USE CASE STUDY REPORT

Group No.: Group 19

Student Names: Krish Maniar and Yash Jain

The goal of the study was to classify the liver patient data and classify the datapoints as healthy and unhealthy. The data was obtained from UCI machine learning repository. Data was cleaned before spliting into training and testing data. The classification techniques used were KNN, Logistic Regression and CART. CART was found to be the most effective technique with and accuracy of 81%

I. Background and Introduction

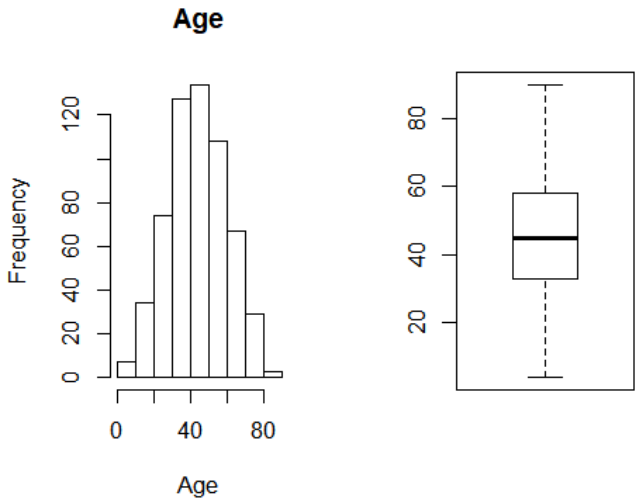
1. The problem: Liver disease affects 1 in 5 people in India. About 1 million patients are newly diagnosed every year in India. It is a silent disease meaning that up to 50% of individuals with liver disease have no visible symptoms. The most common symptoms are nonspecific and may include fatigue and tiredness and thus doctors need time to go through the reports and diagnose the patients as positive or negative.
2. The goal of your study: To predict if the patient under consideration needs to be diagnosed for liver disease or not based on the levels of compounds like bilirubin, albumin, proteins, alkaline phosphatase. Use these patient records to determine which patients have liver disease and which ones do not to reduce burden on doctors·
3. The possible solution: Since the problem under scrutiny is a classification problem, the following three models are used: KNN, Logistic Regression, CART and SVM.

II. Data Exploration and Visualization

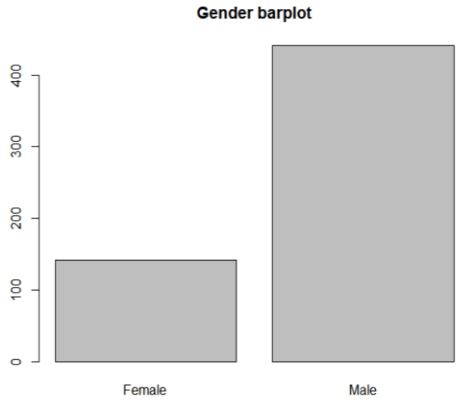
The sourced data contains 11 variables out of which the target variable is chosen as ‘Dataset’. 4 null values are observed in the column ‘Albumin_and_Globulin_Ratio’. The corresponding records are eventually cleaned.

Observing data distribution using histograms, barplots and boxplots:

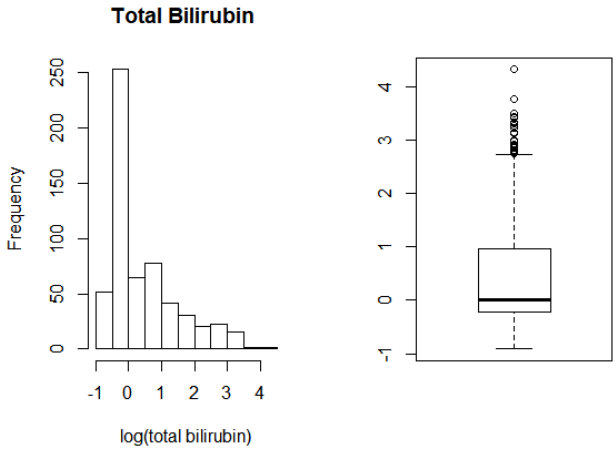
- Age



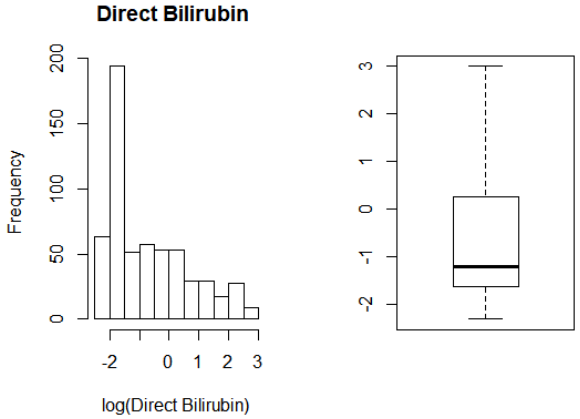
- Gender



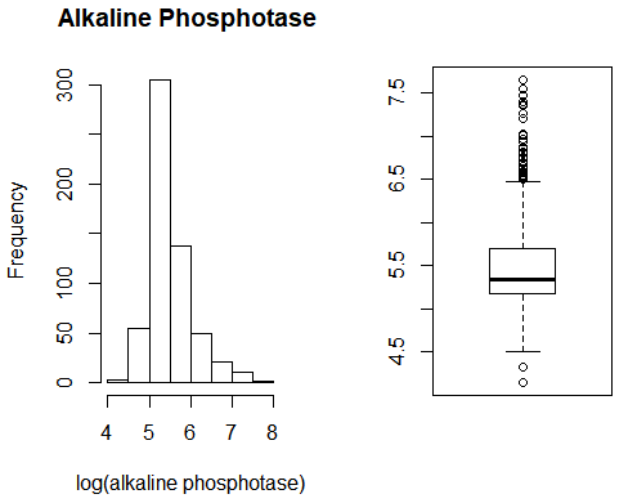
- Total Bilirubin



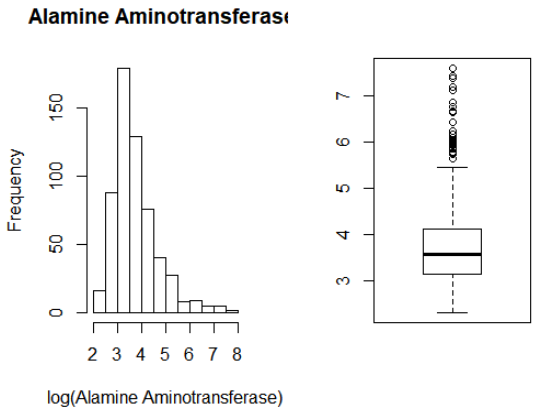
- Direct Bilirubin



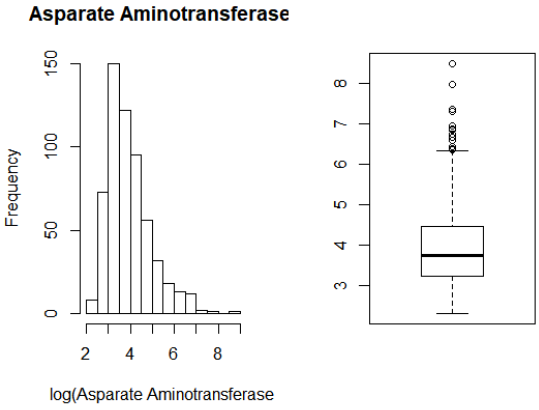
- Alkaline Phosphotase



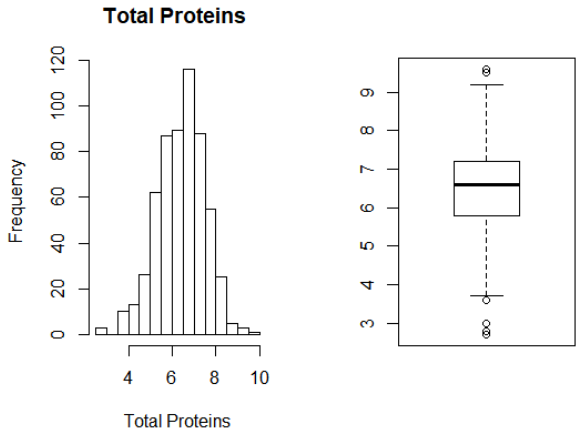
- Alamline Aminotransferase



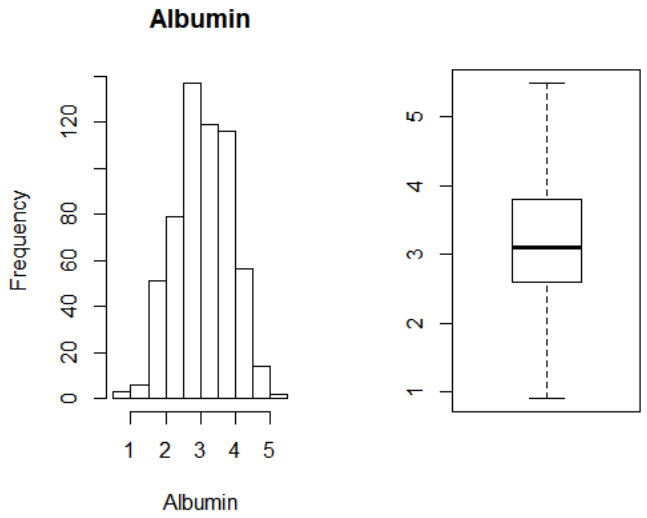
- Asparate Amintransferase



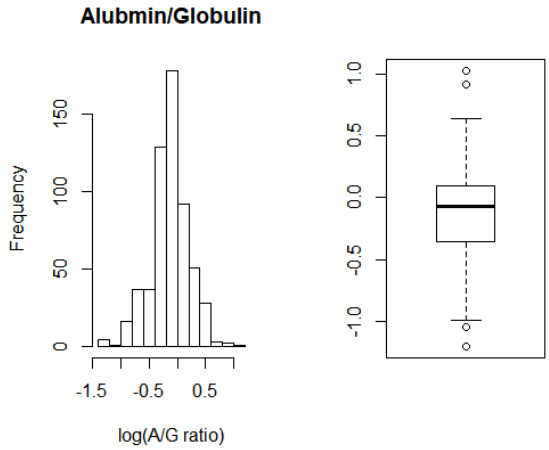
- Total Proteins



- Albumin

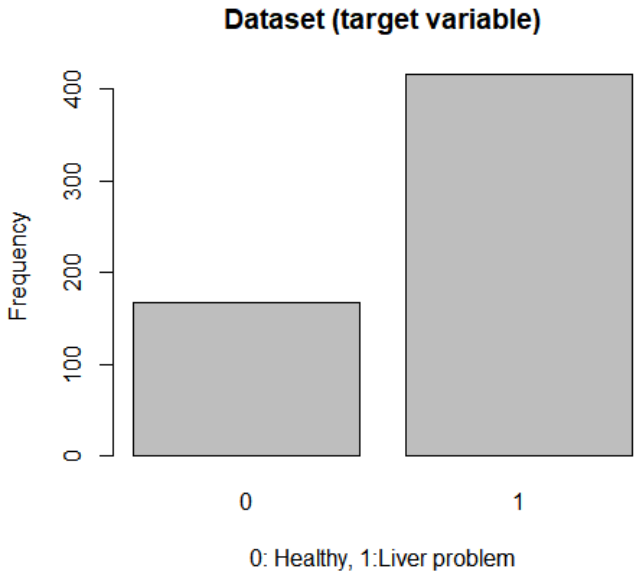


- Albumin and Globulin ratio



- Dataset (target variable)

```
#Change 2 to 0 in target variable
liver_data[liver_data$Dataset == 2,]$Dataset <- 0
```



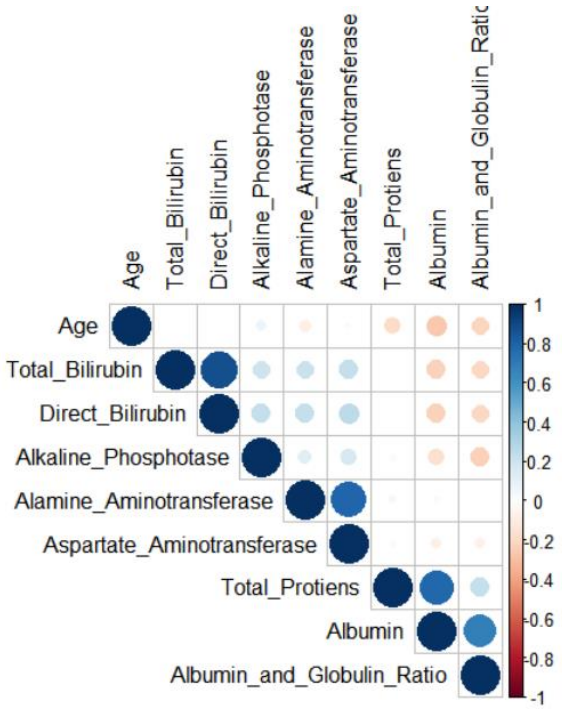
III. Data Preparation and Preprocessing

Data Summary & Structure:

```
> summary(liver_data)
      Age            Gender          Total_Bilirubin  Direct_Bilirubin Alkaline_Phosphotase
 Min.   : 4.00   Length:583         Min.   : 0.400   Min.   : 0.100   Min.   :  63.0
 1st Qu.:33.00   Class :character   1st Qu.: 0.800   1st Qu.: 0.200   1st Qu.: 175.5
 Median :45.00   Mode  :character   Median : 1.000   Median : 0.300   Median : 208.0
 Mean   :44.75                      Mean   : 3.299   Mean   : 1.486   Mean   : 290.6
 3rd Qu.:58.00                      3rd Qu.: 2.600   3rd Qu.: 1.300   3rd Qu.: 298.0
 Max.   :90.00                      Max.   :75.000   Max.   :19.700   Max.   :2110.0

 Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens     Albumin
 Min.   :  10.00          Min.   :  10.0             Min.   :2.700   Min.   :0.900
 1st Qu.:  23.00          1st Qu.:  25.0             1st Qu.:5.800   1st Qu.:2.600
 Median :  35.00          Median :  42.0             Median :6.600   Median :3.100
 Mean   :  80.71          Mean   : 109.9             Mean   :6.483   Mean   :3.142
 3rd Qu.:  60.50          3rd Qu.:  87.0             3rd Qu.:7.200   3rd Qu.:3.800
 Max.   :2000.00          Max.   :4929.0             Max.   :9.600   Max.   :5.500

 Albumin_and_Globulin_Ratio    Dataset
 Min.   :0.3000             Min.   :0.0000
 1st Qu.:0.7000             1st Qu.:0.0000
 Median :0.9300             Median :1.0000
 Mean   :0.9471             Mean   :0.7136
 3rd Qu.:1.1000             3rd Qu.:1.0000
 Max.   :2.8000             Max.   :1.0000
 NA's   :4
>
```

Correlation:

- Positive correlations are displayed in blue and negative correlations in red color.
- Color intensity and the size of the circle are proportional to the correlation coefficients.

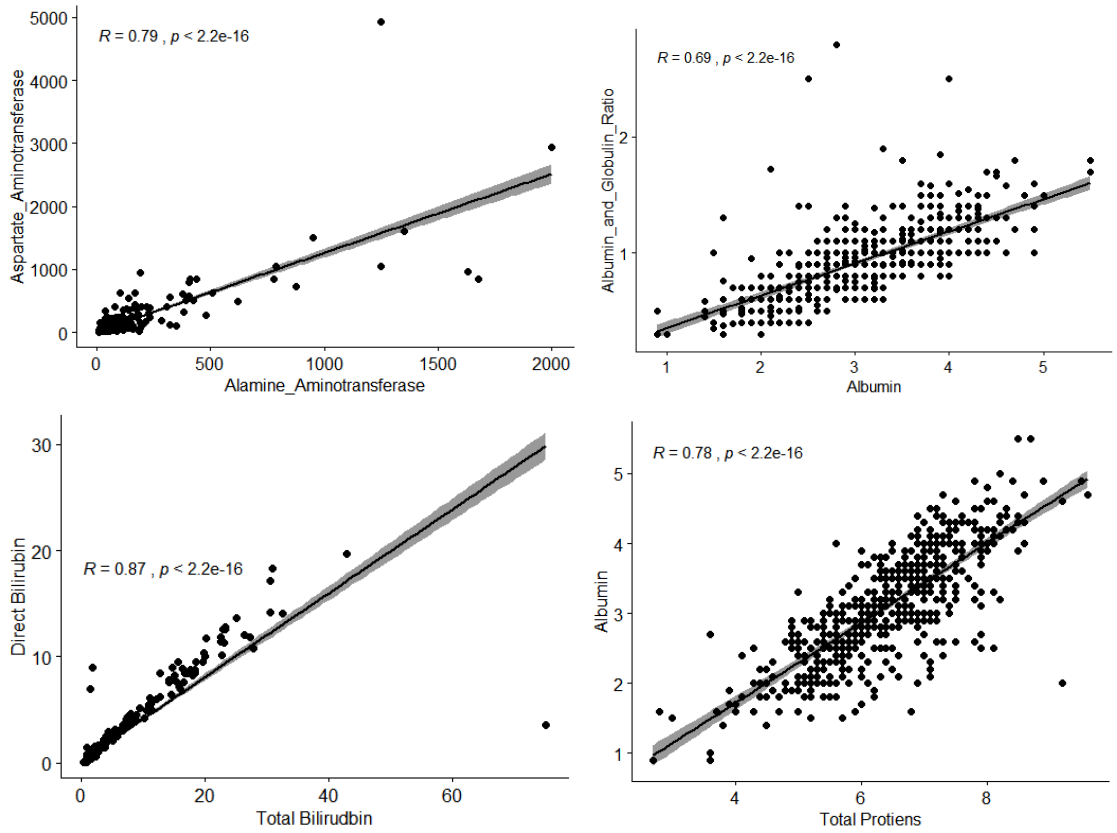


	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase
Age	1.000000000	0.011000374	6.784303e-03	0.07887835
Total_Bilirubin	0.011000374	1.000000000	8.744810e-01	0.20573917
Direct_Bilirubin	0.006784303	0.874480969	1.000000e+00	0.23400757
Alkaline_Phosphotase	0.078878350	0.205739173	2.340076e-01	1.00000000
Alamine_Aminotransferase	-0.087799162	0.213375493	2.331801e-01	0.12477671
Aspartate_Aminotransferase	-0.020498946	0.237323055	2.570224e-01	0.16657999
Total_Protiens	-0.186248122	-0.007905923	3.270877e-05	-0.02706202
Albumin	-0.264210935	-0.222086570	-2.284092e-01	-0.16341865
Albumin_and_Globulin_Ratio	-0.216408346	-0.206267186	-2.001247e-01	-0.23416650

	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens
Age	-0.08779916	-0.02049895	-1.862481e-01
Total_Bilirubin	0.21337549	0.23732305	-7.905923e-03
Direct_Bilirubin	0.23318008	0.25702239	3.270877e-05
Alkaline_Phosphotase	0.12477671	0.16657999	-2.706202e-02
Alamine_Aminotransferase	1.00000000	0.79186215	-4.243210e-02
Aspartate_Aminotransferase	0.79186215	1.00000000	-2.575101e-02
Total_Protiens	-0.04243210	-0.02575101	1.000000e+00
Albumin	-0.02865750	-0.08491457	7.831122e-01
Albumin_and_Globulin_Ratio	-0.00237499	-0.07003983	2.348872e-01

	Albumin	Albumin_and_Globulin_Ratio
Age	-0.26421094	-0.21640835
Total_Bilirubin	-0.22208657	-0.20626719
Direct_Bilirubin	-0.22840915	-0.20012469
Alkaline_Phosphotase	-0.16341865	-0.23416650
Alamine_Aminotransferase	-0.02865750	-0.00237499
Aspartate_Aminotransferase	-0.08491457	-0.07003983
Total_Protiens	0.78311217	0.23488718
Albumin	1.00000000	0.68963234
Albumin_and_Globulin_Ratio	0.68963234	1.00000000

Variable Selection:



Either variable from the pair of the 4 pairs can be eliminated since they are highly correlated and have linear relationship.

So the final variables selected are:

```
> colnames(liver_data3)
[1] "Age"                       "Total_Bilirubin"           "Alamine_Aminotransferase"
[4] "Total_Protiens"            "Albumin"                   "Albumin_and_Globulin_Ratio"
[7] "Dataset"
```

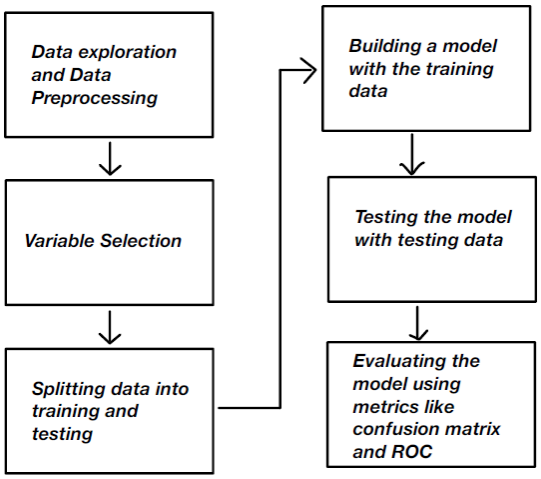
Data Pre-processing:

- Non-numeric data types ‘Gender’ and target variable ‘Dataset’ were converted to factors
- Null values were checked for and the corresponding records were eliminated
- Training and Testing datasets were created with a split ratio of 7:3

```
##Partitioning the dataset

# 70% of the sample size
smp_size <- floor(0.70 * nrow(liver_data3))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(liver_data3)), size = smp_size)
train_data <- liver_data3[train_ind, ]
test_data <- liver_data3[-train_ind, ]
```

IV. Data Mining Techniques and Implementation

Our problem statement is a classification type of problem. So accordingly, we had used classification techniques like KNN, Logistic Regression and Classification trees. SVM was also attempted but it was found out that the dataset istn't suitable for SVM since there didn't exists any hyperplane that could separate the class with substantial accuracy.



1. KNN: K Nearest Neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its K neighbors. The K values we selected are odd values as even values doesn't break ties.

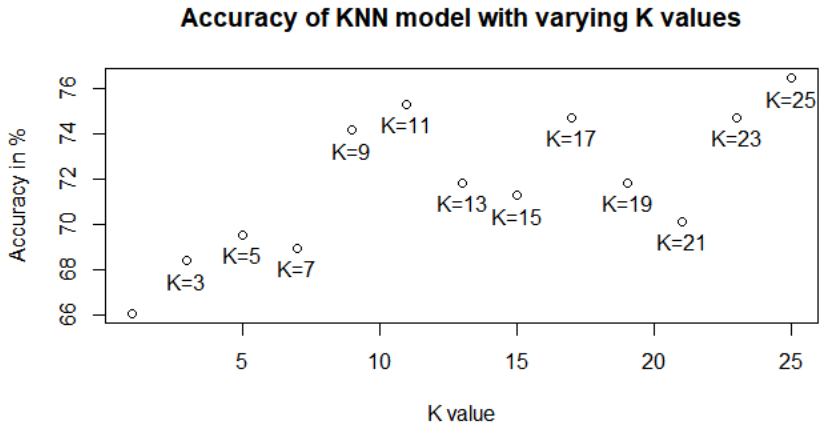
2. Logistic Regression: Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

3. CART: it uses Tree structures to for classification and regression jobs. A classification tree is an algorithm where the target variable is fixed or categorical. The algorithm is then used to identify the "class" within which a target variable would most likely fall.

V. Performance Evaluation

1. KNN

- K=25 was selected as the K value for knn model which resulted in accuracy of 77.011%



- Confusion Matrix:

```
> #Metrics for best K value
> confusionMatrix(k25, test_knn_label)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  11  12
         1  28 123

               Accuracy : 0.7701
                 95% CI : (0.7004, 0.8304)
    No Information Rate : 0.7759
    P-Value [Acc > NIR] : 0.61334

                  Kappa : 0.2262

 Mcnemar's Test P-Value : 0.01771

            Sensitivity : 0.28205
            Specificity : 0.91111
         Pos Pred Value : 0.47826
         Neg Pred Value : 0.81457
             Prevalence : 0.22414
         Detection Rate : 0.06322
   Detection Prevalence : 0.13218
      Balanced Accuracy : 0.59658

       'Positive' Class : 0
```

```
> error_rate
[1] 0.2298851
> false_pos_rate <- table25[2,1]/(table25[2,1] + table25[2,2])
> false_pos_rate
[1] 0.1854305
```

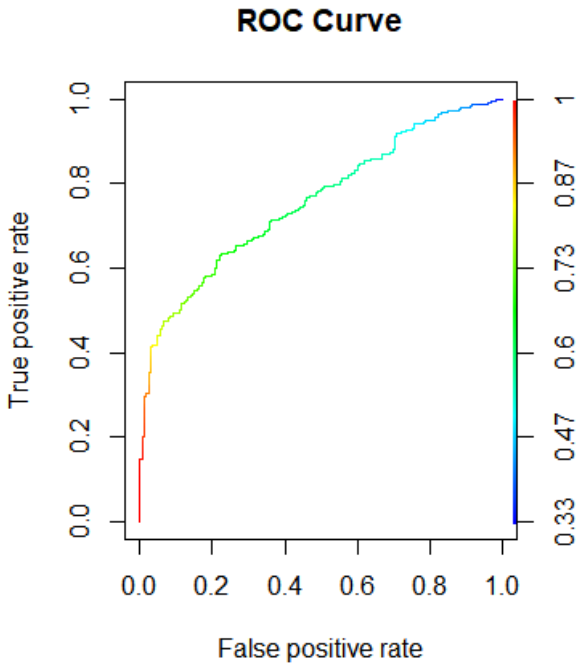
2. Logistic Regression

Confusion matrix:

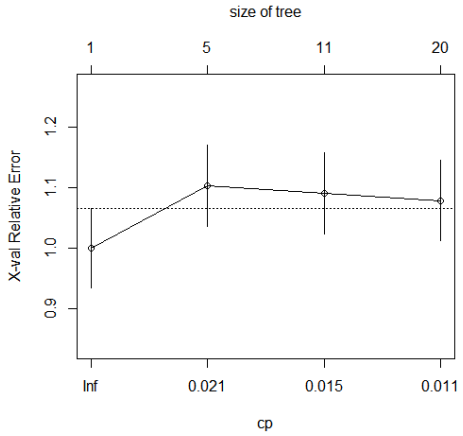
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  40 125
         1  28 386

               Accuracy : 0.7358
                 95% CI : (0.6978, 0.7712)
    No Information Rate : 0.8826
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2123

 Mcnemar's Test P-Value : 8.417e-15

            Sensitivity : 0.58824
            Specificity : 0.75538
         Pos Pred Value : 0.24242
         Neg Pred Value : 0.93237
             Prevalence : 0.11744
         Detection Rate : 0.06908
   Detection Prevalence : 0.28497
      Balanced Accuracy : 0.67181

       'Positive' Class : 0
```

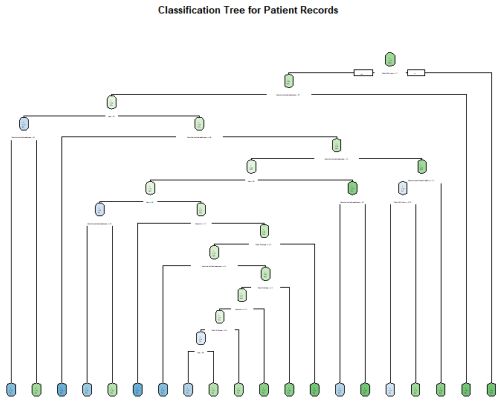
```
> error_rate <- (table_lr[1,2] +  table_lr[2,1]) / 579
> error_rate
[1] 0.2642487
```

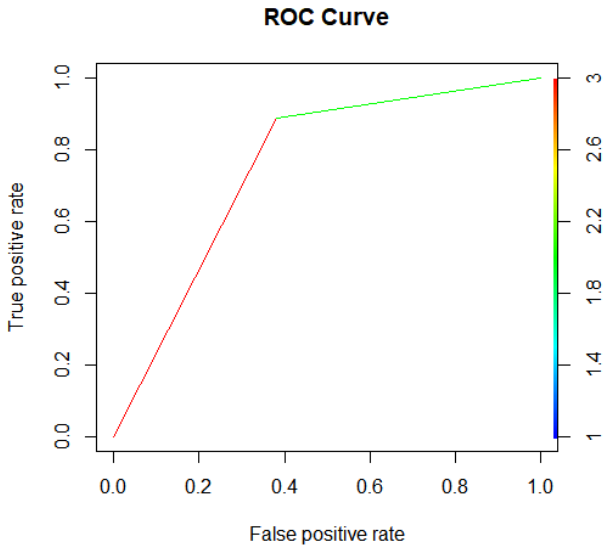
```
> false_pos_rate <- table_lr[2,1]/(table_lr[2,1] + table_lr[2,2])
> false_pos_rate
[1] 0.06763285
```



3.CART


size of tree
1
5
11
20
X-val Relative Error
0.9
1.0
1.1
1.2
Inf
0.021
0.015
0.011
cp


Classification Tree for Patient Records

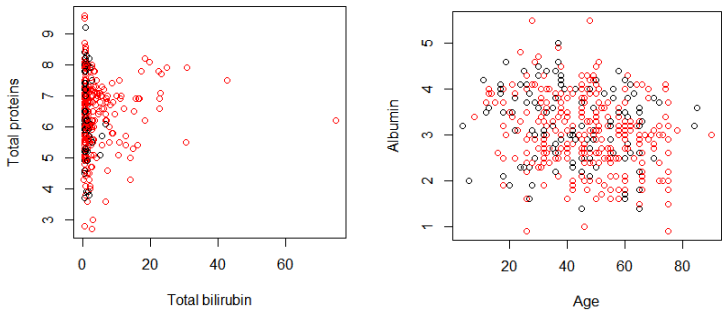

ROC Curve
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
1
1.4
1.8
2.2
2.6
3

Confusion matrix

```
> #Confusion Matrix:
> confusionMatrix(dtpred,as.factor(patients$Diagnosis))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 102  47
         1  63 367

               Accuracy : 0.81
                 95% CI : (0.7756, 0.8412)
    No Information Rate : 0.715
    P-Value [Acc > NIR] : 9.205e-08

                  Kappa : 0.5198

 Mcnemar's Test P-Value : 0.1527

            Sensitivity : 0.6182
            Specificity : 0.8865
         Pos Pred Value : 0.6846
         Neg Pred Value : 0.8535
             Prevalence : 0.2850
         Detection Rate : 0.1762
   Detection Prevalence : 0.2573
      Balanced Accuracy : 0.7523

       'Positive' Class : 0
```

```
> table_cart <- table(dtpred,as.factor(patients$Diagnosis))
> error_rate <- (table_cart[1,2] +  table_cart[2,1]) / 579
> error_rate
[1] 0.1899827
> false_pos_rate <- table_cart[2,1]/(table_cart[2,1] + table_cart[2,2])
> false_pos_rate
[1] 0.1465116
```

4. SVM(failed)



-
- Red is class 1, Black is 0
- SVM couldn't work on this dataset with a good accuracy.
- All the datapoints were being classified into 1 class since there is no clear distinction between predictor values of classes

VI. Discussion and Recommendation

- Data was cleaned and selection of variables was done
- Data was split into 70:30 to overcome the lack of data and give better accuracy.
- Based on our study, we find that Classification and Regression Tress provide us with the most accurate models to classify patient records. It gives us an accuracy of 81%
- To improve the performance of the models the most important thing we can do is gather more data., add more features, experiment with other advanced models..

VII. Summary

Summary of results:

Technique	Accuracy	Sensitivity	Specificity	Error rate	False positive rate
KNN	0.7701	0.282	0.991	0.220	0.185
Logistic Regression	0.7358	0.588	0.753	0.264	0.067
CART	0.81	0.618	0.886	0.189	0.146

Appendix: R Code for use case study

```
library(ggplot2)
library(readr)
library(psych)
library(corrplot)
library(ggpubr)
library(caret)
library(class)
library(gmodels)
library(plyr)

#########################################################################################
###Data Exploration and Preprocessing
#########################################################################################

liver_data <- read_csv("indian_liver_patient.csv")
str(liver_data)

#Change 2 to 0 in target variable
liver_data[liver_data$Dataset == 2,]$Dataset <- 0
str(liver_data)


### Data EXPLORATION
str(liver_data)
summary(liver_data)

#Age
par(mfrow=c(1,2))
hist(liver_data$Age, main = 'Age', xlab = "Age")
boxplot(liver_data$Age)
par(mfrow=c(1,1))

#Gender
par(mfrow=c(1,2))
```

```
barplot(table(liver_data$Gender), main= "Gender barplot")
par(mfrow=c(1,1))

#Total Bilirubin
par(mfrow=c(1,2))
hist(log(liver_data$Total_Bilirubin), main="Total Bilirubin ", xlab="log(total bilirubin)")
boxplot(log(liver_data$Total_Bilirubin))
par(mfrow=c(1,1))

#Direct Bilirubin
par(mfrow=c(1,2))
hist(log(liver_data$Direct_Bilirubin), main = "Direct Bilirubin ", xlab="log(Direct Bilirubin)")
boxplot(log(liver_data$Direct_Bilirubin))
par(mfrow=c(1,1))

#Alkaline Phosphotase
par(mfrow=c(1,2))
hist(log(liver_data$Alkaline_Phosphotase), main="Alkaline Phosphotase ", xlab="log(alkaline
phosphotase)")
boxplot(log(liver_data$Alkaline_Phosphotase))
par(mfrow=c(1,1))

#Alamline Aminotransferase
par(mfrow=c(1,2))
hist(log(liver_data$Alamine_Aminotransferase), main="Alamine Aminotransferase ",
xlab="log(Alamine Aminotransferase)")
boxplot(log(liver_data$Alamine_Aminotransferase))
par(mfrow=c(1,1))

#Asparate Amintransferase
par(mfrow=c(1,2))
hist(log(liver_data$Aspartate_Aminotransferase),main="Asparate Aminotransferase ",
xlab="log(Asparate Aminotransferase")
boxplot(log(liver_data$Aspartate_Aminotransferase))
par(mfrow=c(1,1))

#Total Proteins
par(mfrow=c(1,2))
hist(liver_data$Total_Protiens,main="Total Proteins ",xlab="Total Proteins")
boxplot(liver_data$Total_Protiens)
```

```
par(mfrow=c(1,1))

#Albumin
par(mfrow=c(1,2))
hist(liver_data$Albumin, main="Albumin", xlab="Albumin")
boxplot(liver_data$Albumin)
par(mfrow=c(1,1))

#Albumin and Globulin ratio
par(mfrow=c(1,2))
hist(log(liver_data$Albumin_and_Globulin_Ratio),main="Alubmin/Globulin", xlab="log(A/G
ratio)")
boxplot(log(liver_data$Albumin_and_Globulin_Ratio))
par(mfrow=c(1,1))

#Dataset(target variable)
par(mfrow=c(1,1))
barplot(table(liver_data$Dataset), main="Dataset (target variable)",
        xlab = "0: Healthy, 1:Liver problem", ylab = "Frequency")
par(mfrow=c(1,1))

#In column 'Dataset':1 <- liver patient, 0 <- healthy
table(liver_data$Dataset)



### Data Pre-processing
#Change non-numeric Gender to factor
liver_data$Gender <- factor(liver_data$Gender)
liver_data$Dataset <- factor(liver_data$Dataset)

#Check for null values
summary(liver_data)

#To remove the rows with missing data from liver_data
liver_data2 <- liver_data[complete.cases(liver_data), ]

#To verify if the null value records are removed
str(liver_data2)
summary(liver_data2)
```

```
#To check if attributes have correlation
cor_matrix <- cor(liver_data2[,-c(2,11)]) # Non-numeric fields are not considered
cor_matrix

#Positive correlations are displayed in blue and negative correlations in red color.
#Color intensity and the size of the circle are proportional to the correlation coefficients.
#In the right side of the correlogram,
#the legend color shows the correlation coefficients and the corresponding colors
corrplot(cor_matrix,title = "Correlation Matrix",tl.col = "black",type = "upper")


#To check for linear relationship b/w two variables with high correlation
par(mfrow=c(2,2))
ggscatter(data = liver_data2, x = "Total_Bilirubin", y = "Direct_Bilirubin",
      add = "reg.line", conf.int = TRUE,
      cor.coef = TRUE, cor.method = "pearson",
      xlab = "Total Bilirudbin", ylab = "Direct Bilirubin")

ggscatter(data = liver_data2, x = "Total_Protiens", y = "Albumin",
      add = "reg.line", conf.int = TRUE,
      cor.coef = TRUE, cor.method = "pearson",
      xlab = "Total Protiens", ylab = "Albumin")

ggscatter(data = liver_data2, x = "Alamine_Aminotransferase", y =
"Aspartate_Aminotransferase",
      add = "reg.line", conf.int = TRUE,
      cor.coef = TRUE, cor.method = "pearson",
      xlab = "Alamine_Aminotransferase", ylab = "Aspartate_Aminotransferase")

ggscatter(data = liver_data2, x = "Albumin", y = "Albumin_and_Globulin_Ratio",
      add = "reg.line", conf.int = TRUE,
      cor.coef = TRUE, cor.method = "pearson",
      xlab = "Albumin", ylab = "Albumin_and_Globulin_Ratio")
par(mfrow=c(1,1))


#Select the columns we need as predictors
str(liver_data2)
liver_data3 <- liver_data2[,c(1,3,6,8,9,10,11)]
colnames(liver_data3)
```

```
##Partitioning the dataset

# 70% of the sample size
smp_size <- floor(0.70 * nrow(liver_data3))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(liver_data3)), size = smp_size)
train_data <- liver_data3[train_ind, ]
test_data <- liver_data3[-train_ind, ]

str(train_data)
str(test_data)
str(liver_data3)
```

```
################################################################################
##############
### K - Nearest Neighbours
################################################################################
##############

train_knn <- train_data
test_knn <- test_data
str(train_knn)
str(test_knn)

#Our target variable is 'Dataset' variable
#To remove target variable from training and testing data
train_knn2 <- train_knn[, -7]
train_knn2 <- scale(train_knn2)
test_knn2 <- test_knn[ ,-7]
test_knn2 <- scale(test_knn2)
#To check if traget variable has been removed
str(train_knn2)
str(test_knn2)

train_knn_label <- train_knn$Dataset
test_knn_label <- test_knn$Dataset
str(train_knn_label)
str(test_knn_label)

#Building a KNN model
library(class)
pred_knn <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k = 1)
table_knn <- table(test_knn_label, pred_knn)
total <- table_knn[1,1] + table_knn[2,2]
accuracy <- (total/174)*100
accuracy


#Choosing best value of K
```

```
k1 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=1)
table1 <- table(k1, test_knn_label)
TotalCorrect1 <- table1[1,1] + table1[2,2]
Accuracy1 <- (TotalCorrect1/174)*100
print(Accuracy1)


k3 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=3)
table3 <- table(k3, test_knn_label)
TotalCorrect3 <- table3[1,1] + table3[2,2]
Accuracy3 <- (TotalCorrect3/174)*100
print(Accuracy3)

k5 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=5)
table5 <- table(k5, test_knn_label)
TotalCorrect5 <- table5[1,1] + table5[2,2]
Accuracy5 <- (TotalCorrect5/174)*100
print(Accuracy5)

k7 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=7)
table7 <- table(k7, test_knn_label)
TotalCorrect7 <- table7[1,1] + table7[2,2]
Accuracy7 <- (TotalCorrect7/174)*100
print(Accuracy7)

k9 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=9)
table9 <- table(k9, test_knn_label)
TotalCorrect9 <- table9[1,1] + table9[2,2]
Accuracy9 <- (TotalCorrect9/174)*100
print(Accuracy9)

k11 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=11)
table11 <- table(k11, test_knn_label)
TotalCorrect11 <- table11[1,1] + table11[2,2]
Accuracy11 <- (TotalCorrect11/174)*100
print(Accuracy11)

k13 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=13)
table13 <- table(k13, test_knn_label)
TotalCorrect13 <- table13[1,1] + table13[2,2]
```

```
Accuracy13 <- (TotalCorrect13/174)*100
print(Accuracy13)

k15 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=15)
table15 <- table(k15, test_knn_label)
TotalCorrect15 <- table15[1,1] + table15[2,2]
Accuracy15 <- (TotalCorrect15/174)*100
print(Accuracy15)


k17 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=17)
table17 <- table(k17, test_knn_label)
TotalCorrect17 <- table17[1,1] + table17[2,2]
Accuracy17 <- (TotalCorrect17/174)*100
print(Accuracy17)

k19 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=19)
table19 <- table(k19, test_knn_label)
TotalCorrect19 <- table19[1,1] + table19[2,2]
Accuracy19 <- (TotalCorrect19/174)*100
print(Accuracy19)

k21 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=21)
table21 <- table(k21, test_knn_label)
TotalCorrect21 <- table21[1,1] + table21[2,2]
Accuracy21 <- (TotalCorrect21/174)*100
print(Accuracy21)

k23 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=23)
table23 <- table(k23, test_knn_label)
TotalCorrect23 <- table23[1,1] + table23[2,2]
Accuracy23 <- (TotalCorrect23/174)*100
print(Accuracy23)

k25 <- knn(train = train_knn2, test = test_knn2, cl = train_knn_label, k=25)
table25 <- table(k25, test_knn_label)
table25
TotalCorrect25 <- table25[1,1] + table25[2,2]
Accuracy25 <- (TotalCorrect25/174)*100
print(Accuracy25)
```

```
#Ploting K values and Accuracy
plot_accuracy <- c(Accuracy1, Accuracy3, Accuracy5, Accuracy7, Accuracy9,
                   Accuracy11, Accuracy13, Accuracy15, Accuracy17, Accuracy19,
                   Accuracy21,Accuracy23,Accuracy25)
plot_Klabels <- c("K=1","K=3","K=5","K=7","K=9",
                  "K=11","K=13","K=15","K=17","K=19",
                  "K=21","K=23","K=25")
K_values <- c(1,3,5,7,9,11,13,15,17,19,21,23,25)
plot(x=K_values,y=plot_accuracy,xlab ="K value", ylab = "Accuracy in %",
   main="Accuracy of KNN model with varying K values")
text(x=K_values,y=plot_accuracy,labels=plot_Klabels, pos =1 )


#Metrics for best K value

error_rate <- (table25[1,2] +  table25[2,1]) / 174
error_rate

false_pos_rate <- table25[2,1]/(table25[2,1] + table25[2,2])
false_pos_rate
```

```
################################################################################
##############
### Logistic Regression
################################################################################
##############

#Libraries:
library(car)
library(caret)
library(lattice)
library(ROCR)
library(party)
#Logistic regression MOdel:
#Creating a "0-1" column:

patients<-read.csv("indian_liver_patient.csv")
patients<-patients[complete.cases(patients),]
patients<-patients[,c(-2,-4,-5,-7)]
patients$Diagnosis<-ifelse(patients$Dataset==2,0,1)
patients<-patients[,-7]
colnames(patients)
#Creating the regression model:
logmod<- glm(Diagnosis~.,data = patients,family = binomial())
#Predicting the values:
pred<-predict(logmod, type = "response")
#Creating a confusion matrix to evaluate the results:
confusionMatrix(as.factor(patients$Diagnosis),as.factor(ifelse(pred>0.5,1,0)))
#Plotting the ROC Curve:
plot(performance(prediction(pred,patients$Diagnosis),"tpr","fpr"),main ="ROC Curve", colorize
= T)

table_lr <- table(as.factor(patients$Diagnosis),as.factor(ifelse(pred>0.5,1,0)))
table_lr

error_rate <- (table_lr[1,2] +  table_lr[2,1]) / 579
error_rate

false_pos_rate <- table_lr[2,1]/(table_lr[2,1] + table_lr[2,2])
```

```
false_pos_rate

################################################################################
##############
### Classification and Regression Trees
################################################################################
##############
#Decision Tress:
#library
library(rpart)
library(rpart.plot)
library(maptree)
library(cluster)
#Creating the model
dtmodel<-rpart(Diagnosis~.,data = patients, method = "class")
printcp(dtmodel)
plotcp(dtmodel)
summary(dtmodel)

#Decision Tree Plot
rpart.plot(dtmodel, uniform=TRUE, main="Classification Tree for Patient Records")

#Testing the model:
dtpred<-predict(dtmodel,type = "class")

#Confusion Matrix:
confusionMatrix(dtpred,as.factor(patients$Diagnosis))
table_cart <- table(dtpred,as.factor(patients$Diagnosis))

error_rate <- (table_lr[1,2] +  table_lr[2,1]) / 579
error_rate

false_pos_rate <- table_lr[2,1]/(table_lr[2,1] + table_lr[2,2])
false_pos_rate

#ROC Plot
plot(performance(prediction(as.numeric(dtpred),patients$Diagnosis),"tpr","fpr"),main ="ROC
Curve", colorize = T)
```