College Admission

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

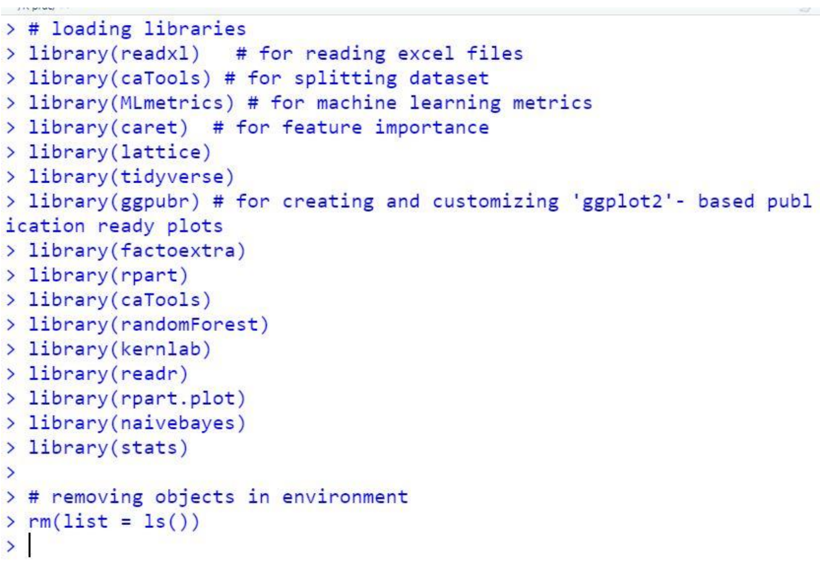
Dataset Description:

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.

Source Code:

```
# =============== Loading Libraries ============

library(readxl)   # for reading excel files

library(caTools) # for splitting dataset

library(MLmetrics) # for machine learning metrics

library(caret)  # for feature importance

library(lattice)

library(tidyverse)

library(ggpubr) # for creating and customizing 'ggplot2'- based publication ready plots

library(factoextra)

library(rpart)

library(caTools)

library(randomForest)

library(kernlab)

library(readr)

library(rpart.plot)

library(naivebayes)

library(stats)

# ============= Removing Existing R-Objects in Environment ===========

rm(list = ls())
```



```
> # loading libraries
> library(readxl)   # for reading excel files
> library(caTools) # for splitting dataset
> library(MLmetrics) # for machine learning metrics
> library(caret)  # for feature importance
> library(lattice)
> library(tidyverse)
> library(ggpubr) # for creating and customizing 'ggplot2'- based publ
ication ready plots
> library(factoextra)
> library(rpart)
> library(caTools)
> library(randomForest)
> library(kernlab)
> library(readr)
> library(rpart.plot)
> library(naivebayes)
> library(stats)
>
> # removing objects in environment
> rm(list = ls())
>
```

===================== Loading Dataset ===============

df <- read.csv("College_admission.csv", header = T)

View(df)

==================== EDA ========================

dim(df)

str(df)

summary(df)

```
~/R prac/
> # loading dataset
> df <- read.csv("College_admission.csv", header = T)
> View(df)
>
>
> # EDA
> dim(df)
[1] 400   7
> str(df)
'data.frame':   400 obs. of  7 variables:
 $ admit      : int  0 1 1 1 0 1 1 0 1 0 ...
 $ gre        : int  380 660 800 640 520 760 560 400 540 700 ...
 $ gpa        : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
 $ ses        : int  1 2 2 1 3 2 2 2 1 1 ...
 $ Gender_Male: int  0 0 0 1 1 1 1 0 1 0 ...
 $ Race       : int  3 2 2 2 2 1 2 2 1 2 ...
 $ rank       : int  3 3 1 4 4 2 1 2 3 2 ...
> summary(df)
     admit             gre             gpa              ses       
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:1.000  
 Median :0.0000   Median :580.0   Median :3.395   Median :2.000  
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :1.992  
 3rd Qu.:1.0000   3rd Qu.:660.0   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.   :0.000   Min.   :1.000   Min.   :1.000  
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:2.000  
 Median :0.000   Median :2.000   Median :2.000  
 Mean   :0.475   Mean   :1.962   Mean   :2.485  
 3rd Qu.:1.000   3rd Qu.:3.000   3rd Qu.:3.000  
 Max.   :1.000   Max.   :3.000   Max.   :4.000  
> 
```

========== Question-1 ===================================

checking and handling NA

sapply(df,function(x) sum(is.na(x))) #no NAs

```
~/R prac/
> # checking and handling NA
> sapply(df,function(x) sum(is.na(x))) #no NAs
      admit         gre         gpa         ses Gender_Male 
          0           0           0           0           0 
       Race        rank 
          0           0 
> |
```

From the result we can say that there are no missing values in the data frame.

========== Question-2 ===================================

checking for outliers

boxplot(df)

boxplot.stats(df$gre)$out

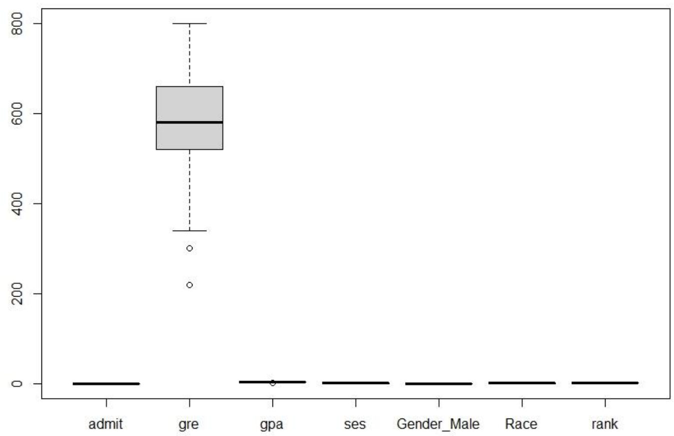
boxplot.stats(df$gpa)$out

df <- df[(df$gre >300 & df$gpa != 2.26),]

boxplot.stats(df$gre)$out

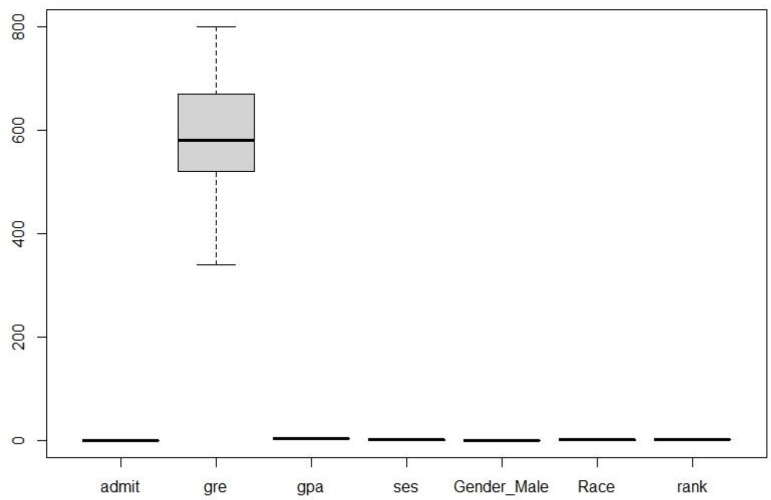
boxplot.stats(df$gpa)$out

boxplot(df)



From the figure , we can say that data contains outliers

```
> boxplot.stats(df$gre)$out
[1] 300 300 220 300
> boxplot.stats(df$gpa)$out
[1] 2.26
> df <- df[(df$gre >300 & df$gpa != 2.26),]
> boxplot.stats(df$gre)$out
integer(0)
> boxplot.stats(df$gpa)$out
numeric(0)
>
```



We remove gre data values which are <=300 and gpa data value which is 2.26.

We only check for gre and gpa features because these are continuous columns else are categorical.

creating grade column

df$grade <- ifelse(df$gre<=440,'Low', ifelse(df$gre>440&df$gre<=580,'Medium', ifelse(df$gre>580,'High','Other')))

table(df$grade)

View(df)

#plotting barplot for grade and admit side-by-side

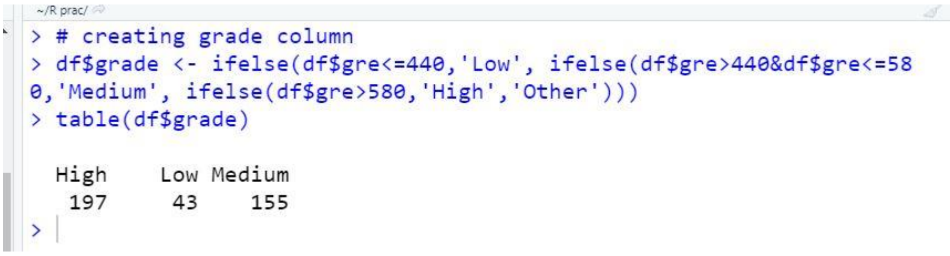
barplot(table(df$admit, df$grade),

beside = T,

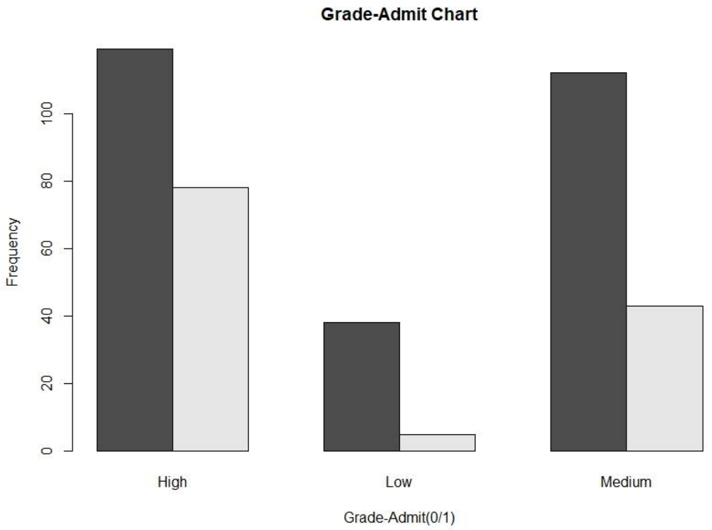
axisnames = T,

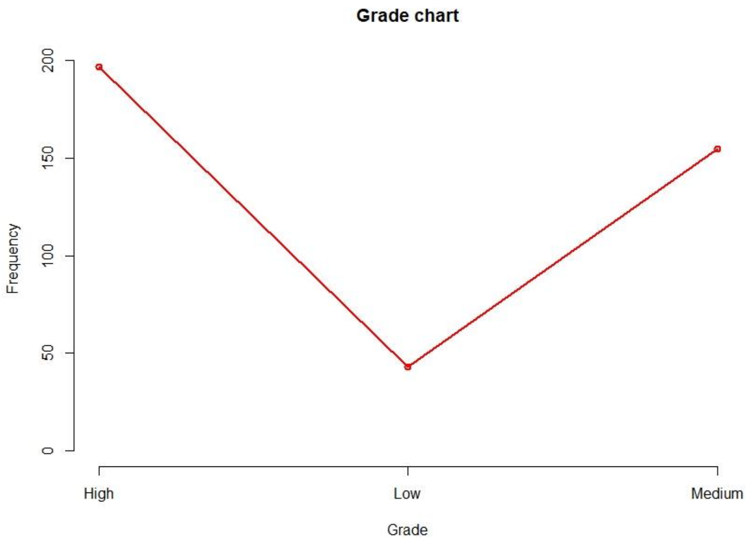
xlab = 'Grade-Admit(0/1)',

```
        ylab = 'Frequency',
        main = 'Grade-Admit Chart'
)
#Line Chart for Grade
plot(table(df$grade),type = "o",col = "red", xlab = "Grade", ylab = "Frequency",
     main = "Grade chart")
```



```
~/R prac/
> # creating grade column
> df$grade <- ifelse(df$gre<=440,'Low', ifelse(df$gre>440&df$gre<=58
0,'Medium', ifelse(df$gre>580,'High','Other')))
> table(df$grade)

  High    Low Medium
   197     43    155
>
```





========== Question-3 ====================================

structure of data-frame and converting required numeric column to factor and vice-verse

str(df)

df <- df %>% mutate_at(c(1,5,6), funs(factor(.)))

df$ses <- factor(df$ses, ordered = T, levels = c(1:3))

df$grade <- factor(df$grade, ordered = T, levels = c('Low','Medium','High'))

df$rank <- factor(df$rank, ordered = T, levels = c(4:1))

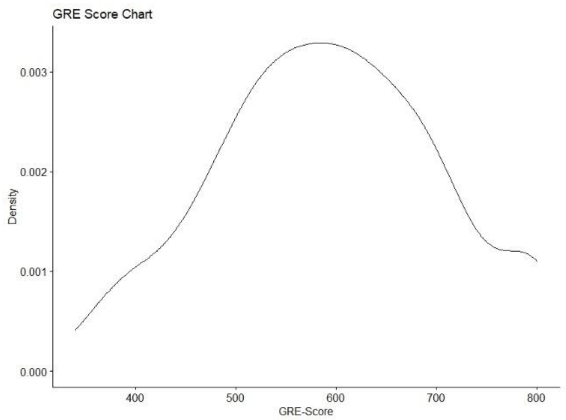
str(df)

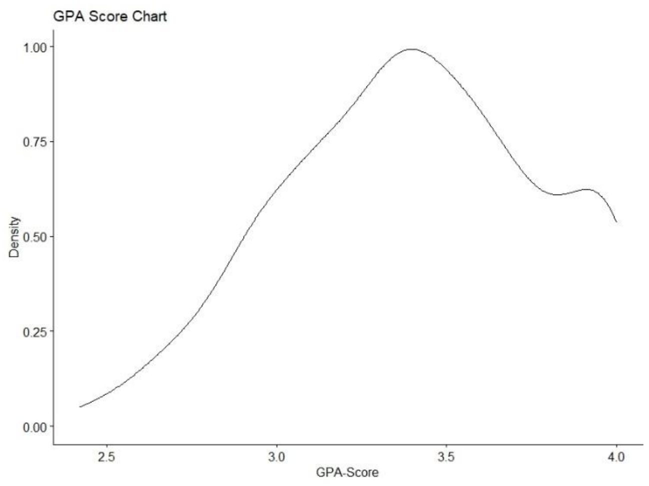
View(df)

```
~/R prac/
> # structure of data-frame and converting required numeric column to
 factor and vice-verse
> str(df)
'data.frame':   395 obs. of  8 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 ...
 $ grade      : chr  "Low" "High" "High" "High" ...
> df <- df %>% mutate_at(c(1,5,6), funs(factor(.)))
> df$ses <- factor(df$ses, ordered = T, levels = c(1:3))
> df$grade <- factor(df$grade, ordered = T, levels = c('Low','Mediu
m','High'))
> df$rank <- factor(df$rank, ordered = T, levels = c(4:1))
> str(df)
'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        : Ord.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       : Ord.factor w/ 4 levels "4"<"3"<"2"<"1": 2 2 4 1 1 3 4
 3 2 3 ...
 $ grade      : Ord.factor w/ 3 levels "Low"<"Medium"<..: 1 3 3 3 2 3
 2 1 2 3 ...
>
```

Converting Admit, SES, Race, Rank,Grade features into factors.

```
# ========== Question-4 ====================================
# checking normality
ggdensity(df$gre,
        xlab = 'GRE-Score',
        ylab = 'Density',
        main = 'GRE Score Chart')
ggdensity(df$gpa,
        xlab = 'GPA-Score',
        ylab = 'Density',
        main = 'GPA Score Chart')
sapply(df[2:3], function(x) shapiro.test(x))
```





From the graphs we can say that data is not normally distributed.

```
~/R prac/
> sapply(df[2:3], function(x) shapiro.test(x))
          gre
statistic 0.9828245
p.value   0.0001223489
method    "Shapiro-Wilk normality test"
data.name "x"
          gpa
statistic 0.9764637
p.value   5.004451e-06
method    "Shapiro-Wilk normality test"
data.name "x"
>
```

In Shapiro-Wilk Test as p -value is not > 0.05 for both gre and gpa we can say that the data is not normally distributed.

========== Question-5 ====================================

Standardization Data

df[2:3] <- sapply(df[2:3], function(x) scale(x, center = T, scale = T))

summary(df)

View(df)

mydf <- df #dataframe for decision-tree

```
> # Standardization Data
> df[2:3] <- sapply(df[2:3], function(x) scale(x, center = T, sc
T))
> summary(df)
 admit        gre                gpa               ses     Gender_Ma
 0:269   Min.   :-2.2511   Min.   :-2.604919   1:130   0:209
 1:126   1st Qu.:-0.6383   1st Qu.:-0.699963   2:137   1:186
         Median :-0.1007   Median : 0.006071   3:128
         Mean   : 0.0000   Mean   : 0.000000
         3rd Qu.: 0.7057   3rd Qu.: 0.725425
         Max.   : 1.8705   Max.   : 1.604636
 Race    rank      grade
 1:140   4: 65   Low   : 43
 2:128   3:119   Medium:155
 3:127   2:150   High  :197
         1: 61


>
```

Scaling data Z = $\frac{(x - \mu)}{\sigma^2}$ **such that mean=0 and standard deviation = 1**

Hence Data is Normalized using scale().

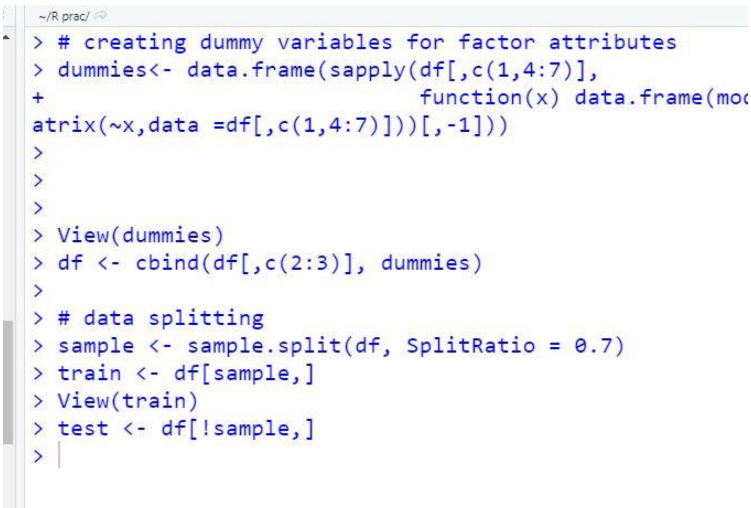
========== Question-6 ====================================

creating dummy variables for factor attributes

dummies<- data.frame(sapply(df[,c(1,4:7)],

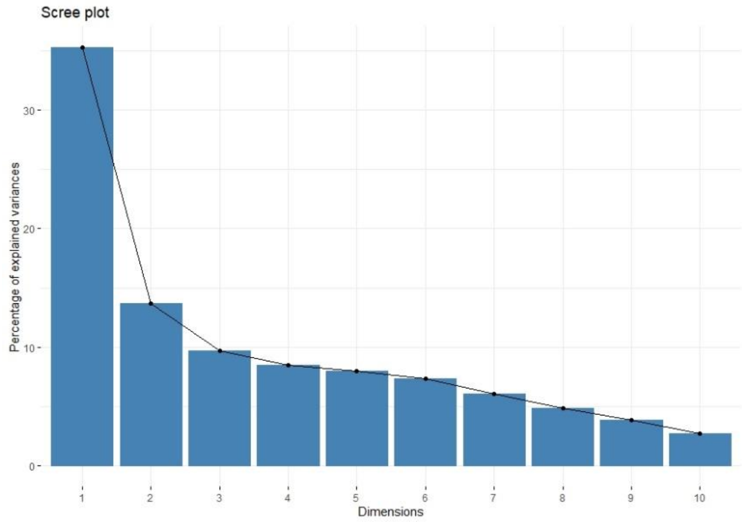
function(x) data.frame(model.matrix(~x,data =df[,c(1,4:7)]))[,-1]))

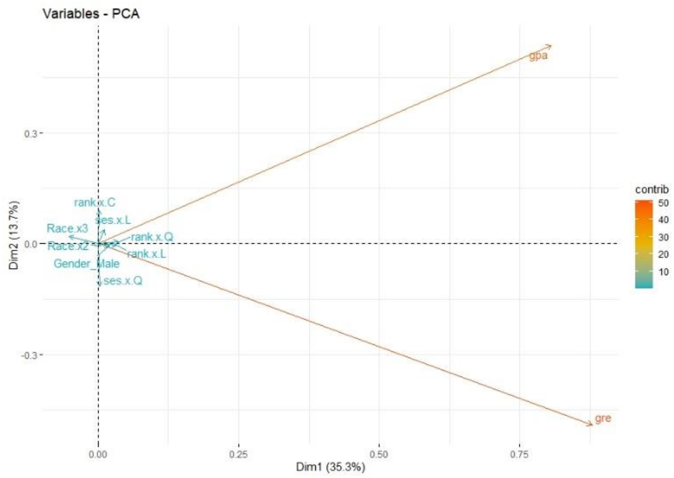
```
View(dummies)
df <- cbind(df[,c(2:3)], dummies)

# data splitting
sample <- sample.split(df, SplitRatio = 0.7)
train <- df[sample,]
View(train)
test <- df[!sample,]

# Feature Importance
pca <- prcomp(train[,-3])
summary(pca)
fviz_eig(pca)
fviz_pca_var(pca,
        col.var = "contrib", # Color by contributions to the PC
        gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
        repel = TRUE     # Avoid text overlapping
)
```



```
~/R prac/
> # creating dummy variables for factor attributes
> dummies<- data.frame(sapply(df[,c(1,4:7)],
+                             function(x) data.frame(moc
atrix(~x,data =df[,c(1,4:7)]))[,-1]))
>
>
>
> View(dummies)
> df <- cbind(df[,c(2:3)], dummies)
>
> # data splitting
> sample <- sample.split(df, SplitRatio = 0.7)
> train <- df[sample,]
> View(train)
> test <- df[!sample,]
>
```

```
> # Feature Importance
> pca <- prcomp(train[,-3])
> summary(pca)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6
Standard deviation     1.1943 0.7439 0.62709 0.58532 0.56777 0.54462
Proportion of Variance 0.3527 0.1368 0.09724 0.08471 0.07971 0.07334
Cumulative Proportion  0.3527 0.4895 0.58676 0.67147 0.75118 0.82452
                           PC7     PC8     PC9    PC10
Standard deviation     0.49639 0.44228 0.39627 0.33265
Proportion of Variance 0.06093 0.04837 0.03883 0.02736
Cumulative Proportion  0.88545 0.93381 0.97264 1.00000
> |
```





From the Figures we can say that GRE and GPA are important features.

From the summary of PCA we can say that PC1 to PC6 Contributes around 82% of importance features Cumulatively.

```
# ========== Question-7 & 8 ====================================
# logistic model
set.seed(123) #for randomness

#building model
lmodel <- glm(formula = admit~.,
              data = train,
              family = 'binomial')
summary(lmodel)

#testing model on test data
pred <- predict(lmodel,
                type = 'response',
                newdata = test[,-3])
test$actual <- factor(ifelse(test$admit == 1,'Yes','No'))
test$pred <- factor(ifelse(pred >= 0.5, 'Yes','No')) #taking cutoff as 0.5
View(test)

#confusion-matrix
confmatrix <- confmatirx <- confusionMatrix(test$pred,
                              test$actual,
                              positive = 'Yes')
confmatirx

#function to get optimal cut-off
perform_fn <- function(cutoff)
{
  pred <- factor(ifelse(pred >= cutoff, "Yes", "No"))
  conf <- confusionMatrix(pred,test$actual, positive = "Yes")
  acc <- conf$overall[1]
```

```
  sens <- conf$byClass[1]
  spec <- conf$byClass[2]
  out <- t(as.matrix(c(sens, spec, cutoff,acc)))
  colnames(out) <- c("sensitivity", "specificity", 'cutoff', 'accuracy')
  return(out)
}



s = seq(.01,.80,length=100)
s

OUT = matrix(0,100,4)
OUT
for(i in 1:100)
{
  OUT[i,] = perform_fn(s[i])
}
OUT

# Let's choose a cutoff value of 0.52070707  for final model
test_cutoff <- factor(ifelse(pred >=0.52070707 , "Yes", "No"))
conf_final <- confusionMatrix(test_cutoff, test$actual, positive = "Yes")
pval <- conf_final$overall[6]
acc <- conf_final$overall[1]
sens <- conf_final$byClass[1]
spec <- conf_final$byClass[2]
pval
acc
sens
spec
```

```
> confmatrix <- confmatirx <- confusionMatrix(test$pred,
+                                 test$actual,
+                                 positive = 'Yes')
> # logistic model
> set.seed(123) #for randomness
> confmatirx
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  87  37
       Yes  8  11

               Accuracy : 0.6853
                 95% CI : (0.6024, 0.7603)
    No Information Rate : 0.6643
    P-Value [Acc > NIR] : 0.3319

                  Kappa : 0.1704

 Mcnemar's Test P-Value : 2.993e-05

            Sensitivity : 0.22917
            Specificity : 0.91579
         Pos Pred Value : 0.57895
         Neg Pred Value : 0.70161
             Prevalence : 0.33566
         Detection Rate : 0.07692
   Detection Prevalence : 0.13287
      Balanced Accuracy : 0.57248

       'Positive' Class : Yes
```

```
Console  Terminal  Jobs
~/R prac/
> # Let's choose a cutoff value of 0.52070707  for final model
> test_cutoff <- factor(ifelse(pred >=0.52070707 , "Yes", "No"))
> conf_final <- confusionMatrix(test_cutoff, test$actual, positive = "Ye
s")
>
> pval <- conf_final$overall[6]
> acc <- conf_final$overall[1]
> sens <- conf_final$byClass[1]
> spec <- conf_final$byClass[2]
>
> pval
AccuracyPValue
     0.5424842
> acc
 Accuracy
0.7202797
> sens
Sensitivity
        0.4
> spec
Specificity
  0.8446602
>
```

Running the model by randomly taking cut-ff value as 0.5 gives 68% accuracy and Sensitivity=0.22 Specificity=0.91

So as to have balance between Sensitivity and Specificity, running model on different cut-off values from 0.01 to 0.8.

Choosing the best optimal Cut-off value from those values.

Best Optimal Cut-off value for this model based on the dataset is 0.5207. With this accuracy increased from 68.53% to 72.02%.

```
# ========== Question- 9, 10, 11, 12 =====================================
#random forest model

#train-control for random forest
control <- trainControl(method="repeatedcv",
                        number=10,
                        repeats=3,
                        savePredictions=TRUE,
                        classProbs=TRUE,
                        summaryFunction = twoClassSummary)
#building model
rfmodel <- rpart(admit~.,
                 data = train,
                 method = 'class')

# make predictions on the test set
predrf <- predict(rfmodel,
                  test,
                  type = 'class')

test$predrf <- factor(ifelse(predrf == 1,'Yes','No'))
#confusion-matrix
rfconmatrix <- confusionMatrix(test$predrf,test$actual,positive = 'Yes')
rfconmatrix
```

```
+                    type = 'class')
> test$predrf <- factor(ifelse(predrf == 1,'Yes','No'))
> #confusion-matrix
> rfconmatrix <- confusionMatrix(test$predrf,test$actual,positive =
s')
> rfconmatrix
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  87  39
       Yes  6  12

               Accuracy : 0.6875
                 95% CI : (0.605, 0.7621)
    No Information Rate : 0.6458
    P-Value [Acc > NIR] : 0.1691

                  Kappa : 0.2

 Mcnemar's Test P-Value : 1.84e-06

            Sensitivity : 0.23529
            Specificity : 0.93548
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.69048
             Prevalence : 0.35417
         Detection Rate : 0.08333
   Detection Prevalence : 0.12500
      Balanced Accuracy : 0.58539

       'Positive' Class : Yes
```

```
# ==================================================================
#decision tree model


dttrain <- mydf[sample,]
dttest <- mydf[!sample,]


#building model
dtmodel <- rpart(admit ~ .,
         data = dttrain,
         method = "class",
         control = rpart.control(minsplit = 500,
                   minbucket = 250,
                   cp = 0.05))


# make predictions on the test set
tree.predict <- predict(dtmodel, dttest, type = "class")


#confusion-matrix
```

confusionMatrix(tree.predict, as.factor(dttest$admit), positive = '1')

```
› # make predictions on the test set
› tree.predict <- predict(dtmodel, dttest, type = "class")
›
› #confusion-matrix
› confusionMatrix(tree.predict, as.factor(dttest$admit), positive = '1')
:onfusion Matrix and Statistics

          Reference
'rediction  0  1
         0 93 51
         1  0  0

               Accuracy : 0.6458
                 95% CI : (0.5619, 0.7237)
    No Information Rate : 0.6458
    P-Value [Acc > NIR] : 0.538

                  Kappa : 0

 Mcnemar's Test P-Value : 2.534e-12

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.6458
             Prevalence : 0.3542
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 1

›
```

==

#naive bayes

#building model

nbmodel <- naive_bayes(admit ~ ., data = dttrain, usekernel = T)

make predictions on the test set

nbpred <- predict(nbmodel, dttest)

#confusion-matrix

test_conf2 <- confusionMatrix(nbpred, factor(dttest$admit),positive = '1')

test_conf2

```
>
> #confusion-matrix
> test_conf2 <- confusionMatrix(nbpred, factor(dttest$admit),positive =
 '1')
> test_conf2
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 78 35
         1 15 16

               Accuracy : 0.6528
                 95% CI : (0.569, 0.7301)
    No Information Rate : 0.6458
    P-Value [Acc > NIR] : 0.46866

                  Kappa : 0.1672

 Mcnemar's Test P-Value : 0.00721

            Sensitivity : 0.3137
            Specificity : 0.8387
         Pos Pred Value : 0.5161
         Neg Pred Value : 0.6903
             Prevalence : 0.3542
         Detection Rate : 0.1111
   Detection Prevalence : 0.2153
      Balanced Accuracy : 0.5762

       'Positive' Class : 1
```

==

Parameters/Model	**Logistic**	**Random Forest**	**Decision Tree**	**Naïve Baye'**
Specificity	0.8446	0.9354	1.0	0.8387
Sensitivity	0.400	0.2352	0	0.3137
P-Value	0.542	0.1691	0.538	0.4686
Accuracy	72.02%	68.75%	64.58%	65.28%

From the above the best model is Logistic Model