Final Project Report

STAT 515 – 006

Group 3

Koustav Roy, Indu Priya Sharma, and Carter Forry

Project Purpose

The focus of this report is to analyze death knowing heart failure has been diagnosed and given predictive variables based on Chicco and Jurman (Ref) data recorded in 2020. Since the desired outcome of death is binary, logistical regressions and tree-based models are appropriate analytical techniques to implement. The overall questions that are to be addressed are as follows:

1. What predictive variables best predict if a patient dies given heart failure?
2. What modeling technique produces the most accurate outcome: logistical regression or tree-based modeling?
3. Can we draw the same conclusions as the Chicco and Jurman (Ref) claim in their research article?

The research article concluded that gender is not a significant variable and found that age, ejection fraction, serum sodium, anemia, and blood pressure are the most influential variables to predict death. Chicco and Jurman (Ref) used a Cox regression versus a standard logistical regression; therefore, some differences in conclusions are expected.

Method of Analysis

To compare logistical regression and tree-based modeling, the data set is split into a training data set and a testing data set. The training data set includes 80% of the initial observations and the testing data set includes 20% of the initial observations. This breakdown is shown in Table 1.

Table 1: Train/Test Data Breakdown

Number of Observations	
Training Data Set	Testing Data Set
239	60

The logistical regression model is found using R's "glm" function. In this approach, all predictive variables are used. This produces a prediction percentage between 0 and 1. If the probability is greater than or equal to 0.5, the model prediction indicates death. If the calculated probability is less than 0.5, the patient lives.

Using K-Nearest Neighbour approach we try to find optimal number of clusters to predict the categorical data and the accuracy rate. This is done "knn" function and our assumption is that data should not be properly concentrated into any region where optimal clustering is not possible.

Using tree-based methods and random forest techniques, the same approach is taken as the logistical regression: if the predicted value is greater than 0.5, a Death has occurred. R has powerful built-in tree and random forest functions that will be utilized for this analysis.

Initial Data Set Investigation

The predictive heart failure data set comes from Chicco and Jurman (Ref) and uses 12 predictive variables listed below in Table 2.

Table 2: Heart Failure Predictive Variables

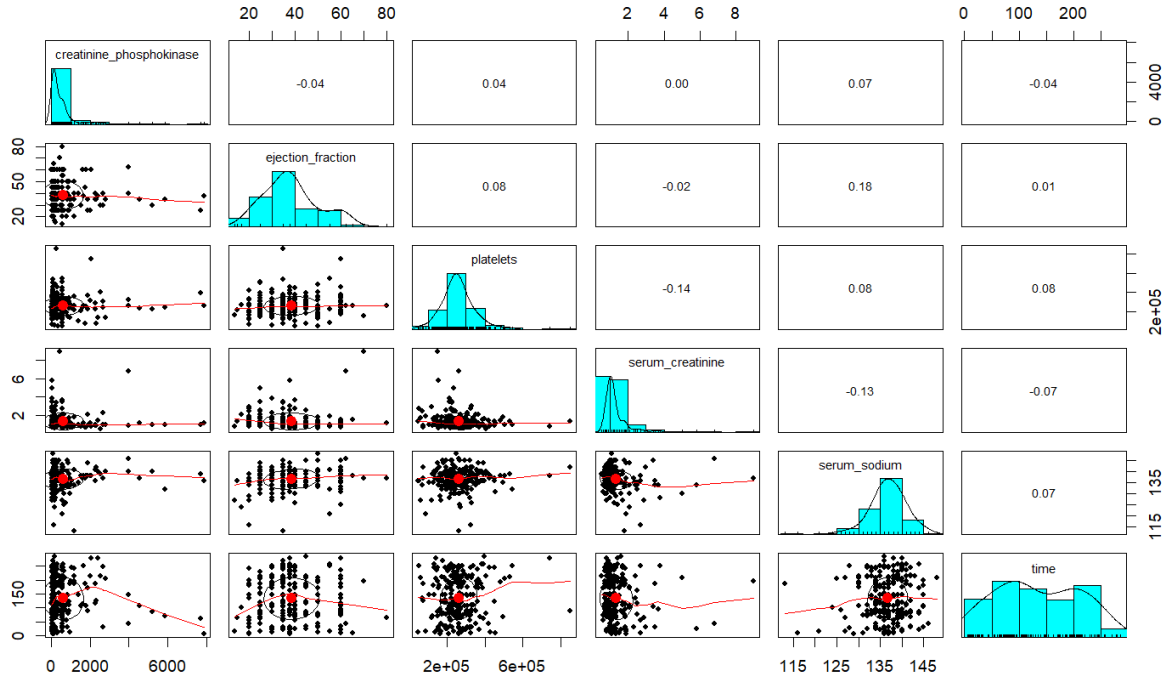
Age	Anemia	High blood pressure
Creatinine Phosphokinase (CPK)	Diabetes	Ejection fraction
Platelets	Sex	Serum creatinine
Serum sodium	Smoking	Time

The definition of each variable is as follows:

- Age: age of the patient (years)
- Anaemia: decrease of red blood cells or hemoglobin (boolean)
- High blood pressure: if the patient has hypertension (boolean)
- Creatinine phosphokinase (CPK): level of the CPK enzyme in the blood (mcg/L)
- Diabetes: if the patient has diabetes (boolean)
- Ejection fraction: percentage of blood leaving the heart at each contraction (percentage)
- Platelets: platelets in the blood (kiloplatelets/mL)
- Sex: woman or man (binary)
- Serum creatinine: level of serum creatinine in the blood (mg/dL)
- Serum sodium: level of serum sodium in the blood (mEq/L)
- Smoking: if the patient smokes or not (boolean)
- Time: follow-up period (days) *after diagnosis

Before any modeling is applied to the data set, correlations between the numerical predictive variables need to be determined (Figure 1). Categorical variables are left out of this preliminary analysis.

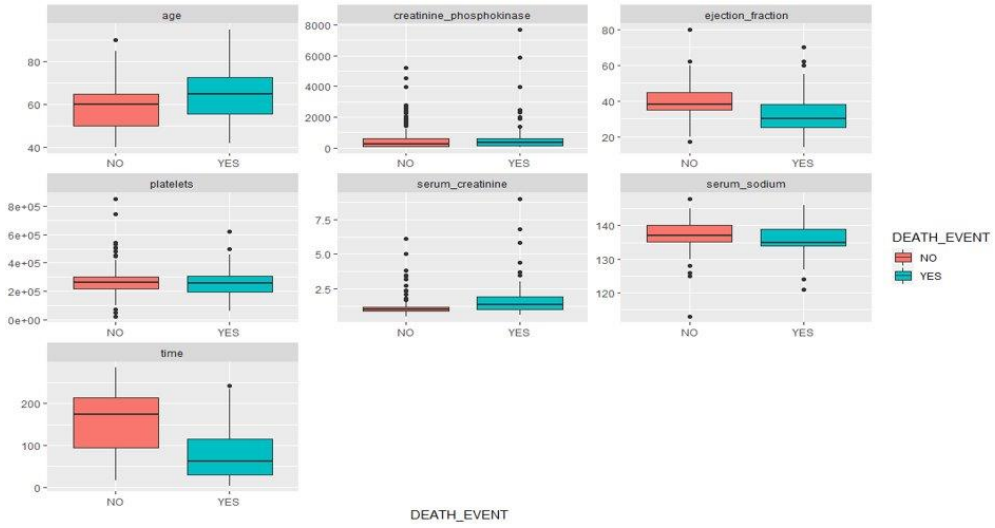
Figure 1: Correlation between Predictive Variables



Based on Figure 1, no relationships exist between predictive variables which is desired. One thing to note is that the observations are skewed towards male patients and there are more survivals than deaths which could lead to a slightly imbalanced dataset. In the study there are 194 male observations and only 104 female observations. There is no missing data, so each observation corresponds to an outcome (death or survival).

Box plots can also be used to show potential imbalance in the data set. Figure 2 summarizes numeric variables and their given outputs. Age, Ejection Fraction, and Time variable are slightly imbalanced

Figure 2: Box-Plot Analysis of Data Set and Potential Imbalance



Logistical Regression

Using a logistical regression detailed in the method described in the previous section, the statistically significant variables to predict death given heart failure are shown in Table 3 (not shown in order of significance).

Table 3: Logistical Regression Significant Variables

Variable Name	P Value
Age	0.002802
Ejection Fraction	0.000158
Serum Creatinine	0.000521
Sex	0.055341
Time	3.01e-09

Based on the P values in Table 3, the top three most statistically significant variables are in order: time, ejection fraction, and serum creatinine. Chicco and Jurman (Ref) also concluded in their research paper that these are influential predictive variables. One important observation is that Sex is not a relatively good predictive variable which Chicco and Jurman (Ref) also concluded. The importance of time and serum creatinine on the logarithmic regression are shown in Figure 3 and 4 respectively.

Figure 3: Probability of Death given Time since Diagnosis

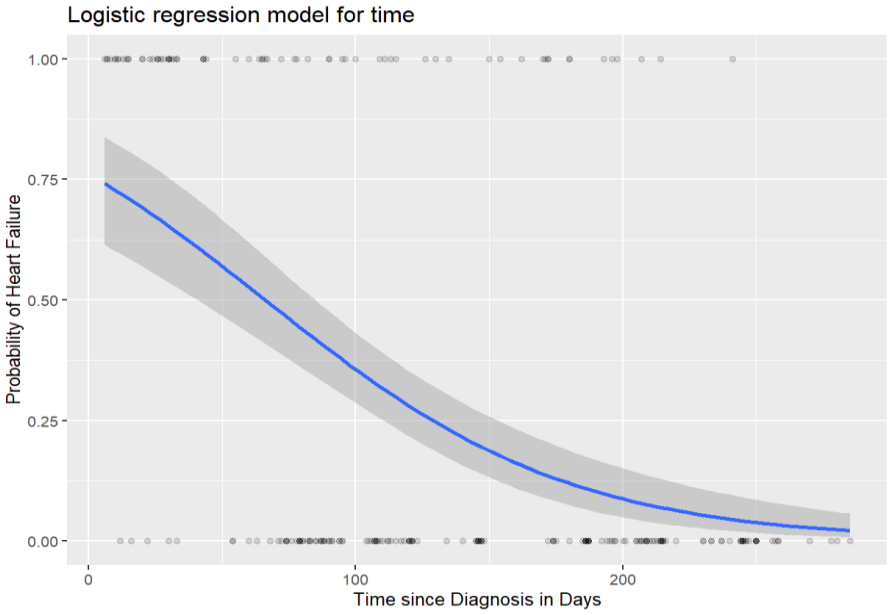
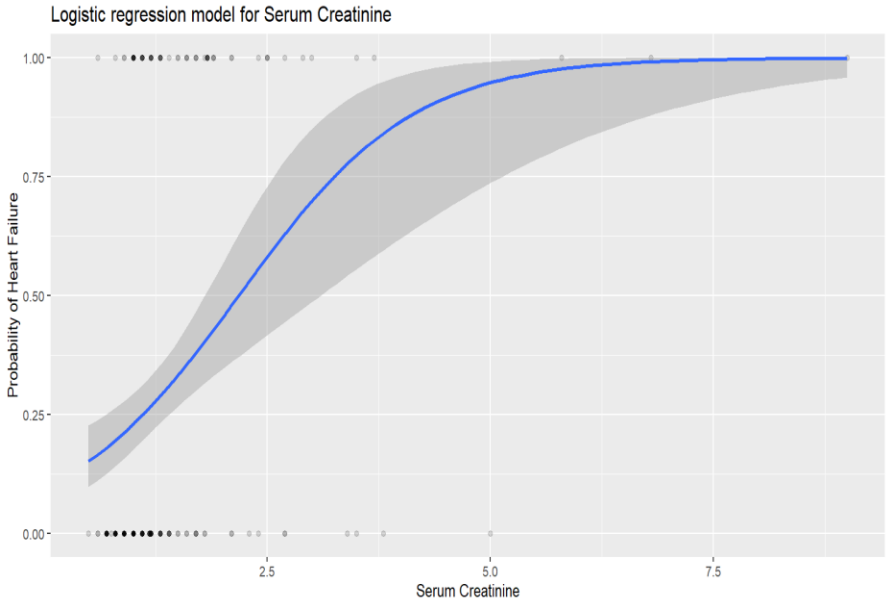


Figure 4: Probability of Death given Serum Creatinine



Using the logistical regression model and the test training data set, the average correct prediction can be calculated. The training and testing datasets yield similar results in terms of correct predictions of death.

Average Correct Prediction [Train Data set] ≈ **84%**

Average Correct Prediction [Test Data set] ≈ **85%**

The confusion matrix for the same logarithmic model using the training and testing data set are shown in Table 4 and 5 respectively.

Table 4: Logistical Model Train Data Confusion Matrix

Actual Outcome	**Model Prediction**	
	No Heart Failure	Heart Failure
No Heart Failure	155	13
Heart Failure	25	46

Table 5: Logistical Model Test Data Confusion Matrix

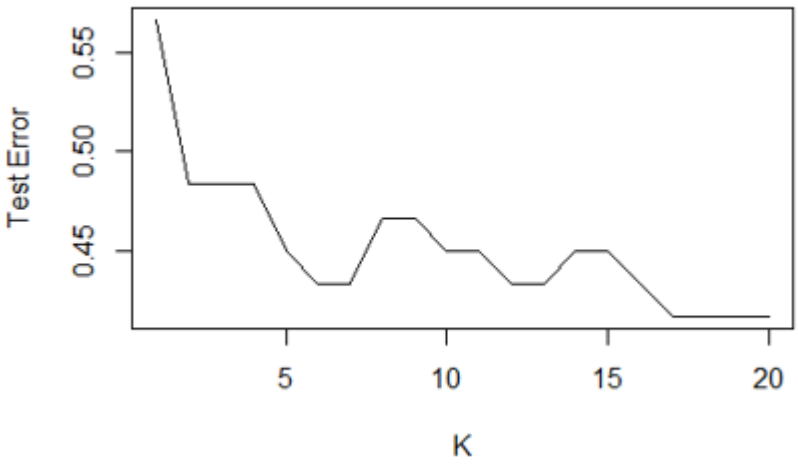
Actual Outcome	**Model Prediction**	
	No Heart Failure	Heart Failure
No Heart Failure	31	4
Heart Failure	5	20

In the training data set the false negative rate is much larger than the false positive rate. It is unclear what is causing the difference in error types. In the test confusion matrix, the false negative and false positive rate are closer; however, a significant number of observations are not used to generate the table compared to the training data set. Similar differences could occur if the test data set were to increase in size.

K-Nearest Neighbor

The k-nearest neighbors (KNN) algorithm is a data classification method to estimating the likelihood that a data point will become a member of one class or another based on what group the nearest neighbors belong to. The algorithm decides nearest K points from which data points distance is measured in multidimensional environment and assigned to groups nearest to that. Creating too many points might overfit the data whereas too few points might have biased results. Selecting the number of points considering model complexity and biasness is the main motive. So, we have performed the comparative performance with points from 1 to 20 shown in Figure 5:

Figure 5: K-Nearest Neighbor Test Error for different clusters



From the figure we can conclude that the error rate is quite high for all number of points. So, this can be concluded that K-Nearest Neighbor algorithm might not be a good fit for our data.

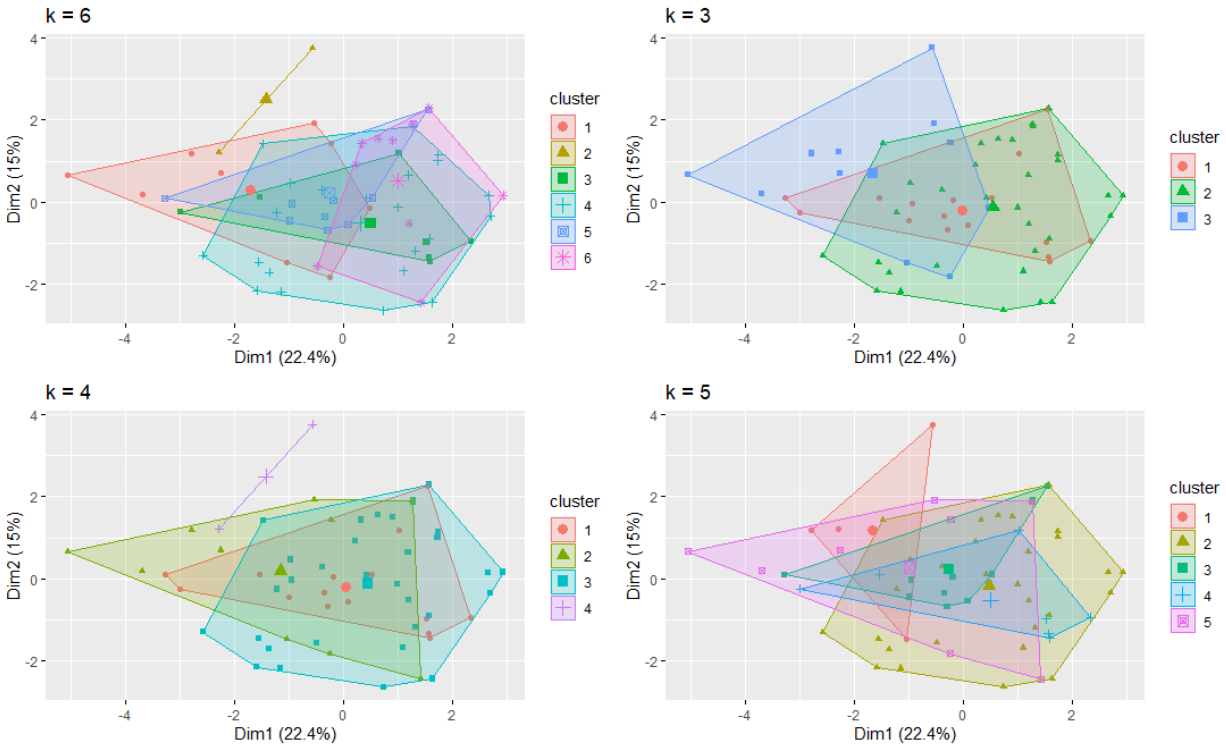
Here we are selecting 6 nearest points as this will reduce the model complexity significantly and the biasness is almost name. The confusion matrix with 6 points is as shown in Table 6:

Table 6: K-Nearest Neighbor Confusion Matrix for 6 clusters

K-Nearest Neighbor Test Data Confusion Matrix		
Actual Outcome	**Model Prediction**	
	No Heart Failure	Heart Failure
No Heart Failure	31	4
Heart Failure	23	2

The accuracy rate is around 58%, which is not satisfactory. One more notable point is we have high false negative cases which is highly undesirable in medicare data analysis. We tried to dig into further why the algorithm did not do well with the data. We tried some visualization to understand it in Figure 6

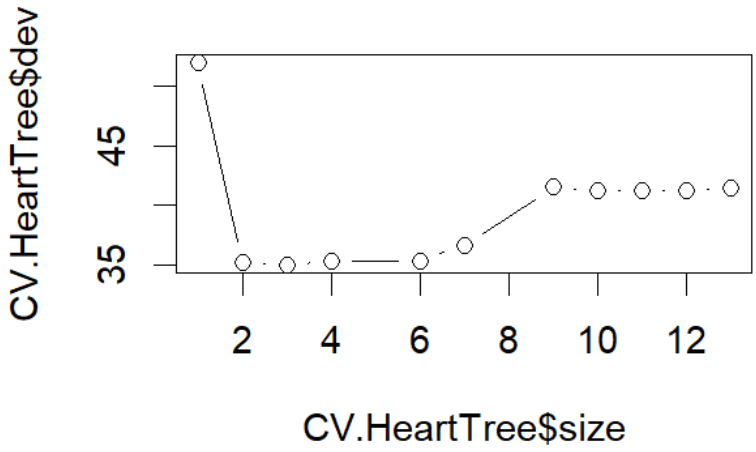
Figure 6: K-Means Cluster Data Visualization



Tree Based Modeling

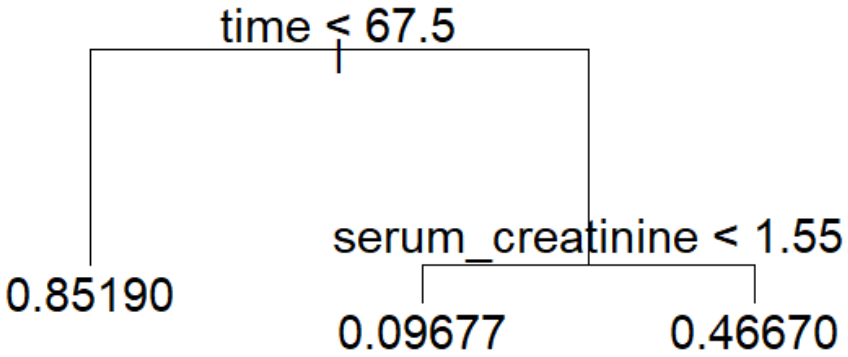
With Tree based modeling, two approaches are taken utilizing R's built-in functions for trees and random forests. To create the most efficient tree, cross validation is done to determine the appropriate number of branches. The summary of the cross validation is shown in Figure 7 and concludes that three branches are sufficient for the tree model.

Figure 7: Tree Model Cross-Validation



Using three branches on the tree model yields the following predictive tree (Figure 8). Where an outcome greater than 0.5 indicates a death event which is the same approach the logistical regression model uses.

Figure 8: Cross-Validated Tree Model



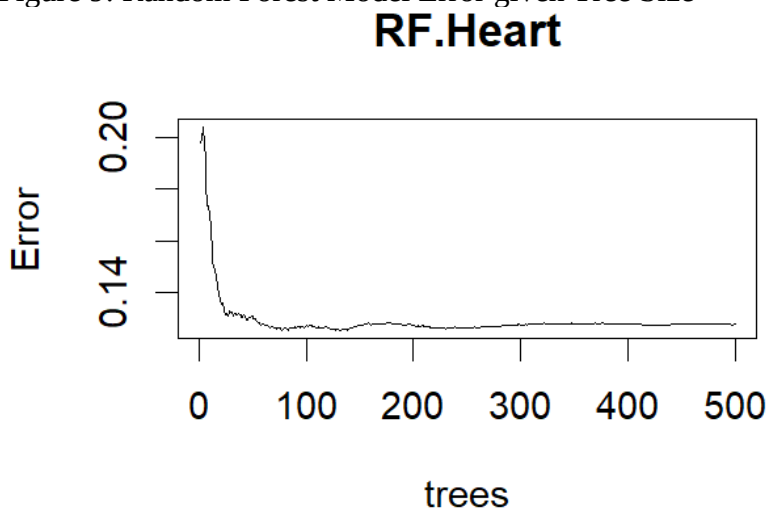
This model yields an accuracy of approximately 80% (lower than logistical regression). The associated confusion matrix is shown in Table 7

Table 7: Tree Model Test Data Confusion Matrix

Actual Outcome	**Model Prediction**	
	No Heart Failure	Heart Failure
No Heart Failure	31	8
Heart Failure	4	17

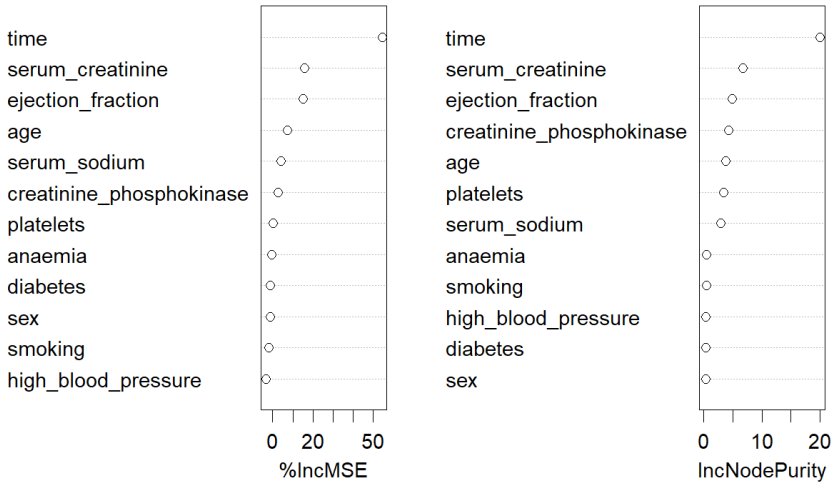
The next step is to include a random forest approach which uses a more complex number of variable interactions. The model error and the number of included trees is summarized in Figure 9

Figure 9: Random Forest Model Error given Tree Size



Random Forest is capable of a model accuracy of 89.3% which is an improvement over the cross-validated tree model and the logistical regression method. Random Forest also outputs a summary of how influential each variable is to the model outcome (Figure 10). Based on this Figure, the three most significant variables are time, serum creatinine, and ejection fraction. These are the same variables that the logistical regression model produced as well.

Figure 10: Random Forest Significant Variables



Conclusions

Using a logistical regression and tree-based model, the following conclusions can be drawn, answering the project purpose:

1. Based on the logarithmic regression analysis, the most influential variables in predicting heart failure are: Time, Ejection Fraction, and Serum Creatinine.
2. Out of the three models (logistical, tree, and random forest), random forest yielded the highest model accuracy.

Model Accuracy		
Logistical	Tree-Based	Random Forest
85%	80%	89.3%

3. Using a logistical and tree-based model, similar conclusions to Chicco and Jurman (Ref) can be found. One major finding of Chicco and Jurman is that gender does not play as important role as other studies suggest (Ref) in determining whether the patient dies given heart failure. The logistical model and tree-based model also concluded that the more influential variables in the model are: age, ejection fraction, serum creatinine, sex, and time.

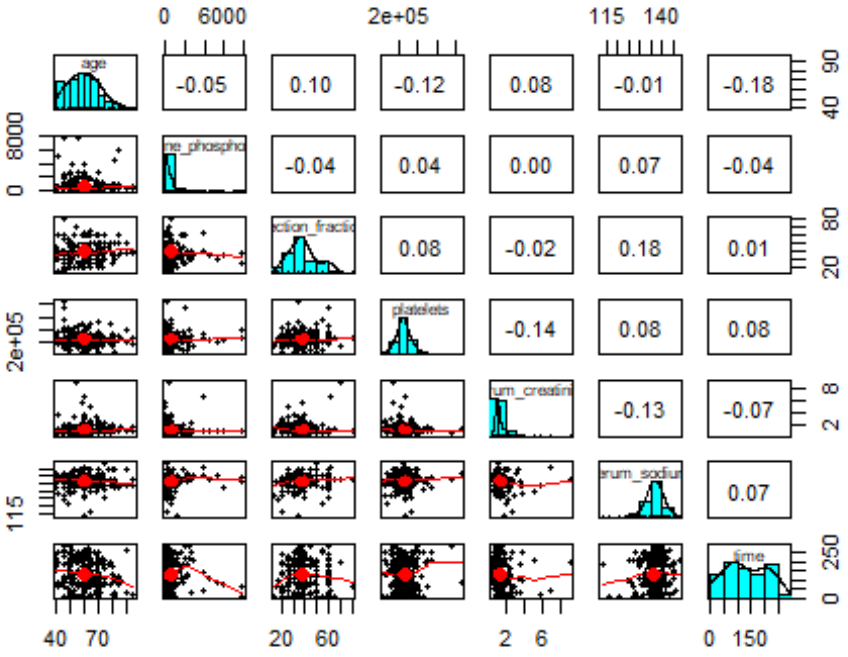
References

Davide Chicco, Giuseppe Jurman: "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone". BMC Medical Informatics and Decision Making 20, 16 (2020).

Appendix

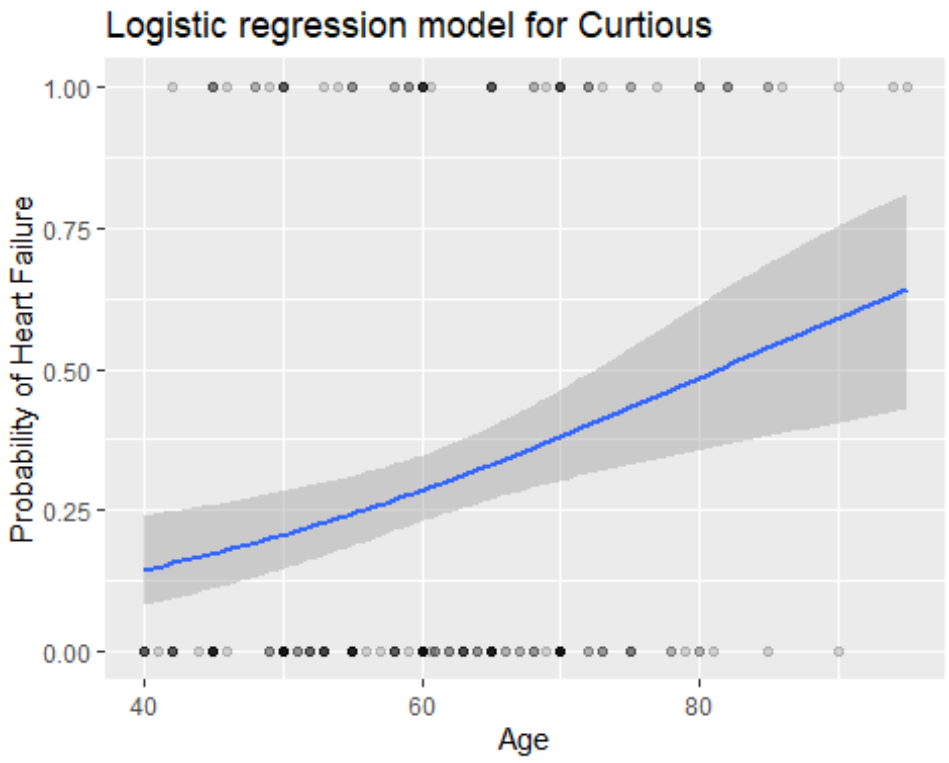
Logistical Regression R Code and Outputs

```
# linear regression of death rate data

# libraries
library(ISLR)
library(GGally)
library(tidyverse)
library("reshape2")
library(psych)
library(ggplot2)

# import given style plotting
source('hw.R')

# import data
TrainData = read.csv('TrainData.csv')
colnames(TrainData)[1] = c('age')
TestData = read.csv('TestData.csv')
colnames(TestData)[1] = c('age')

# correlation for numerical vairables
CorrData = TrainData
CorrData = subset(CorrData, select = -c(anaemia,diabetes,high_blood_pressure,
sex,smoking,DEATH_EVENT) )
# Output to see the data
pairs.panels(CorrData)
```

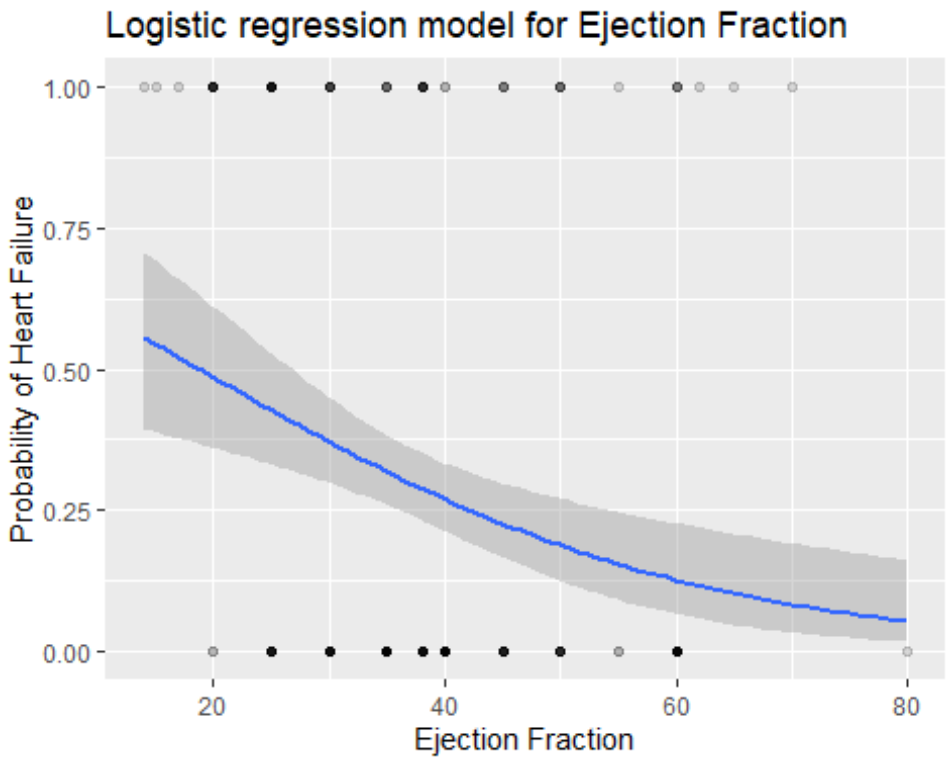


```
####################################
# Log regression using all variables
TrainAllVariables = glm(DEATH_EVENT~.,data=TrainData,family="binomial")
summary(TrainAllVariables)

##
## Call:
## glm(formula = DEATH_EVENT ~ ., family = "binomial", data = TrainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1855  -0.5524  -0.2521   0.4385   2.5831
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                9.249e+00  6.147e+00   1.505 0.132431
## age                        5.481e-02  1.834e-02   2.989 0.002802 **
## anaemia                    2.954e-01  4.105e-01   0.719 0.471844
## creatinine_phosphokinase   2.562e-04  1.783e-04   1.437 0.150682
## diabetes                   9.997e-02  3.958e-01   0.253 0.800580
## ejection_fraction         -6.904e-02  1.828e-02  -3.778 0.000158 ***
## high_blood_pressure       -3.854e-01  4.144e-01  -0.930 0.352450
## platelets                 -1.991e-06  2.346e-06  -0.849 0.396096
## serum_creatinine           8.540e-01  2.461e-01   3.470 0.000521 ***
## serum_sodium              -6.539e-02  4.331e-02  -1.510 0.131103
## sex                       -9.093e-01  4.745e-01  -1.916 0.055341 .
## smoking                   -3.593e-02  4.823e-01  -0.074 0.940619
## time                      -1.995e-02  3.364e-03  -5.931 3.01e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 290.80  on 238  degrees of freedom
## Residual deviance: 173.87  on 226  degrees of freedom
## AIC: 199.87
##
## Number of Fisher Scoring iterations: 6

exp(coef(TrainAllVariables))

##              (Intercept)                       age                   anaemia
##             1.038939e+04              1.056341e+00             1.343623e+00
## creatinine_phosphokinase                  diabetes          ejection_fraction
##             1.000256e+00              1.105141e+00             9.332867e-01
##      high_blood_pressure                 platelets           serum_creatinine
##             6.802090e-01              9.999980e-01             2.348911e+00
##             serum_sodium                       sex                   smoking
##             9.367043e-01              4.028017e-01             9.647118e-01
```

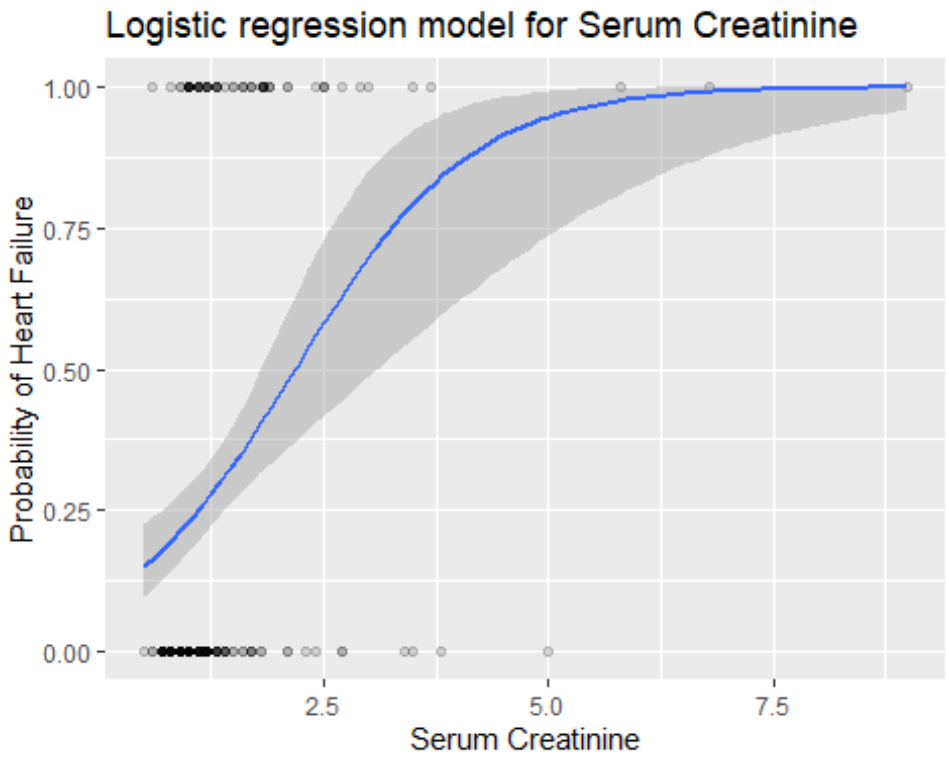
```
##                       time
##              9.802446e-01

confint(TrainAllVariables)

##                                    2.5 %        97.5 %
## (Intercept)                -2.730217e+00  2.169017e+01
## age                         2.016980e-02  9.247460e-02
## anaemia                    -5.128723e-01  1.105929e+00
## creatinine_phosphokinase -7.366832e-05  6.458264e-04
## diabetes                   -6.815564e-01  8.780581e-01
## ejection_fraction          -1.066461e-01 -3.459893e-02
## high_blood_pressure        -1.216699e+00  4.167669e-01
## platelets                  -6.777209e-06  2.381760e-06
## serum_creatinine            4.364223e-01  1.434204e+00
## serum_sodium               -1.522424e-01  2.009022e-02
## sex                        -1.867933e+00  3.719182e-03
## smoking                    -9.881724e-01  9.143024e-01
## time                       -2.705403e-02 -1.377881e-02

# log plots of significant variables
TrainData %>%
  mutate(prob = ifelse(DEATH_EVENT == "1", 1, 0)) %>%
  ggplot(aes(TrainData$age, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model for Curtious") +
  xlab("Age") +
  ylab("Probability of Heart Failure")
```



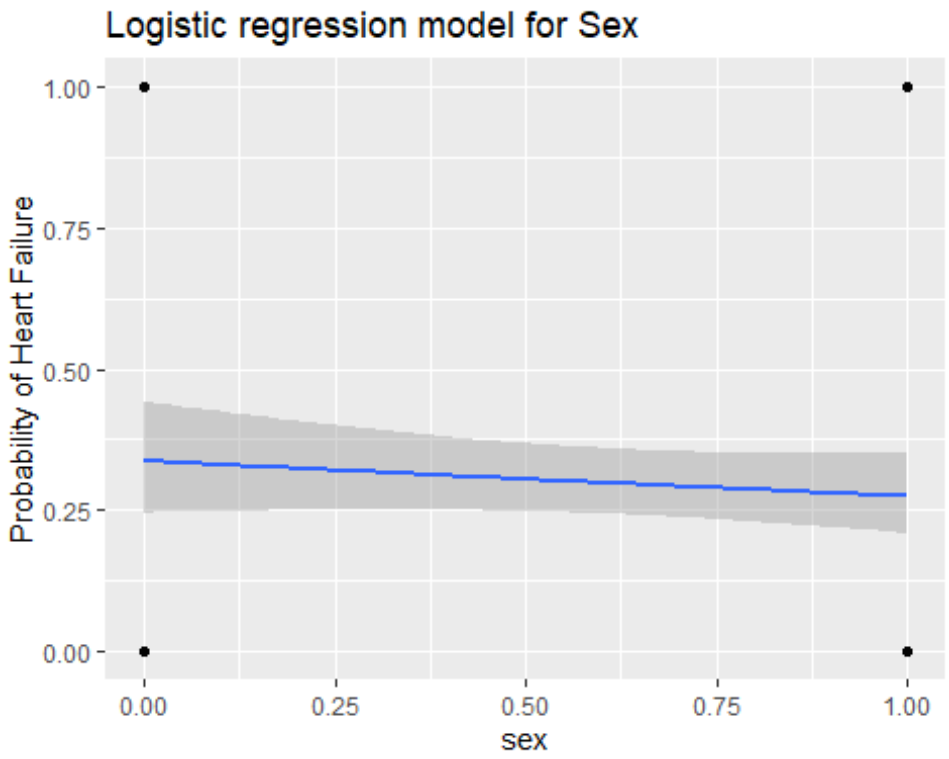

```
TrainData %>%
  mutate(prob = ifelse(DEATH_EVENT == "1", 1, 0)) %>%
  ggplot(aes(TrainData$ejection_fraction, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model for Ejection Fraction") +
  xlab("Ejection Fraction") +
  ylab("Probability of Heart Failure")
```




```
TrainData %>%
  mutate(prob = ifelse(DEATH_EVENT == "1", 1, 0)) %>%
  ggplot(aes(TrainData$serum_creatinine, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model for Serum Creatinine") +
  xlab("Serum Creatinine") +
  ylab("Probability of Heart Failure")
```




```
TrainData %>%
  mutate(prob = ifelse(DEATH_EVENT == "1", 1, 0)) %>%
  ggplot(aes(TrainData$sex, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model for Sex") +
  xlab("sex") +
  ylab("Probability of Heart Failure")
```




```
TrainData %>%
  mutate(prob = ifelse(DEATH_EVENT == "1", 1, 0)) %>%
  ggplot(aes(TrainData$time, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model for time") +
  xlab("Time since Diagnosis in Days") +
  ylab("Probability of Heart Failure")
```

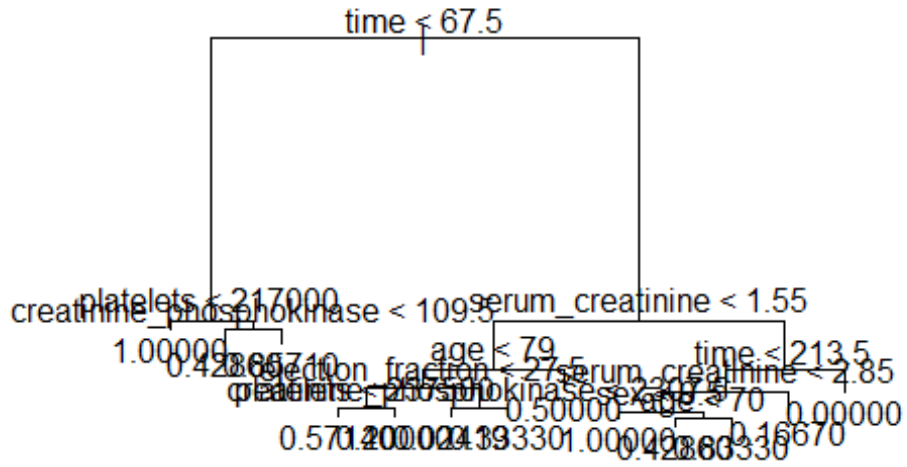

```
# Training Outcome
PredictionTraining = predict(TrainAllVariables, TrainData, type = "response")
TrainPrediction = rep("0",nrow(TrainData))
TrainPrediction[PredictionTraining>.5]="1"
TrainPrediction[PredictionTraining<=.5]="0"

mean(TrainPrediction==TrainData$DEATH_EVENT)

## [1] 0.8410042

table(TrainData$DEATH_EVENT,TrainPrediction)

##    TrainPrediction
##       0   1
##   0 155  13
##   1  25  46

# Test Outcome
dim(TestData)

## [1] 60 13

Prediction = predict(TrainAllVariables, TestData, type = "response")
TestPrediction = rep("0",nrow(TestData))
TestPrediction[Prediction>.5]="1"
TestPrediction[Prediction<=.5]="0"

mean(TestPrediction==TestData$DEATH_EVENT)
```

```
## [1] 0.85

table(TestData$DEATH_EVENT,TestPrediction)

##    TestPrediction
##      0  1
##   0 31  4
##   1  5 20

##########################################
```

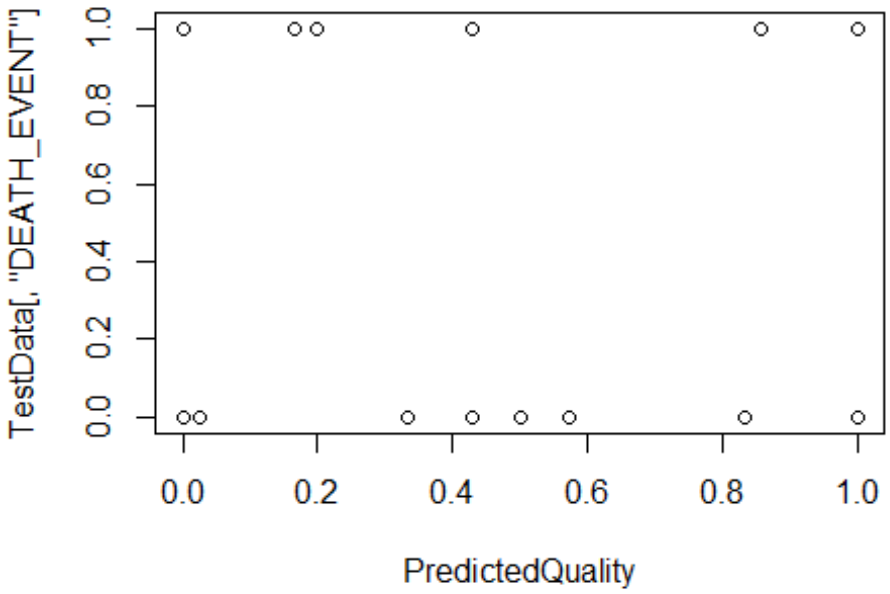
Tree Model and Random Forest

```
# Included Libraries
library(tree)
library(ISLR)
library(tidyr)
library(tidyverse)
library(randomForest)
library(leaps)
library(psych)
library(ggplot2)

# import data set and add color variable
AllData = read.csv('AllData.csv',sep=',')
names(AllData)[1] <- 'age'

# Split data into train and test (80% train 20% test)
set.seed(29)
TrainIndex = sample(1:nrow(AllData), (max(lengths(AllData))*0.8))
TrainData = AllData[TrainIndex,]
TestData = AllData[-TrainIndex,]

# create tree
HeartTree = tree(DEATH_EVENT~.,TrainData)
summary(HeartTree)

##
## Regression tree:
## tree(formula = DEATH_EVENT ~ ., data = TrainData)
## Variables actually used in tree construction:
## [1] "time"                    "platelets"
## [3] "creatinine_phosphokinase" "serum_creatinine"
## [5] "age"                     "ejection_fraction"
## [7] "sex"
## Number of terminal nodes:  13
## Residual mean deviance:  0.08008 = 18.1 / 226
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.85710 -0.02419 -0.02419  0.00000  0.00000  0.97580
```

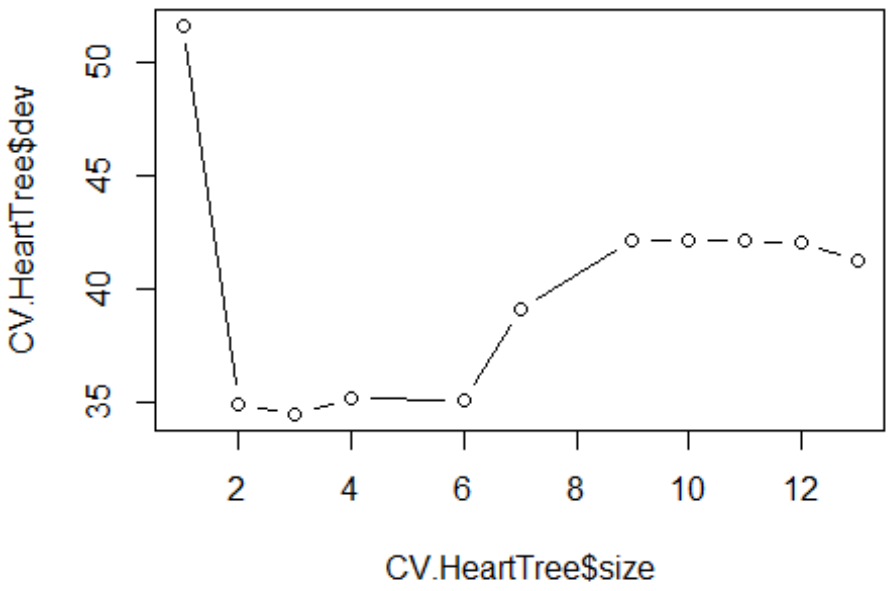
```
# Plot tree
plot(HeartTree)
text(HeartTree,pretty=0)
```



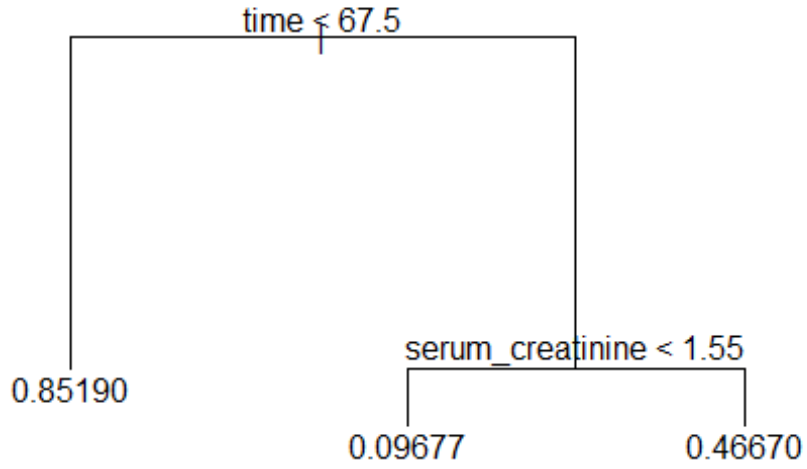

```
# Calculate MSE (mean squared error)
PredictedQuality = predict(HeartTree,newdata=TestData)
plot(PredictedQuality,TestData[,'DEATH_EVENT'])
```



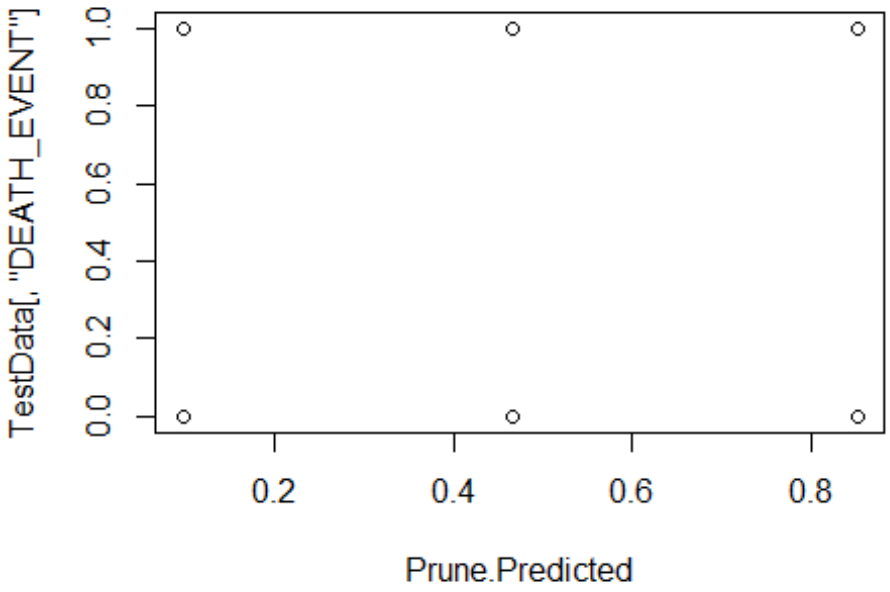

```
differenceTestData = PredictedQuality-TestData[,'DEATH_EVENT']
MSE1 = mean((PredictedQuality-TestData[,'DEATH_EVENT'])^2)


# Cross Validation to determine optimal level of tree complexity
CV.HeartTree=cv.tree(HeartTree)
summary(CV.HeartTree)

##        Length Class  Mode
## size   11     -none- numeric
## dev    11     -none- numeric
## k      11     -none- numeric
## method  1     -none- character

plot(CV.HeartTree$size,CV.HeartTree$dev,type='b')
```



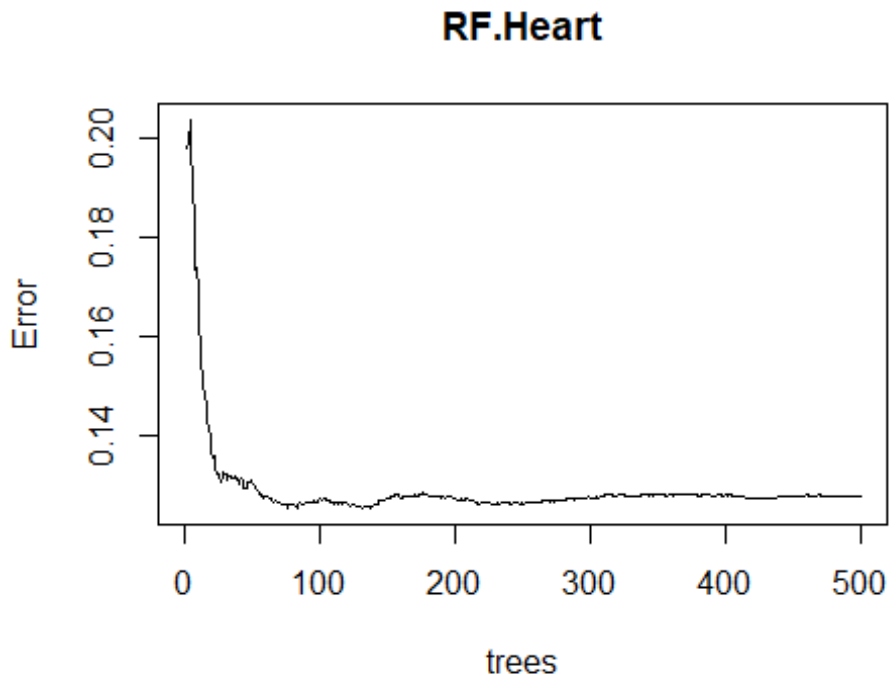

```
# Prune tree based on cross validation
Prune.Heart = prune.tree(HeartTree,best=3)
plot(Prune.Heart)
text(Prune.Heart,pretty=0)
```




```
Prune.Predicted = predict(Prune.Heart,newdata=TestData)
plot(Prune.Predicted,TestData['DEATH_EVENT'])
```



```
Prune.MSE = mean((Prune.Predicted-TestData[,'DEATH_EVENT'])^2)

# Random Forests fit
set.seed(1)
RF.Heart=randomForest(DEATH_EVENT~.,data=AllData,subset=TrainIndex,mtry=6,imp
ortance=TRUE)
RF.Predicted = predict(RF.Heart,newdata=TestData)
MSE.RF = mean((RF.Predicted-TestData[,'DEATH_EVENT'])^2) #test set MSE
plot(RF.Heart)
```




```
# use importance function to determine important variables
#variable importance
importance(RF.Heart)
```

```
##                             %IncMSE IncNodePurity
## age                       7.2946688     3.7662025
## anaemia                  -0.3661244     0.4515941
## creatinine_phosphokinase  2.7213090     4.2113344
## diabetes                 -1.0765932     0.3221530
## ejection_fraction        15.2454365     4.8347420
## high_blood_pressure      -3.3928571     0.3252903
## platelets                 0.2156617     3.3619677
## serum_creatinine         15.6951749     6.7638185
## serum_sodium              4.0357812     2.9360344
## sex                      -1.1446478     0.3175625
## smoking                  -1.7978461     0.4086603
## time                     54.4064559    20.0497866
```

```
varImpPlot(RF.Heart)
```