```
getwd()
setwd('//home//labsuser')
clg=read.csv('College_admission.csv')

# ***ANALYSIS TASK***

# Categorizing GRE

library(dplyr)

C=rename(clg,GRE=gre)
C$Gre_category=ifelse(C$GRE<=440,'Low',ifelse(C$GRE<=580,'Medium','High')
)
head(C)

table(C$Gre_category)
table(C$admit)

gre_set=filter(C,admit==1) # All three categories of students are taken
into consideration for admission
head(gre_set)

check<-filter(C,admit==1&Gre_category=='Low') # Only 6 students with low
Gre_category are admitted
head(check)

# Insight_1: Out of 6 students, 3 of them are having high socioeconomic
status and the other 3 students are having African-American race.
#           So, SES and Race are also the key drivers for admission

# KEY DRIVERS FOR ADMISSION:
#  => GRE
#  => gpa
#  => ses
#  => Race
#  => rank

# ***PREDICTIVE***

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

# Finding the missing values
sapply(clg, function(x) sum(is.na(x))) # => There's no missing values

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

# Finding outliers
boxplot(clg)
boxplot(clg$admit)
boxplot(clg$gre)   # There are 2 outliers in gre
boxplot(clg$gpa)   # There is 1 outlier in gpa
boxplot(clg$ses)
boxplot(clg$Gender_Male)
boxplot(clg$rank)
boxplot(clg$Race)

quantile(clg$gre,c(0,0.05,0.1,0.25,0.50,0.75,0.90,0.95,0.99,0.995,1.0))

data1 <- clg[clg$gre <800, ]
boxplot(data1$gre)
```

```
data2 <- data1[data1$gre > 220, ]
boxplot(data2$gre)  # Outliers are removed

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

str(clg)
a=as.factor(clg$gpa)

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

set.seed(1234)
dplyr::sample_n(C, 10)

# 1.admit data

library("ggpubr")
ggdensity(C$admit,
          main = "Density plot of admit",
          xlab = "admit")  # density plot

ggqqplot(C$admit)          # Q-Q plot

shapiro.test(C$admit) # Shappiro-Wik normality test
# p<0.05 , so not normally distributed
# For admit, data is not normally distributed

# 2. GRE data
ggdensity(C$GRE,
          main = "Density plot of admit",
          xlab = "GRE")

ggqqplot(C$GRE) # All data points didn't fall inside the reference line

shapiro.test(C$GRE)
#For GRE, data is not normally distributed

# 3. gpa data
ggdensity(C$gpa,
          main = "Density plot of gpa",
          xlab = "gpa")

ggqqplot(C$gpa)

shapiro.test(C$gpa)
#For gpa, data is not normally distributed

# 4. ses data
ggdensity(C$ses,
          main = "Density plot of ses",
          xlab = "ses")

ggqqplot(C$ses)

shapiro.test(C$ses)
#For ses, data is not normally distributed

# 5. Race data
```

```
ggdensity(C$Race,
          main = "Density plot of Race",
          xlab = "Race")

ggqqplot(C$Race)

shapiro.test(C$Race)
#For Race, data is not normally distributed
# >>> Data is not normally distributed <<<

# •   Normalize the data if not normally distributed.

# Normalizing the data using feature scaling
scale_data <- as.data.frame(scale(clg))
head(scale_data)
summary(scale_data) # ( ata is normally distributed)

# •   Use variable reduction techniques to identify significant
variables.
library(caret)
fit<- lm(admit ~.
         , data=scale_data)
summary(fit)
# >>> gre, gpa, and rank are significant variables

# •   Run logistic model to determine the factors that influence the
admission process of a student (Drop insignificant variables)

# Install and load caTools package
install.packages("caTools")
library(caTools)

# Randomly split data
set.seed(88)
split = sample.split(clg$admit, SplitRatio = 0.75)
head(split)

# Create training and testing sets
clgTrain = subset(clg, split == TRUE)
clgTest = subset(clg, split == FALSE)

# Logistic Regression Model
clgLog = glm(admit ~ gre + gpa + rank, data=clgTrain, family=binomial)
summary(clgLog)
# From logistic model, the factors that influence the admission process
of a student are gpa and rank

# Dropping insignificant variables
AdmnLog = glm(admit ~ gre + rank, data=clgTrain, family=binomial)
summary(AdmnLog)

# •   Calculate the accuracy of the model and run validation techniques.

# Make predictions on training set
predictTrain = predict(AdmnLog, type="response")

# Analyze predictions
summary(predictTrain)
tapply(predictTrain, clgTrain$admit, mean)
```

```
# Confusion matrix for threshold of 0.5
table(clgTrain$admit, predictTrain > 0.5)
# Accuracy = 71.6%
# Sensitivity and specificity
# Sensitivity = 21/32  = 0.656
# Specificity = 194/268 = 0.723

# Confusion matrix for threshold of 0.7
table(clgTrain$admit, predictTrain > 0.7)
# Accuracy = 68.3%
# Sensitivity and specificity
# Sensitivity = 0
# Specificity = 205/300 = 0.683


# Confusion matrix for threshold of 0.2
table(clgTrain$admit, predictTrain > 0.2)
# Accuracy = 46.6%
# Sensitivity and specificity
# Sensitivity = 87/239  = 0.364
# Specificity = 53/61 = 0.868

# Install and load ROCR package
install.packages("ROCR")
library(ROCR)

# Prediction function
ROCRpred = prediction(predictTrain, clgTrain$admit)

# Performance function
ROCRperf = performance(ROCRpred, "tpr", "fpr")

# Plot ROC curve
plot(ROCRperf)

# Add colors
plot(ROCRperf, colorize=TRUE)

# Add threshold labels
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1),
text.adj=c(-0.2,1.7))


# Prediction on Test Set
predictTest = predict(AdmnLog, type = "response", newdata = clgTest)
table(clgTest$admit,predictTest >= 0.3)
#Accuracy= 58%

# >>> Acccuracy of this logistic model is 58%

# •   Validation techniques.

# Multicorelation
# VIF => Variance Influence Factor
library(car)
vif(AdmnLog)
# Here, vif is less than 2. So, it's good

# Serial correlation or auto correlation
# DWT => Durbin Watson test
```

```r
library("lmtest")
dwtest(AdmnLog)#Auto corelation
# Here, p-value>0.05, so it's good

# •   Try other modelling techniques like decision tree and SVM and
select a champion model
# Other modelling techniques: KNN , decision tree

# >>> KNN - K Nearest Neighbor <<<

# Installing Packages
install.packages("e1071")   # For statistics and probability theory
install.packages("caTools")
install.packages("class")   # For classification

# Loading package
library(e1071)
library(caTools)
library(class)

# Loading data
head(clg)

# Splitting data into train
# and test data
split <- sample.split(clg, SplitRatio = 0.7)
train_cl <- subset(clg, split == "TRUE")
test_cl <- subset(clg, split == "FALSE")

# Feature Scaling
train_scale <- scale(train_cl[, 1:7]) # Scaling all rows of first four
columns
test_scale <- scale(test_cl[, 1:7])
# NOTE: We do scaling standardization. i.e., converting all values into a
single format
# BACK END FORMULA FOR SCALING : (Actual value-Mean)/Standard deviation
# Values can in both pos and neg

# Fitting KNN Model
# to training dataset
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$admit,
                      k = 5)
classifier_knn
# Here, cl is classification, what to classify? Species
# k=1 -> one neighbor
# Preferably, choose k=5 for stabilization

# Confusion Matrix
cm <- table(test_cl$admit, classifier_knn)
cm
# no errors

# Model Evaluation - Choosing K
# Calculate out of Sample error
misClassError <- mean(classifier_knn != test_cl$admit)
print(paste('Accuracy =', 1-misClassError))
# Accuracy = 99.4% [KNN model]
# >>> Decision tree <<<
library(rpart)
library(rpart.plot)
clg$status<-ifelse(clg$admit==1,'Admitted','Not Admitted')
v <- clg$admit

table(v)
set.seed(522)

# runif function returns a uniform distribution which can be further
conditionally split into 75-25 ratio
clg[, 'train'] <- ifelse(runif(nrow(clg)) < 0.75, 1, 0)

trainSet <- clg[clg$train == 1,]
testSet <- clg[clg$train == 0, ]

trainColNum <- grep('train', names(trainSet))

trainSet <- trainSet[, -trainColNum]
testSet <- testSet[, -trainColNum]

treeFit <- rpart(admit~.,data=trainSet,method = 'class')
print(treeFit)

rpart.plot(treeFit, box.col=c("red", "green"))

Prediction1 <- predict(treeFit,newdata=testSet,type = 'class')

cm <- table(testSet$admit, Prediction1)
cm
misClassError <- mean(Prediction1 != testSet$admit)
print(paste('Accuracy =', 1-misClassError))
# Accuracy=100%

# •   Determine the accuracy rates for each kind of model

# Logistic model      = 58%
# KNN model           = 99.3%
# Decision tree model = 100%

# Here, the campion models are KNN and Decision tree.
# The most accurate model is Decision tree

# ***DESCRIPTIVE***

# Probability of admitted students = 127/(273+127)= 0.3175
# Acceptance rate = 31.75%

Head(gre_set)
gpa1<-filter(gre_set,gpa>=3.5) #[3.5 to 4]
gpa1 #68

gpa2<-filter(gre_set,gpa>=3&gpa<3.5)  #[3 to 3.49]
gpa2 #44

gpa3<-filter(gre_set,gpa>=2.5&gpa<3)  #[2.5 to 2.99]
gpa3 #14

gpa4<-filter(gre_set,gpa>=2&gpa<2.5)  #[2.0 to 2.49]
gpa4 #1
```

```
gpa5<-filter(gre_set,gpa>=1.5&gpa<2)  #[1.5 to 1.99]
gpa5

C$GPA_category<-
ifelse(C$gpa>=3.5,'High',ifelse(C$gpa>=3&C$gpa<3.5,'Medium','Low'))
head(C)

# Insight_2: Only students with having GPA of atleast 2.5 are admitted
# GPA Categories: 3.5-4.0  => 'High'   - 68 students
#                 3.0-3.49 => 'Medium' - 44 students
#                 2.0-2.99 => 'Low'    - 15 students
```