Heart Failure Analysis

Valentina Gonzalez Bohorquez

5/10/2021

Project Aim

To choose the best model to accurately predict mortality caused by Heart Failure. The models we created were logistic regression, K-Nearest Neighbors, and Random Forest.

I. Introduction

Cardiovascular diseases are the number 1 cause of death, accounting for 31% of deaths globally. Most cardiovascular diseases can be prevented through lifestyle improvements, such as proper diet, exercise, as well as limited tobacco and alcohol intake. Machine Learning can play an essential role in predicting presence, or absence of cardiovascular diseases and more. Such information, if predicted well in advance, can provide important insights to doctors who can then adapt their diagnosis and treatment per patient basis.

II. Dataset

This binary classification dataset contains 12 features that can be used to predict mortality by heart failure.

Categorical Variables	Numeric Variables	Output
anaemia - Decrease of red blood cells/hemoglobin (0:False/1:True)	***age*** - Age of patient (years)	***DEATH_EVENT*** - If patient died during follow up period (0:False/1:True)
diabetes - If patient has diabetes (0:False/1:True)	***creatinine_phosphokinase*** - Level of CPK enzyme in blood (mcg/L)	
high_blood_pressure - If patient has hypertension (0:False/1:True)	***ejection_fraction*** - percentage of blood leaving (%)	
sex - woman or man (0:woman/1:man)	***platelets*** - Platelets in blood (kiloplatelets/mL)	
smoking - If patient smokes or not (0:False/1:True)	***serum_creatinine*** - Level of creatinine in blood (mg/dL)	
	serum_sodium - Level of sodium in blood (mEq/L)	
	time - Follow-up period (Days)	

III. Data Analytic Strategy

Load the libraries.

```r
library("caret")
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
library("class")
library("faraway")
```

```
##
## Attaching package: 'faraway'
```

```
## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
library("InformationValue")
```

```
##
## Attaching package: 'InformationValue'
```

```
## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, sensitivity, specificity
```

```r
library("leaps")
library("randomForest")
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library("caTools")
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.1.0     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::combine()       masks randomForest::combine()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::lift()          masks caret::lift()
## x randomForest::margin() masks ggplot2::margin()
```

```
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

Load the dataset.

```
heart_disease <- read.csv("~/Documents/Second Semester/Stat Mod & Comp/Final Project/heart_failure_clini
```

Use the summary function for viewing metrics related to our data.

```
summary(heart_disease)
```

```
##       age           anaemia       creatinine_phosphokinase    diabetes
##  Min.   :40.00   Min.   :0.0000   Min.   :  23.0           Min.   :0.0000
##  1st Qu.:51.00   1st Qu.:0.0000   1st Qu.: 116.5           1st Qu.:0.0000
##  Median :60.00   Median :0.0000   Median : 250.0           Median :0.0000
##  Mean   :60.83   Mean   :0.4314   Mean   : 581.8           Mean   :0.4181
##  3rd Qu.:70.00   3rd Qu.:1.0000   3rd Qu.: 582.0           3rd Qu.:1.0000
##  Max.   :95.00   Max.   :1.0000   Max.   :7861.0           Max.   :1.0000
##  ejection_fraction high_blood_pressure   platelets      serum_creatinine
##  Min.   :14.00     Min.   :0.0000      Min.   : 25100   Min.   :0.500
##  1st Qu.:30.00     1st Qu.:0.0000      1st Qu.:212500   1st Qu.:0.900
##  Median :38.00     Median :0.0000      Median :262000   Median :1.100
##  Mean   :38.08     Mean   :0.3512      Mean   :263358   Mean   :1.394
##  3rd Qu.:45.00     3rd Qu.:1.0000      3rd Qu.:303500   3rd Qu.:1.400
##  Max.   :80.00     Max.   :1.0000      Max.   :850000   Max.   :9.400
##   serum_sodium        sex            smoking            time
##  Min.   :113.0   Min.   :0.0000   Min.   :0.0000   Min.   :  4.0
##  1st Qu.:134.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 73.0
##  Median :137.0   Median :1.0000   Median :0.0000   Median :115.0
##  Mean   :136.6   Mean   :0.6488   Mean   :0.3211   Mean   :130.3
##  3rd Qu.:140.0   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:203.0
##  Max.   :148.0   Max.   :1.0000   Max.   :1.0000   Max.   :285.0
##   DEATH_EVENT
```

```
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3211
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

Upon closer analysis, an issue with the column types is detected. In R, a categorical variable is a factor. However, summary(heart_disease) revealed that "sex" was incorrectly treated as a number, as opposed to a factor (0 = Female, 1 = Male).

Use the sapply function to view each column type.

```
sapply(heart_disease, class)
```

```
##                      age                  anaemia creatinine_phosphokinase
##                "numeric"                "integer"                "integer"
##                 diabetes        ejection_fraction      high_blood_pressure
##                "integer"                "integer"                "integer"
##                platelets         serum_creatinine             serum_sodium
##                "numeric"                "numeric"                "integer"
##                      sex                  smoking                     time
##                "integer"                "integer"                "integer"
##              DEATH_EVENT
##                "integer"
```

Use the transform function to change the in-built type of each feature.

```
heart_disease <- transform(
  heart_disease,
  age=as.integer(age),
  sex=as.factor(sex),
  diabetes=as.factor(diabetes),
  anaemia=as.factor(anaemia),
  high_blood_pressure=as.factor(high_blood_pressure),
  smoking=as.factor(smoking),
  DEATH_EVENT=as.factor(DEATH_EVENT)
)
```

View the corrected column types.

```
sapply(heart_disease, class)
```

```
##                      age                  anaemia creatinine_phosphokinase
##                "integer"                 "factor"                "integer"
##                 diabetes        ejection_fraction      high_blood_pressure
##                 "factor"                "integer"                 "factor"
```

```
##                platelets         serum_creatinine             serum_sodium
##                "numeric"               "numeric"                "integer"
##                      sex                 smoking                     time
##                 "factor"                "factor"                "integer"
##              DEATH_EVENT
##                "factor"
```

```
summary(heart_disease)
```

```
##       age        anaemia creatinine_phosphokinase diabetes ejection_fraction
##  Min.   :40.00   0:170   Min.   :  23.0           0:174    Min.   :14.00
##  1st Qu.:51.00   1:129   1st Qu.: 116.5           1:125    1st Qu.:30.00
##  Median :60.00           Median : 250.0                     Median :38.00
##  Mean   :60.83           Mean   : 581.8                     Mean   :38.08
##  3rd Qu.:70.00           3rd Qu.: 582.0                     3rd Qu.:45.00
##  Max.   :95.00           Max.   :7861.0                     Max.   :80.00
##  high_blood_pressure   platelets      serum_creatinine  serum_sodium   sex
##  0:194               Min.   : 25100   Min.   :0.500    Min.   :113.0   0:105
##  1:105               1st Qu.:212500   1st Qu.:0.900    1st Qu.:134.0   1:194
##                      Median :262000   Median :1.100    Median :137.0
##                      Mean   :263358   Mean   :1.394    Mean   :136.6
##                      3rd Qu.:303500   3rd Qu.:1.400    3rd Qu.:140.0
##                      Max.   :850000   Max.   :9.400    Max.   :148.0
##  smoking      time       DEATH_EVENT
##  0:203   Min.   :  4.0   0:203
##  1: 96   1st Qu.: 73.0   1: 96
##          Median :115.0
##          Mean   :130.3
##          3rd Qu.:203.0
##          Max.   :285.0
```

Split the data into train and test.

```
trainIndex <- createDataPartition(heart_disease$DEATH_EVENT, p=0.70, list=FALSE, times=1)

heart_disease_train <- heart_disease[trainIndex,]
heart_disease_test <- heart_disease[-trainIndex,]
```

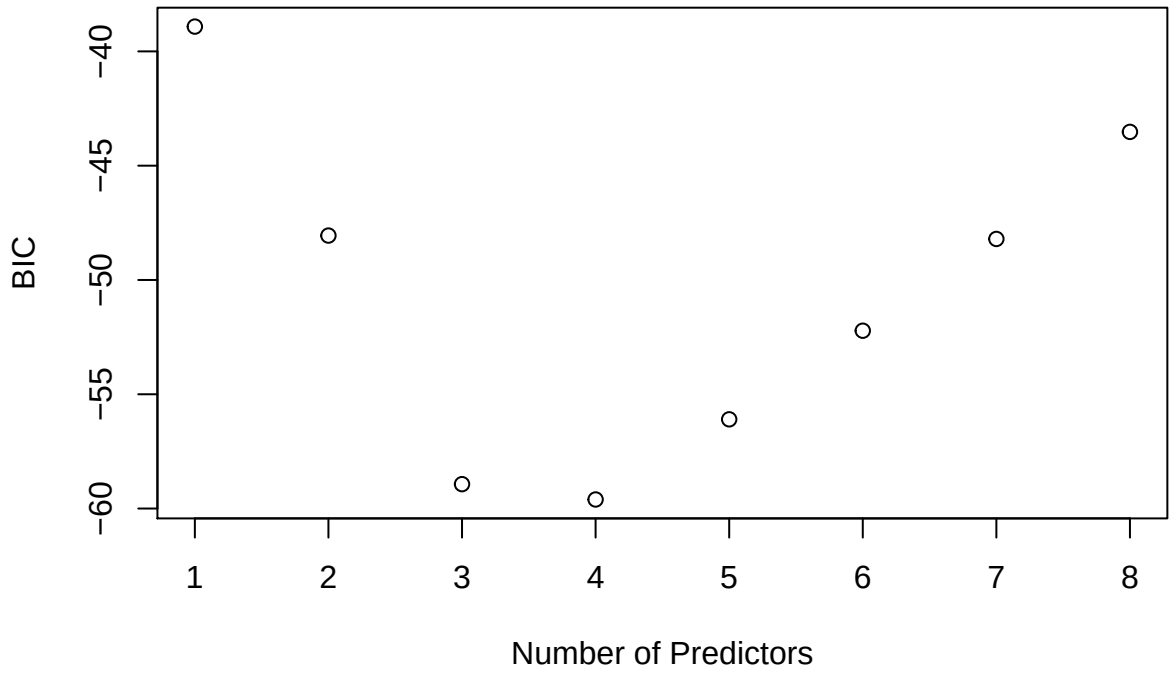
BIC Model Selection

```
heart_disease_train_BIC <- regsubsets(DEATH_EVENT ~ ., data=heart_disease_train)
heart_disease_train_BIC_sum <- summary(heart_disease_train_BIC)
heart_disease_train_BIC_sum$which
```

```
##   (Intercept)   age anaemia1 creatinine_phosphokinase diabetes1
## 1        TRUE FALSE    FALSE                    FALSE     FALSE
## 2        TRUE FALSE    FALSE                    FALSE     FALSE
## 3        TRUE FALSE    FALSE                    FALSE     FALSE
## 4        TRUE  TRUE    FALSE                    FALSE     FALSE
```

```
## 5        TRUE  TRUE    FALSE                    FALSE     FALSE
## 6        TRUE  TRUE    FALSE                    FALSE     FALSE
## 7        TRUE  TRUE    FALSE                    FALSE     FALSE
## 8        TRUE  TRUE    FALSE                     TRUE     FALSE
##   ejection_fraction high_blood_pressure1 platelets serum_creatinine
## 1             FALSE                FALSE     FALSE            FALSE
## 2             FALSE                FALSE     FALSE             TRUE
## 3              TRUE                FALSE     FALSE             TRUE
## 4              TRUE                FALSE     FALSE             TRUE
## 5              TRUE                FALSE     FALSE             TRUE
## 6              TRUE                FALSE      TRUE             TRUE
## 7              TRUE                FALSE      TRUE             TRUE
## 8              TRUE                FALSE      TRUE             TRUE
##   serum_sodium  sex1 smoking1 time
## 1        FALSE FALSE    FALSE TRUE
## 2        FALSE FALSE    FALSE TRUE
## 3        FALSE FALSE    FALSE TRUE
## 4        FALSE FALSE    FALSE TRUE
## 5         TRUE FALSE    FALSE TRUE
## 6         TRUE FALSE    FALSE TRUE
## 7         TRUE  TRUE    FALSE TRUE
## 8         TRUE  TRUE    FALSE TRUE
```

```
plot(heart_disease_train_BIC_sum$bic, ylab="BIC", xlab="Number of Predictors")
```



Through the Bayesian Information Criterion (BIC), we choose the parameters to create the models. BIC can measure the efficiency of the parameterized model in terms of predicting the data. Limiting the model to

statistically sufficient variables can improve prediction accuracy. Additionally, it is computationally cheaper in the end. For this dataset, the minimum BIC occurs when there are 3 predictors: ejection_fraction, serum_creatinine, and time.

IV. Predictive Models and Results

Logistic Regression Model

```
lmod_heart_disease <- glm(DEATH_EVENT ~ ejection_fraction + serum_creatinine + time,
                          family=binomial, heart_disease_train)
summary(lmod_heart_disease)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ ejection_fraction + serum_creatinine +
##     time, family = binomial, data = heart_disease_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0406  -0.7771  -0.3436   0.7409   2.5543
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.584684   0.758781   3.406 0.000658 ***
## ejection_fraction -0.062506   0.017166  -3.641 0.000271 ***
## serum_creatinine   0.671535   0.178746   3.757 0.000172 ***
## time              -0.017939   0.003062  -5.858 4.69e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 265.26  on 210  degrees of freedom
## Residual deviance: 186.16  on 207  degrees of freedom
## AIC: 194.16
##
## Number of Fisher Scoring iterations: 5
```

Created a logistic regression model with the parameters ejection_fraction, serum_creatinine, and time. The 'summary' function revealed a null deviance of 256.56 on 209 degrees of freedom, a residual deviance of 161.96 on 206 degrees of freedom, and a AIC of 169.96.

Prediction

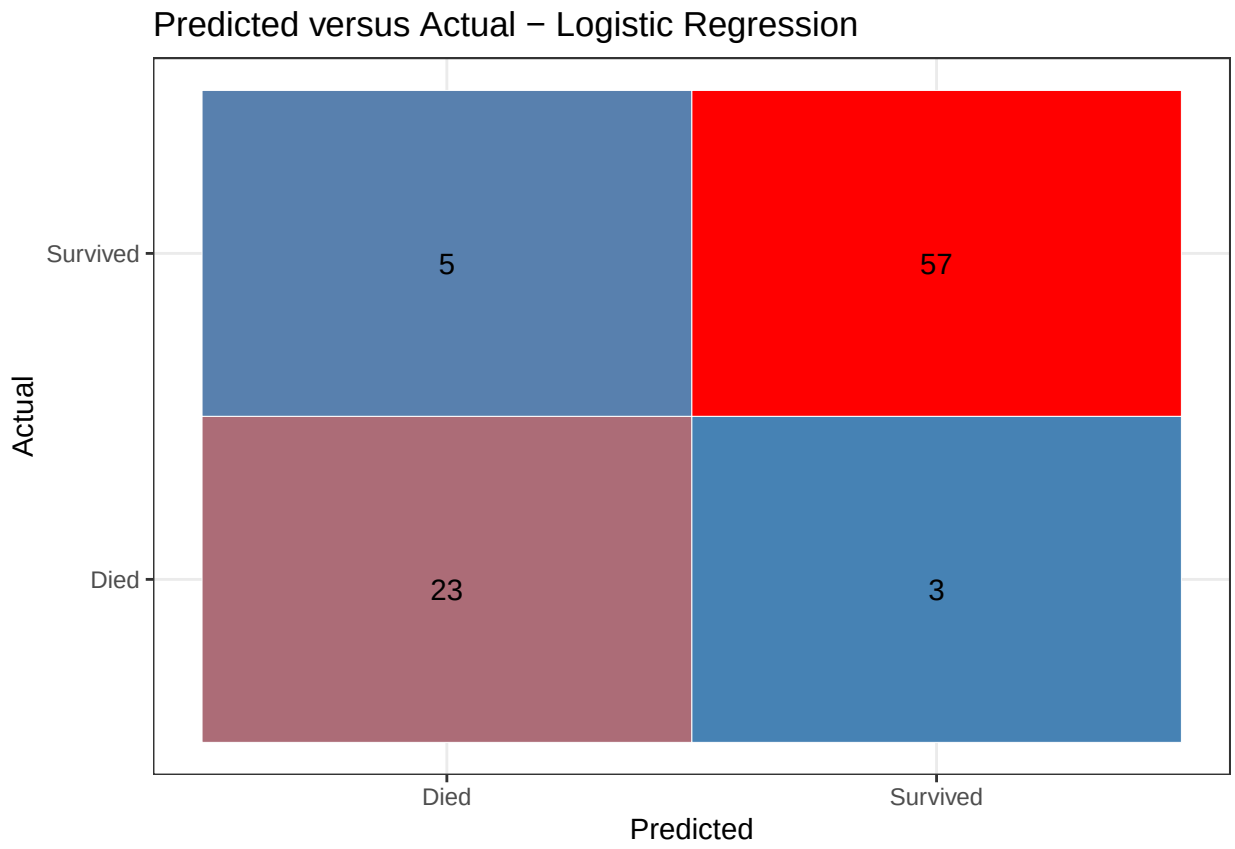
```
lmod_heart_disease_prob <- predict(lmod_heart_disease, heart_disease_test, type="response")
optCutoff <- optimalCutoff(heart_disease_test$DEATH_EVENT,lmod_heart_disease_prob)

heart_disease_test_pred_lmod <- heart_disease_test %>%
```

```
  mutate(predict=1*(lmod_heart_disease_prob > optCutoff)) %>%
  mutate(accurate=1*(predict==DEATH_EVENT))
lmod_acc <- sum(heart_disease_test_pred_lmod$accurate)/nrow(heart_disease_test_pred_lmod)
```

The 'optimalCutoff' determines the optimal decision threshold for the logistic model. In this case, it was 0.3696721. Moreover, we measured the accuracy of the model by adding the taking the heart failure test prediction accuracy divided by the number of rows. The accuracy of the Logistic model was about 83.15%.

Creating the Logistic confusion matrix.

```
confusion_matrix_lmod <- as.data.frame(table(heart_disease_test_pred_lmod$DEATH_EVENT,heart_disease_tes
confusion_matrix_lmod$Var1 <- as.character(confusion_matrix_lmod$Var1)
confusion_matrix_lmod$Var2 <- as.character(confusion_matrix_lmod$Var2)
confusion_matrix_lmod$Var1[confusion_matrix_lmod$Var1 == 0] <- "Survived"
confusion_matrix_lmod$Var1[confusion_matrix_lmod$Var1 == 1] <- "Died"
confusion_matrix_lmod$Var2[confusion_matrix_lmod$Var2 == 0] <- "Survived"
confusion_matrix_lmod$Var2[confusion_matrix_lmod$Var2 == 1] <- "Died"
```



K-Nearest Neighbors Model

```
heart_disease_train_filtered_x <- heart_disease_train %>%
  select(ejection_fraction,serum_creatinine,time)
heart_disease_train_filtered_y <- heart_disease_train$DEATH_EVENT
heart_disease_test_filtered_x <- heart_disease_test %>%
  select(ejection_fraction,serum_creatinine,time)
heart_disease_test_filtered_y <- heart_disease_test$DEATH_EVENT
```

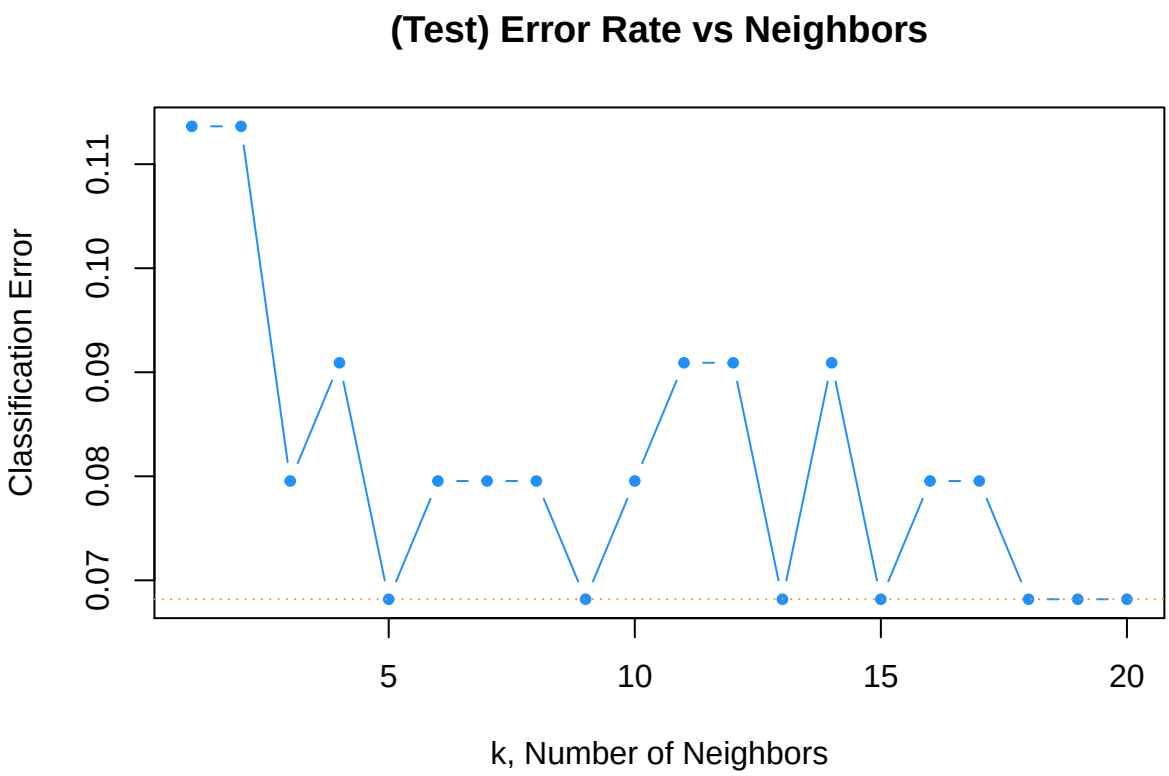
Selecting the same parameters as the logistic model (ejection_fraction, serum_creatinine, and time), filtered the train and test data.

Calculating the error.

```
calc_error <- function(actual, predicted){
  error <- mean(actual != predicted)
  return(error)
}
```

Determining k and finding the minimum error.

```
ks = 1:20
errors <- rep(x=0, times=length(ks))

for (i in seq_along(ks)) {
  prediction <- knn(train=heart_disease_train_filtered_x,
                    test=heart_disease_test_filtered_x,
                    cl=heart_disease_train_filtered_y,
                    k=ks[i])
  errors[i] <- calc_error(heart_disease_test_filtered_y, prediction)
}

plot(errors, type = "b", col = "dodgerblue", cex = 1, pch = 20,
     xlab = "k, Number of Neighbors", ylab = "Classification Error",
     main = "(Test) Error Rate vs Neighbors")
abline(h = min(errors), col = "darkorange", lty = 3)
```




```
min(errors)
```

```
## [1] 0.06818182
```

```
k <- which(errors == min(errors))
k
```

```
## [1]  5  9 13 15 18 19 20
```

At first, we determined the k value through the square root of n method, which gave k ~ 17. However, this method gave a lower prediction accuracy (around 70%). Therefore, we decided to choose k based on a train/test approach, fitting KNN models for k between 1 and 20, and choosing the k with the minimum classification error rate, with k = 6 and error rate of about 13%.

Creating the KNN model.

```
heart_disease_knn <- knn(train=heart_disease_train_filtered_x,
                         test=heart_disease_test_filtered_x,
                         cl=heart_disease_train_filtered_y,
                         k=k)
```

```
## Warning in if (ntr < k) {: the condition has length > 1 and only the first
## element will be used
```

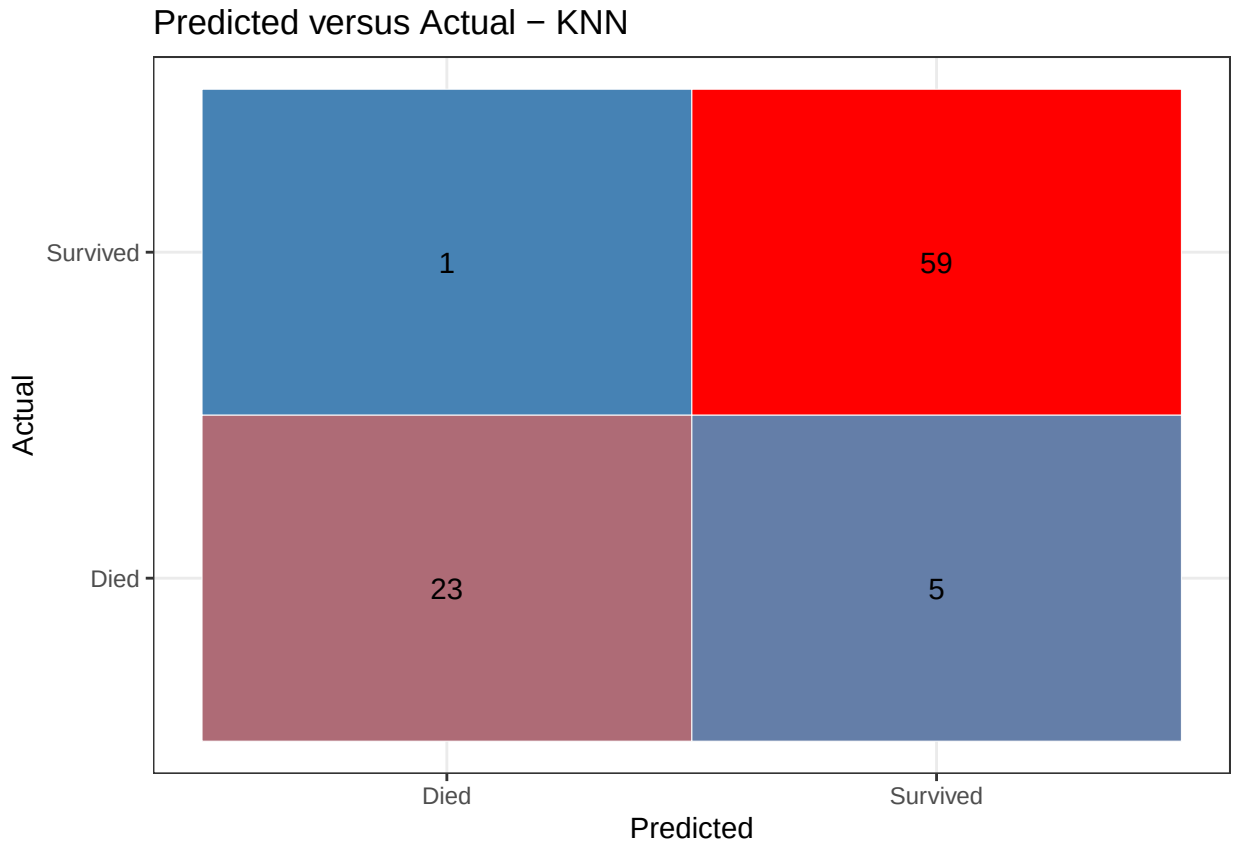
```
## Warning in if (k < 1) stop(gettextf("k = %d must be at least 1", k), domain =
## NA): the condition has length > 1 and only the first element will be used
```

```
knn_acc <- 1-calc_error(heart_disease_test_filtered_y,heart_disease_knn)
```

Used the 'knn' function to create the KNN model. The accuracy of the KNN model was about 86.52%.

Creating the KNN confusion matrix.

```
confusion_matrix_knn <- as.data.frame(table(heart_disease_test_filtered_y,heart_disease_knn))
confusion_matrix_knn$heart_disease_test_filtered_y <- as.character(confusion_matrix_knn$heart_disease_t
confusion_matrix_knn$heart_disease_knn <- as.character(confusion_matrix_knn$heart_disease_knn)
confusion_matrix_knn$heart_disease_test_filtered_y[confusion_matrix_knn$heart_disease_test_filtered_y ==
confusion_matrix_knn$heart_disease_test_filtered_y[confusion_matrix_knn$heart_disease_test_filtered_y ==
confusion_matrix_knn$heart_disease_knn[confusion_matrix_knn$heart_disease_knn == 0] <- "Survived"
confusion_matrix_knn$heart_disease_knn[confusion_matrix_knn$heart_disease_knn == 1] <- "Died"
```



Random Forest Model

Creating the Random Forest model.

```
random_forest_model <- randomForest(formula=as.factor(DEATH_EVENT) ~ ejection_fraction + serum_creatini
```

Utilized the 'randomForest' function to create the Random Forest Model.

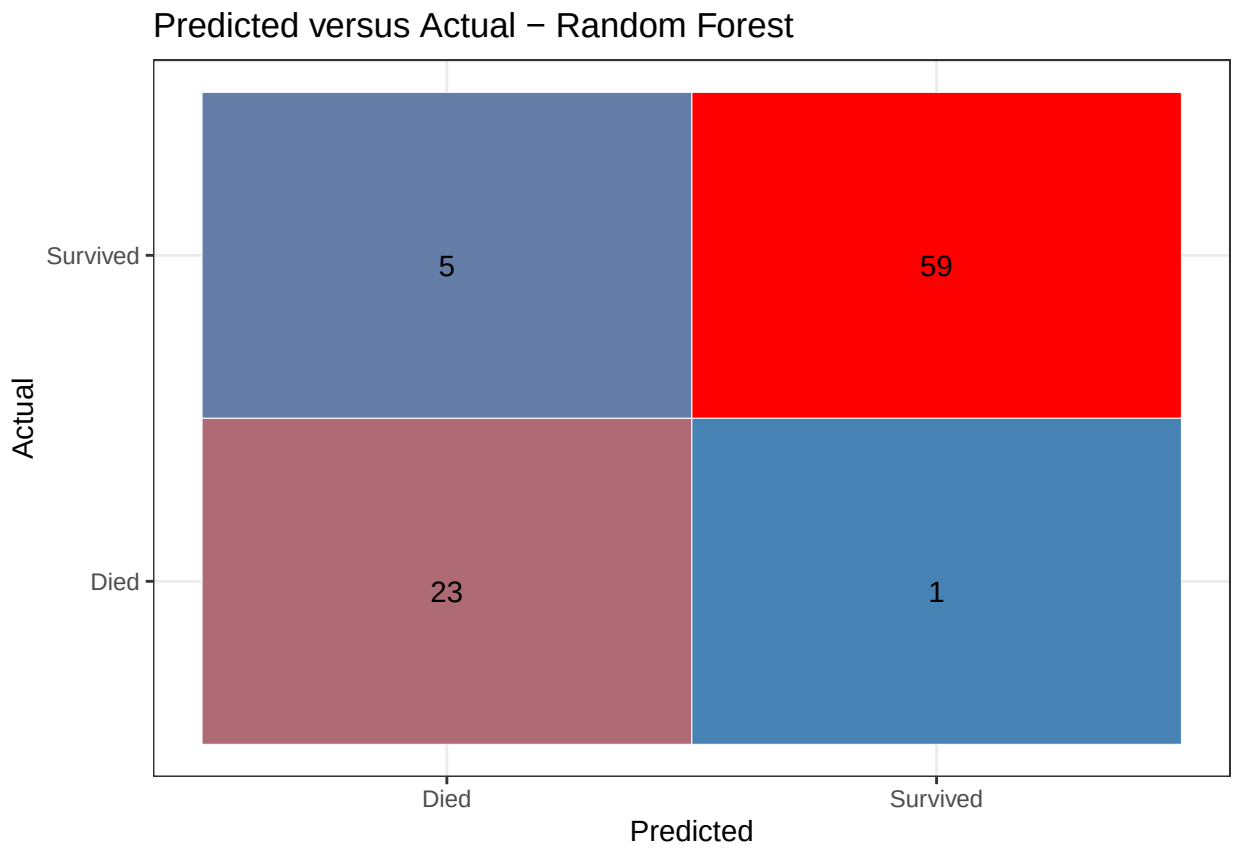
Prediction

```
heart_disease_test_pred_rf <- heart_disease_test %>%
  mutate(pred = predict(random_forest_model, heart_disease_test))%>%
  mutate(accurate=1*(pred==DEATH_EVENT))
rf_acc <- sum(heart_disease_test_pred_rf$accurate)/nrow(heart_disease_test_pred_rf)
```

The accuracy of the Random Forest model was about 87.64%.

Creating the confusion matrix.

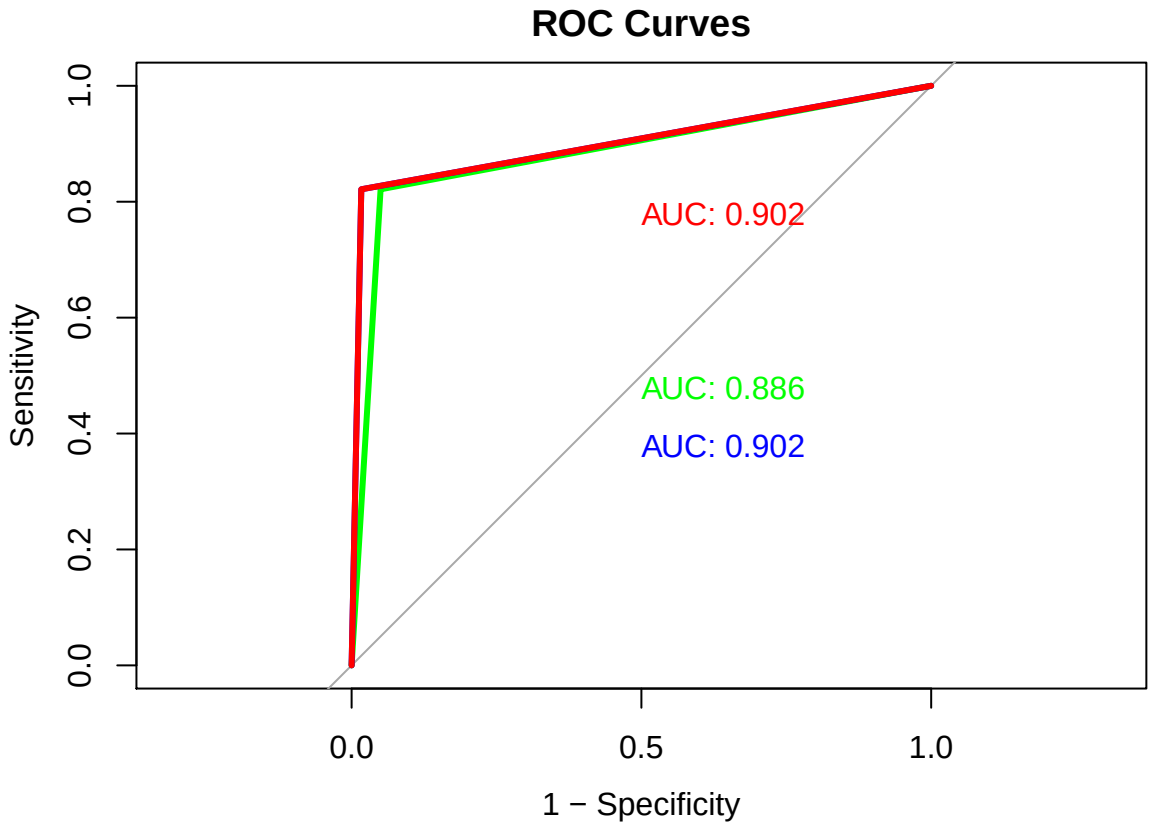
```
confusion_matrix_rf <- as.data.frame(table(heart_disease_test_pred_rf$DEATH_EVENT,heart_disease_test_pr
confusion_matrix_rf$Var1 <- as.character(confusion_matrix_rf$Var1)
confusion_matrix_rf$Var2 <- as.character(confusion_matrix_rf$Var2)
confusion_matrix_rf$Var1[confusion_matrix_rf$Var1 == 0] <- "Survived"
confusion_matrix_rf$Var1[confusion_matrix_rf$Var1 == 1] <- "Died"
confusion_matrix_rf$Var2[confusion_matrix_rf$Var2 == 0] <- "Survived"
confusion_matrix_rf$Var2[confusion_matrix_rf$Var2 == 1] <- "Died"
```



V. Conclusions

ROC Curbs

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



The Receiver Operating Characteristic (ROC) curve plot shows the performance of classification models at all classification thresholds. This curve plots two parameters: True Positive (Y-axis) and False Positive (X-axis) Rate. The ROC curves is useful to visualize and compare the performance of classifier methods. This ROC curve displays that the Random Forest model, with an accuracy percentage of 84.6%, performed the best in comparison to the logistic regression and KNN model. In conclusion, the Random Forest model is the best model to accurately predict mortality caused by Heart Failure.