College Admission

Andrei Enescu

10/23/2020

DESCRIPTION

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 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

Dataset Description:

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

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 modeling 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:

13. 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

These are the libraries I used:

```
library(rio)
library(dplyr)
library(ggplot2)
library(ggpubr)
library(lattice)
library(caret)
library(pROC)
library(rpart)
library(rpart.plot)
library(e1071)
library(class)
library(naivebayes)
library(randomForest)
```

Importing the data using RIO package

```
CollegeDF <- import("College_admission.csv")
```

Predictive:

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

```
which(is.na(CollegeDF))
```

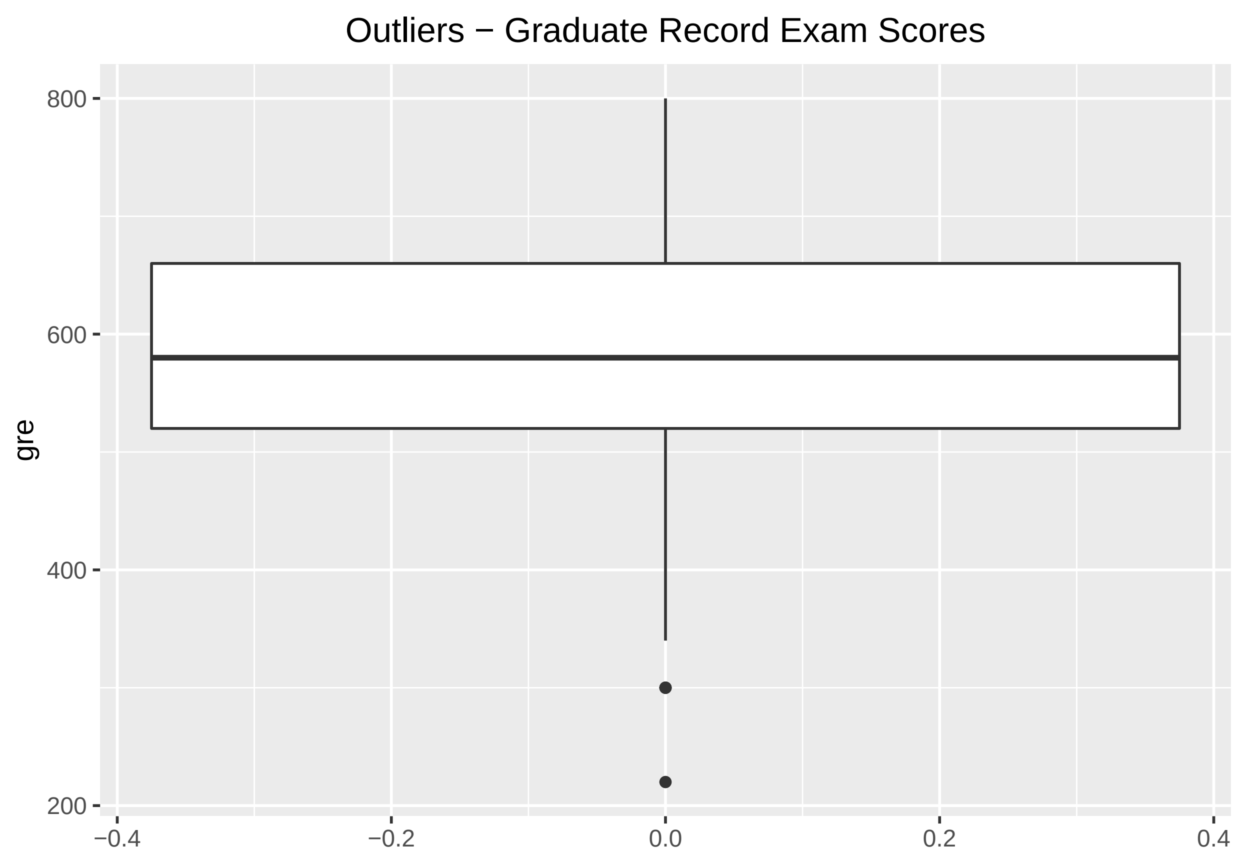
```
## integer(0)
```

Comments:

There are no missing values

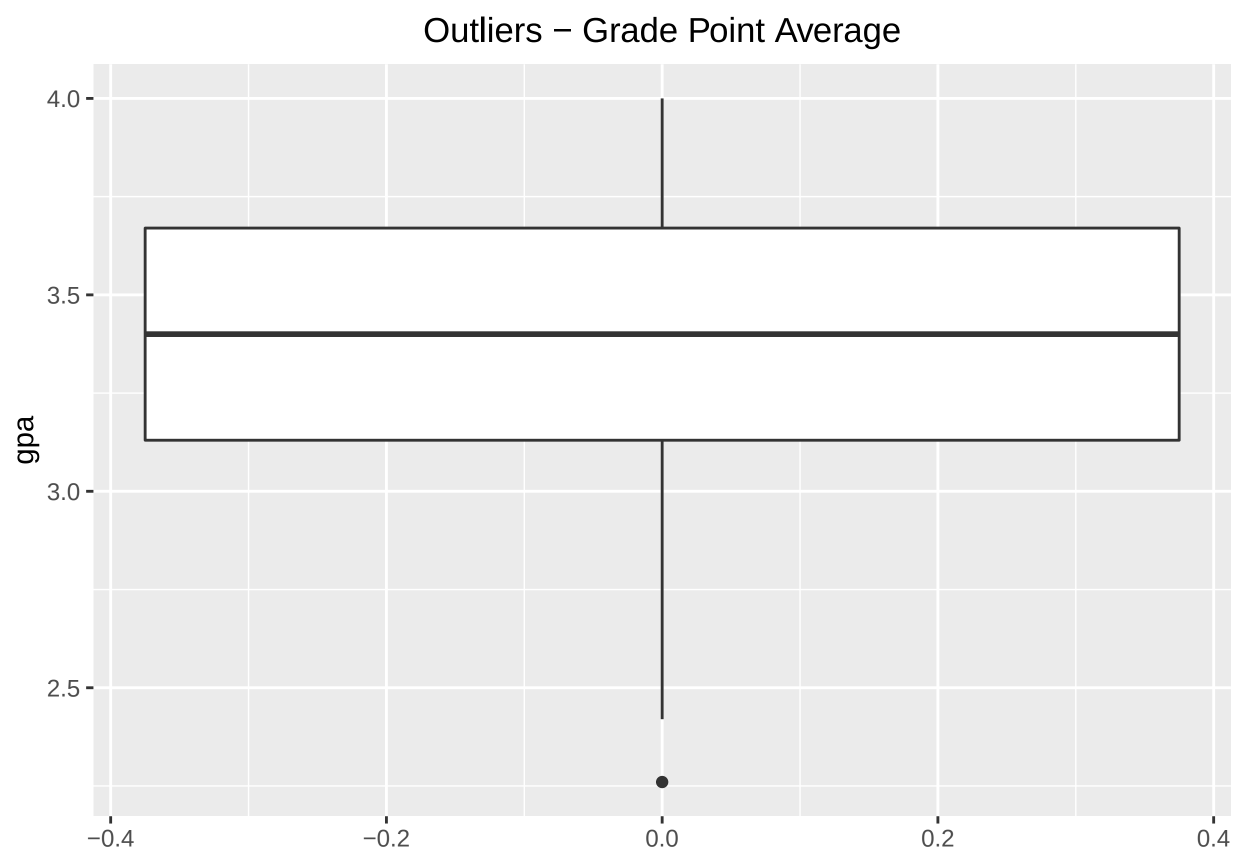
2. Find outliers (if any, then perform outlier treatment)

```
ggplot(CollegeDF,
       aes(y = gre, label = gre)) +
  geom_boxplot() +
  labs(title = "Outliers - Graduate Record Exam Scores") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
CollegeDF <- CollegeDF[CollegeDF$gre > 305, ]

ggplot(CollegeDF,
       aes(y = gpa, label = gpa)) +
  geom_boxplot() +
  labs(title = "Outliers - Grade Point Average") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
CollegeDF <- CollegeDF[CollegeDF$gpa > 2.5, ]
```

Comments:

I used boxplot to check for outliers and removed them.

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

```
summary(CollegeDF)
```

```
##      admit             gre             gpa             ses       
##  Min.   :0.0000   Min.   :340.0   Min.   :2.520   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.140   1st Qu.:1.000  
##  Median :0.0000   Median :590.0   Median :3.400   Median :2.000  
##  Mean   :0.3189   Mean   :591.5   Mean   :3.405   Mean   :1.997  
##  3rd Qu.:1.0000   3rd Qu.:665.0   3rd Qu.:3.670   3rd Qu.:3.000  
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :3.000  
##   Gender_Male          Race            rank      
##  Min.   :0.0000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:2.000  
##  Median :0.0000   Median :2.000   Median :2.000
##  Mean   :0.4694   Mean   :1.964   Mean   :2.477
##  3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :3.000   Max.   :4.000
```

```
str(CollegeDF)
```

```
## 'data.frame':    392 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 ...
```

```
CollegeDF$admit<-as.factor(CollegeDF$admit)
CollegeDF$ses<-as.factor(CollegeDF$ses)
CollegeDF$Gender_Male<-as.factor(CollegeDF$Gender_Male)
CollegeDF$Race<-as.factor(CollegeDF$Race)
CollegeDF$rank<-as.factor(CollegeDF$rank)
str(CollegeDF)
```

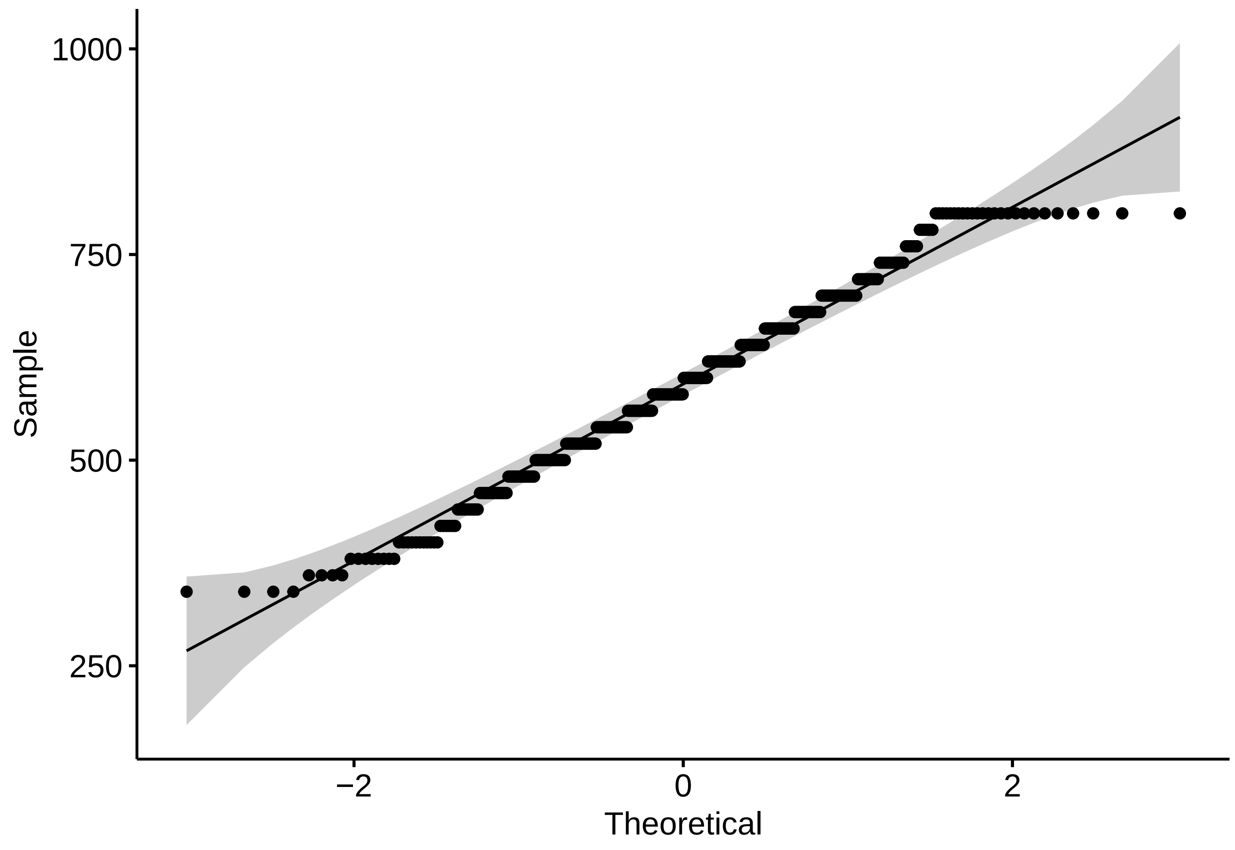
```
## 'data.frame':    392 obs. of  7 variables:
##  $ admit      : Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
##  $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ ses        : Factor w/ 3 levels "1","2","3": 1 2 2 1 3 2 2 2 1 1 ...
##  $ Gender_Male: Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 2 1 ...
##  $ Race       : Factor w/ 3 levels "1","2","3": 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 ...
```

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

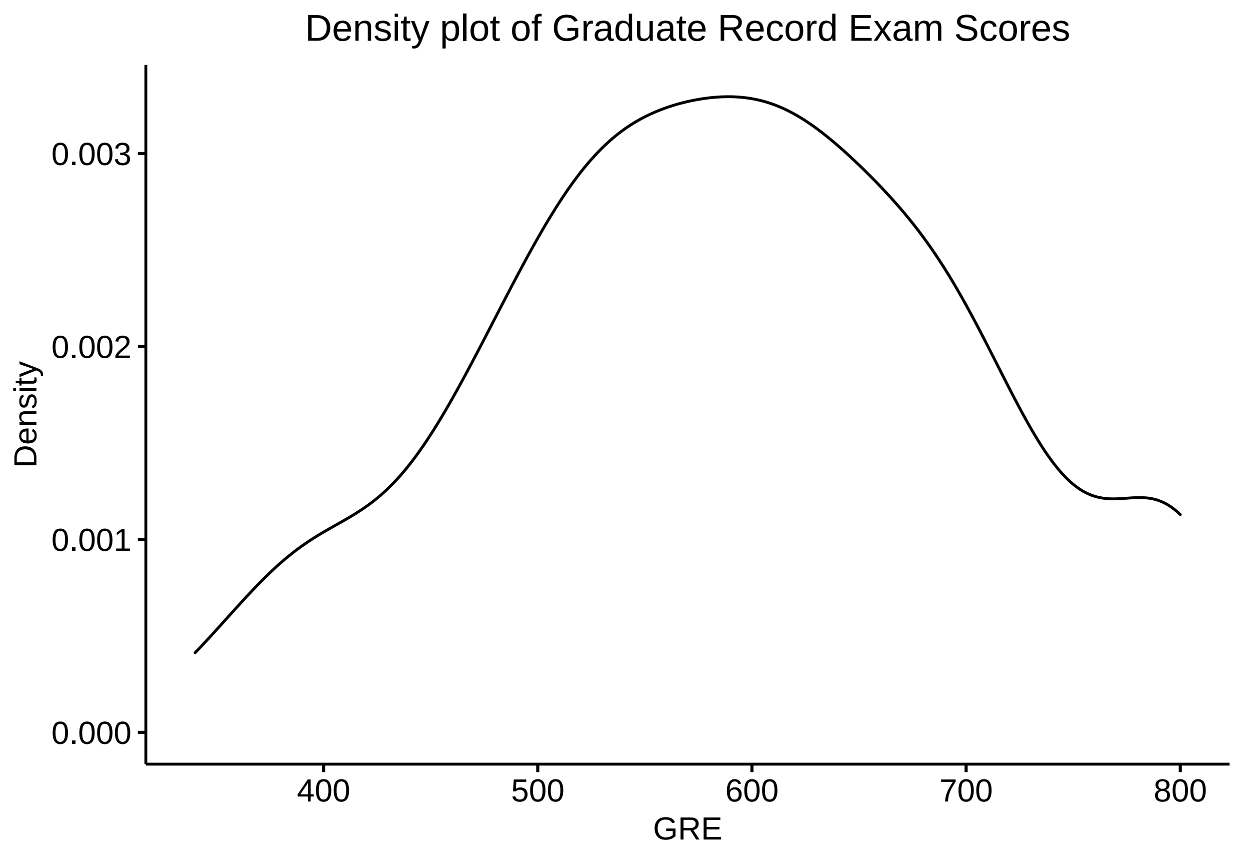
```
shapiro.test(CollegeDF$gre)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  CollegeDF$gre
## W = 0.9827, p-value = 0.0001227
```

```
ggqqplot(CollegeDF$gre)
```



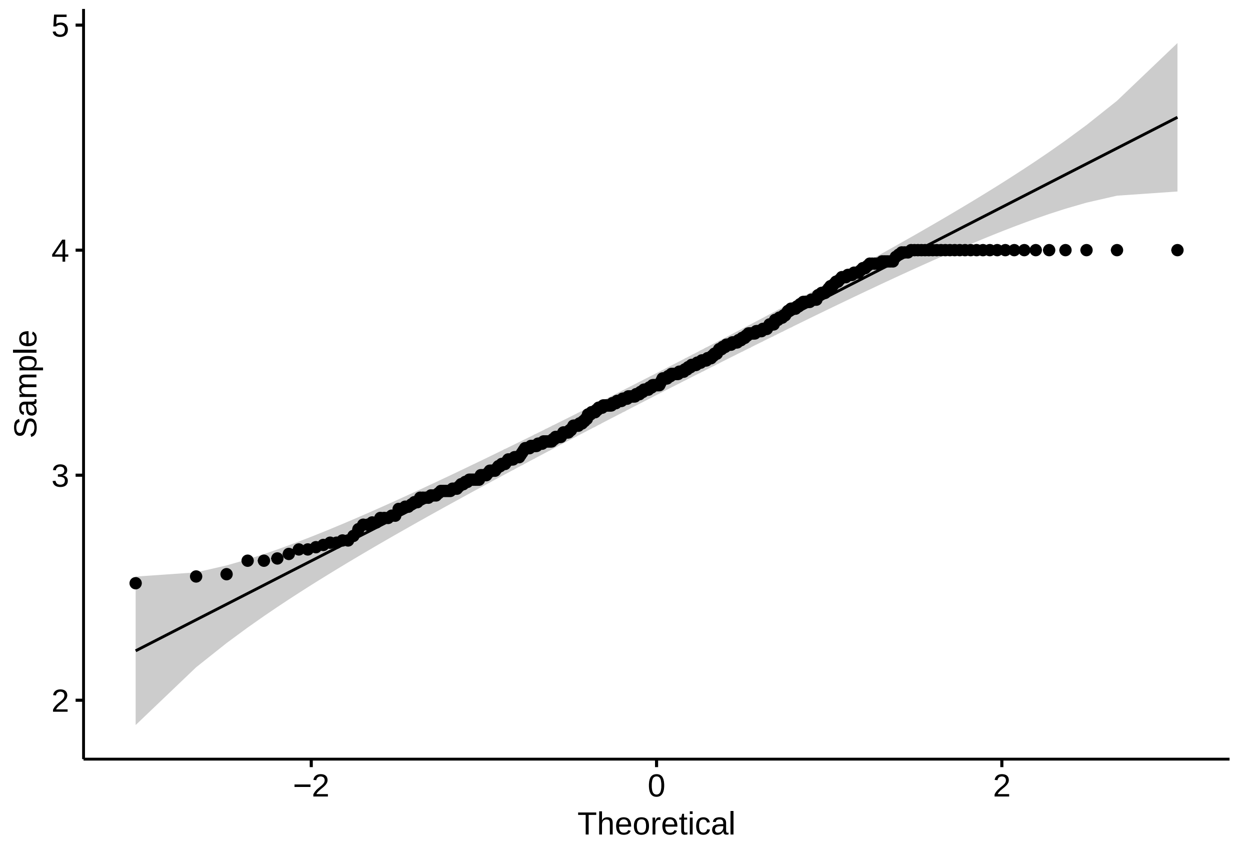
```r
ggdensity(CollegeDF$gre,
          main = "Density plot of Graduate Record Exam Scores",
          xlab = "GRE",
          ylab = "Density") +
  theme(plot.title = element_text(hjust = 0.5))
```



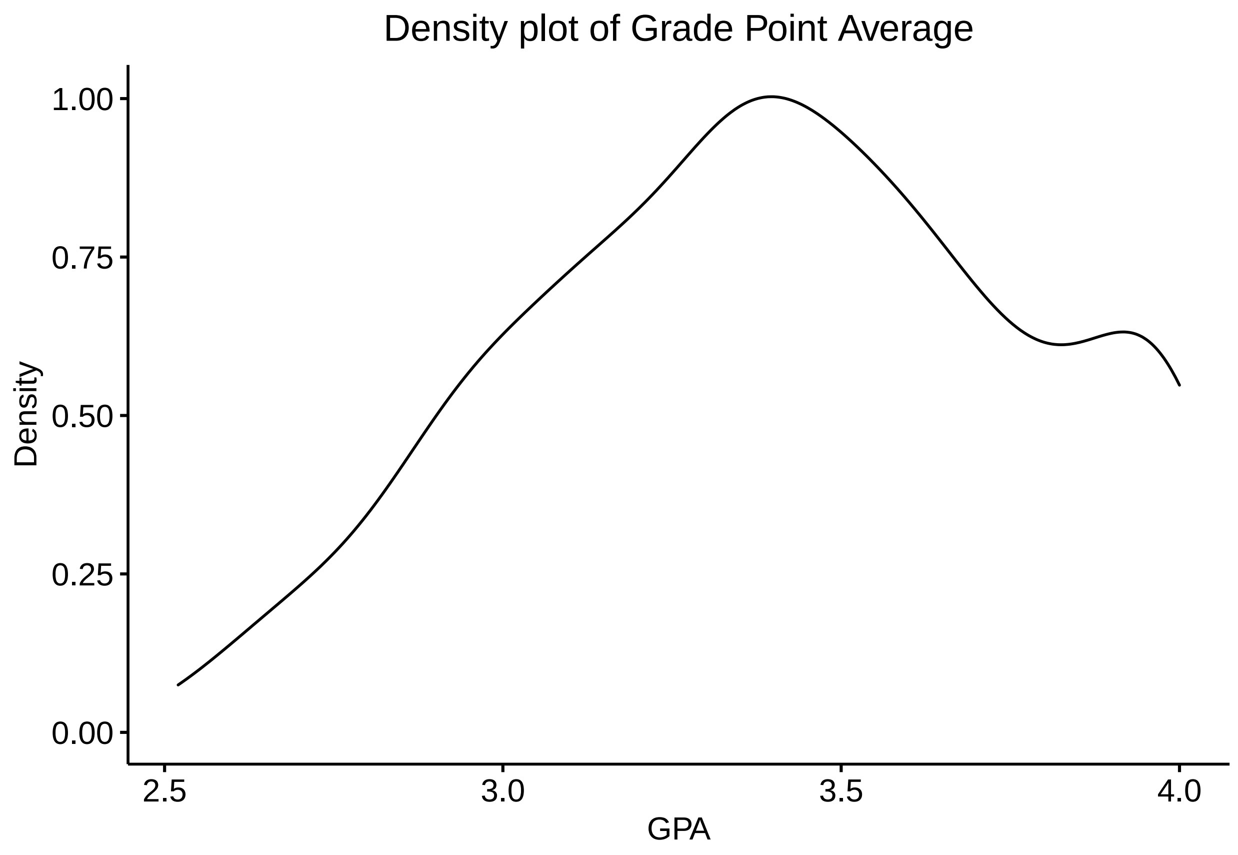
```
shapiro.test(CollegeDF$gpa)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  CollegeDF$gpa
## W = 0.97499, p-value = 2.765e-06
```

```
ggqqplot(CollegeDF$gpa)
```



```r
ggdensity(CollegeDF$gpa,
          main = "Density plot of Grade Point Average",
          xlab = "GPA",
          ylab = "Density") +
  theme(plot.title = element_text(hjust = 0.5))
```



Comments:

Based on the Shapiro test (p-value is lower than 0.05) and the plots we can conclude that the data is not normally distributed.

5. Normalize the data if not normally distributed.

```r
CollegeDF$gre <- scale(CollegeDF$gre)
CollegeDF$gpa <- scale(CollegeDF$gpa)
```

Comments:

Data has been normalized using *scale* function

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)

```r
set.seed(1)
inTrain <-
  createDataPartition(CollegeDF$admit, p = 0.7, list = FALSE)
Training <- CollegeDF[inTrain, ]
Testing <- CollegeDF[-inTrain, ]
```

Linear Regression model

```r
fit1 <- glm(admit ~ ., Training,
               family = "binomial")
summary(fit1)
```

```
##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = Training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7402  -0.8732  -0.6053   1.1630   2.2689
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.63745    0.46547   1.369  0.17085
## gre           0.26340    0.15142   1.740  0.08194 .
## gpa           0.46072    0.15880   2.901  0.00372 **
## ses2          0.05197    0.32929   0.158  0.87460
## ses3         -0.54100    0.35805  -1.511  0.13080
## Gender_Male1 -0.27135    0.28316  -0.958  0.33792
## Race2        -0.43090    0.34647  -1.244  0.21361
## Race3        -0.31438    0.34057  -0.923  0.35596
## rank2        -0.81880    0.40483  -2.023  0.04312 *
## rank3        -1.33209    0.42707  -3.119  0.00181 **
## rank4        -1.60886    0.51375  -3.132  0.00174 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.78  on 274  degrees of freedom
## Residual deviance: 308.60  on 264  degrees of freedom
## AIC: 330.6
##
## Number of Fisher Scoring iterations: 4
```

We keep only the significant variables for a new model

```r
fit2 <- glm(admit ~ gpa + rank + gre, Training,
               family = "binomial")
summary(fit2)
```

```
##
## Call:
## glm(formula = admit ~ gpa + rank + gre, family = "binomial",
##     data = Training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5973  -0.8654  -0.6297   1.1003   2.1611
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.05967    0.32829   0.182  0.85578
## gpa          0.44500    0.15583   2.856  0.00429 **
## rank2       -0.72871    0.39216  -1.858  0.06314 .
## rank3       -1.24478    0.41731  -2.983  0.00286 **
## rank4       -1.49551    0.49998  -2.991  0.00278 **
## gre          0.26495    0.14977   1.769  0.07690 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 344.78  on 274  degrees of freedom
## Residual deviance: 314.09  on 269  degrees of freedom
## AIC: 326.09
##
## Number of Fisher Scoring iterations: 4
```

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

```r
Pred <- predict(fit2,Testing, type="response")
Testing$Predict.recommend <- ifelse(Pred >= 0.5, 1, 0)

table(Testing$admit,
      Testing$Predict.recommend,
      dnn = list("Actual", "Predicted"))
```

```
##       Predicted
## Actual  0  1
##      0 70 10
##      1 25 12
```

```r
table(Testing$admit, Testing$Predict.recommend)
```

```
##
##      0  1
##    0 70 10
##    1 25 12
```

```
Testing$Predict.recommend <- as.factor(Testing$Predict.recommend)
confusionMatrix(Testing$Predict.recommend, Testing$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 25
##          1 10 12
##
##                Accuracy : 0.7009
##                  95% CI : (0.6093, 0.782)
##     No Information Rate : 0.6838
##     P-Value [Acc > NIR] : 0.38712
##
##                   Kappa : 0.2237
##
##  Mcnemar's Test P-Value : 0.01796
##
##             Sensitivity : 0.8750
##             Specificity : 0.3243
##          Pos Pred Value : 0.7368
##          Neg Pred Value : 0.5455
##              Prevalence : 0.6838
##          Detection Rate : 0.5983
##    Detection Prevalence : 0.8120
##       Balanced Accuracy : 0.5997
##
##        'Positive' Class : 0
##
```

Comments:

The Accuracy of the above model is 70.09%

Now we use K-Fold Cross Validation

```
train_control <- trainControl(method = "cv", number = 5)
fit3 <-
  train(
    admit ~ gpa + rank + gre,
    CollegeDF,
    trControl = train_control,
    method = "glm",
    family = "binomial"
  )
fit3$resample
```

```
##    Accuracy     Kappa Resample
## 1 0.6923077 0.1709477    Fold1
## 2 0.7307692 0.2465501    Fold2
## 3 0.7468354 0.3112467    Fold3
## 4 0.6794872 0.1255605    Fold4
## 5 0.6962025 0.1734961    Fold5
```

```
fit3$results
```

```
##   parameter  Accuracy     Kappa AccuracySD    KappaSD
## 1      none 0.7091204 0.2055602 0.02836707 0.07327328
```

Comments:

We get 70.91% accuracy when we use K-Fold Cross-Validation. K-Fold Cross-Validation addresses the problem of over-fitting of our model and is providing us with the right picture by giving us the correct, more unbiased and real evaluation score of our model.

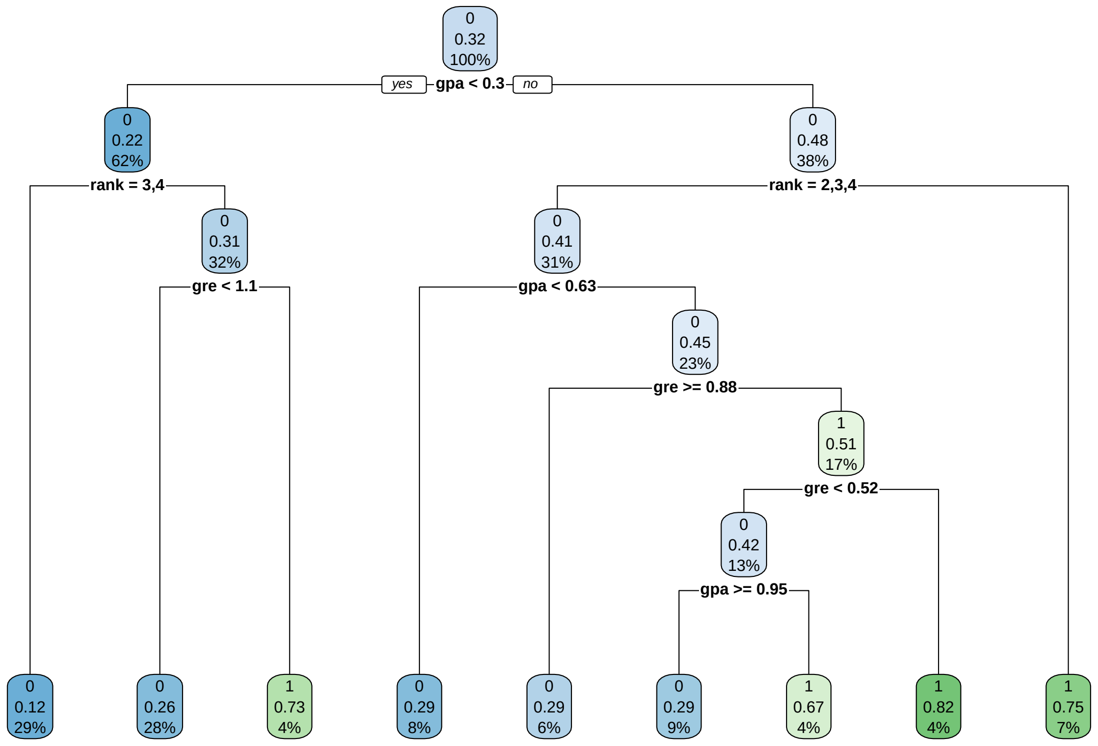
9. Try other modeling techniques like decision tree and SVM and select a champion model

10. Determine the accuracy rates for each kind of model

11. Select the most accurate model

Decision Tree

```
fit4 <-
  rpart(
    admit ~ gpa + rank + gre,
    Training,
    method = "class",
    control = rpart.control(minsplit = 30, cp = 0.01)
  )
rpart.plot(fit4)
```



```
Pred_DT <- predict(fit4, Testing, type="class")

table(Pred_DT, Testing$admit, dnn = list("Actual", "Predicted"))
```

```
##          Predicted
## Actual    0  1
##         0 72 28
##         1  8  9
```

```
confusionMatrix(Pred_DT, Testing$admit)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  0  1
##           0 72 28
##           1  8  9
##
##                Accuracy : 0.6923
##                  95% CI : (0.6003, 0.7743)
##     No Information Rate : 0.6838
##     P-Value [Acc > NIR] : 0.465253
##
##                   Kappa : 0.1676
```

```
##
##  Mcnemar's Test P-Value : 0.001542
##
##             Sensitivity : 0.9000
##             Specificity : 0.2432
##          Pos Pred Value : 0.7200
##          Neg Pred Value : 0.5294
##              Prevalence : 0.6838
##          Detection Rate : 0.6154
##    Detection Prevalence : 0.8547
##       Balanced Accuracy : 0.5716
##
##        'Positive' Class : 0
##
```

Comments:

Accuracy of the model is 69.23%

SVM

```r
fit5 <- svm(admit ~ gpa+rank+gre, Training, kernel="linear",
            scale = T)
summary(fit5)
```

```
##
## Call:
## svm(formula = admit ~ gpa + rank + gre, data = Training, kernel = "linear",
##     scale = T)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  184
##
##  ( 97 87 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
Pred_SVM <- predict(fit5,Testing, type="response")

table(Pred_SVM, Testing$admit, dnn = list("Actual", "Predicted"))
```

```
##       Predicted
## Actual  0  1
##      0 71 27
##      1  9 10
```

```
confusionMatrix(Pred_SVM, Testing$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 71 27
##          1  9 10
##
##                Accuracy : 0.6923
##                  95% CI : (0.6003, 0.7743)
##     No Information Rate : 0.6838
##     P-Value [Acc > NIR] : 0.465253
##
##                   Kappa : 0.1815
##
##  Mcnemar's Test P-Value : 0.004607
##
##             Sensitivity : 0.8875
##             Specificity : 0.2703
##          Pos Pred Value : 0.7245
##          Neg Pred Value : 0.5263
##              Prevalence : 0.6838
##          Detection Rate : 0.6068
##    Detection Prevalence : 0.8376
##       Balanced Accuracy : 0.5789
##
##        'Positive' Class : 0
##
```

Comments:

Accuracy of the model is 69.23%

The most accurate model is the linear regression with a validated accuracy of 70.91%.

12. Identify other Machine learning or statistical techniques

kNN

```
sqrt(275)
```

```
## [1] 16.58312
```

```
fit6 <-
  knn(Training[, -1], Testing[-c(1, 8)], cl = Training[, 1], k = 16)

table(Testing[, 1], fit6, dnn = list("Actual", "Predicted"))
```

```
##       Predicted
## Actual  0  1
##      0 75  5
##      1 30  7
```

```r
confusionMatrix(fit6,Testing$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 75 30
##          1  5  7
##
##                Accuracy : 0.7009
##                  95% CI : (0.6093, 0.782)
##     No Information Rate : 0.6838
##     P-Value [Acc > NIR] : 0.3871
##
##                   Kappa : 0.1548
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.9375
##             Specificity : 0.1892
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.5833
##              Prevalence : 0.6838
##          Detection Rate : 0.6410
##    Detection Prevalence : 0.8974
##       Balanced Accuracy : 0.5633
##
##        'Positive' Class : 0
##
```

Comments:

Accuracy of the model is 70.09%

Naive Bayes

```r
fit7 <- naive_bayes(admit ~ gpa + rank + gre, Training)
Pred_NV <- predict(fit7, Testing)
```

```
## Warning: predict.naive_bayes(): more features in the newdata are provided as
## there are probability tables in the object. Calculation is performed based on
## features to be found in the tables.
```

```r
table(Pred_NV, Testing$admit, dnn = list("Actual", "Predicted"))
```

```
##       Predicted
## Actual  0  1
##      0 64 26
##      1 16 11
```

```r
confusionMatrix(Pred_NV, Testing$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 64 26
##          1 16 11
##
##                Accuracy : 0.641
##                  95% CI : (0.5471, 0.7276)
##     No Information Rate : 0.6838
##     P-Value [Acc > NIR] : 0.8625
##
##                   Kappa : 0.1049
##
##  Mcnemar's Test P-Value : 0.1649
##
##             Sensitivity : 0.8000
##             Specificity : 0.2973
##          Pos Pred Value : 0.7111
##          Neg Pred Value : 0.4074
##              Prevalence : 0.6838
##          Detection Rate : 0.5470
##    Detection Prevalence : 0.7692
##       Balanced Accuracy : 0.5486
##
##        'Positive' Class : 0
##
```

Comments:

Accuracy of the model is 64.10%

Random Forest

```r
fit8 <- randomForest(admit ~ gpa+rank+gre, Training)
Pred_RF <- predict(fit8,Testing)

table(Pred_RF, Testing$admit, dnn = list("Actual", "Predicted"))
```

```
##       Predicted
## Actual  0  1
##      0 72 26
##      1  8 11
```

```r
confusionMatrix(Pred_RF, Testing$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 72 26
##          1  8 11
##
##                Accuracy : 0.7094
##                  95% CI : (0.6183, 0.7896)
##     No Information Rate : 0.6838
##     P-Value [Acc > NIR] : 0.312950
##
##                   Kappa : 0.227
##
##  Mcnemar's Test P-Value : 0.003551
##
##             Sensitivity : 0.9000
##             Specificity : 0.2973
##          Pos Pred Value : 0.7347
##          Neg Pred Value : 0.5789
##              Prevalence : 0.6838
##          Detection Rate : 0.6154
##    Detection Prevalence : 0.8376
##       Balanced Accuracy : 0.5986
##
##        'Positive' Class : 0
##
```

Comments:

Accuracy of the model is 70.94%

Descriptive:

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

```
College <- import("College_admission.csv")
CollegeGRE <-
  College %>% mutate(Categorized = case_when(
    gre < 440 ~ "Low",
    gre < 580 ~ "Medium",
    gre >= 580 ~ "High"))

count(filter(CollegeGRE, Categorized == "Low"))
```

```
##    n
## 1 38
```

```
count(filter(CollegeGRE, Categorized == "Medium"))
```

```
##     n
## 1 136
```

```
count(filter(CollegeGRE, Categorized == "High"))
```

```
##     n
## 1 226
```

```
Admit_GPAlow <- filter(CollegeGRE, Categorized == "Low")
count(filter(CollegeGRE, Categorized == "Low"))
```

```
##    n
## 1 38
```

```
sum(Admit_GPAlow$admit == 1)
```

```
## [1] 4
```

```
(4 / 38) * 100
```

```
## [1] 10.52632
```

Comments:

For category "Low" the admission probability is 10.53%

```
Admit_GPAmedium <- filter(CollegeGRE, Categorized == "Medium")
count(filter(CollegeGRE, Categorized == "Medium"))
```

```
##     n
## 1 136
```

```
sum(Admit_GPAmedium$admit == 1)
```

```
## [1] 39
```

```
(39 / 136) * 100
```

```
## [1] 28.67647
```

Comments:

For category "Medium" the admission probability is 28.68%

```
Admit_GPAhigh <- filter(CollegeGRE, Categorized == "High")
count(filter(CollegeGRE, Categorized == "High"))
```

```
##     n
## 1 226
```

```r
sum(Admit_GPAhigh$admit == 1)
```

```
## [1] 84
```

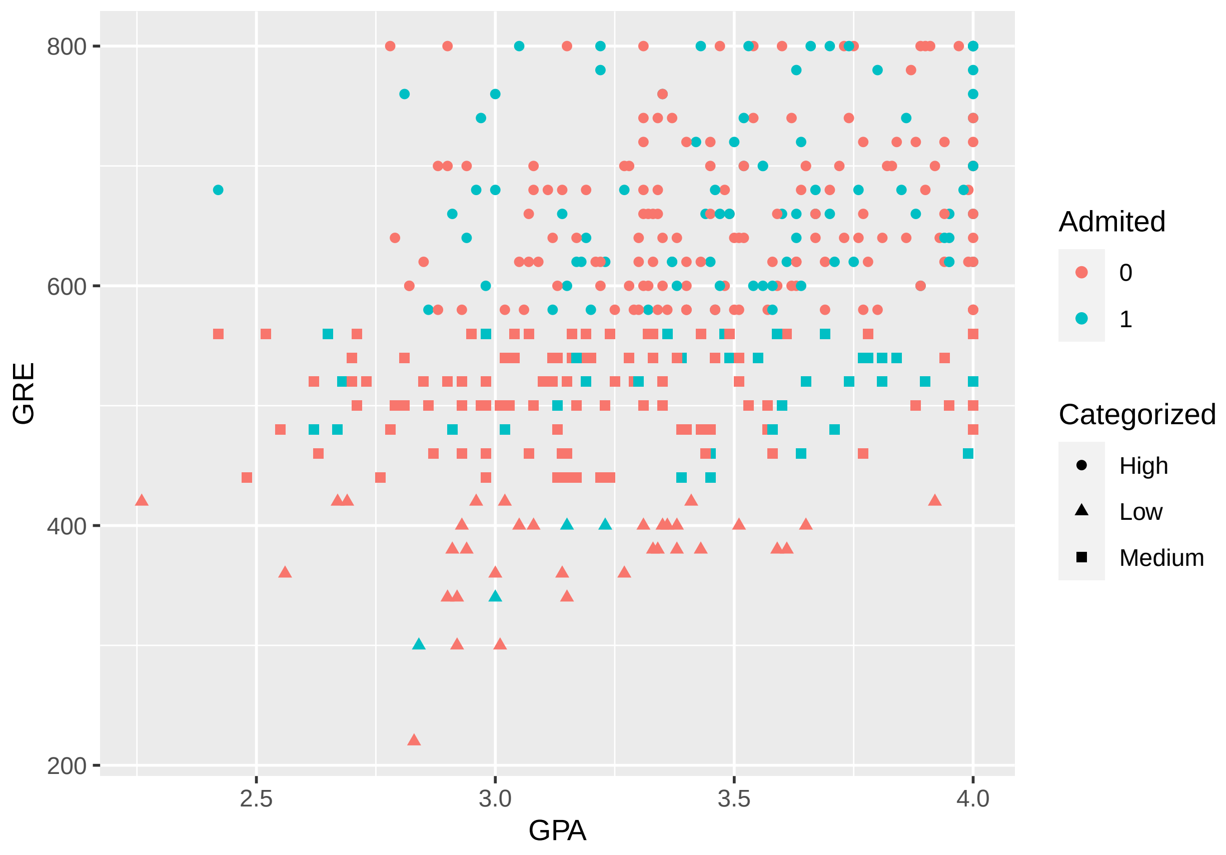
```r
(84 / 226) * 100
```

```
## [1] 37.16814
```

Comments:

For category "High" the admission probability is 37.17%

```r
ggplot(CollegeGRE,
       aes(x = gpa, y = gre, colour = factor(admit), shape = factor(Categorized))) +
  geom_point() +
  labs(x = "GPA",
       y = "GRE",
       shape = "Categorized",
       colour = "Admited")
```


GRE
GPA
Admited
Categorized
High
Low
Medium