Early Liver Disease Diagnosis Prediction using R – Surbhi Anand

**Table of Contents**

[Introduction](#_bookmark0) [4](#_bookmark0)

[Business Understanding](#_bookmark1) [5](#_bookmark1)

[Data Exploration & Visualization](#_bookmark2) [5](#_bookmark2)

[Data Preparation](#_bookmark3) [7](#_bookmark3)

[Applying data mining methods using R](#_bookmark4) [10](#_bookmark4)

[Conclusion](#_bookmark5) [12](#_bookmark5)

[Future considerations for the Project](#_bookmark6) [13](#_bookmark6)

[References](#_bookmark7) [13](#_bookmark7)

[Appendix](#_bookmark8) [13](#_bookmark8)

**List of Figures**

### Figure 1: Pie chart of the response variable Figure 2: Histogram of gender

### Figure 3: Missing values

### Figure 4: Boxplot to identify outliers Figure 5: Correlation matrix

### Figure 6: Identifying outliers using Crooks Distance

**List of Tables**

### Table 1: Columns Description

### Table 2: Correlation between predictor columns

# Introduction

### The project aims to utilize the techniques gained in Data Mining lectures in the field of healthcare. Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles, and drugs. Chronic liver disease is a leading cause of morbidity and mortality nowadays. This dataset provides insights for using patient records to classify which patients have liver disease and which ones do not.

### The data for this project is obtained from Kaggle. <https://www.kaggle.com/uciml/indian-liver-patient-records>

### Certain factors as age gender and certain chemical compounds are taken into consideration as mentioned in the table:

|  |  |
| --- | --- |
| **Column** | **Description** |

|  |  |
| --- | --- |
| Age | Patient’s age. |
| Gender | Patient’s gender. |
| Total\_Bilirubin | Total bilirubin is a combination of direct and indirect bilirubin. High levels indicate liver problems. |
| Direct\_Bilirubin | Bilirubin is an orange-yellow substance made during the normal breakdown of red blood cells. High levels indicate liver problems. |
| Alkaline\_Phosphotase | An enzyme is mostly found in cells of bone and liver. |
| Alamine\_Aminotransferase | An enzyme is mostly found in cells of liver and kidney. |
| Aspartate\_Aminotransferase | It is found in cells throughout the body and its test measures the level of the enzyme. |
| Total\_Protiens | It measures the total amount of albumin and globulin in the body. |
| Albumin | It is a protein produced by the liver that circulates in plasma. |
| Albumin\_and\_Globulin\_Ratio | A high ratio is an indicator of disease in the liver, kidney, and intestines. |
| Dataset | Indicates whether a person is diagnosed with liver disease or not. |

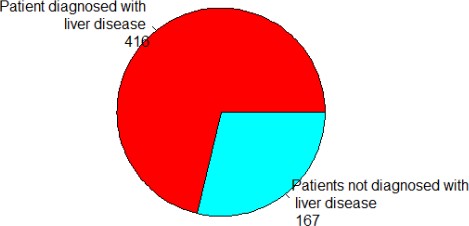
#### Table 1.1 Column description

# Business Understanding

### According to CDC statistics, Liver disease accounts for approximately 2 million deaths per year. Early diagnosis of liver disease may help in reducing the current death rate. Patient's health records, including blood test reports, can be analyzed using data mining methods, to determine whether the person will suffer from liver disease in the future or not. This will be an added advantage for doctors, who can suggest lifestyle changes and medication to potential liver disease patients at an early stage.

# Data Exploration & Visualization

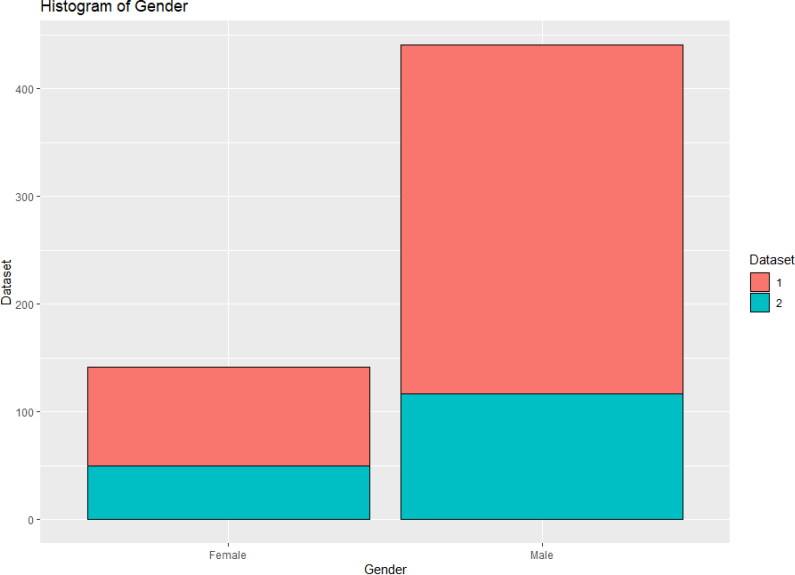
## Data Structure:



#### Figure 1: Pie chart of the response variable

### The data set consists of 11 columns and 583 rows having 416 liver patient records and 167 non-liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patients (Dataset = 1) or not (Dataset = 2).

## Histogram of gender according to the response variable:



#### Figure 2: Histogram of gender

### The histogram shows that the number of male patients is more than that of female patients. Also, male patients are more risk-prone to liver disease.

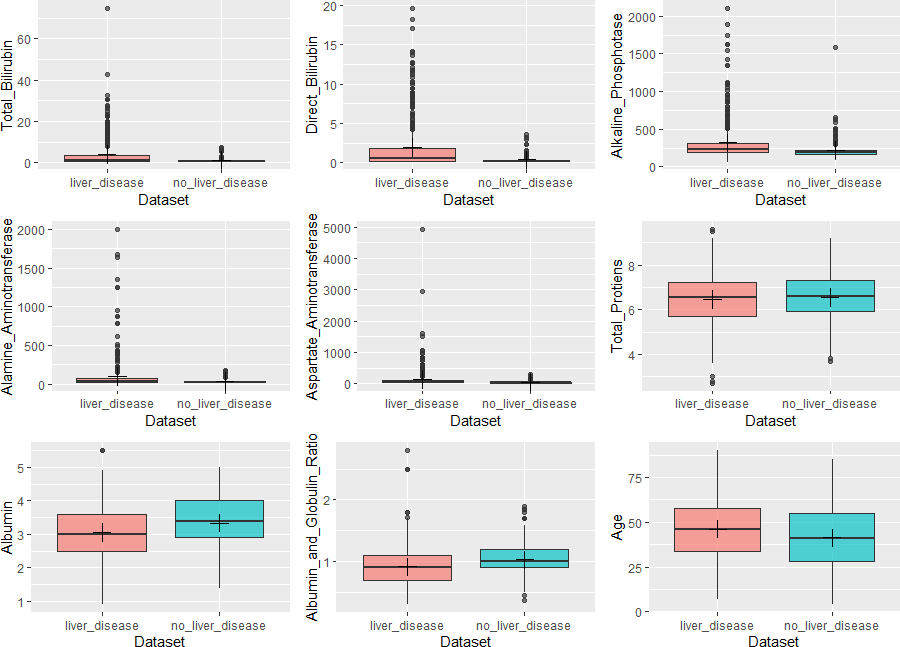
## Missing values in the dataset:



#### Figure 3: Missing values

### The above graph shows that there are a total of 4 missing values in column “Albumin\_and\_globulin\_ratio”. It will be handled in the data preparation phase.

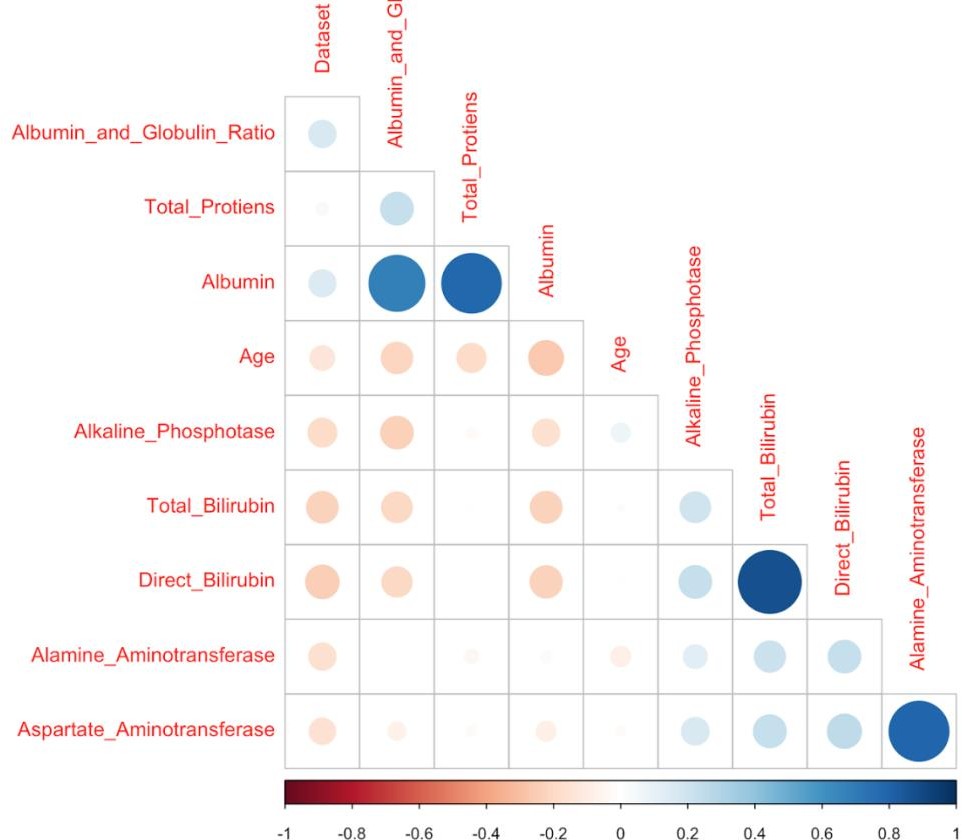
## Identifying outliers in the dataset:



#### Figure 4: Box plot to identify outliers

### The boxplot shows that some variables have skewed distribution (some extreme values), which could bias the statistical inferences and reduce the accuracy of prediction models. This is handled in the data preparation phase.

## Correlation between predictor columns:



#### Figure 5: Correlation between predictor columns

### The correlation matrix is used to identify predictor columns having a strong positive correlation.

### Total\_Bilirubin and Direct\_Bilirubin Alamine\_Aminotransferase and Aspartate\_Aminotransferase Total\_Protiens and Albumin



# Data Preparation

### Data collection in a practical scenario has flaws due to several reasons. Also depending upon the problem one is solving; the entire dataset might not be of his/her interest. There can be certain columns or features that are redundant for the prediction or classification problem.

### For our project, we did a literature survey of the techniques which are used for data cleaning and data preparation. We performed the following steps in the data preparation phase:

## Creating categorical variables

### We created a new factor variable, Gender.d using the Gender column. The new column has levels of ‘1’ for ‘Male’ and ‘0’ for females. Also, we created a new column ‘Result’, for the response variable, i.e. Dataset. It has two levels, 1- having a liver disease and 0- not having liver disease.

## Handling missing values

### Imputation in statistics is a method to substitute values in place of missing values. Discarding any case with missing values can introduce bias or affect the representativeness of the results. Imputation preserves these cases by replacing missing values with estimated values.

### We used the Predicted Mean matching imputation method for filling missing entries. In this technique, a set of candidate donors are chosen whose predicted values closest to the missing value. The algorithm then picks one donor from a set of candidate donors randomly. The assumption is that the distribution of missing entry is the same as the observed data of the candidate donors.

### Predicted mean matching is similar to the hot deck method where a missing value is imputed from a randomly selected similar record.

## Handling outliers

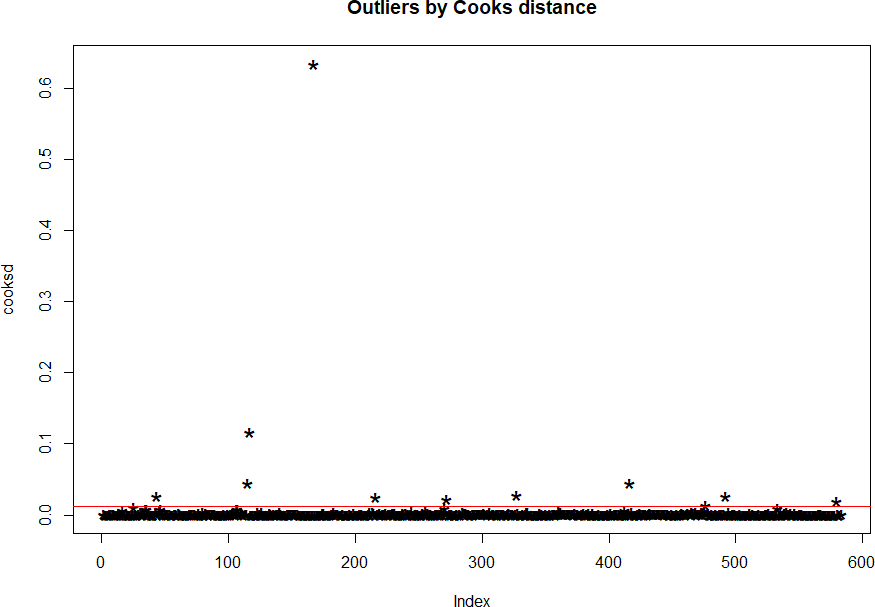
### We have used Cook’s distance to find outliers that negatively affect the regression model. If Cook’s distance exceeds a certain threshold, the data point can be considered an outlier. These influential records were excluded from the data. Below is the formula for Cook’s distance.

𝑛

(𝑌𝑗 − (𝑌𝑗𝑖)

∑ (𝑝 + 1)𝜎2

𝑗=1



#### Figure 6: Identifying outliers using Crooks Distance

### These influential records are excluded from the clean dataset.

## Dropping columns based on the correlation

|  |  |
| --- | --- |
| **Predictor columns** | **Correlation (r)** |
| Total\_Bilirubin and Direct\_Bilirubin | 0.87 |
| Alamine\_Aminotransferase and Aspartate\_Aminotransferase | 0.79 |
| Total\_Protiens and Albumin | 0.78 |

#### Table 2: Correlation between predictor columns

### Since there is a direct relationship between the above pair of predictor columns, we dropped one of them from the dataset. We included the following columns in data mining models:

### Age Gender.d



### Total\_Bilirubin Alkaline\_Phosphotase Alamine\_Aminotransferase Total\_Protiens Albumin\_and\_Globulin\_Ratio Result (Response variable)

## Partitioning

### We have divided the data into training records, validation records, and test records. We have taken random samples of data from the dataset for each category using 𝑠𝑎𝑚𝑝𝑙𝑒 function in R. This is simple random sampling. The benefit of this method is it leads to a low bias of model performance.

### Validation partition is used to evaluate the model developed on training data. This phase can be iterative, where different models are tried out. The validation partition is used to compare the performance of different models and select the best one. Overfitting of training data can be detected through the validation set.

### The test partition is used to assess the performance of the final chosen model with unknown data. Test data will not be used in training or validation so that model is assessed unbiasedly. Overfitting problem with validation data can be identified with test partition

### Dataset Size: 583

### Training records size: 50% of the dataset Validation records size: 30% of the dataset Test records size: 20% of the dataset

### There is no intersection between training, validation, and test data set.

# Applying data mining methods using R

## Baseline:

### Every candidate of the data set is assumed to be a liver patient, irrespective of their age, gender, or compounds present in the body.

|  |  |
| --- | --- |
| **Measures** | **Dataset** |
| Accuracy | 0.725 |
| Sensitivity | 1 |
| Specificity | 0 |
| FPR | 1 |

## Logistic Regression:

### Logistic regression is analogous to multiple linear regression, except the outcome is binary. The response of the Logistic regression formula is the log of odds of a binary outcome of 1. The summary of the logistic regression model showed that columns – Gender, Total\_Bilirubin, Alkaline\_Phosphotase, and Alamine\_Aminotransferase are significant, having p-value (0.5).

|  |  |  |
| --- | --- | --- |
| **Measures** | **Validation Dataset** | **Test Dataset** |
| Accuracy | 0.731 | 0.730 |
| Sensitivity | 0.909 | 0.831 |
| Specificity | 0.300 | 0.384 |
| FPR | 0.700 | 0.616 |

## Classification Trees:

### We started with a full tree, overriding default parameters of cp (complexity parameter), and min split. The full tree had a minimum x error for 27 splits and a cp of 0.12. To avoid overfitting, we pruned the full tree with the smallest x error.

|  |  |  |
| --- | --- | --- |
| **Measures** | **Validation Dataset** | **Test Dataset** |
| Accuracy | 0.748 | 0.730 |
| Sensitivity | 0.917 | 0.876 |
| Specificity | 0.340 | 0.231 |
| FPR | 0.660 | 0.769 |

## Random Forest:

### Random forest is based on applying bagging to decision trees with one important extension, in addition to sampling the records, the algorithm also samples the variables. We used different number of trees, i.e. 11,25, 101,401,1601. Although the method produced an optimal result for the default number of trees, i.e. 25.

|  |  |  |
| --- | --- | --- |
| **Measures** | **Validation Dataset** | **Test Dataset** |
| Accuracy | 0.719 | 0.704 |
| Sensitivity | 0.893 | 0.820 |
| Specificity | 0.300 | 0.307 |
| FPR | 0.700 | 0.693 |

## Boosting:

### Boosting is a general technique to create an ensemble of models. We used different number of iterations, i.e. 11,25, 101,401,1601. Although the method produced an optimal result for the default number of iterations, i.e. 25.

|  |  |  |
| --- | --- | --- |
| **Measures** | **Validation Dataset** | **Test Dataset** |
| Accuracy | 0.701 | 0.652 |
| Sensitivity | 0.876 | 0.764 |
| Specificity | 0.280 | 0.269 |
| FPR | 0.720 | 0.731 |

## K – nearest neighbors

### The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve classification problems. To find the best value of k, we used the function knn.cv. We identified k=15 as the optimal number of neighbors.

|  |  |  |
| --- | --- | --- |
| **Measures** | **Validation Dataset** | **Test Dataset** |
| Accuracy | 0.707 | 0.643 |
| Sensitivity | 0.909 | 0.764 |
| Specificity | 0.220 | 0.230 |
| FPR | 0.780 | 0.770 |

# Conclusion

### We tried five classification methods to determine whether a person will be diagnosed with liver disease or not based on his/her health records. Summarized comparative analysis of different data mining methods on:

## Validation data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Mining Method** | **Accuracy** | **Sensitivity** | **Specificity** | **FPR** |
| Baseline | 0.725 | 1.000 | 0.000 | 1.000 |
| Logistic  Regression | 0.731 | 0.909 | 0.300 | 0.700 |
| Classification  Trees | 0.748 | 0.917 | 0.340 | 0.660 |
| Random Forest | 0.719 | 0.893 | 0.300 | 0.700 |
| Boosting | 0.701 | 0.876 | 0.280 | 0.720 |
| K – nearest  neighbours | 0.707 | 0.909 | 0.220 | 0.780 |

## Test data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Mining Method** | **Accuracy** | **Sensitivity** | **Specificity** | **FPR** |
| Baseline | 0.725 | 1.000 | 0.000 | 1.000 |
| Logistic  Regression | 0.730 | 0.832 | 0.385 | 0.615 |
| Classification Trees | 0.730 | 0.876 | 0.231 | 0.769 |
| Random Forest | 0.704 | 0.820 | 0.307 | 0.693 |
| Boosting | 0.652 | 0.764 | 0.269 | 0.731 |
| K – nearest  neighbours | 0.643 | 0.764 | 0.230 | 0.770 |

### Based on the summary of performance measures, we would recommend **logistic regression** as the most suitable method for the classification of patients. Accuracy of logistic

### regression and classification tree is comparable to the testing dataset, but the FPR of later is higher than the former.

# Future considerations for the Project

### Dataset could have been more informative if it would have captured data for other predictor columns like weight, fitness level, and other diseases patients are suffering with. Hidden causes could be revealed by analyzing such a dataset. Also, we can try other data mining methods, like neural networks and bagging.

# References

### <https://www.cdc.gov/nchs/fastats/liver-disease.htm> <https://www.kaggle.com/netzone/indian-liver-disease-modeling> <https://www.r-project.org/>

### <https://www.rdocumentation.org/> <https://www.cdc.gov/nchs/fastats/liver-disease.htm>

# Appendix

### R code:

*#Loading required packages*

**if** (!require("pacman")) install.packages("pacman","caret","e1071","lift") pacman::p\_load(pacman,rio,tidyverse,caret,e1071,lift)

df <- import("indianliver.xlsx")

*# DATA EXPLORATION*

*# Data structure* nrow(df) str(df)

*# Pie chart of response variable* count <- table(df$Dataset) count

library(ggplot2)

label <- c("Patient diagnosed with \n liver disease","Patients not diagnosed with \n liver disease")

label

label <- paste(label,"\n", count) label

length(label)

pie(count, labels = label, main = "Distribution in Dataset", col = rainbow(length(label)))

*# Histogram for gender*

plot\_temp <- df

plot\_temp$Dataset <- as.factor(plot\_temp$Dataset) plot\_temp$Gender <- as.factor(plot\_temp$Gender)

ggplot(data = plot\_temp, aes(x=Gender)) + geom\_bar(color = "black", aes(fill = Dataset)) + xlab("Gender") + ylab("Dataset")+ ggtitle("Histogram of Gender")

*# Missing values* library(naniar) library(visdat) vis\_miss(df)

*# Identifying Outliers*

plot\_temp1 <- df %>% filter(!is.na(Albumin\_and\_Globulin\_Ratio))

*# unique(plot\_temp1$Gender) # Verify the unique values in Gender variable* plot\_temp1$Gender <- as.factor(plot\_temp1$Gender) *# Transform chr to factor* plot\_temp1$Dataset <- as.factor(plot\_temp1$Dataset) *# Transform chr to factor*

levels(plot\_temp1$Dataset) <- c("liver\_disease", "no\_liver\_disease") *#*

*Rename*

plot\_temp1.o <- plot\_temp1 *# Original Dataframe*

plot\_temp1.o <- plot\_temp1.o %>% *# create a new variable (AST/ALT ratio)*

mutate(AST\_ALT\_ratio = Aspartate\_Aminotransferase/Alamine\_Aminotransferase)

plot\_temp1.o <- plot\_temp1.o[,c(1:10, 12, 11)] *#change columns order*

pb1 <- ggplot(plot\_temp1.o, aes(Dataset, Total\_Bilirubin)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb2 <- ggplot(plot\_temp1.o, aes(Dataset, Direct\_Bilirubin)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb3 <- ggplot(plot\_temp1.o, aes(Dataset, Alkaline\_Phosphotase)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb4 <- ggplot(plot\_temp1.o, aes(Dataset, Alamine\_Aminotransferase)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb5 <- ggplot(plot\_temp1.o, aes(Dataset, Aspartate\_Aminotransferase)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb6 <- ggplot(plot\_temp1.o, aes(Dataset, Total\_Protiens)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb7 <- ggplot(plot\_temp1.o, aes(Dataset, Albumin)) + geom\_boxplot(aes(fill

= Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb8 <- ggplot(plot\_temp1.o, aes(Dataset, Albumin\_and\_Globulin\_Ratio)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

pb9 <- ggplot(plot\_temp1.o, aes(Dataset, Age)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")

*#pb10 <- ggplot(plot\_temp1.o, aes(Dataset, AST\_ALT\_ratio)) + geom\_boxplot(aes(fill = Dataset), alpha = 2/3) + stat\_summary(fun.y=mean, geom="point", shape=3, size=4) + theme(legend.position = "none")*

library(gridExtra) library(magrittr) library(dplyr)

grid.arrange(pb1, pb2, pb3, pb4, pb5, pb6, pb7, pb8, pb9, ncol=3)

*# Identifying correlation*

plot\_temp2 <- na.omit(df) library(corrplot)

M <- cor(plot\_temp2[sapply(plot\_temp2, **function**(x) !is.character(x))]) corrplot(M, method = "circle",order = "hclust",type='lower', diag=**F**, addCoefasPercent=**T**)

M <- cor(plot\_temp2[sapply(plot\_temp2, **function**(x) !is.character(x))]) corrplot(M, method = "number",order = "hclust",type='lower', diag=**F**, addCoefasPercent=**T**)

*# DATA PREPARATION*

*# Creating categorical values*

df$Result <- factor(ifelse(df$Dataset == 2,0,1 )) df$Gender.d <- factor(ifelse(df$Gender == "Male",1,0 )) df <- df[,-c(2,11)]

str(df)

*# Handling missing data* sum(is.na(df)) library(mice) md.pattern(df)

imputed.df <- mice(data = df, m=5, method = "pmm", maxit = 50, seed = 500) summary(imputed.df)

imputed.df$imp$Albumin\_and\_Globulin\_Ratio completed.df <- complete(imputed.df,2)

*# Handling outliers*

mod <- glm(Result ~ ., data = completed.df, family = "binomial") cooksd <- cooks.distance(mod)

plot(cooksd, pch="\*", cex=2, main = "Outliers by Cooks distance") abline(h = 4\*mean(cooksd, na.rm = **T**), col = "red")

influential <- as.numeric(names(cooksd)[cooksd > 4\*mean(cooksd, na.rm = 'T')])

influential

completed.df <- completed.df[-influential,] nrow(completed.df)

*# Dropping columns based on correlation*

completed.df <- completed.df[,-c(3,6,8)]

*# Partitioning*

training.rows <- sample(rownames(completed.df),dim(completed.df)[1]\*0.5) valid.rows <- sample(setdiff(rownames(completed.df),training.rows),dim(completed.df)[1]\*0

.3)

test.rows <- setdiff(rownames(completed.df), union(training.rows,valid.rows))

train.data <- completed.df[training.rows,] nrow(train.data)

valid.data <- completed.df[valid.rows,] nrow(valid.data)

test.data <- completed.df[test.rows,] nrow(test.data)

*# APPLYING DATA MINING METHODS USING R*

*#0. Baseline classification*

*#Every candidate is considered as liver patient.*

temp <- completed.df[completed.df$Result==0,] nrow(temp)

nrow(completed.df)

*# Accuracy*

nrow(temp)/nrow(completed.df)

baseline.accuracy <- 1-(nrow(temp)/nrow(completed.df)) baseline.accuracy

*#1. Logistic Regression*

logit.model <- glm(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase + Alamine\_Aminotransferase

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio, data = train.data, family = "binomial")

summary(logit.model)

*# Applying model on Validation Set*

validation.prob <- predict(logit.model, valid.data, type = "response") validation.classifications <- ifelse(validation.prob < 0.5, 0, 1) confusionMatrix(as.factor(validation.classifications),

as.factor(valid.data$Result), positive = "1")

*# Applying model on test Set*

test.prob <- predict(logit.model, test.data, type = "response") test.classifications <- ifelse(test.prob < 0.5, 0, 1) confusionMatrix(as.factor(test.classifications),

as.factor(test.data$Result), positive = "1")

*# 2. Classification Trees*

library(rpart)

liver.full.tree <- rpart(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase + Alamine\_Aminotransferase

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio, data = train.data,

method = "class",

control = rpart.control(cp = 0.0, minsplit = 0))

library(rattle) fancyRpartPlot(liver.full.tree)

printcp(liver.full.tree)

prunned.liver.tree <- prune(liver.full.tree, cp = 0.015) fancyRpartPlot(prunned.liver.tree)

*# Applying model on training data*

actual.liver.train <- train.data$Result

predict.liver.train <- predict(prunned.liver.tree, type = "class") table(actual.liver.train,predict.liver.train)

*#accuracy*

sum(actual.liver.train == predict.liver.train)/nrow(train.data)

*# Applying model on validation data*

actual.liver.valid <- valid.data$Result

predict.liver.valid <- predict(prunned.liver.tree, type = "class", newdata

= valid.data) table(actual.liver.valid,predict.liver.valid) *#accuracy*

sum(actual.liver.valid == predict.liver.valid)/nrow(valid.data)

*#sensitivity*

sum(actual.liver.valid == 1 & predict.liver.valid == 1)/sum(actual.liver.valid == 1)

*#specificity*

sum(actual.liver.valid == 0 & predict.liver.valid == 0)/sum(actual.liver.valid == 0)

*# Applying model on test data*

actual.liver.test <- test.data$Result

predict.liver.test <- predict(prunned.liver.tree, type = "class", newdata = test.data)

table(actual.liver.test,predict.liver.test)

*#accuracy*

sum(actual.liver.test == predict.liver.test)/nrow(test.data)

*#sensitivity*

sum(actual.liver.test == 1 & predict.liver.test == 1)/sum(actual.liver.test

== 1)

*#specificity*

sum(actual.liver.test == 0 & predict.liver.test == 0)/sum(actual.liver.test

== 0)

*#3. Random Forests*

*# ntree = 25 gives the optimal accuracy for validation as well as test data.*

library(randomForest)

*# ntree = 5*

liver.forest.model.5 <- randomForest(Result ~ Age + Gender.d + Total\_Bilirubin

Alamine\_Aminotransferase

data=train.data, ntree=5)

+ Alkaline\_Phosphotase +

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio,

*# Applying model on training data*

predict.liver.train.rf5 <- predict(liver.forest.model.5, newdata = train.data)

sum(actual.liver.train == predict.liver.train.rf5)/nrow(train.data)

*# Applying model on validation data*

predict.liver.valid.rf5 <- predict(liver.forest.model.5, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.rf5)/nrow(valid.data)

*# Applying model on test data*

predict.liver.test.rf5 <- predict(liver.forest.model.5, newdata = test.data)

sum(actual.liver.test == predict.liver.test.rf5)/nrow(test.data)

*# ntree = 11*

liver.forest.model.11 <- randomForest(Result ~ Age + Gender.d + Total\_Bilirubin

Alamine\_Aminotransferase

data=train.data, ntree=11)

+ Alkaline\_Phosphotase +

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio,

*# Applying model on training data*

predict.liver.train.rf11 <- predict(liver.forest.model.11, newdata = train.data)

sum(actual.liver.train == predict.liver.train.rf11)/nrow(train.data)

*# Applying model on validation data*

predict.liver.valid.rf11 <- predict(liver.forest.model.11, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.rf11)/nrow(valid.data)

*# Applying model on test data*

predict.liver.test.rf11 <- predict(liver.forest.model.11, newdata = test.data)

sum(actual.liver.test == predict.liver.test.rf11)/nrow(test.data)

*# ntree = 25*

liver.forest.model.25 <- randomForest(Result ~ Age + Gender.d + Total\_Bilirubin

Alamine\_Aminotransferase

data=train.data, ntree=25)

+ Alkaline\_Phosphotase +

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio,

*# Applying model on training data*

predict.liver.train.rf25 <- predict(liver.forest.model.25, newdata = train.data)

sum(actual.liver.train == predict.liver.train.rf25)/nrow(train.data)

*# Applying model on validation data*

predict.liver.valid.rf25 <- predict(liver.forest.model.25, newdata = valid.data)

*#accuracy*

sum(actual.liver.valid == predict.liver.valid.rf25)/nrow(valid.data)

*#sensitivity*

sum(actual.liver.valid == 1 & predict.liver.valid.rf25 == 1)/sum(actual.liver.valid == 1)

*#specificity*

sum(actual.liver.valid == 0 & predict.liver.valid.rf25 == 0)/sum(actual.liver.valid == 0)

*# Applying model on test data*

predict.liver.test.rf25 <- predict(liver.forest.model.25, newdata = test.data)

*#accuracy*

sum(actual.liver.test == predict.liver.test.rf25)/nrow(test.data)

*#sensitivity*

sum(actual.liver.test == 1 & predict.liver.test.rf25 == 1)/sum(actual.liver.test == 1)

*#specificity*

sum(actual.liver.test == 0 & predict.liver.test.rf25 == 0)/sum(actual.liver.test == 0)

*# ntree = 101*

liver.forest.model.101 <- randomForest(Result ~ Age + Gender.d + Total\_Bilirubin

Alamine\_Aminotransferase

data=train.data, ntree=101)

+ Alkaline\_Phosphotase +

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio,

*# Applying model on training data*

predict.liver.train.rf101 <- predict(liver.forest.model.101, newdata = train.data)

sum(actual.liver.train == predict.liver.train.rf101)/nrow(train.data)

*# Applying model on validation data*

predict.liver.valid.rf101 <- predict(liver.forest.model.101, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.rf101)/nrow(valid.data)

*# Applying model on test data*

predict.liver.test.rf101 <- predict(liver.forest.model.101, newdata = test.data)

sum(actual.liver.test == predict.liver.test.rf101)/nrow(test.data)

*# ntree = 401*

liver.forest.model.401 <- randomForest(Result ~ Age + Gender.d + Total\_Bilirubin

Alamine\_Aminotransferase

data=train.data, ntree=401)

+ Alkaline\_Phosphotase +

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio,

*# Applying model on training data*

predict.liver.train.rf401 <- predict(liver.forest.model.401, newdata = train.data)

sum(actual.liver.train == predict.liver.train.rf401)/nrow(train.data)

*# Applying model on validation data*

predict.liver.valid.rf401 <- predict(liver.forest.model.401, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.rf401)/nrow(valid.data)

*# Applying model on test data*

predict.liver.test.rf401 <- predict(liver.forest.model.401, newdata = test.data)

sum(actual.liver.test == predict.liver.test.rf401)/nrow(test.data)

*# ntree = 1601*

liver.forest.model.1601 <- randomForest(Result ~ Age + Gender.d + Total\_Bilirubin

Alamine\_Aminotransferase

data=train.data, ntree=1601)

+ Alkaline\_Phosphotase +

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio,

*# Applying model on training data*

predict.liver.train.rf1601 <- predict(liver.forest.model.1601, newdata = train.data)

sum(actual.liver.train == predict.liver.train.rf1601)/nrow(train.data)

*# Applying model on validation data*

predict.liver.valid.rf1601 <- predict(liver.forest.model.1601, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.rf1601)/nrow(valid.data)

*# Applying model on test data*

predict.liver.test.rf1601 <- predict(liver.forest.model.1601, newdata = test.data)

sum(actual.liver.test == predict.liver.test.rf1601)/nrow(test.data)

*#4. Boosting*

*# iter = 25 gives the optimal accuracy for validation and testing data.*

library(ada)

*# iter =5*

liver.boost.model.5 <- ada(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase +

Alamine\_Aminotransferase

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio , data =

train.data, iter=5)

*#Applying model on training data*

predict.liver.train.boost5 <- predict(liver.boost.model.5, newdata = train.data)

sum(actual.liver.train == predict.liver.train.boost5)/nrow(train.data)

*#Applying model on validation data*

predict.liver.valid.boost5 <- predict(liver.boost.model.5, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.boost5)/nrow(valid.data)

*#Applying model on test data*

predict.liver.test.boost5 <- predict(liver.boost.model.5, newdata = test.data)

sum(actual.liver.test == predict.liver.test.boost5)/nrow(test.data)

*# iter =11*

liver.boost.model.11 <- ada(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase +

Alamine\_Aminotransferase

train.data, iter=11)

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio , data =

*#Applying model on training data*

predict.liver.train.boost11 <- predict(liver.boost.model.11, newdata = train.data)

sum(actual.liver.train == predict.liver.train.boost11)/nrow(train.data)

*#Applying model on validation data*

predict.liver.valid.boost11 <- predict(liver.boost.model.11, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.boost11)/nrow(valid.data)

*#Applying model on test data*

predict.liver.test.boost11 <- predict(liver.boost.model.11, newdata = test.data)

sum(actual.liver.test == predict.liver.test.boost11)/nrow(test.data)

*# iter =25*

liver.boost.model.25 <- ada(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase +

Alamine\_Aminotransferase

train.data, iter=25)

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio , data =

*#Applying model on training data*

predict.liver.train.boost25 <- predict(liver.boost.model.25, newdata = train.data)

sum(actual.liver.train == predict.liver.train.boost25)/nrow(train.data)

*#Applying model on validation data*

predict.liver.valid.boost25 <- predict(liver.boost.model.25, newdata = valid.data)

*#accuracy*

sum(actual.liver.valid == predict.liver.valid.boost25)/nrow(valid.data)

*#sensitivity*

sum(actual.liver.valid == 1 & predict.liver.valid.boost25 == 1)/sum(actual.liver.valid == 1)

*#specificity*

sum(actual.liver.valid == 0 & predict.liver.valid.boost25 == 0)/sum(actual.liver.valid == 0)

*#Applying model on test data*

predict.liver.test.boost25 <- predict(liver.boost.model.25, newdata = test.data)

*#accuracy*

sum(actual.liver.test == predict.liver.test.boost25)/nrow(test.data)

*#sensitivity*

sum(actual.liver.test == 1 & predict.liver.test.boost25 == 1)/sum(actual.liver.test == 1)

*#specificity*

sum(actual.liver.test == 0 & predict.liver.test.boost25 == 0)/sum(actual.liver.test == 0)

*# iter =101*

liver.boost.model.101 <- ada(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase +

Alamine\_Aminotransferase

train.data, iter=101)

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio , data =

*#Applying model on training data*

predict.liver.train.boost101 <- predict(liver.boost.model.101, newdata = train.data)

sum(actual.liver.train == predict.liver.train.boost101)/nrow(train.data)

*#Applying model on validation data*

predict.liver.valid.boost101 <- predict(liver.boost.model.101, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.boost101)/nrow(valid.data)

*#Applying model on test data*

predict.liver.test.boost101 <- predict(liver.boost.model.101, newdata = test.data)

sum(actual.liver.test == predict.liver.test.boost101)/nrow(test.data)

*# iter =401*

liver.boost.model.401 <- ada(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase +

Alamine\_Aminotransferase

train.data, iter=401)

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio , data =

*#Applying model on training data*

predict.liver.train.boost401 <- predict(liver.boost.model.401, newdata = train.data)

sum(actual.liver.train == predict.liver.train.boost401)/nrow(train.data)

*#Applying model on validation data*

predict.liver.valid.boost401 <- predict(liver.boost.model.401, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.boost401)/nrow(valid.data)

*#Applying model on test data*

predict.liver.test.boost401 <- predict(liver.boost.model.401, newdata = test.data)

sum(actual.liver.test == predict.liver.test.boost401)/nrow(test.data)

*# iter =1601*

liver.boost.model.1601 <- ada(Result ~ Age + Gender.d + Total\_Bilirubin

+ Alkaline\_Phosphotase +

Alamine\_Aminotransferase

train.data, iter=1601)

+ Total\_Protiens

+ Albumin\_and\_Globulin\_Ratio , data =

*#Applying model on training data*

predict.liver.train.boost1601 <- predict(liver.boost.model.1601, newdata = train.data)

sum(actual.liver.train == predict.liver.train.boost1601)/nrow(train.data)

*#Applying model on validation data*

predict.liver.valid.boost1601 <- predict(liver.boost.model.1601, newdata = valid.data)

sum(actual.liver.valid == predict.liver.valid.boost1601)/nrow(valid.data)

*#Applying model on test data*

predict.liver.test.boost1601 <- predict(liver.boost.model.1601, newdata = test.data)

sum(actual.liver.test == predict.liver.test.boost1601)/nrow(test.data)

*#5. k nearest - accuracy is optimal with 15 neighbors*

library(class)

x.train.data <- sapply(train.data[,-7], as.numeric) y.train.data <- train.data[,7]

x.valid.data <- sapply(valid.data[,-7], as.numeric) y.valid.data <- valid.data[,7]

x.test.data <- sapply(test.data[,-7], as.numeric) y.test.data <- test.data[,7]

normalize <- **function**(numbers){(numbers - mean(numbers))/sd(numbers)}

x.train.data.normalized <- apply(x.train.data, 2, normalize) apply(x.train.data.normalized, 2, mean)

apply(x.train.data.normalized, 2, sd)

x.valid.data.normalized <- apply(x.valid.data, 2, normalize) apply(x.valid.data.normalized, 2, mean)

apply(x.valid.data.normalized, 2, sd)

x.test.data.normalized <- apply(x.test.data, 2, normalize) apply(x.test.data.normalized, 2, mean)

apply(x.test.data.normalized, 2, sd)

*#training - accuracy is optimal with 11 neighbors*

**for**(k **in** c(1,3,5,7,9,11,13,15,17,19)){

knn.predicted.train.data <- knn.cv(x.train.data.normalized, y.train.data, k)

print(paste("With", k, "neighbours the accuracy is", sum(y.train.data == knn.predicted.train.data)/nrow(x.train.data.normalized)))

}

*#valid - accuracy is optimal with 15 neighbors*

**for**(k **in** c(1,3,5,7,9,11,13,15,17,19)){

knn.predicted.valid.data <- knn.cv(x.valid.data.normalized, y.valid.data, k)

print(paste("With", k, "neighbours the accuracy is", sum(y.valid.data == knn.predicted.valid.data)/nrow(x.valid.data.normalized)))

}

valid.predicted <- knn(x.train.data.normalized, x.valid.data.normalized, y.train.data, 15)

*#accuracy*

sum(y.valid.data == valid.predicted)/nrow(x.valid.data.normalized)

*#sensitivity*

sum(y.valid.data == 1 & valid.predicted == 1)/sum(y.valid.data == 1)

*#specificity*

sum(y.valid.data == 0 & valid.predicted == 0)/sum(y.valid.data == 0)

*#test - accuracy is optimal with 11 neighbors*

test.predicted <- knn(x.valid.data.normalized, x.test.data.normalized, y.valid.data, 11)

*#accuracy*

sum(y.test.data == test.predicted)/nrow(x.test.data.normalized)

*#sensitivity*

sum(y.test.data == 1 & test.predicted == 1)/sum(y.test.data == 1)

*#specificity*

sum(y.test.data == 0 & test.predicted == 0)/sum(y.test.data == 0)