Project_College_Admission.R

might

2022-10-30

```
#importing library
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caTools)
library(ROCR)
library(MASS)
library(caret)
```

```
## Loading required package: lattice
```

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(readr)
library(ggpubr)

#Clearing the r environment
rm(list = ls(all = TRUE))


#loading data
clg_ad <- read.csv("College_admission.csv")
head(clg_ad)
```

```
##   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
```

```r
tail(clg_ad)
```

```
##     admit gre  gpa ses Gender_Male Race rank
## 395     1 460 3.99   3           1    3    3
## 396     0 620 4.00   2           0    2    2
## 397     0 560 3.04   2           0    1    3
## 398     0 460 2.63   3           0    2    2
## 399     0 700 3.65   1           1    1    2
## 400     0 600 3.89   2           1    3    3
```

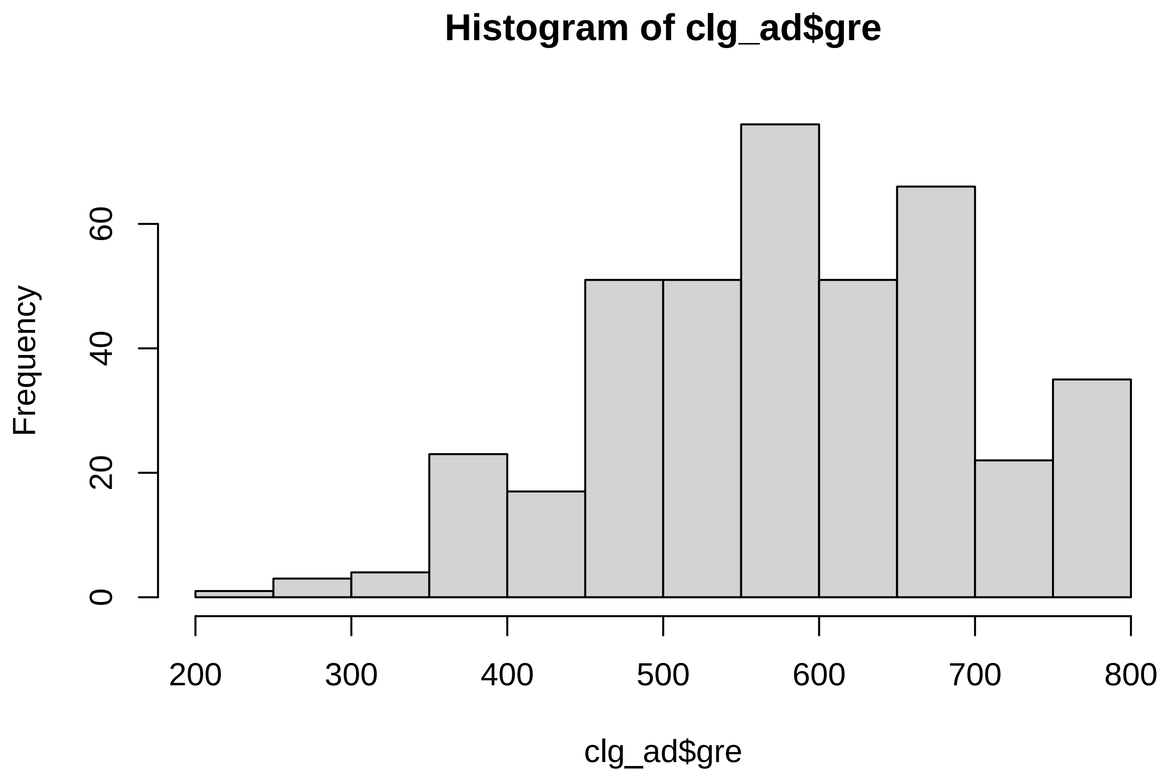
```r
#descriptive statistics
summary(clg_ad$gpa)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.260   3.130   3.395   3.390   3.670   4.000
```

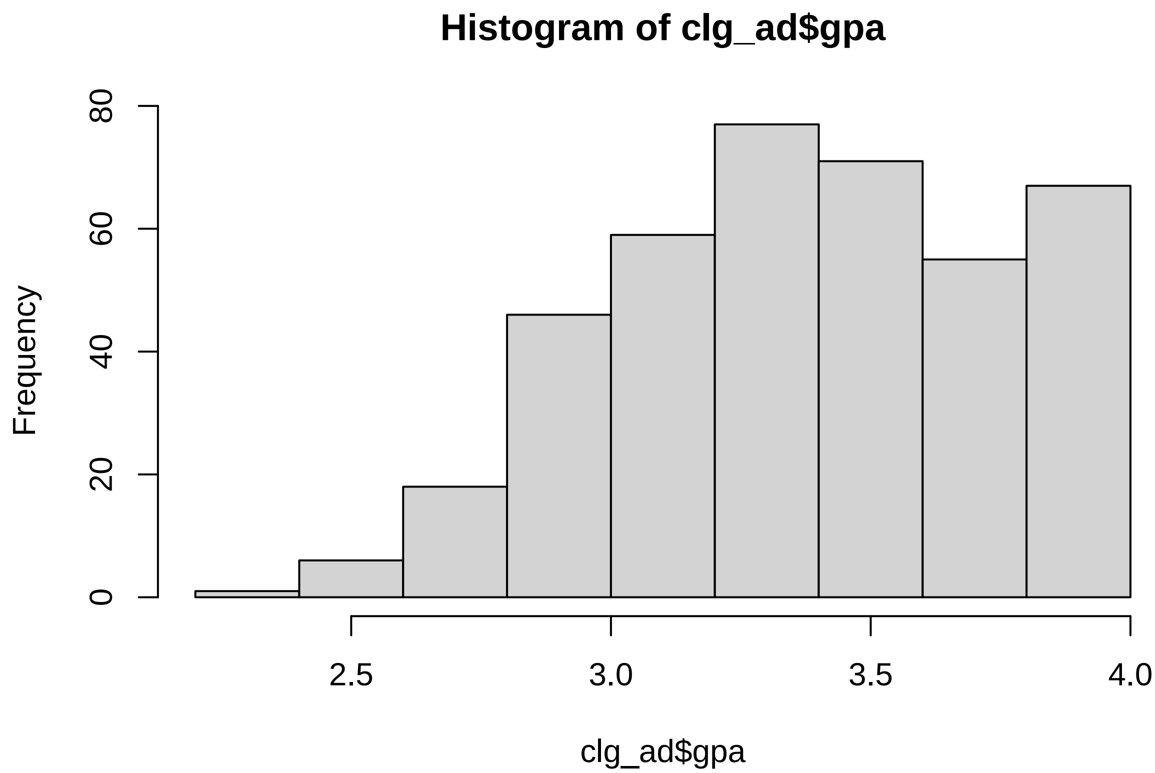
```r
#1. Find the missing values. (if any, perform missing value treatment)

sum(is.null(clg_ad))
```

```
## [1] 0
```

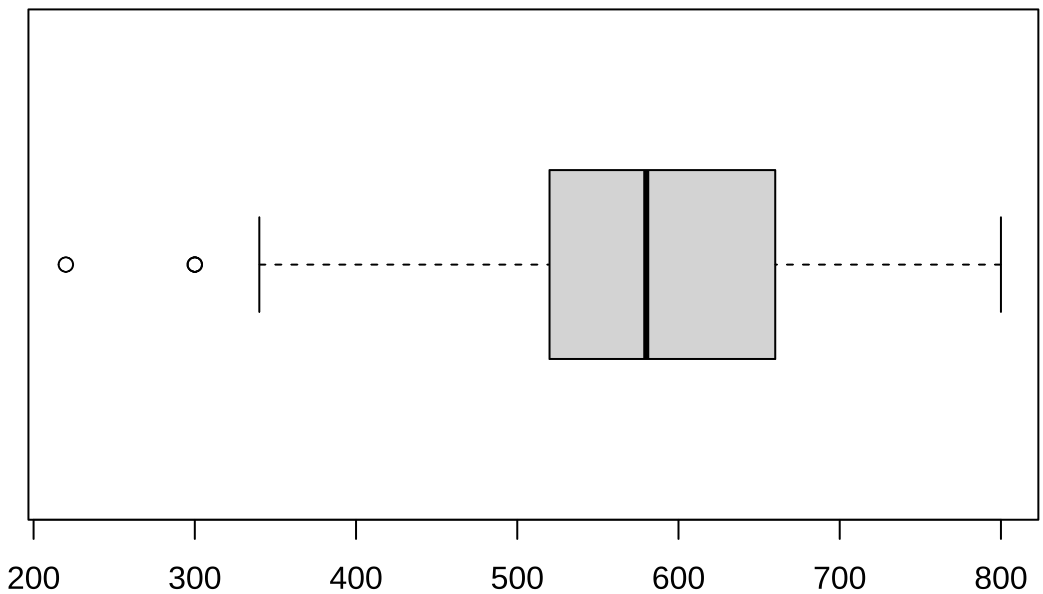
```r
#Since there are no Null values in the dataset, no need for missing value treatment



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

hist(clg_ad$gre)
```



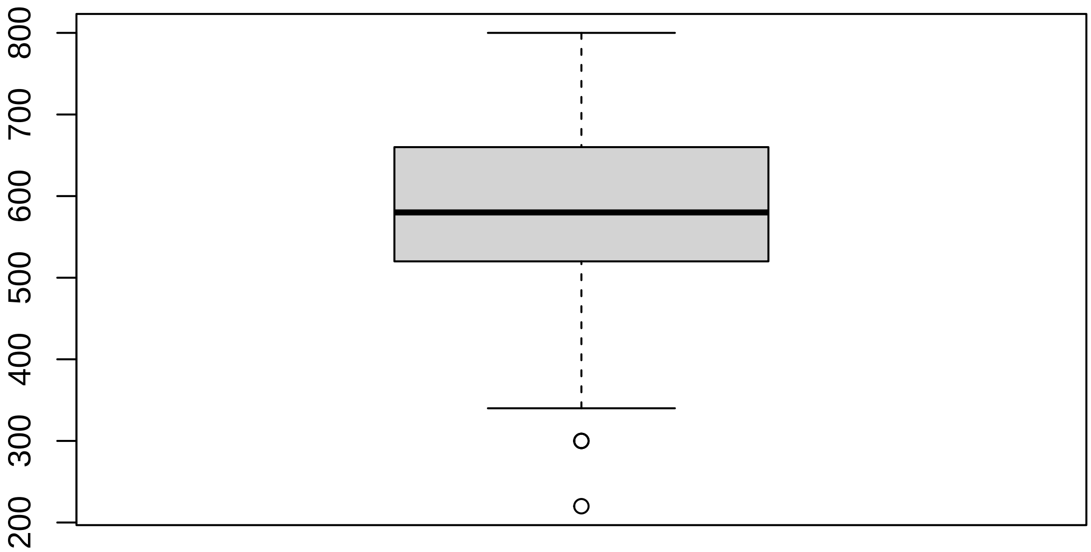

```
hist(clg_ad$gpa)
```




```r
#Using Boxplot to understand if there are any outliers present in GRE variable
boxplot(clg_ad$gre, horizontal = T)
```




```
# By looking at the Boxplot, we can see there are outliers in GRE variable
greoutlier <- boxplot(clg_ad$gre)$out
```



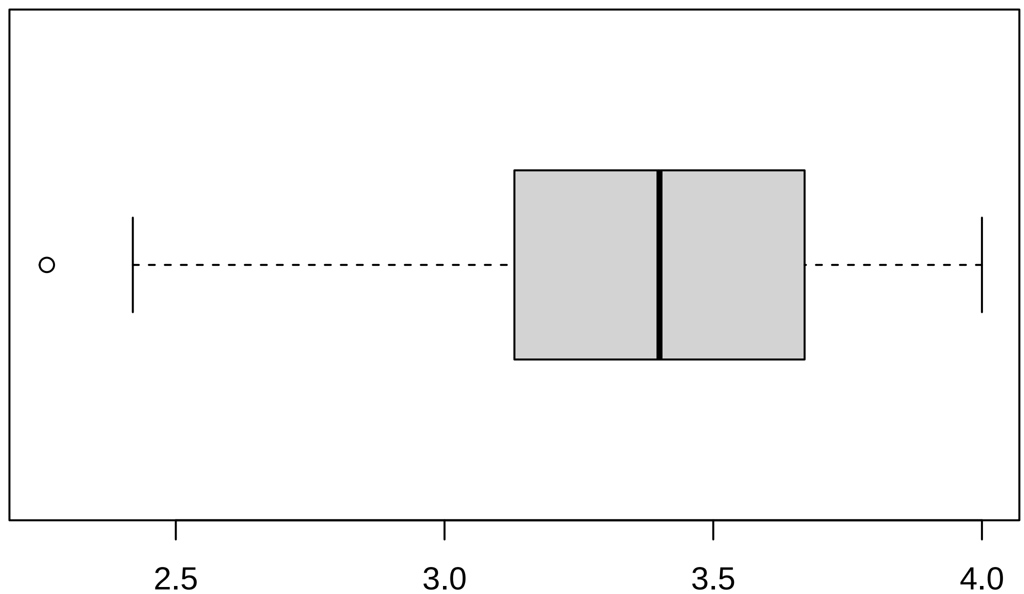

```r
#checking length of GRE outliers
length(greoutlier)
```

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

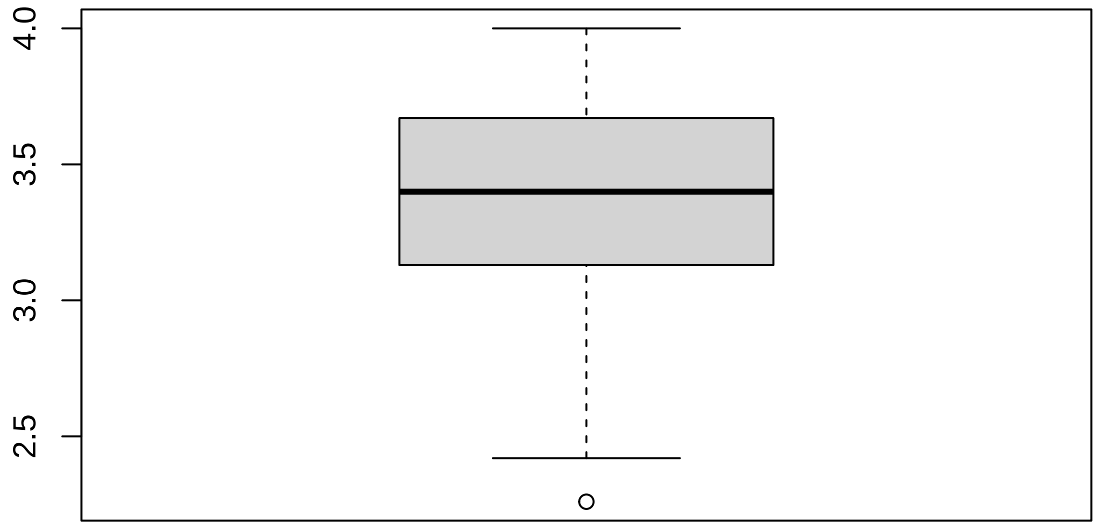
```r
#Removing outliers from GRE variable
Q <- quantile(clg_ad$gre, probs=c(.25, .75), na.rm = FALSE)
iqr <- IQR(clg_ad$gre)

up <- Q[2]+1.5*iqr # Upper Range
low<- Q[1]-1.5*iqr # Lower Range

clg_ad <- subset(clg_ad, clg_ad$gre > (Q[1] - 1.5*iqr) & clg_ad$gre < (Q[2]+1.5*iqr))
head(clg_ad)
```

```
##   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
```

```r
#Using Boxplot to understand if there are any outliers present in GPA variable
boxplot(clg_ad$gpa, horizontal = T)
```




```r
# By looking at the Boxplot, we can see there are outliers in GPA variable
gpaoutlier <- boxplot(clg_ad$gpa)$out
```




```
length(gpaoutlier)
```

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

```
#There is only one outlier present in this Variable, we can remove it.

#Removing outliers from GPA variable
Q <- quantile(clg_ad$gpa, probs=c(.25, .75), na.rm = FALSE)

iqr <- IQR(clg_ad$gpa)

up <-  Q[2]+1.5*iqr # Upper Range
low<- Q[1]-1.5*iqr # Lower Range

clg_ad <- subset(clg_ad, clg_ad$gpa > (Q[1] - 1.5*iqr) & clg_ad$gpa < (Q[2]+1.5*iqr))
head(clg_ad)
```

```
##   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
```

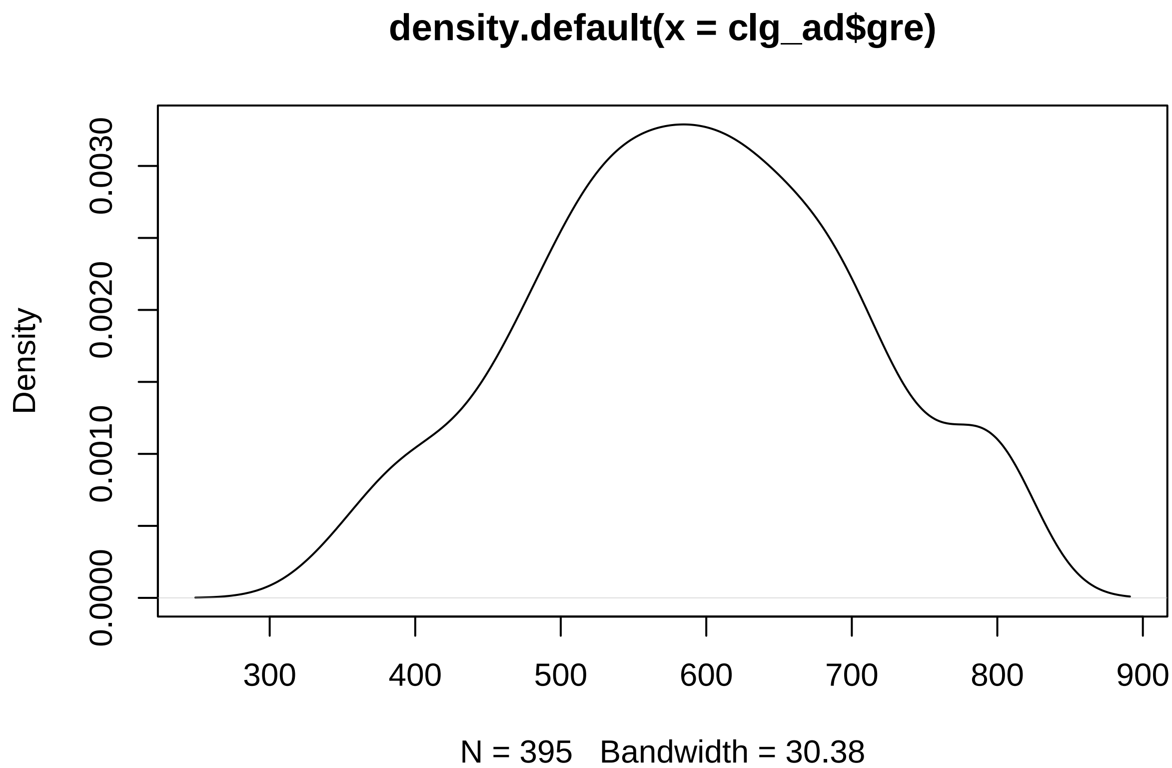
```r
#now outliers have been removed from data, After removing outliers we have 395 data points

#3. Find the structure of the data set and if required, transform the numeric data type to factor and v
#structure of data
str(clg_ad)
```

```
## 'data.frame':    395 obs. of  7 variables:
##  $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
##  $ Gender_Male: int  0 0 0 1 1 1 1 0 1 0 ...
##  $ Race       : int  3 2 2 2 2 1 2 2 1 2 ...
##  $ rank       : int  3 3 1 4 4 2 1 2 3 2 ...
```

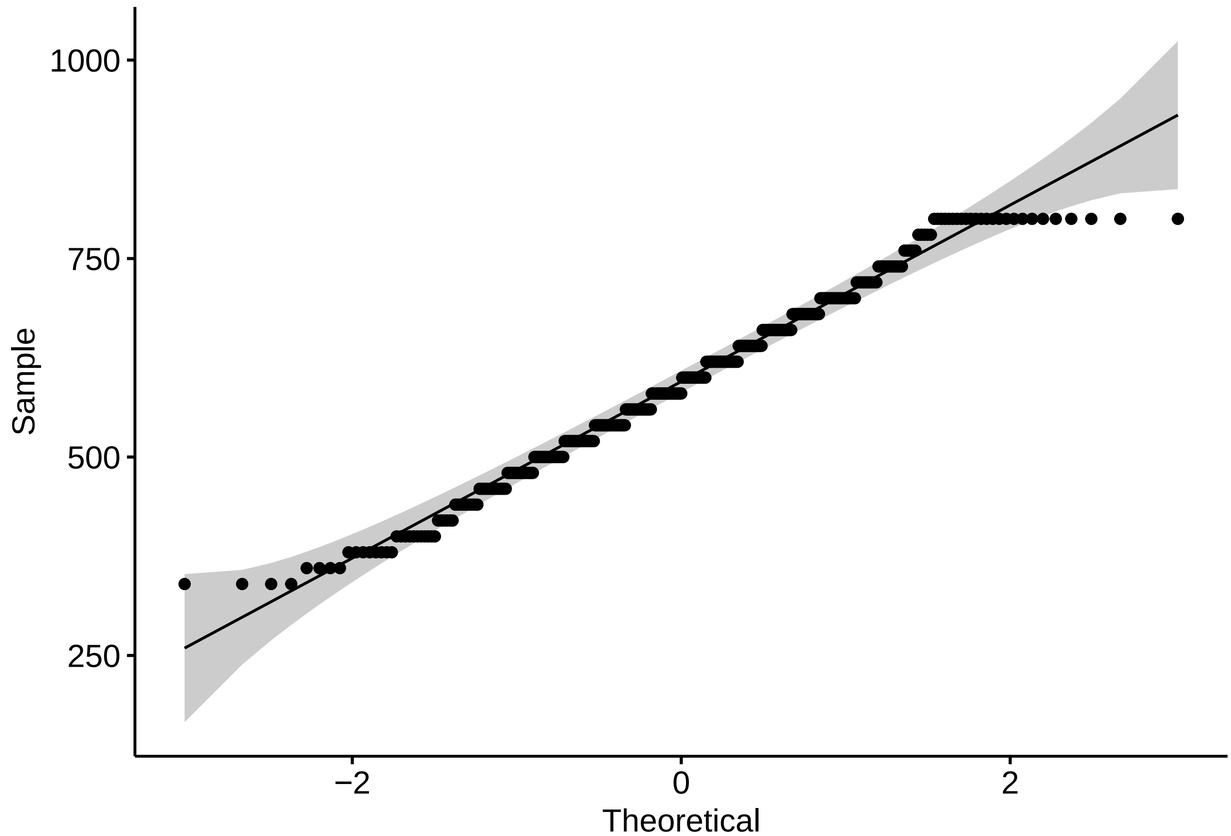
```r
clg_ad$admit <- as.factor(clg_ad$admit)
clg_ad$ses <- as.factor(clg_ad$ses)
clg_ad$Gender_Male <- as.factor(clg_ad$Gender_Male)
clg_ad$Race <- as.factor(clg_ad$Race)
clg_ad$rank <- as.factor(clg_ad$rank)
str(clg_ad)
```

```
## 'data.frame':    395 obs. of  7 variables:
##  $ admit      : Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
##  $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ ses        : 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 ...
```

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

#checking distribution of GRE variable
plot(density(clg_ad$gre))
```



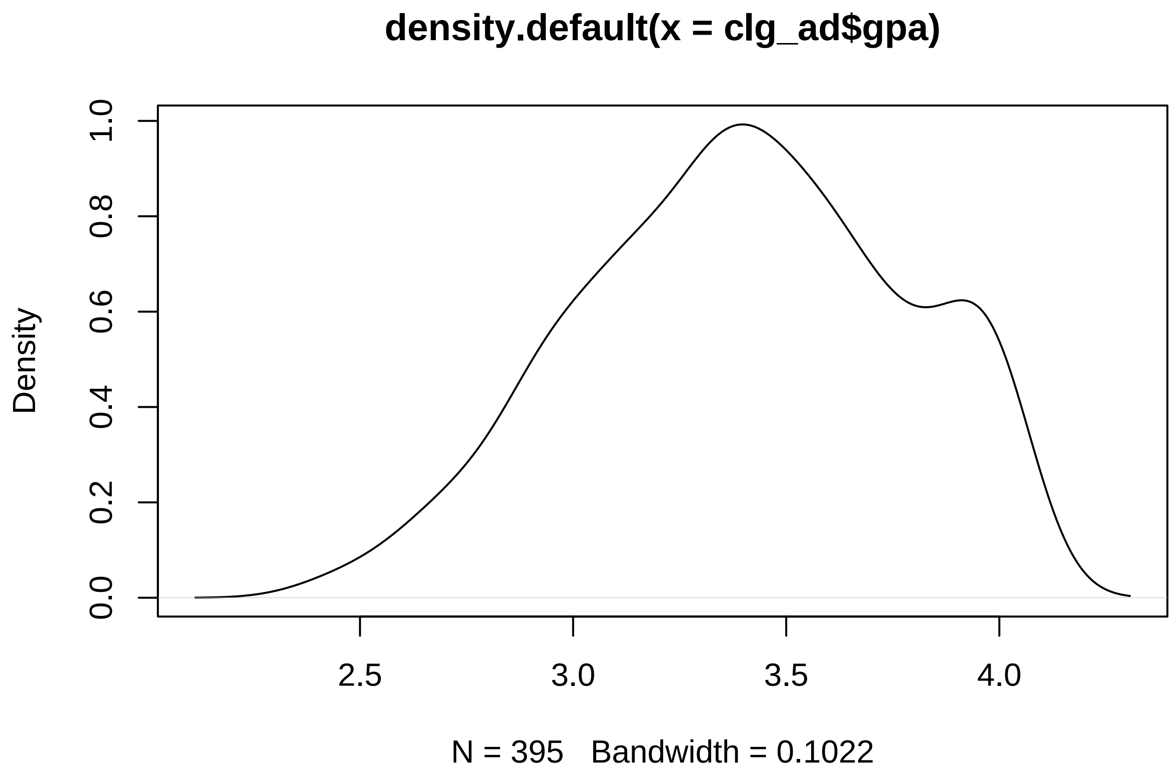

```
ggqqplot(clg_ad$gre)
```



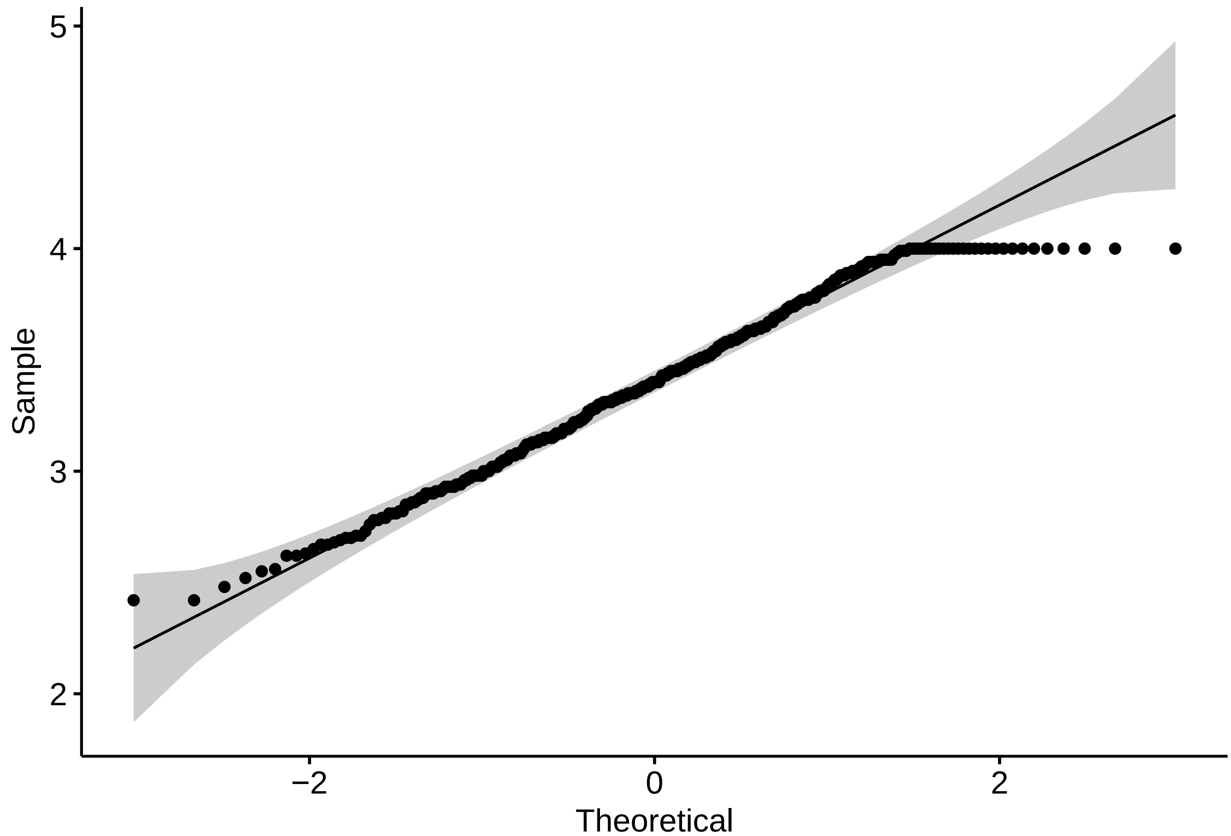

```
#Since Majority of the data points fall on the line,
#we can assume that the data is normally distributed

##Normality test using Shapiro test
shapiro.test(clg_ad$gre)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  clg_ad$gre
## W = 0.98282, p-value = 0.0001223
```

```
#checking distribution of GPA variable
plot(density(clg_ad$gpa))
```




```
##Since Majority of the data points fall on the line, we can assume that the data is normally distribut
ggqqplot(clg_ad$gpa)
```




```
#Normality test using Shapiro test
shapiro.test(clg_ad$gpa)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  clg_ad$gpa
## W = 0.97646, p-value = 5.004e-06
```

```
#5. Normalize the data if not normally distributed
#Since the data is normally distributed, data tranformation is not required
#But however there is variance in GRE and GPA variable, we can clearly see
  #that gre values are in hundred times more than the gpa values.
#In this case when we build model, gre will internally influence
  #the result more due to its larger value.
#To avoid problem and for accurate model,
  #we can scale down the variables to avoid problem and accurate model.

#Creating a copy of the data
clgad <- clg_ad
clg_ad$gre <- scale(clg_ad$gre, center = T, scale = T)
clg_ad$gpa <- scale(clg_ad$gpa, center = T, scale = T)
head(clg_ad)
```

```
##   admit        gre        gpa ses Gender_Male Race rank
```

```
## 1      0 -1.8927212  0.5655683   1           0    3    3
## 2      1  0.6160871  0.7254249   2           0    2    3
## 3      1  1.8704913  1.6046358   2           0    2    1
## 4      1  0.4368865 -0.5534273   1           1    2    4
## 5      0 -0.6383171 -1.2461390   3           1    2    4
## 6      1  1.5120901 -1.0596397   2           1    1    2
```

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

#We can build logistic regression model and identify significant variables.
set.seed(1234)
sampledata <- sample.split(clg_ad$admit, SplitRatio = 0.7)
train <- clg_ad[sampledata==T,]
test <- clg_ad[sampledata==F,]

test_without_admit <- test[,-1]

# fit the model
class(train$admit)
```

```
## [1] "factor"
```

```
log_reg <- glm(admit ~ . , data = train, family = 'binomial')
summary(log_reg)
```

```
##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8138  -0.8617  -0.5745   1.0071   2.2491
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.7374     0.4309    1.711 0.087000 .
## gre            0.3871     0.1556    2.488 0.012833 *
## gpa            0.2469     0.1525    1.619 0.105418
## ses2          -0.2934     0.3504   -0.837 0.402436
## ses3          -0.2998     0.3430   -0.874 0.382196
## Gender_Male1  -0.2494     0.2898   -0.861 0.389398
## Race2         -0.5247     0.3472   -1.511 0.130761
## Race3         -0.3224     0.3429   -0.940 0.347027
## rank2         -0.7151     0.3855   -1.855 0.063561 .
## rank3         -1.5240     0.4226   -3.606 0.000311 ***
## rank4         -2.0875     0.5807   -3.595 0.000325 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 345.55  on 275  degrees of freedom
## Residual deviance: 301.79  on 265  degrees of freedom
```

```
## AIC: 323.79
##
## Number of Fisher Scoring iterations: 4
```

```
#Using step AIC method to identify significant variables
model_AIC = stepAIC(object = log_reg,direction = "both")
```

```
## Start:  AIC=323.79
## admit ~ gre + gpa + ses + Gender_Male + Race + rank
##
##               Df Deviance    AIC
## - ses          2   302.78 320.78
## - Race         2   304.16 322.16
## - Gender_Male  1   302.53 322.53
## <none>             301.79 323.79
## - gpa          1   304.44 324.44
## - gre          1   308.24 328.24
## - rank         3   323.76 339.76
##
## Step:  AIC=320.78
## admit ~ gre + gpa + Gender_Male + Race + rank
##
##               Df Deviance    AIC
## - Gender_Male  1   303.44 319.44
## - Race         2   305.47 319.47
## <none>             302.78 320.78
## - gpa          1   305.40 321.40
## + ses          2   301.79 323.79
## - gre          1   309.18 325.18
## - rank         3   325.26 337.26
##
## Step:  AIC=319.44
## admit ~ gre + gpa + Race + rank
##
##               Df Deviance    AIC
## - Race         2   305.88 317.88
## <none>             303.44 319.44
## - gpa          1   306.15 320.15
## + Gender_Male  1   302.78 320.78
## + ses          2   302.53 322.53
## - gre          1   309.63 323.63
## - rank         3   325.48 335.48
##
## Step:  AIC=317.88
## admit ~ gre + gpa + rank
##
##               Df Deviance    AIC
## <none>             305.88 317.88
## - gpa          1   308.62 318.62
## + Race         2   303.44 319.44
## + Gender_Male  1   305.47 319.47
## + ses          2   304.69 320.69
## - gre          1   311.84 321.84
## - rank         3   328.35 334.35
```

```r
summary(model_AIC)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6582  -0.8584  -0.5817   1.0534   2.3413
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1806     0.3145   0.574 0.565843
## gre           0.3652     0.1524   2.396 0.016563 *
## gpa           0.2482     0.1511   1.642 0.100525
## rank2        -0.7396     0.3751  -1.972 0.048646 *
## rank3        -1.5088     0.4161  -3.626 0.000287 ***
## rank4        -2.1181     0.5753  -3.682 0.000231 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 345.55  on 275  degrees of freedom
## Residual deviance: 305.88  on 270  degrees of freedom
## AIC: 317.88
##
## Number of Fisher Scoring iterations: 4
```

```r
#From the results above GRE, GPA and Rank are the most significant variables.


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

```
##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8138  -0.8617  -0.5745   1.0071   2.2491
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.7374     0.4309   1.711 0.087000 .
## gre            0.3871     0.1556   2.488 0.012833 *
## gpa            0.2469     0.1525   1.619 0.105418
## ses2          -0.2934     0.3504  -0.837 0.402436
## ses3          -0.2998     0.3430  -0.874 0.382196
```

```
## Gender_Male1  -0.2494     0.2898  -0.861 0.389398
## Race2         -0.5247     0.3472  -1.511 0.130761
## Race3         -0.3224     0.3429  -0.940 0.347027
## rank2         -0.7151     0.3855  -1.855 0.063561 .
## rank3         -1.5240     0.4226  -3.606 0.000311 ***
## rank4         -2.0875     0.5807  -3.595 0.000325 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 345.55  on 275  degrees of freedom
## Residual deviance: 301.79  on 265  degrees of freedom
## AIC: 323.79
##
## Number of Fisher Scoring iterations: 4
```

```
#After looking at the summary results obtained from logistic regression,
  #factors that influence admission processs are GRE, GPA and RANK

# Dropping variables that are insignificant.
clg_ad1 <- subset(clg_ad, select = -c(ses,Gender_Male,Race))
head(clg_ad1)
```

```
##   admit        gre        gpa rank
## 1     0 -1.8927212  0.5655683    3
## 2     1  0.6160871  0.7254249    3
## 3     1  1.8704913  1.6046358    1
## 4     1  0.4368865 -0.5534273    4
## 5     0 -0.6383171 -1.2461390    4
## 6     1  1.5120901 -1.0596397    2
```

```
#8. Calculate the accuracy of the model and run validation techniques
#Using the result of logistic regresssion model to calculate the accuracy
#Using only the significant variables for Building models
class(clg_ad1)
```

```
## [1] "data.frame"
```

```
head(clg_ad1)
```


```
##   admit        gre        gpa rank
## 1     0 -1.8927212  0.5655683    3
## 2     1  0.6160871  0.7254249    3
## 3     1  1.8704913  1.6046358    1
## 4     1  0.4368865 -0.5534273    4
## 5     0 -0.6383171 -1.2461390    4
## 6     1  1.5120901 -1.0596397    2
```


```
set.seed(123)
```

```r
sample_data <- sample.split(clg_ad1$admit, SplitRatio = 0.7)
Train <- clg_ad1[sample_data==T,]
Test <- clg_ad1[sample_data==F,]

test_without_admit <- Test[,-1]

logreg_model <- glm(admit ~ . , data = Train, family = 'binomial')

summary(logreg_model)
```

```
##
## Call:
## glm(formula = admit ~ ., family = "binomial", data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4134  -0.8949  -0.6772   1.2222   2.0431
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.2022     0.3247  -0.623   0.5334
## gre           0.2153     0.1411   1.526   0.1269
## gpa           0.2224     0.1421   1.565   0.1177
## rank2        -0.2949     0.3792  -0.778   0.4368
## rank3        -1.0567     0.4252  -2.485   0.0130 *
## rank4        -1.2841     0.5104  -2.516   0.0119 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 345.55  on 275  degrees of freedom
## Residual deviance: 326.50  on 270  degrees of freedom
## AIC: 338.5
##
## Number of Fisher Scoring iterations: 4
```

```r
#Predicting on test data and calculating the accuracy using confusion matrix
## Model Evaluation
prob_train <- predict(logreg_model,newdata = Test[,-1], type = "response")
preds_train <- ifelse(prob_train > 0.49,1,0) # use 0.5 or 0.49 to get the best accuracy
comp = table(Test$admit,preds_train)
confusionMatrix(comp,positive = "0")
```

```
## Confusion Matrix and Statistics
##
##    preds_train
##      0  1
##   0 77  4
##   1 28 10
##
##                Accuracy : 0.7311
```

```
##                  95% CI : (0.6421, 0.8082)
##     No Information Rate : 0.8824
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2568
##
##  Mcnemar's Test P-Value : 4.785e-05
##
##             Sensitivity : 0.7333
##             Specificity : 0.7143
##          Pos Pred Value : 0.9506
##          Neg Pred Value : 0.2632
##              Prevalence : 0.8824
##          Detection Rate : 0.6471
##    Detection Prevalence : 0.6807
##       Balanced Accuracy : 0.7238
##
##        'Positive' Class : 0
##
```

```r
#From the above model we can see that
  #logistic regression model is predicting the admission rate is 73.11%



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

#Decision Tree
#Using library C50
library(C50)


#Building the model and printing the summary
c5_tree_model <- C5.0(admit~., Train, rules = T)
c5_tree_model
```

```
##
## Call:
## C5.0.formula(formula = admit ~ ., data = Train, rules = T)
##
## Rule-Based Model
## Number of samples: 276
## Number of predictors: 3
##
## Number of Rules: 0
##
## Non-standard options: attempt to group attributes
```

```r
summary(c5_tree_model)
```

```
##
## Call:
## C5.0.formula(formula = admit ~ ., data = Train, rules = T)
```

```
##
##
## C5.0 [Release 2.07 GPL Edition]      Sun Oct 30 21:08:29 2022
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 276 cases (4 attributes) from undefined.data
##
## Rules:
##
## Default class: 0
##
##
## Evaluation on training data (276 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##       0   88(31.9%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     188          (a): class 0
##      88          (b): class 1
##
##
## Time: 0.0 secs
```

```
#Predicting on test data
prob_pred <- predict(c5_tree_model, Test[,-1])
prob_pred
```

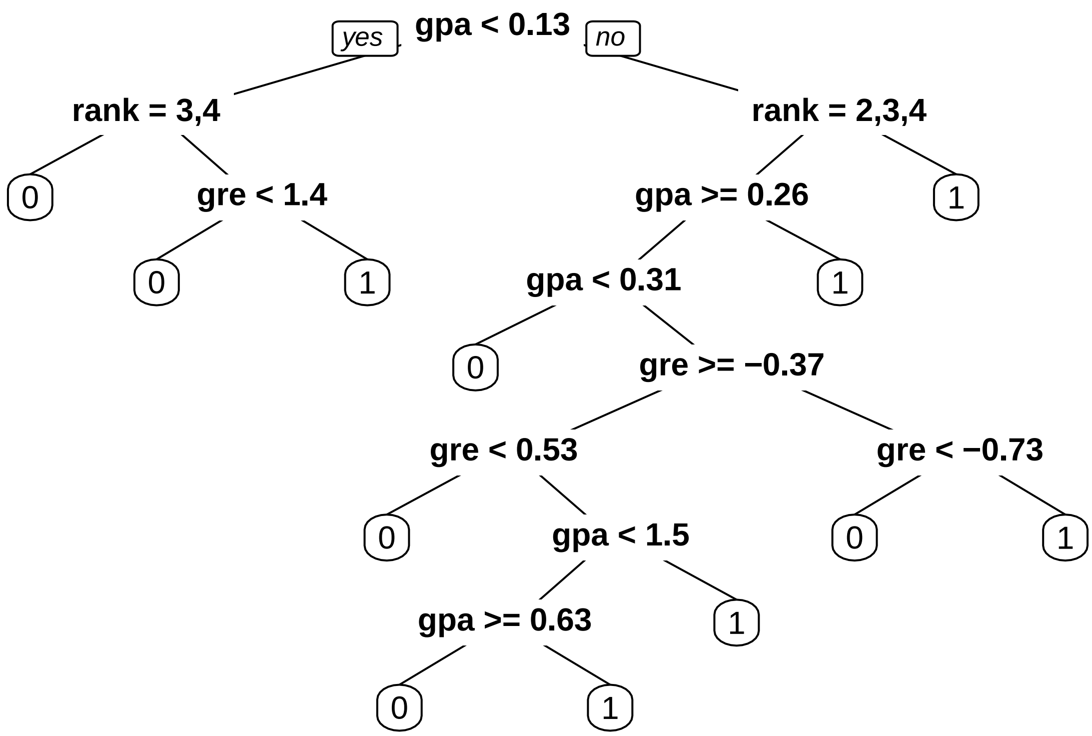
```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```
#Using confusion metric to calculate accuracy
confusionMatrix(prob_pred,Test$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 81 38
##          1  0  0
##
##                Accuracy : 0.6807
##                  95% CI : (0.589, 0.7631)
```

```
##     No Information Rate : 0.6807
##     P-Value [Acc > NIR] : 0.5438
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.947e-09
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6807
##          Neg Pred Value :    NaN
##              Prevalence : 0.6807
##          Detection Rate : 0.6807
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```r
#using Decision Tree (library C50) is giving 68.07% accuracy

#Decision Tree using rpart
#Model Building
rpart_tree_model <- rpart(admit ~ .,
                    data = Train,
                    method = "class")


# display decision tree
prp(rpart_tree_model)
```




```r
# make predictions on the test set
tree_predict <- predict(rpart_tree_model, Test[,-1], type = "class")
tree_predict
```

```
##   3   5   9  10  11  13  15  16  19  23  26  31  36  37  38  44  48  53  55  57
##   1   0   0   0   1   1   1   0   0   0   1   1   0   0   0   0   0   0   0   0
##  58  61  62  64  77  78  80  82  88  89  92  95  98 100 101 102 104 106 107 109
##   0   0   0   0   0   1   1   0   1   0   1   0   0   0   0   0   1   0   1   0
## 124 128 129 130 139 148 152 154 156 158 160 164 165 168 176 178 179 185 187 189
##   0   0   0   0   0   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0
## 193 194 196 203 207 208 213 214 218 220 239 242 247 248 256 260 262 263 271 275
##   0   0   0   1   1   1   0   0   1   0   0   1   0   0   0   0   0   0   0   0
## 277 279 283 286 288 297 301 303 314 315 321 324 325 327 329 330 337 338 340 348
##   0   0   1   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   1   0
## 350 353 355 361 363 365 366 367 368 372 374 379 384 386 388 390 391 393 394
##   0   0   1   1   0   0   0   0   0   0   0   0   1   0   0   0   1   0   0
## Levels: 0 1
```

```r
# evaluate the results
confusionMatrix(tree_predict, as.factor(Test$admit), positive = "0")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

```
##          0 72 20
##          1  9 18
##
##                Accuracy : 0.7563
##                  95% CI : (0.6691, 0.8303)
##     No Information Rate : 0.6807
##     P-Value [Acc > NIR] : 0.04490
##
##                   Kappa : 0.3928
##
##  Mcnemar's Test P-Value : 0.06332
##
##             Sensitivity : 0.8889
##             Specificity : 0.4737
##          Pos Pred Value : 0.7826
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6807
##          Detection Rate : 0.6050
##    Detection Prevalence : 0.7731
##       Balanced Accuracy : 0.6813
##
##        'Positive' Class : 0
##
```

```
#using Decision Tree (rpart) model is giving 75.63% accuracy

### Build a random forest
rf_model <- randomForest(admit ~ ., data=Train, proximity=FALSE,
                         ntree=15, mtry=3, na.action=na.omit)
rf_model
```

```
##
## Call:
##  randomForest(formula = admit ~ ., data = Train, proximity = FALSE,      ntree = 15, mtry = 3, na.ac
##                Type of random forest: classification
##                      Number of trees: 15
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 36.96%
## Confusion matrix:
##     0  1 class.error
## 0 141 47       0.250
## 1  55 33       0.625
```

```
summary(rf_model)
```

```
##                 Length Class  Mode
## call              7    -none- call
## type              1    -none- character
## predicted       276    factor numeric
## err.rate         45    -none- numeric
## confusion         6    -none- numeric
## votes           552    matrix numeric
```

```
## oob.times       276    -none- numeric
## classes           2    -none- character
## importance        3    -none- numeric
## importanceSD      0    -none- NULL
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           14    -none- list
## y               276    factor numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```
#using the model to predict on test data and using confusion matrix to calculate accuracy
rf_pred <- predict(rf_model, newdata=Test[,-1], type='Class')

confusionMatrix(rf_pred, Test$admit, positive = "0")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 27
##          1 11 11
##
##                Accuracy : 0.6807
##                  95% CI : (0.589, 0.7631)
##     No Information Rate : 0.6807
##     P-Value [Acc > NIR] : 0.54380
##
##                   Kappa : 0.173
##
##  Mcnemar's Test P-Value : 0.01496
##
##             Sensitivity : 0.8642
##             Specificity : 0.2895
##          Pos Pred Value : 0.7216
##          Neg Pred Value : 0.5000
##              Prevalence : 0.6807
##          Detection Rate : 0.5882
##    Detection Prevalence : 0.8151
##       Balanced Accuracy : 0.5768
##
##        'Positive' Class : 0
##
```

```
#using Random Forrest model is giving 68.07% accuracy

# build the Support Vector Machines(SVM) model
svm_model<- ksvm(admit ~ ., data = Train,scale = FALSE , C=25)
summary(svm_model)
```

```
## Length  Class   Mode
```

```
##      1   ksvm     S4
```

```r
# Predicting the model results
svm_predict <- predict(svm_model, Test[,-1])

#SVM model accuracy using confusion matrix
confusionMatrix(svm_predict, Test$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 75 30
##          1  6  8
##
##                Accuracy : 0.6975
##                  95% CI : (0.6065, 0.7783)
##     No Information Rate : 0.6807
##     P-Value [Acc > NIR] : 0.3882638
##
##                   Kappa : 0.1639
##
##  Mcnemar's Test P-Value : 0.0001264
##
##             Sensitivity : 0.9259
##             Specificity : 0.2105
##          Pos Pred Value : 0.7143
##          Neg Pred Value : 0.5714
##              Prevalence : 0.6807
##          Detection Rate : 0.6303
##    Detection Prevalence : 0.8824
##       Balanced Accuracy : 0.5682
##
##        'Positive' Class : 0
##
```

```r
#using SVM model is giving 69.75% accuracy


# 10. Determine the accuracy rates for each kind of model
# a) Logistic Regression : 73.11%
# b) Decision Tree (C50) : 68.07%
# c) Decision Tree (rpart) : 75.63%
# d) Random Forest : 68.07%
# e) SVM : 69.75%


# 11. Select the most accurate model
# Decision - Decision Tree using library rpart is giving best accuracy
 # which is 75.63% compared to other models
# So Most accurate model is Decision Tree(Rpart) Model
```

```r
# 12. Identify other Machine learning or statistical techniques
# I have already used Random Forest. Other than random forest,
# we can apply Naive Bayes and Boosting Algorithms.

# Naive Bayes
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```r
naive_bayes_model <- naive_bayes(admit~.,Train)
naive_bayes_model
```

```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = admit ~ ., data = Train)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
##         0         1
## 0.6811594 0.3188406
##
## ---------------------------------------------------------------------------------
##
##  Tables:
##
## ---------------------------------------------------------------------------------
##  ::: gre (Gaussian)
## ---------------------------------------------------------------------------------
##
## gre              0           1
##   mean -0.07402582  0.23935859
##   sd    1.03548586  0.97246981
##
## ---------------------------------------------------------------------------------
##  ::: gpa (Gaussian)
## ---------------------------------------------------------------------------------
##
## gpa             0          1
##   mean -0.0538757  0.1880284
##   sd    1.0229455  1.0078305
##
## ---------------------------------------------------------------------------------
##  ::: rank (Categorical)
## ---------------------------------------------------------------------------------
```

```
##
## rank          0          1
##    1 0.11702128 0.20454545
##    2 0.36702128 0.48863636
##    3 0.32446809 0.21590909
##    4 0.19148936 0.09090909
##
## ---------------------------------------------------------------------------------
```

```
pred_nb <- predict(naive_bayes_model,Test[,-1])
confusionMatrix(pred_nb,Test$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 76 28
##          1  5 10
##
##                Accuracy : 0.7227
##                  95% CI : (0.6332, 0.8008)
##     No Information Rate : 0.6807
##     P-Value [Acc > NIR] : 0.1888403
##
##                   Kappa : 0.24
##
##  Mcnemar's Test P-Value : 0.0001283
##
##             Sensitivity : 0.9383
##             Specificity : 0.2632
##          Pos Pred Value : 0.7308
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6807
##          Detection Rate : 0.6387
##    Detection Prevalence : 0.8739
##       Balanced Accuracy : 0.6007
##
##        'Positive' Class : 0
##
```

```
train.control <- trainControl(method = "repeatedcv", number = 5, repeats = 3)
model_xgbTree <- train(admit ~ .,data=Train, method = "xgbTree",
                       trControl = train.control, verbosity = 0)
print(model_xgbTree)
```

```
## eXtreme Gradient Boosting
##
## 276 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
```

```
## Summary of sample sizes: 222, 221, 220, 221, 220, 221, ...
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  Accuracy
##   0.3  1          0.6               0.50        50      0.6508842
##   0.3  1          0.6               0.50       100      0.6533750
##   0.3  1          0.6               0.50       150      0.6558907
##   0.3  1          0.6               0.75        50      0.6545863
##   0.3  1          0.6               0.75       100      0.6593931
##   0.3  1          0.6               0.75       150      0.6543899
##   0.3  1          0.6               1.00        50      0.6377481
##   0.3  1          0.6               1.00       100      0.6520314
##   0.3  1          0.6               1.00       150      0.6533101
##   0.3  1          0.8               0.50        50      0.6667797
##   0.3  1          0.8               0.50       100      0.6435915
##   0.3  1          0.8               0.50       150      0.6579181
##   0.3  1          0.8               0.75        50      0.6460574
##   0.3  1          0.8               0.75       100      0.6520539
##   0.3  1          0.8               0.75       150      0.6507095
##   0.3  1          0.8               1.00        50      0.6498060
##   0.3  1          0.8               1.00       100      0.6472703
##   0.3  1          0.8               1.00       150      0.6520314
##   0.3  2          0.6               0.50        50      0.6520771
##   0.3  2          0.6               0.50       100      0.6557359
##   0.3  2          0.6               0.50       150      0.6435490
##   0.3  2          0.6               0.75        50      0.6484159
##   0.3  2          0.6               0.75       100      0.6218350
##   0.3  2          0.6               0.75       150      0.6302998
##   0.3  2          0.6               1.00        50      0.6386957
##   0.3  2          0.6               1.00       100      0.6327209
##   0.3  2          0.6               1.00       150      0.6338015
##   0.3  2          0.8               0.50        50      0.6437903
##   0.3  2          0.8               0.50       100      0.6304970
##   0.3  2          0.8               0.50       150      0.6449118
##   0.3  2          0.8               0.75        50      0.6399319
##   0.3  2          0.8               0.75       100      0.6449118
##   0.3  2          0.8               0.75       150      0.6399976
##   0.3  2          0.8               1.00        50      0.6241711
##   0.3  2          0.8               1.00       100      0.6228467
##   0.3  2          0.8               1.00       150      0.6240580
##   0.3  3          0.6               0.50        50      0.6330319
##   0.3  3          0.6               0.50       100      0.6485714
##   0.3  3          0.6               0.50       150      0.6305820
##   0.3  3          0.6               0.75        50      0.6447379
##   0.3  3          0.6               0.75       100      0.6544813
##   0.3  3          0.6               0.75       150      0.6411456
##   0.3  3          0.6               1.00        50      0.6448244
##   0.3  3          0.6               1.00       100      0.6315095
##   0.3  3          0.6               1.00       150      0.6350802
##   0.3  3          0.8               0.50        50      0.6498244
##   0.3  3          0.8               0.50       100      0.6534400
##   0.3  3          0.8               0.50       150      0.6364895
##   0.3  3          0.8               0.75        50      0.6243250
##   0.3  3          0.8               0.75       100      0.6377056
```

```
## 0.3 3 0.8 0.75 150 0.6400425
## 0.3 3 0.8 1.00 50 0.6204209
## 0.3 3 0.8 1.00 100 0.6242360
## 0.3 3 0.8 1.00 150 0.6182820
## 0.4 1 0.6 0.50 50 0.6559548
## 0.4 1 0.6 0.50 100 0.6557568
## 0.4 1 0.6 0.50 150 0.6303399
## 0.4 1 0.6 0.75 50 0.6497379
## 0.4 1 0.6 0.75 100 0.6582027
## 0.4 1 0.6 0.75 150 0.6556918
## 0.4 1 0.6 1.00 50 0.6438087
## 0.4 1 0.6 1.00 100 0.6520539
## 0.4 1 0.6 1.00 150 0.6496296
## 0.4 1 0.8 0.50 50 0.6472936
## 0.4 1 0.8 0.50 100 0.6582027
## 0.4 1 0.8 0.50 150 0.6486813
## 0.4 1 0.8 0.75 50 0.6520531
## 0.4 1 0.8 0.75 100 0.6435458
## 0.4 1 0.8 0.75 150 0.6543699
## 0.4 1 0.8 1.00 50 0.6461889
## 0.4 1 0.8 1.00 100 0.6520539
## 0.4 1 0.8 1.00 150 0.6556253
## 0.4 2 0.6 0.50 50 0.6388520
## 0.4 2 0.6 0.50 100 0.6342007
## 0.4 2 0.6 0.50 150 0.6316450
## 0.4 2 0.6 0.75 50 0.6459492
## 0.4 2 0.6 0.75 100 0.6396697
## 0.4 2 0.6 0.75 150 0.6412570
## 0.4 2 0.6 1.00 50 0.6277625
## 0.4 2 0.6 1.00 100 0.6324587
## 0.4 2 0.6 1.00 150 0.6374619
## 0.4 2 0.8 0.50 50 0.6532668
## 0.4 2 0.8 0.50 100 0.6364510
## 0.4 2 0.8 0.50 150 0.6350184
## 0.4 2 0.8 0.75 50 0.6302301
## 0.4 2 0.8 0.75 100 0.6169857
## 0.4 2 0.8 0.75 150 0.6252734
## 0.4 2 0.8 1.00 50 0.6350794
## 0.4 2 0.8 1.00 100 0.6159027
## 0.4 2 0.8 1.00 150 0.6169160
## 0.4 3 0.6 0.50 50 0.6498068
## 0.4 3 0.6 0.50 100 0.6461905
## 0.4 3 0.6 0.50 150 0.6317981
## 0.4 3 0.6 0.75 50 0.6351908
## 0.4 3 0.6 0.75 100 0.6364727
## 0.4 3 0.6 0.75 150 0.6412530
## 0.4 3 0.6 1.00 50 0.6266178
## 0.4 3 0.6 1.00 100 0.6291077
## 0.4 3 0.6 1.00 150 0.6327882
## 0.4 3 0.8 0.50 50 0.6376583
## 0.4 3 0.8 0.50 100 0.6447603
## 0.4 3 0.8 0.50 150 0.6292200
## 0.4 3 0.8 0.75 50 0.6486155
## 0.4 3 0.8 0.75 100 0.6354089
```

```
##   0.4  3          0.8              0.75       150       0.6366651
##   0.4  3          0.8              1.00        50       0.6172046
##   0.4  3          0.8              1.00       100       0.6232660
##   0.4  3          0.8              1.00       150       0.6220747
##   Kappa
##    3.685579e-02
##    7.534871e-02
##    1.216743e-01
##    2.627343e-02
##    7.990407e-02
##    8.780889e-02
##   -3.917632e-02
##    3.860905e-02
##    5.118630e-02
##    6.364288e-02
##    5.952866e-02
##    1.084836e-01
##    1.002143e-02
##    6.068873e-02
##    9.130938e-02
##   -9.634290e-03
##    1.828988e-02
##    4.189686e-02
##    1.019261e-01
##    1.333316e-01
##    1.227399e-01
##    9.345025e-02
##    5.982175e-02
##    9.212853e-02
##    6.529361e-02
##    7.392589e-02
##    9.559793e-02
##    5.799685e-02
##    7.372039e-02
##    1.193502e-01
##    5.170855e-02
##    1.080348e-01
##    1.065808e-01
##    1.542224e-02
##    5.630381e-02
##    7.500519e-02
##    7.964340e-02
##    1.358438e-01
##    1.036282e-01
##    1.121338e-01
##    1.511013e-01
##    1.404052e-01
##    9.523142e-02
##    9.093560e-02
##    1.074978e-01
##    1.270032e-01
##    1.486643e-01
##    1.181953e-01
##    5.508229e-02
```

```
##    1.158355e-01
##    1.361926e-01
##    3.660436e-02
##    7.677235e-02
##    7.197338e-02
##    8.040067e-02
##    1.178564e-01
##    5.526494e-02
##    4.298114e-02
##    8.940785e-02
##    1.123139e-01
##    4.762013e-04
##    5.563008e-02
##    5.862813e-02
##    3.689814e-02
##    9.612135e-02
##    1.014035e-01
##    5.354440e-02
##    7.950124e-02
##    1.293429e-01
##    3.069677e-05
##    4.192773e-02
##    6.904182e-02
##    8.881188e-02
##    9.973840e-02
##    1.137594e-01
##    9.052183e-02
##    1.097942e-01
##    1.229047e-01
##    4.372097e-02
##    8.909687e-02
##    1.141158e-01
##    1.191130e-01
##    1.086639e-01
##    1.151491e-01
##    5.801564e-02
##    7.065358e-02
##    1.007069e-01
##    6.411381e-02
##    4.156881e-02
##    5.861449e-02
##    1.339567e-01
##    1.312748e-01
##    1.065693e-01
##    9.716368e-02
##    1.247375e-01
##    1.326003e-01
##    7.230958e-02
##    9.559686e-02
##    1.165391e-01
##    8.226904e-02
##    1.309609e-01
##    1.050572e-01
##    1.229370e-01
```

```
##    1.080579e-01
##    1.297676e-01
##    6.259913e-02
##    8.575536e-02
##    8.882034e-02
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##  parameter 'min_child_weight' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 50, max_depth = 1, eta
##  = 0.3, gamma = 0, colsample_bytree = 0.8, min_child_weight = 1 and subsample
##  = 0.5.
```

```
pred_xgboost <- predict(model_xgbTree,Test[,-1])
confusionMatrix(pred_xgboost,Test$admit)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 77 29
##          1  4  9
##
##                Accuracy : 0.7227
##                  95% CI : (0.6332, 0.8008)
##     No Information Rate : 0.6807
##     P-Value [Acc > NIR] : 0.1888
##
##                   Kappa : 0.2271
##
##  Mcnemar's Test P-Value : 2.943e-05
##
##             Sensitivity : 0.9506
##             Specificity : 0.2368
##          Pos Pred Value : 0.7264
##          Neg Pred Value : 0.6923
##              Prevalence : 0.6807
##          Detection Rate : 0.6471
##    Detection Prevalence : 0.8908
##       Balanced Accuracy : 0.5937
##
##        'Positive' Class : 0
##
```

```
# 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

head(clgad)
```

```
##   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
```

```
clgad$Category[clgad$gre < 440] <- "Low"
clgad$Category[clgad$gre >= 440 & clgad$gre < 580] <- "Medium"
clgad$Category[clgad$gre >= 580] <- "High"
head(clgad)
```

```
##   admit gre  gpa ses Gender_Male Race rank Category
## 1     0 380 3.61   1           0    3    3      Low
## 2     1 660 3.67   2           0    2    3     High
## 3     1 800 4.00   2           0    2    1     High
## 4     1 640 3.19   1           1    2    4     High
## 5     0 520 2.93   3           1    2    4   Medium
## 6     1 760 3.00   2           1    1    2     High
```

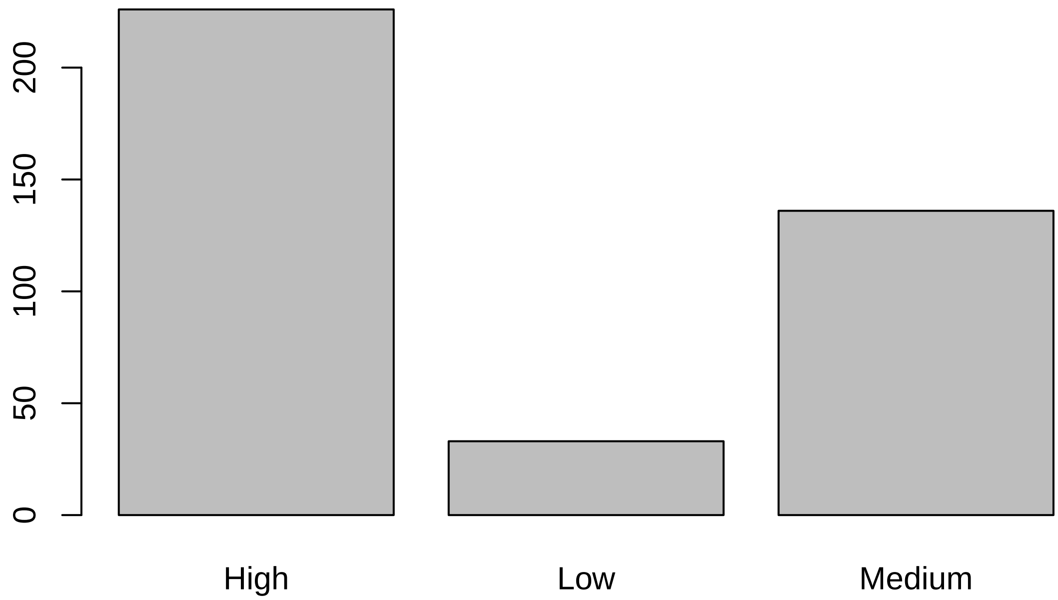
```
clgad$Category <- as.factor(clgad$Category)
str(clgad)
```

```
## 'data.frame':    395 obs. of  8 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 ...
##  $ Category   : Factor w/ 3 levels "High","Low","Medium": 2 1 1 1 3 1 3 2 3 1 ...
```

```
summary(clgad$Category)
```

```
##   High    Low Medium
##    226     33    136
```

```
plot(clgad$Category)
```


0
50
100
150
200
High
Low
Medium