Final Project

Will, the Hepatitis Patient will Survive or Not? A Statistical Machine Learning Approach

Submitted by

Md Salman Rahman

Graduate Student

University of Texas Rio Grande Valley

Submitted to

Dr. Tamer Oraby

December 6, 2021

Contents

1 Introduction

In this study, I will explore the research question whether a hepatitis patient will survive or not and will make future prediction about the survivable of a hepatitis patient. The meaning of hepatitis is inflammation of the liver. Liver is one of the most important part of human body because it process the nutrients, fights with infections things, and filter the blood of the body. Toxins, heavy alcohol use may affect the functionality of liver and cause hepatitis. Basically, there are 5 main types of hepatitis viruses, referred as A, B, C, D, E and among them type B and C cause chronic diseases that may affect millions of people in the world and together it may cause liver cirrhosis and cancer. One of the important characteristics of this disease is that it is asymptomatic and its difficult for the people to understand about this disease in the early stage.

So, my aim here is to develop a machine learning model which can able to help the physician to identify the hepatitis patient survivable condition in the early stage so that we can prevent the death of the patients.

2 Literature Review

There are several work on survivable condition of the hepatitis patients done in the past. Geamsakul et.al analysis the hepatitis dataset using tree based and graph based induction method [2]. They conclude that the average error rate is 22.60%. Also, sudha et.al investigate the performance of 15 data mining classification such as Rnd tree, Quinlan decision tree, and KNN on hepatitis dataset [7]. They claim that the error rate is less than 10% of different data mining algorithm such as KNN, Rnd tree, and BVM. Naive Bayes, Bayes-net, multi layer perceptron, and random forest algorithm applied to hepatitis patient by Karthikeyan et.al and they find that naive Bayes perform better than other classification techniques for hepatitis patient [5].

It is obvious that there are lots of works done on this famous dataset but the novelty of this work is that, I try to conclude almost all the classification algorithm and make a details comparison of different model using different metric such as testing error, sensitivity, specificity, accuracy, AUC, precision, recall, and F-1 score. And, this gives a clear picture of performance of different machine learning model on hepatitis patients.

3 Material and Methods

3.1 Data Description and Data Preparation

The data set is about Hepatitis data sets taken from UCI ML repository. The data set contains a total of 155 cases where there are 19 features and 1 response (class) variable. The attribute consists of class, age, sex, steroid, antivirals, fatigue, malaise, anorexia, liver big, liver firm, spleen palpable, spiders, ascites, varices, bilirubin, alk, phosphate, sgot, albumin, protime, and histology. The set consist of missing value especially there was 67 missing data in the PROTIME (prothrombin time) variable. And the missing value of the data set is replaced with median value of the column. Because, missing value drastically reduce the performance of the machine learning model. Also, I check the potential outlier in the data set by drawing the box-plot.

3.2 K Nearest Neighbour (KNN)

K Nearest Neighbor is a non parametric simple and easy-to-implement supervised machine learning algorithm that is used widely in classification problems. Non-parametric methods are very flexible since they do not follow any certain form but it suffer from over fitting and required big data as large number of parameter involved. The KNN algorithm assumes that the similar things exist in close proximity near to each other[3].

3.3 Logistic Regression (LR)

Linear regression based classification task is not effective because if the output is close to zero then there is a chance of predicting negative probability of the output and predicting probability greater than zero if the output value is bigger.So this is problem, especially fitting a straight line to a binary response that is coded as 0 and 1, in principle we can always predict probability of input $p(X) < 0$ for some values and $p(X) > 1$ for others unless the range of input variable is limited[6].

Logistic regression tackle this problem using logistic function in which the output of the model is always between 0 and 1[1]. Logit model is widely used to model the probability of certain class or events. Logistic function is given by,

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

We have to use multiple logistic regression for our data-set as for our problem there are multiple predictors. For multiple predictors logistic functions becomes

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

3.4 Linear Discriminant Analysis (LDA)

Linear discriminant analysis find the linear combination of features that separate two or more classes and it is a generalization of Fisher's linear discriminant method. LDA arises in the special situation when we make the assumption that the classes have a common covariance matrix [4]. LDA is commonly used if we have more than two response classes. LDA is closely related to factor analysis and principal component analysis.

3.5 Quadratic Discriminant Analysis (QDA)

Quadratic discriminant analysis is closely related to LDA. In LDA, we make the assumption that the observations within each class are drawn from a multivariate Gaussian distribution. Similarly, the QDA classifier results from assuming that the observations from each class are drawn from a Gaussian distribution, and we put estimates for the parameters into the Bayes theorem to make predictions. But, QDA assumes that each class has its own covariance matrix. QDA widely used for finding the classifier.

3.6 Support Vector Machine (SVM)

Support Vector Machine basically construct a one/set of hyper plane in an high dimensional space, which is used for classification, regression, and outliers detection. In a p dimensional space, a hyper-plane is a flat affine subspace of dimension p-1. SVM separates data using a separating line and furthest away from the closet data point and this make SVM unique from other algorithm. SVM are widely used as it can find a complex relationship between data without having lots of information about the data set. But if the number of features is bigger than the number of data point SVM trends to overfit.

3.7 Random Forest (RF)

Random forest is widely used ensemble learning method where the classification output of the random forest is the class selected by most trees. Other ensemble learning methods such as bagging methods get success for reducing the variance of an essential prediction function. Bagging especially works better for trees where variance is high and bias is low. Random forest is modification of the bagging technique that builds a large collection of de-correlated trees. The principle difference between bagging and random forest is the choice of predictor of subset size say m.

For example, if for random forest model m=p then the model is simply to bagging. Random forest generally outperform than the decision trees but in comparison to accuracy it is lower than the gradient boosted trees.

3.8 Classification Tree

Classification tree is widely used predictive modeling machine learning technique which uses decision trees to go from predictor variable of an item to output variable to draw conclusion. Basically applying a classification trees involved some steps referred as, determining the root of the tree, calculating the entropy for the classes, calculating entropy after splitting for each attribute, calculating information gain for each split, performing the split, and performing further split followed by completing the decision trees.

3.9 Artificial Neural Network(ANN)

Neural networks are computing system which is highly inspired by biological neuron. Generally, neural network are trained with example by passing input in the input layer and it results forming probability weighted associations between the two and store within the data structure inside the net itself. For example, a neural network takes an input vector of p variables $X = (X_1, X_2, \cdots, X_p)$ and builds a nonlinear function f(X) to predict a response variable Y [3]. Figure 1, shows a feed forward neural network use in this study. The neural network model have the form of

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

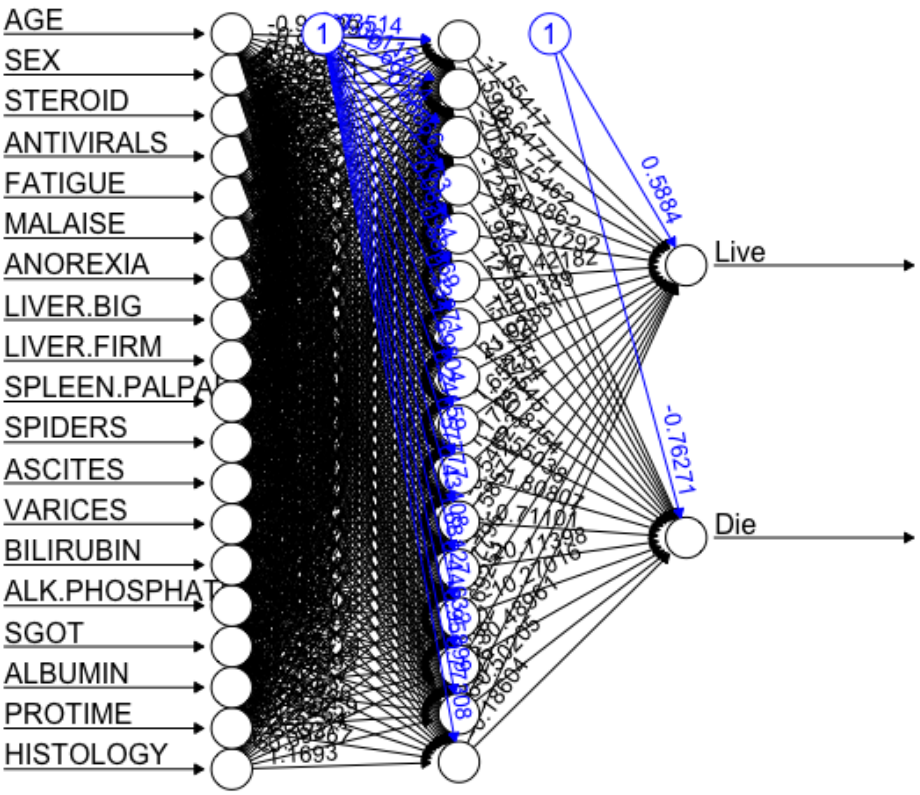

Figure 1: Neural Network Architecture

4 Results

4.1 Descriptive Analysis

In the figure 2, we see that there are 32 people who died due to hepatitis and 123 alive people in the response variable. It is noticeable that the data set is slightly imbalanced as there are only 26% patient who are died due to hepatitis. The training and testing sample is split into 65% training and 35% testing sample so that there will be enough proportion of the die and live people in the training and testing data-sets. I explore the data-set by plotting histogram, pie chart, and box plot and the complete descriptive analysis is available in the R code.

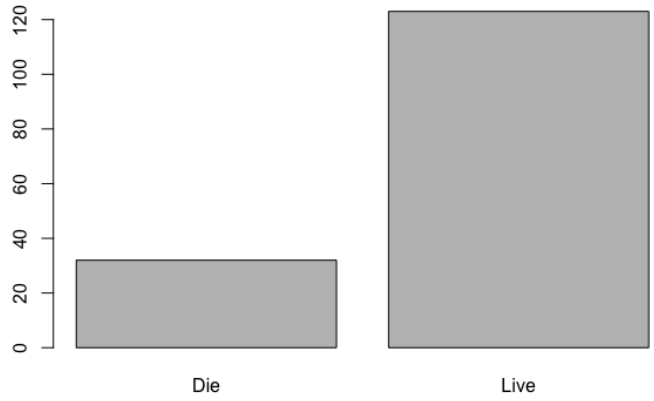

Figure 2: Outcome Count of the Response Variable

In this figure, we find the correlation plot of different predictors and response variable.

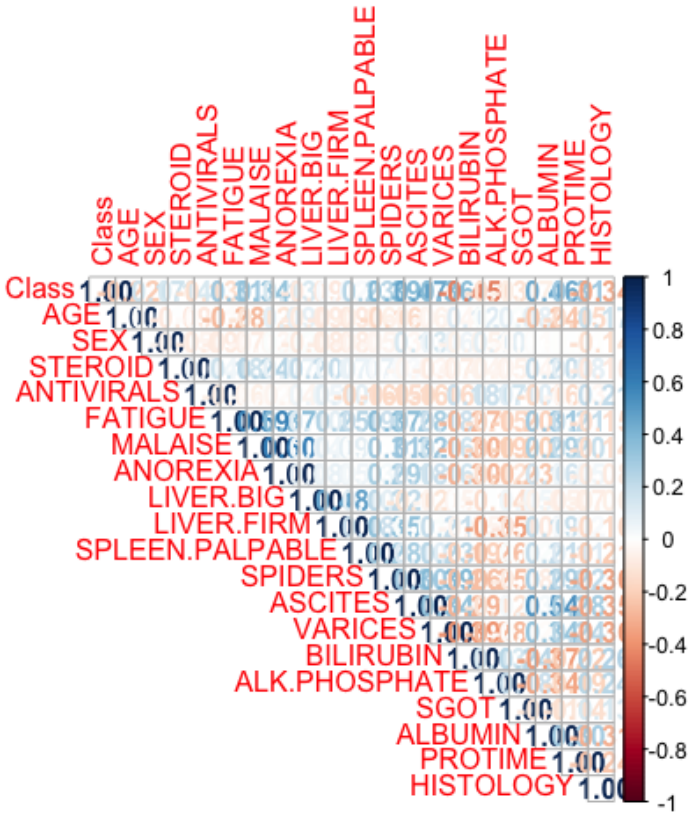

Figure 3: Correlation Plot

4.2 Outcome of Different Machine Learning Models

4.2.1 K Nearest Neighbour (KNN)

Here, I use knn function from the library class to build the KNN model. Figure 4 depicts the K value vs cross validation and we find that optimal K value is 9 for the project. After that, the model is trained and tested considering K=9. Table 1, shows the confusion matrix of the K nearest neighbour algorithm. It indicate that KNN model misclassified 8 die people with live and 1 live people misclassified with die people. For KNN model the testing error is 0.161, sensitivity is 0.385, and specificity is 0.977, and accuracy of the model is 0.839. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease. Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.681 for the KNN model and the ROC plot is given in Figure 5.

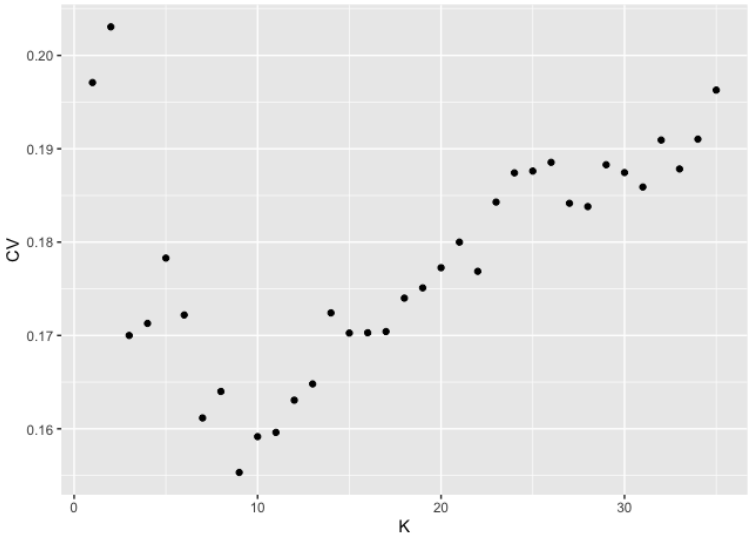

Figure 4: Optimal K value

Table 1: Confusion Matrix of KNN

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 5$	$b = 1$	$a + b$
	Live	$c = 8$	$d = 42$	$c + d$
	Total	$a + c$	$b + d$	N

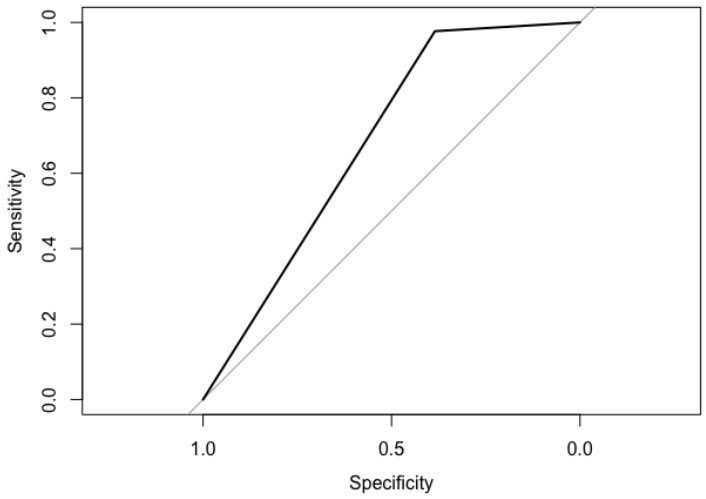

Figure 5: ROC Curve for KNN

Table 2: Confusion Matrix of Logistic Regression

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 8$	$b = 2$	$a + b$
	Live	$c = 5$	$d = 41$	$c + d$
	Total	$a + c$	$b + d$	N

4.2.2 Logistic Regression (LR)

Logistic regression model is deployed in R using the glm function using the binomial family. Table 2, shows the confusion matrix of the logistic regression algorithm. It indicate that LR model misclassified 5 die people with live and 2 live people misclassified with die people. For LR model the testing error is 0.125, sensitivity is 0.615, and specificity is 0.934, and accuracy of the model is 0.875. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease.

Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.941 for the LR model and the ROC plot is given in Figure 6.

4.2.3 Linear Discriminant Analysis (LDA)

LDA model is deployed in R using lda() function from the MASS library and I use predict() function to make the prediction with the testing dataset and predict functions in LDA returns a list. Table 3, shows the confusion matrix of the LDA algorithm. It indicate that LDA model misclassified 3 die people with live and 1 live people misclassified with die people. For LDA model the testing error is 0.071, sensitivity is 0.769, and specificity is 0.977, and accuracy of the model is 0.875. Here the sensitivity indicates the ability of a test to correctly identify the patient with

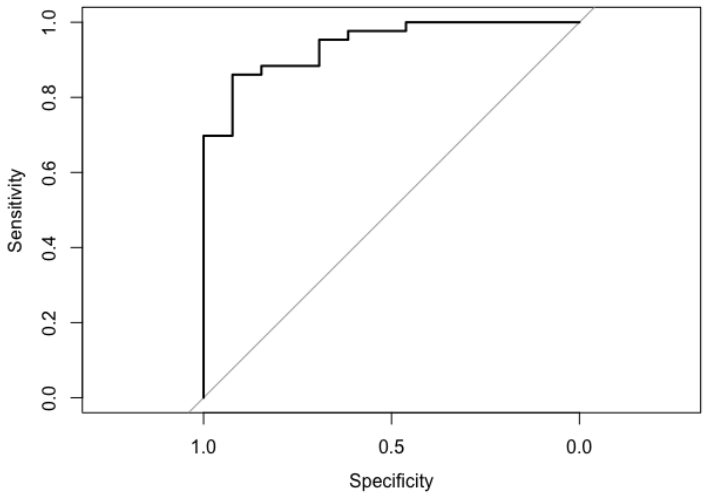

Figure 6: ROC Curve for Logistic Regression

Table 3: Confusion Matrix of LDA

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 10$	$b = 1$	$a + b$
	Live	$c = 3$	$d = 42$	$c + d$
	Total	$a + c$	$b + d$	N

disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease.

Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.873 for the LDA model and the ROC plot is given in Figure 7.

4.2.4 Naive Bayes (NB)

Naive Bayes is deployed in R using the naiveBayes() functions, which is a part of the e1071 library. The syntax for the naive Bayes is similar to lda() and qda(). I use naive Bayes classifier models each quantitative feature using the Gaussian

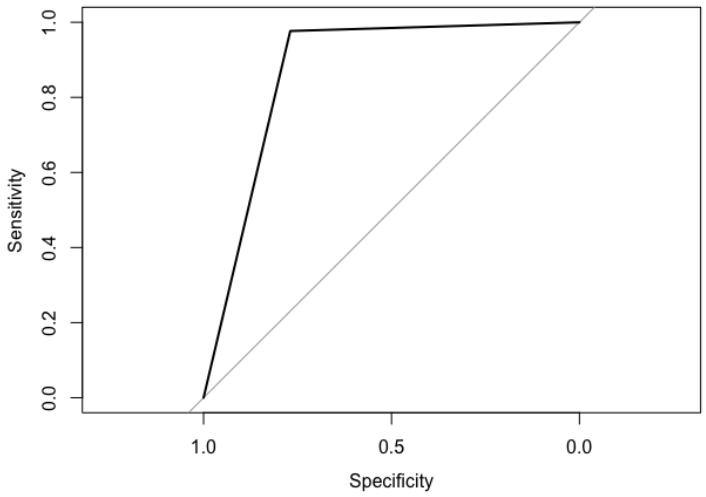

Figure 7: ROC Curve for Linear Discriminant Analysis

Table 4: Confusion Matrix of NB

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 11$	$b = 8$	$a + b$
	Live	$c = 2$	$d = 35$	$c + d$
	Total	$a + c$	$b + d$	N

distribution. Table 4, shows the confusion matrix of the NB algorithm. It indicate that NB model misclassified 2 die people with live and 8 live people misclassified with die people. For NB model the testing error is 0.179, sensitivity is 0.846, and specificity is 0.814, and accuracy of the model is 0.821. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease. Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.830 for the NB model and the ROC plot is given in Figure 8.

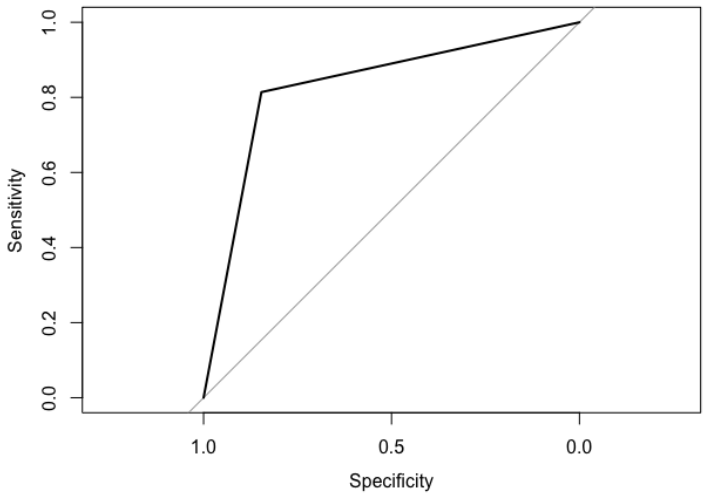

Figure 8: ROC Curve for Naive Bayes

4.2.5 Support Vector Machine (SVM)

I use R e1071 package to build SVM classification model using radial kernel. Table 5, shows the confusion matrix of the SVM algorithm. It indicate that SVM model misclassified 9 die people with live. For SVM model the testing error is 0.160, sensitivity is 0.308, and specificity is 1, and accuracy of the model is 0.839. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease.

Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.654 for the SVM model and the ROC plot is given in Figure 9.

4.2.6 Random Forest (RF)

A random forest model using is build with 10000 trees using the training data. I try different mtry (number of variables randomly sampled as candidates at each split) values varying from 1 to 15 and calculate the Out of Bag(OOB) error for each model.

Table 5: Confusion Matrix of SVM

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 4$	$b = 0$	$a + b$
	Live	$c = 9$	$d = 43$	$c + d$
	Total	$a + c$	$b + d$	N

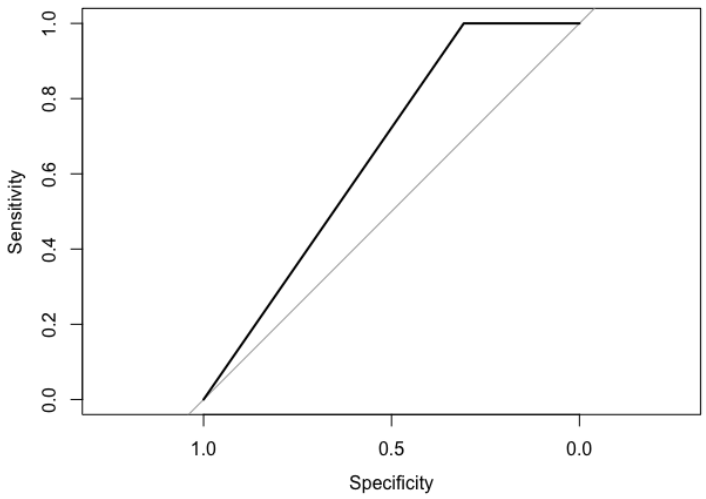

Figure 9: ROC Curve for Support Vector Machine

Table 6: Confusion Matrix of RF

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 5$	$b = 1$	$a + b$
	Live	$c = 8$	$d = 42$	$c + d$
	Total	$a + c$	$b + d$	N

I consider the lowest OOB error as the best model and mtry = 9 gives the lowest Out of Bag(OOB) error and it is consider as the best random forest model. I also create a variable importance plot to decide on the importance of the variable. Age, sex, and steroid are the three most important variable find my variable importance plot.

Table 6, shows the confusion matrix of the RF algorithm. It indicate that SVM model misclassified 8 die people with live and 1 live people misclassified with die people. For RF model the testing error is 0.161, sensitivity is 0.385, and specificity is 0.977, and accuracy of the model is 0.839. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease.

Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.681 for the RF model and the ROC plot is given in Figure 10.

4.2.7 Classification Tree (CT)

Tree package is used from the tree library to build this classification tree. Table 7, shows the confusion matrix of the Classification tree algorithm. It indicate that CT model misclassified 6 die people with live and 4 live people misclassified with die people. For CT model the testing error is 0.179, sensitivity is 0.539, and specificity is 0.907, and accuracy of the model is 0.821. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity

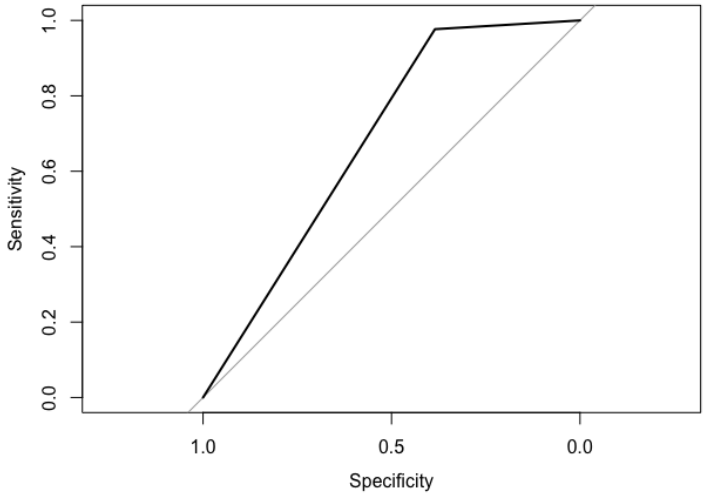

Figure 10: ROC Curve for Random Forest

Table 7: Confusion Matrix of Classification Tree

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 7$	$b = 4$	$a + b$
	Live	$c = 6$	$d = 39$	$c + d$
	Total	$a + c$	$b + d$	N

is the ability of a test to correctly identify people without the disease.
Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.723 for the CT model and the ROC plot is given in Figure 11.

4.2.8 Artificial Neural Network (ANN)

Neural net function is used from the neuralnet library to build the Neural Network model. In the ANN model 16 hidden neurons is used and the architecture of the model is shown in Figure 1. The logistic activation function is used in the model as it is best suited for predicting classification problem with binary outcome.

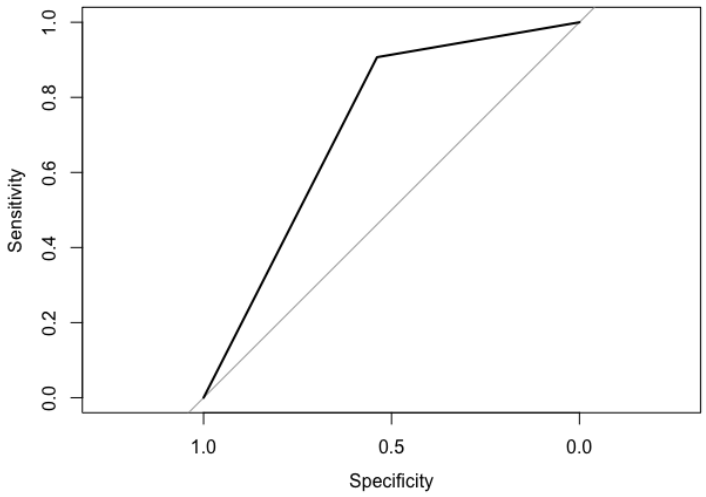

Figure 11: ROC Curve for Classification Tree

Table 8: Confusion Matrix of ANN

		Reference Level		
		Die	Live	Total
Prediction	Die	$a = 8$	$b = 3$	$a + b$
	Live	$c = 5$	$d = 40$	$c + d$
	Total	$a + c$	$b + d$	N

Table 8, shows the confusion matrix of the ANN. It indicate that CT model misclassified 5 die people with live and 3 live people misclassified with die people. For ANN model the testing error is 0.125, sensitivity is 0.615, and specificity is 0.930, and accuracy of the model is 0.857. Here the sensitivity indicates the ability of a test to correctly identify the patient with disease. On the other hand, specificity is the ability of a test to correctly identify people without the disease.

Also, I create the ROC curve and evaluate the AUC value. The AUC (Area under the curve) is 0.834 for the ANN model and the ROC plot is given in Figure 12.

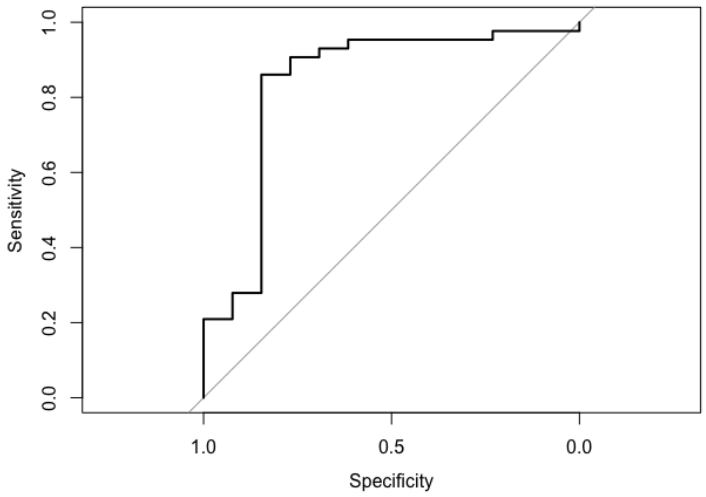

Figure 12: ROC Curve for Artificial Neural Network

Table 9: Performance Evaluation of Different Model

Criteria	KNN	LR	LDA	NB	SVM	RF	CT	ANN
Testing Error	0.161	0.125	0.071	0.179	0.160	0.161	0.179	0.125
Sensitivity	0.385	0.615	0.769	0.846	0.308	0.385	0.539	0.615
Specificity	0.977	0.934	0.977	0.814	1.00	0.977	0.907	0.930
Accuracy	0.839	0.875	0.929	0.821	0.839	0.839	0.821	0.857
AUC	0.6807	0.941	0.873	0.830	0.654	0.681	0.723	0.834
Precision	0.833	0.80	0.909	0.579	1.00	0.833	0.636	0.727
Recall	0.385	0.615	0.769	0.846	0.308	0.385	0.538	0.615
F1 - Score	0.526	0.696	0.833	0.688	0.471	0.526	0.583	0.667

5 Discussion and Conclusion

In this study, machine learning model is applied to attain the objective of the study. Table 9 shows the results summary of the widely used classification algorithm showing the testing error, sensitivity, specificity, accuracy, AUC, precision, recall, and

F-1 score. I perform quadratic discriminate analysis (QDA) also but as the response variable of the data-set is imbalanced, QDA perform poorly and that's why I exclude the part from the manuscript and keep it only in R code.
Precision generally indicate the number of positive class predictions that originally belongs to positive class. And, recall indicates the number of positive class predictions which is made from all the positive class example given in the data sets. AUC (Area under the ROC curve) measures the entire two dimensional area underneath the entire ROC curve. Accuracy is simply the proportion of individual who correctly classified the proportion of true positive and true negative.
F1 score is needed when we want to seek a balance between precision and recall. And, F1 score is calculated by taking the weighted average of the precision and recall. Among all the model, based on the different performance metrics listed in table 9, I can say that Neural network (feed-forward) and logistic regression are performs best as it have lowest testing error with high accuracy. Support vector machine and random forest gives almost similar results.
All in all I will go with neural network algorithm because it has the room to increase the accuracy by increasing the hidden neuron and layer but as our data-set is small we didn't increase hidden neuron very much because it trends to over-fit. In future if the cases number of the data-set may increase, I am sure the neural network can able to decrease the error to a negligible quantity.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Warodom Geamsakul, Takashi Matsuda, Tetsuya Yoshida, Kouzou Ohara, Hiroshi Motoda, Takashi Washio, Hideto Yokoi, and Katsuhiko Takabayashi. Anal-

ysis of hepatitis dataset by decision tree based on graph-based induction. In *New Frontiers in Artificial Intelligence*, pages 5–28. Springer, 2003.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[4] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[5] T Karthikeyan and P Thangaraju. Analysis of classification algorithms applied to hepatitis patients. *International Journal of Computer Applications*, 62(15), 2013.

[6] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[7] P Nancy, V Sudha, and R Akiladevi. Analysis of feature selection and classification algorithms on hepatitis data. *Int. J Advanced Res Comp. Eng. Technol*, 6(1):19–23, 2017.

6 Appendix

6.1 R Code

```
"
Author: Md Salman Rahman
Course: MATH 6333 Statistical Learning
Course Instructor: Dr. Tamer Oraby
Final Project
```

```
"
setwd("/Users/salman/OneDrive - The University of Texas-Rio Grande Valley/Course_vi

library(ggplot2)
library(e1071)
library(ROCR)
library(irr)
library(caret)
#loading data set

hepa_data <-  read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/h

colnames(hepa_data) <- c('Class','AGE','SEX','STEROID',
            'ANTIVIRALS','FATIGUE','MALAISE','ANOREXIA','LIVER BIG','LIVER FIRM','S
            'VARICES','BILIRUBIN','ALK PHOSPHATE','SGOT','ALBUMIN','PROTIME','HISTO

str(hepa_data)

# histogram
hist(hepa_df$Class)
ggplot(data=hepa_df, aes(hepa_data$Class)) +
  geom_histogram()

plot(hepa_df$Class)
# checking NA value
table(is.na(hepa_data))

summary(hepa_data)
```

```
hepa_data[hepa_data == '?'] <- NA

hepa <- function(a){
  a <-as.numeric(as.character(a))
  a[is.na(a)] = median(a,na.rm = TRUE)
  a
}

hepa_data <- data.frame(apply(hepa_data,2,hepa))

summary(hepa_data)
str(hepa_data)

# Correlation plot
pairs(hepa_df[,])

library(corrplot)
# correlation matrix
corrplot(cor(hepa_data[,]),
         method = "number",
         type = "upper" # show only upper side
)

#identifying outlier
boxplot(hepa_data$AGE~hepa_data$Class, main="Age",ylab="",xlab="")
boxplot(hepa_data$ALK.PHOSPHATE~hepa_data$Class, main="Alk_Phosphate",ylab="",xlab=
```

```
norm <- function(p){
  return((p-min(p))/max(p)-min(p))
}

hepa_df <- as.data.frame(lapply(hepa_data, norm))

summary(hepa_df)
str(hepa_df)



hepa_df$Class <- ifelse(hepa_df$Class == -0.5, 'Live','Die')
hepa_df$Class <- as.factor(hepa_df$Class)
hepa.df.final <- hepa_df
str(hepa.df.final)

# training and testing data
set.seed(123)
ind = sample(2,nrow(hepa_df),replace=TRUE,prob=c(0.65,0.35))
training = hepa_df[ind==1,]
testing = hepa_df[ind==2,]
train <- as.data.frame(training)
test <- as.data.frame(testing)



# K nearest neighbor
```

```
library(class)
library(MASS)
Choose_K<-function(K,M=1000){
  MSE<-0
  for(i in 1:M){
    train_knn<-sample(155,155-31) # for 5-fold CV
    KNN<-knn(hepa_df[train_knn,2:20], hepa_df[-train_knn,2:20], hepa_df[train_knn,1
             k=K,prob=TRUE,use.all=FALSE)
    MSE<-MSE+mean(KNN != hepa_df[-train_knn,1])
  }
  CVK<-MSE/M
  return(CVK)
}


library("purrr")

LK<-1:35
vCVK<-LK %>% map(function(K) Choose_K(K))

vCVK<-unlist(vCVK)
vCVK<-as.data.frame(vCVK)
colnames(vCVK)<-c("CVK")

ggplot()+
  geom_point(data=vCVK,aes(x=LK,y=CVK))+
  xlab("K")+
  ylab("CV")
```

```
LK[which.min(vCVK$CVK)]

library(class)
KNN <- knn(train[,2:20],train[,2:20], train[,1],
           k=9, prob=TRUE,use.all=TRUE) # with appropriate k = 9


# prediction using test data

library(class)
knn_test <- knn(train[,2:20],test[,2:20],train[,1],
                k=9, prob=TRUE,use.all = TRUE)

# error

error <- mean(knn_test != test[,1])
error

# confusion matrix
library(caret)
knn.conf <- confusionMatrix(data=as.factor(knn_test),reference = test$Class)
knn.conf$byClass


# ROC curve

library(pROC)
```

```
roc.knn <- plot.roc(test$Class, as.numeric(knn_test))

#AUC
roc.1 <- roc(test$Class, as.numeric(knn_test))
auc(roc.1)



# logistic regression

glm.fit.case <- glm(Class ~., data=train, family = "binomial")
summary(glm.fit.case)
coef(glm.fit.case)
# prediction
logis.pred <- predict(glm.fit.case, newdata=test, type = "response")

logis.pred.1 <-ifelse(logis.pred>=0.5,"Live","Die")

# confusion matrix
library(caret)
logis.confu <- confusionMatrix(data=as.factor(logis.pred.1),reference = test$Class)
logis.confu$byClass
# ROC curve

library(pROC)
roc.logis <- plot.roc(test$Class, as.numeric(logis.pred))

#AUC
roc.1 <- roc(test$Class, as.numeric(logis.pred))
```

```
auc(roc.1)

# error

error <- mean(logis.pred.1 != test[,1])
error



# Linear Discriminant Analysis

library(MASS)

lda_fit <- lda(Class ~., data=train)
lda_fit

plot(lda_fit, col = as.integer(train$Class))

library(klaR)

partimat(Class ~ ., data=train, method="lda")

# prediction

lda_predict <- predict(lda_fit,test)$class
error_lda <- mean(lda_predict != test[,1])

# confusion matrix
```

```
library(caret)
expected <- factor(test[,1])
predicted <- factor(lda_predict)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)
results$byClass

# ROC curve
library(pROC)
roc.lda <- plot.roc(test$Class, as.numeric(lda_predict))


#AUC
roc.1 <- roc(test$Class, as.numeric(lda_predict))
auc(roc.1)


# Quadratic Discriminant Analysis

qda_fit <- qda(Class ~., data=train)
qda_fit

# prediction

qda_predict <- predict(qda_fit,test)$class
error_qda <- mean(qda_predict != test[,1])
```

```
# Naive Bayes Classification
library(e1071)

nb_fit <- naiveBayes(Class ~., data=train)

nb_fit

# prediction

naive_predict <- predict(nb_fit,test)
error_nb <- mean(naive_predict != test[,1])

# confusion matrix

table(naive_predict,test[,1])
library(caret)

expected <- factor(test[,1])
predicted <- factor(naive_predict)
results <- confusionMatrix(data=predicted, reference=expected)
print(results)

results$byClass

# ROC curve
```

```
library(pROC)
roc.nb <- plot.roc(test$Class, as.numeric(naive_predict))


#AUC
roc.1 <- roc(test$Class, as.numeric(naive_predict))
auc(roc.1)

# Support Vector Machine

set.seed(101)
library(e1071)

#support vector machine
svm.fit <- svm(Class ~ ., data=train, kernel = "radial")

svm.pred <- predict(svm.fit, newdata=test, outcome="test")

error_svm <- mean(svm.pred != test[,1])


# confusion matrix
library(caret)
svm.confu <- confusionMatrix(data=as.factor(svm.pred),reference = test$Class)
svm.confu$byClass

# ROC curve
```

```
library(pROC)
plot.roc(test$Class, as.numeric(svm.pred))

#AUC
roc.curve <- roc(test$Class, as.numeric(svm.pred))
auc(roc.curve)



# Random Forest
library(randomForest)
set.seed(101)

set.seed(101)
oob_error <- function(tree = 10000, mtr){
  mar.rf<- randomForest(Class ~., data=train, ntree=tree, mtry=mtr)
  error <- tail(mar.rf$err.rate,1)[,1]
  return(error)
}
x =seq(1:15)
y=c()

for (i in x) {
  p <- oob_error(tree = 10000, mtr =i)
  y=c(y,p)

}
lowest.x <- y[x]==min(y)
```

```
# mar 8 gives the lowest oob error
mar_rf.8 <- randomForest(Class ~ ., data = train, ntree=10000, mtry=8)
importance <- mar_rf.8$importance
plot(importance)

rf_pred.8 <- predict(mar_rf.8, newdata=test)
error <- mean(rf_pred.8 != test[,1])

# confusion matrix
library(caret)
rf.confu <- confusionMatrix(data=as.factor(rf_pred.8),reference = test[,1])
rf.confu$byClass

# ROC curve
library(pROC)
plot.roc(test$Class, as.numeric(rf_pred.8))

#AUC
roc.2 <- roc(test$Class, as.numeric(rf_pred.8))
auc(roc.2)


# Neural Network

require(neuralnet)

nn <- neuralnet(Class ~ ., data=train, hidden = 16, act.fct = "logistic", linear.ou
```

```
plot(nn)

# prediction
nn.predict <- predict(nn, test,type = "class")
nn.pred <-ifelse(nn.predict>=0.5,"Live","Die")
error <- mean(nn.pred[,2] != test[,1])




# confusion matrix
library(caret)
nn.confu <- confusionMatrix(data=as.factor(nn.pred[,2]),reference = test$Class)
nn.confu$byClass

# ROC curve
library(pROC)
plot.roc(test$Class, as.numeric(nn.predict[,2]))

#AUC
roc.2 <- roc(test$Class, as.numeric(nn.predict[,2]))
auc(roc.2)



# classification tree
```

```
library(tree)
tree.fit <- tree(Class ~., data=train)

summary(tree.fit)

# prediction

tree.pred <- predict(tree.fit, test, type = "class")
error.tree <- mean(tree.pred != test[,1])

# confusion matrix
library(caret)
tree.confu <- confusionMatrix(data=as.factor(tree.pred),reference = test$Class)
tree.confu$byClass

# ROC curve
library(pROC)
plot.roc(test$Class, as.numeric(tree.pred))

#AUC
roc.2 <- roc(test$Class, as.numeric(tree.pred))
auc(roc.2)
```