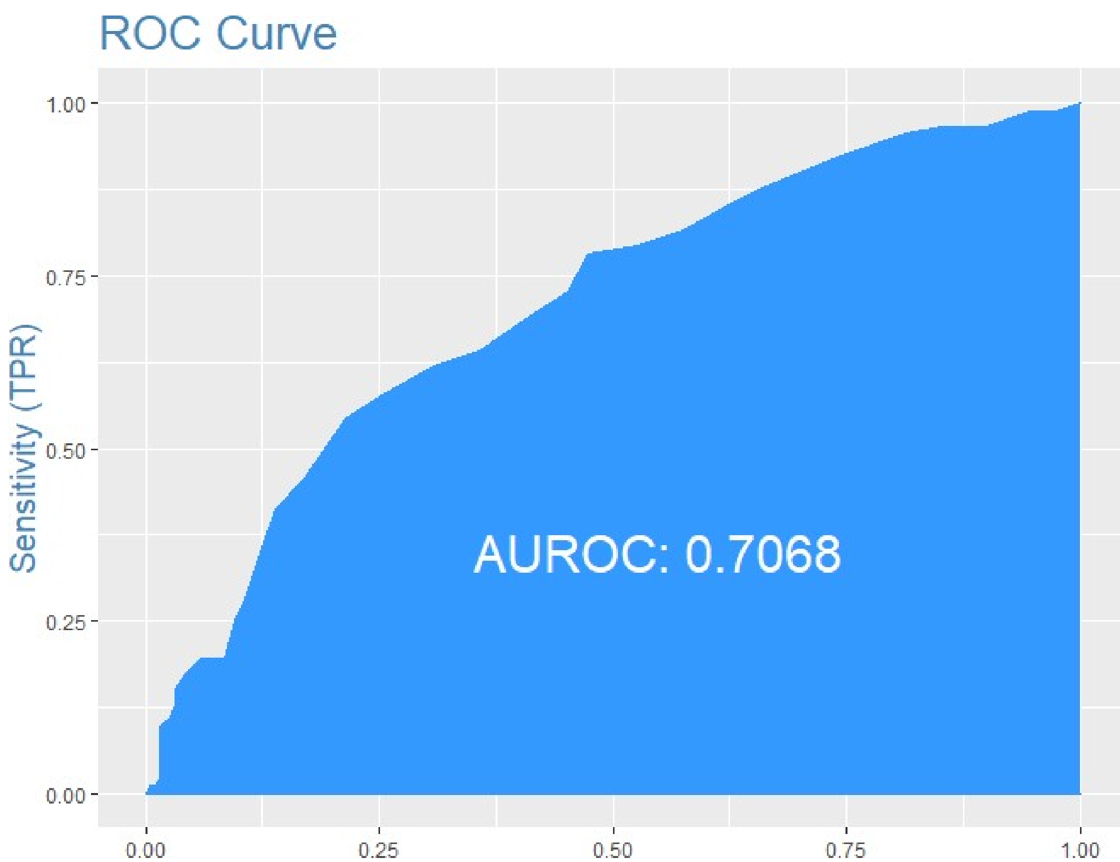
```
#crosstab confusion matrix for Testing data set for fitA
TestingResult=table(Testing$admit,Pred_fitAtest,dnn=list('actual','predicted'))
caret::confusionMatrix(TestingResult,positive='1')
```

```
> caret::confusionMatrix(TestingResult,positive=
Confusion Matrix and Statistics

      predicted
actual  0  1
     0 72 13
     1 21 14

               Accuracy : 0.7167
                 95% CI : (0.6272, 0.7951)
    No Information Rate : 0.775
    P-Value [Acc > NIR] : 0.9464

                  Kappa : 0.2649

 Mcnemar's Test P-Value : 0.2299

            Sensitivity : 0.5185
            Specificity : 0.7742
         Pos Pred Value : 0.4000
         Neg Pred Value : 0.8471
             Prevalence : 0.2250
         Detection Rate : 0.1167
   Detection Prevalence : 0.2917
      Balanced Accuracy : 0.6464

       'Positive' Class : 1

>
```

Accuracy -71.67%,Sensitivity -51.85% and Specificity -77.42%

ROC curve for fitA model

```
#AOC curve for fitA
plotROC(actuals = Training$admit,predictedScores = as.numeric(fitted(fitA)))
```



ROC -70.68%

Summary

Model	Variable	AIC	Res. Dev	max p value	LoC	Train Accu	Test_Accu	AOC	Priority
FitA	~gre,gpa,ses,rank	332.21	322.21	0,13	0,87	68,9	71,67	70,68	2
FitB	~gre,gpa,rank	332.43	324.43	0,07	0,93	70,36	74,17	70,04	1

Model Fit B is better because of higher level of confidence and better accuracy .

- Objective 9 - Try other modelling techniques like decision tree and SVM and select a champion model
- Objective 10 - Determine the accuracy rates for each kind of model
- Objective 12 - Select the most accurate model

Decision Trees

- Data transformation- changing the numeric variables in to factors

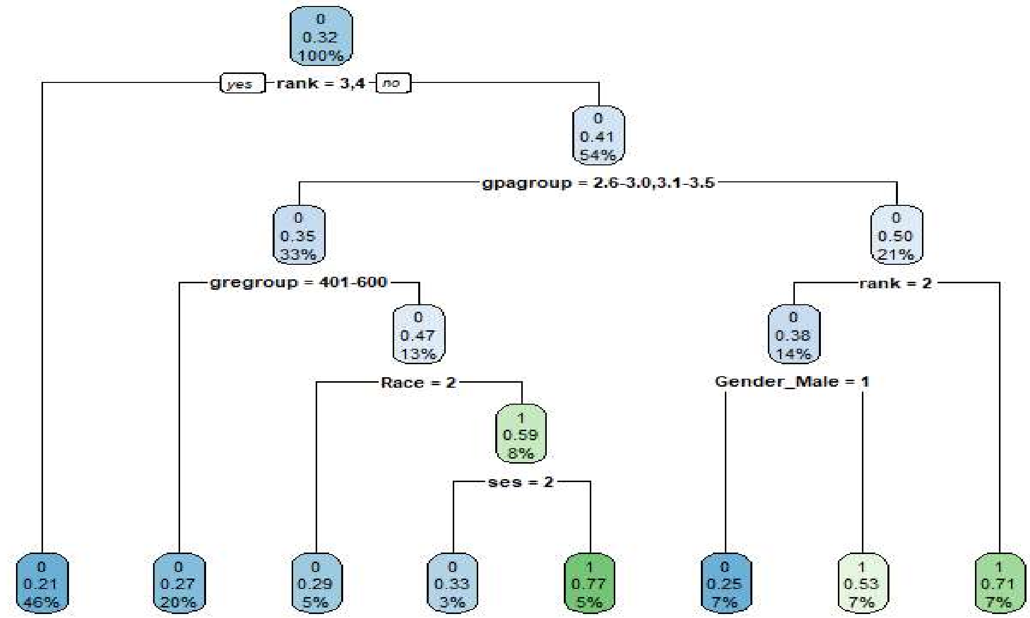
```
# Decision tree

#Transforming numeric variable gre in to Factors
College_Admission$gregroup=cut(College_Admission$gre,breaks=c(200,400,600,800),labels = c("200-400","401-600","601-800"))


#Transforming numeric variable gpa in to Factors
College_Admission$gpagroup=cut(College_Admission$gpa,breaks=c(2.0,2.5,3,3.5,4),labels = c("2.0-2.5","2.6-3.0","3.1-3.5","3.6-4.0"))

#Removing gre and gpa column
Admission= College_Admission[,-c(2,3)]
```

```
#convert variables as factors variables that are specific units
Admission$admit=as.factor(Admission$admit)
Admission$ses=as.factor(Admission$ses)
Admission$Gender_Male=as.factor(Admission$Gender_Male)
Admission$Race=as.factor(Admission$Race)
Admission$rank=as.factor(Admission$rank)
```

Creating Data Partition set in 70:30 ratio for Training and Test data set

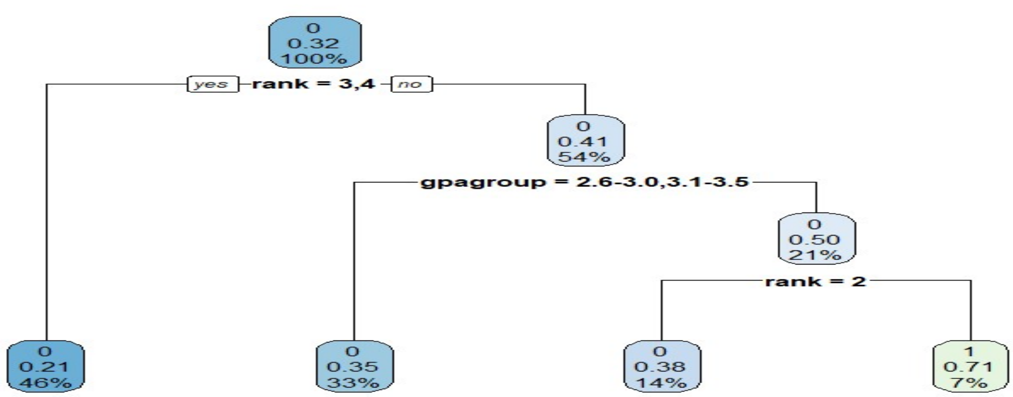
```
library(caret)
set.seed(2.34)
inTrain=createDataPartition(Admission$admit, p=0.7,list=FALSE)

training_DT=Admission[inTrain,]
testing_DT=Admission[-inTrain,]
library(rpart)

library(rpart.plot)


names(training_DT)
fit=rpart(admit ~ .,  method="class",data=training_DT)
rpart.plot(fit)
```

Using rpart function building decision tree model for admit variable



From the decision trees it is clear that admission is dependent on the variabes gpa,rank and gre

```
#Pruning of trees
fit1=rpart(admit~ ., method="class", data=training_DT,
           control=rpart.control(minsplit=50))
rpart.plot(fit1)
```



Validating the accuracy of the model

Confusion matrix for Training set for Decision tree model

```
# Test the accuracy of the model
Pred_Training=predict(fit, newdata=training_DT[,-1],type="class")
Pred_Training
Pred_TrainDT=table(training_DT$admit, Pred_Training, dnn=list("actual","predicted"))

caret::confusionMatrix(Pred_TrainDT,positive='1')
```

```
> caret::confusionMatrix(Pred_TrainDT,positive='1')
Confusion Matrix and Statistics

      predicted
actual   0   1
     0 174  18
     1  54  35

               Accuracy : 0.7438
                 95% CI : (0.6885, 0.7938)
    No Information Rate : 0.8114
    P-Value [Acc > NIR] : 0.9979

                  Kappa : 0.336

 Mcnemar's Test P-Value : 3.711e-05

            Sensitivity : 0.6604
            Specificity : 0.7632
         Pos Pred Value : 0.3933
         Neg Pred Value : 0.9062
             Prevalence : 0.1886
         Detection Rate : 0.1246
   Detection Prevalence : 0.3167
      Balanced Accuracy : 0.7118

       'Positive' Class : 1
```

Accuracy – 74.38%,Sensitivity – 66.04% and Specificity – 76.32%

Confusion matrix for Test data set for Decision tree model

```
# Test the accuracy of the model
Pred_Test=predict(fit, newdata=testing_DT[,-1],type="class")

Pred_TestDT=table(testing_DT$admit, Pred_Test, dnn=list("actual","predicted"))

caret::confusionMatrix(Pred_TestDT,positive='1')
```

```
Confusion Matrix and Statistics

      predicted
actual  0  1
     0 69 12
     1 26 12

               Accuracy : 0.6807
                 95% CI : (0.589, 0.7631)
    No Information Rate : 0.7983
    P-Value [Acc > NIR] : 0.99915

                  Kappa : 0.1858

 Mcnemar's Test P-Value : 0.03496

            Sensitivity : 0.5000
            Specificity : 0.7263
         Pos Pred Value : 0.3158
         Neg Pred Value : 0.8519
             Prevalence : 0.2017
         Detection Rate : 0.1008
   Detection Prevalence : 0.3193
      Balanced Accuracy : 0.6132

       'Positive' Class : 1

>
```

Accuracy -68.07% ,Sensitivity - 50%,Specificity – 72.63%

SVM model

Creating Data Partition in 70:30 ratio for Training and Test Data set for SVM model

```
#SVM
College_Admission$admit=as.factor(College_Admission$admit)
library(caret)
set.seed(2.45)

inTrain=createDataPartition(College_Admission$admit, p=0.7,list=FALSE)

training_SVM=College_Admission[inTrain,]
testing_SVM=College_Admission[-inTrain,]


Fit_SVM=svm(admit~.,training_SVM)
predTr=predict(Fit_SVM,newdata=training_SVM)
caret::confusionMatrix(training_SVM$admit,predTr,positive='1')
```

Confusion Matrix for SVM Model using Training data set

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 186   6
         1  68  21

               Accuracy : 0.737
                 95% CI : (0.681, 0.787)
    No Information Rate : 0.904
    P-Value [Acc > NIR] : 1

                  Kappa : 0.252

 Mcnemar's Test P-Value : 1.33e-12

            Sensitivity : 0.7778
            Specificity : 0.7323
         Pos Pred Value : 0.2360
         Neg Pred Value : 0.9687
             Prevalence : 0.0961
         Detection Rate : 0.0747
   Detection Prevalence : 0.3167
      Balanced Accuracy : 0.7550

       'Positive' Class : 1

>
```

Accuracy -73.67% , Sensitivity – 77.77% ,Specificity – 73.22%

Confusion matrix for Test data set for SVM model

```
predictT=predict(Fit_SVM,newdata=testing_SVM)
caret::confusionMatrix(testing_SVM$admit,predictT,positive='1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 72  9
         1 33  5

               Accuracy : 0.647
                 95% CI : (0.554, 0.732)
    No Information Rate : 0.882
    P-Value [Acc > NIR] : 1.000000

                  Kappa : 0.025

 Mcnemar's Test P-Value : 0.000387

            Sensitivity : 0.357
            Specificity : 0.686
         Pos Pred Value : 0.132
         Neg Pred Value : 0.889
             Prevalence : 0.118
         Detection Rate : 0.042
   Detection Prevalence : 0.319
      Balanced Accuracy : 0.521

       'Positive' Class : 1

> 
```

Accuracy – 64.7% , Sensitivity – 35.7% ,Specificity – 68.6%

Summary

Model	Approach	TrainAccu	Test Accuracy	Difference b/w Test Accu and Train Accu	Priority
FitB	Logistic Regression	70,36	74,17	3,81	1
Fit	Decision Tree	74,38	68,07	6,31	2
Fit_SVM	SVM	73,67	64,7	8,97	3

Logistic Regression model has a best Test accuracy and difference between Test and Training accuracy is smallest among the three models. So Logistic Regression is the better model out of these 3 models.

- Objective 12 - Identify other Machine learning or statistical techniques
  Random Forest

```r
#RandomForest
library(randomForest)
?randomForest
F1=randomForest(as.factor(admit)~., data = training_DT) # important to declare dependent variable as a Factor or
#the discrete variable to be converted as Factor


Test_randomforest=predict(F1,testing_DT,type = "vote")# 500 models voted
View(Test_randomforest)

Test=ifelse(Test_randomforest[,2]>0.5,1,0)

table(testing_DT$admit,Test)

library(e1071)
library(caret)
caret::confusionMatrix(table(Test,testing_DT$admit))
?confusionMatrix
```

Confusion matrix for Test data set for Random Forest

```
> caret::confusionMatrix(table(Test,testing_DT$admit))
Confusion Matrix and Statistics


Test  0  1
   0 70 29
   1 11  9

               Accuracy : 0.6639
                 95% CI : (0.5715, 0.7478)
    No Information Rate : 0.6807
    P-Value [Acc > NIR] : 0.69157

                  Kappa : 0.1156

 Mcnemar's Test P-Value : 0.00719

            Sensitivity : 0.8642
            Specificity : 0.2368
         Pos Pred Value : 0.7071
         Neg Pred Value : 0.4500
             Prevalence : 0.6807
         Detection Rate : 0.5882
   Detection Prevalence : 0.8319
      Balanced Accuracy : 0.5505

       'Positive' Class : 0
```

Accuracy - 66.39% ,Sensitivity – 86.42%,Specificity -23.68%

Confusion matrix for Training Data Set for Random Forest model

```
Train_randomforest=predict(F1,training_DT,type = "vote")# 500 models voted
View(Train_randomforest)

Train=ifelse(Train_randomforest[,2]>0.5,1,0)

table(training_DT$admit,Train)

library(e1071)
library(caret)
caret::confusionMatrix(table(Train,training_DT$admit))
?confusionMatrix
```

```
> caret::confusionMatrix(table(Train,training_DT$admit))
Confusion Matrix and Statistics


Train   0   1
    0 191  39
    1   1  50

               Accuracy : 0.8577
                 95% CI : (0.8112, 0.8963)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 1.327e-11

                  Kappa : 0.6286

 Mcnemar's Test P-Value : 4.909e-09

            Sensitivity : 0.9948
            Specificity : 0.5618
         Pos Pred Value : 0.8304
         Neg Pred Value : 0.9804
             Prevalence : 0.6833
         Detection Rate : 0.6797
   Detection Prevalence : 0.8185
      Balanced Accuracy : 0.7783

       'Positive' Class : 0
```

Accuracy – 85.77%, Sensitivity – 99.48% ,Specificity – 56.18%

Naive Bayes

```
#Naive Bayes
Fit_NB=naiveBayes(admit~.,training_SVM)
print(Fit_NB)
Pred_NB_train=predict(Fit_NB,newdata =training_SVM)

caret::confusionMatrix(training_SVM$admit,Pred_NB_train)
```

Confusion matrix for Naive Bayes model Training data set

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 175  17
         1  68  21

               Accuracy : 0.6975
                 95% CI : (0.6401, 0.7507)
    No Information Rate : 0.8648
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1742

 Mcnemar's Test P-Value : 5.852e-08

            Sensitivity : 0.7202
            Specificity : 0.5526
         Pos Pred Value : 0.9115
         Neg Pred Value : 0.2360
             Prevalence : 0.8648
         Detection Rate : 0.6228
   Detection Prevalence : 0.6833
      Balanced Accuracy : 0.6364

       'Positive' Class : 0
```

Accuracy – 69.75%,Sensitivity -72.02%,Specificity – 55.26%

Confusion matrix for Test Data Set

```
#Naive Bayes
Fit_NB=naiveBayes(admit~.,training_SVM)
print(Fit_NB)
Pred_NB_test=predict(Fit_NB,newdata =testing_SVM)

caret::confusionMatrix(testing_SVM$admit,Pred_NB_test)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 72  9
         1 28 10

               Accuracy : 0.6891
                 95% CI : (0.5977, 0.7707)
    No Information Rate : 0.8403
    P-Value [Acc > NIR] : 0.999988

                  Kappa : 0.1753

 Mcnemar's Test P-Value : 0.003085

            Sensitivity : 0.7200
            Specificity : 0.5263
         Pos Pred Value : 0.8889
         Neg Pred Value : 0.2632
             Prevalence : 0.8403
         Detection Rate : 0.6050
   Detection Prevalence : 0.6807
      Balanced Accuracy : 0.6232

       'Positive' Class : 0
```

Accuracy – 68.91%,Sensitivity – 72%,Specificity – 52.63%

Summary

Model	Approach	TrainAccu	Test Accuracy	Difference b/w Test Accu and Train Accu	Priority
FitB	Logistic Regression	70,36	74,17	3,81	1
Fit	Decision Tree	74,38	68,07	6,31	3
Fit_SVM	SVM	73,67	64,7	8,97	4
F1	Random Forest	85,77	66,39	19,38	5
Fit_NB	Naive Bayes	69,75	68,91	0,84	2

Logistic Regression is the preferred model as it has a high Test accuracy.

Descriptive

- Objective 1 - 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:

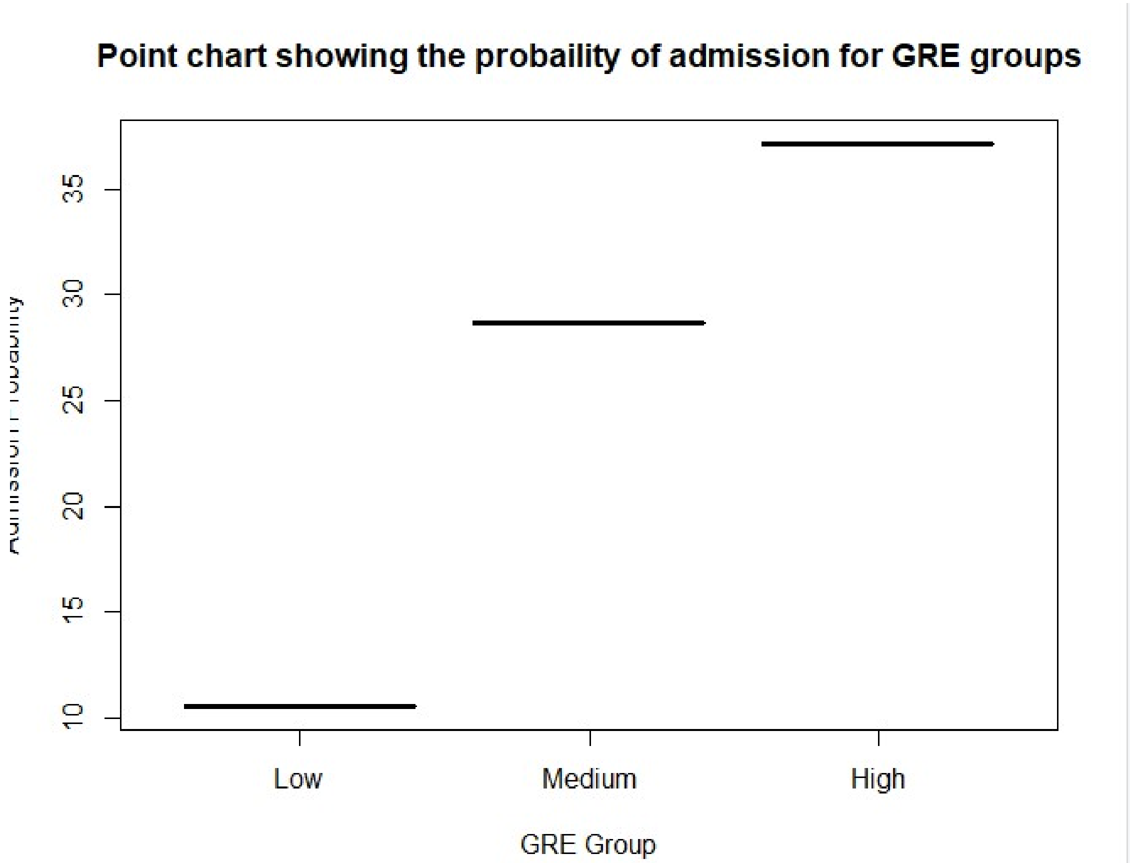
GRE	Categorized
0-440	Low
440-580	Medium
580+	High

```
College_Admission$gregroup=cut(College_Admission$gre,breaks=c(0,439,579,800),labels = c("Low","Medium","High"))
```

```
t=table (College_Admission$admit,College_Admission$gregroup)

sum=aggregate(admit~gregroup,data=College_Admission,sum)
len=aggregate(admit~gregroup,data=College_Admission,length)
  gre_admission=cbind(sum,l=len[,2])
  gre_admission$p=(gre_admission$admit/gre_admission$l)*100
 gre_admission
plot(gre_admission$gregroup,gre_admission$p)
```

```
> table (College_Admission$admit,College_Admission$gregroup)

    Low Medium High
  0  34     97  142
  1   4     39   84

> gre_admission=cbind(sum,l=len[,2])
> gre_admission
  gregroup admit   l
1      Low     4  38
2   Medium    39 136
3     High    84 226
```

```
>   gre_admission$p=(gre_admission$admit/gre_admission$l)*100
>  gre_admission
  gregroup admit   l        p
1      Low     4  38 10.52632
2   Medium    39 136 28.67647
3     High    84 226 37.16814
> plot(gre_admission$gregroup,gre_admission$p)
```

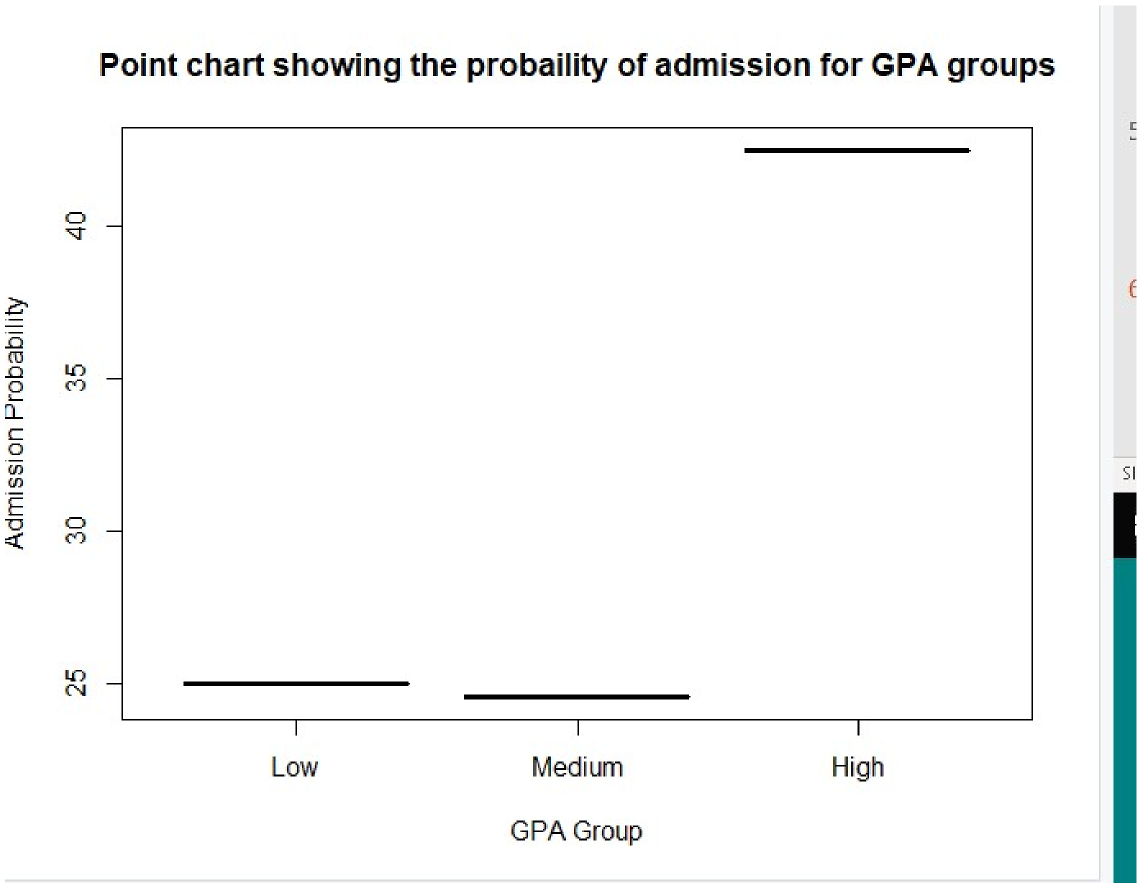


Point chart for GPA

```
College_Admission$gpagroup=cut(College_Admission$gpa,breaks=c(0,2.49,3.49,4.00),labels = c("Low","Medium","High"))
```

```
library(dplyr)
sumgpa=aggregate(admit~gpagroup,data=College_Admission,sum)
len=aggregate(admit~gpagroup,data=College_Admission,length)
gpa_admission=cbind(sumgpa,l=len[,2])
gpa_admission$p=(gpa_admission$admit/gpa_admission$l)*100
gpa_admission
plot(gpa_admission$gpagroup,gpa_admission$p,main= "Point chart showing the probaility of admission for GPA groups",xlab="GPA Group",ylab="Admi
```

```
> gpa_admission
  gpagroup admit   l
1      Low     1   4
2   Medium    58 236
3     High    68 160
> gpa_admission$p=(gpa_admission$admit/gpa_admission$l)*100
> gpa_admission
  gpagroup admit   l        p
1      Low     1   4 25.00000
2   Medium    58 236 24.57627
3     High    68 160 42.50000
> plot(gpa_admission$gpagroup,gpa_admission$p,main= "Point chart showing the probaility of admissi
```



Result

Analysis Task

- ❖ Objective 1 - Find the missing value

No missing values found(Refer page 6)

- ❖ Objective 2 - Find outliers (if any, then perform outlier treatment)

Outliers present below lower threshold value for GRE attribute which was removed(Refer page 7)

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

Numeric variables in GRE and GPA attributes were transformed in to Factor(Refer page 8)

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

GRE attribute has a nearly normal distribution and GPA attribute has a negative skewed distribution(Refer pages 9-11)

- ❖ Objective 5 - Normalize the data if not normally distributed.

Log function was used to normalize the data(Refer page 11)

- ❖ Objective 6 - Use variable reduction techniques to identify significant variables.

Significant variables are gre,gpa and rank(Refer pages 11-14)

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

Factors that influence the admission process of a student are GRE ,GPA and Rank as per logistic regression model (Refer pages 11-14)

- ❖ Objective 8 - Calculate the accuracy of the model and run validation techniques.

Model Fit B is better because of higher level of confidence and better accuracy . (Refer pages 14 -20)

- ❖ Objective 9 - Try other modelling techniques like decision tree and SVM and select a champion model
- ❖ Objective 10 - Determine the accuracy rates for each kind of model
- ❖ Objective 11 - Select the most accurate model

  Decision tree and SVM approach was used .Confusion matrix was used to find the accuracy of the models Logistic regression was the better model because of better Test accuracy and lower difference between the Test and Training accuracy (Refer pages 20-25)
- ❖ Objective 12 - Identify other Machine learning or statistical techniques

  Random Forest and Naive Bayes techniques were used (Refer pages 26 29)

Descriptive Task

- Objective 1 - Categorize the average of grade point into High, Medium, and Low (with admission probability percentages) and plot it on a point chart.
  Point chart has been plotted for GPA and GRE (Refer page (30-32) .It was found that applicants with high GRE and GPA have higher probability of getting admission