**Heart Failure Death Prediction**

**Brian Lee, Jamie Boyd, Kenneth VanGemert, Gary Williams**

## 1. Problem Statement and Background

The accessible data was from the website: "https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data". The actual collection of the data was from Faisalabad Institute of Cardiology and at the Allied Hospital in Faisalabad (Punjab, Pakistan), during April to December 2015. According to "About this dataset" from the Kaggle URL provided above, "Cardiovascular disease is a number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide." Furthermore, "heart failure is a common event caused by cardiovascular disease". Since cardiovascular disease is the main factor causing heart failure, which is a critical cause of death world-wide, many people who had or are prone to heart failure would care deeply about this problem. If we can generate a model that can accurately predict the threshold of feature values that is highly correlated with death or survived among people who had heart failure, then it can help doctors such as cardiologists, nurses, surgeons, and other medical professionals to identify, diagnose, and treat patients who suffered from heart failure and save their lives.

The informal success measures will include the predictive accuracy of the model from the cross-validated data to ensure a robust model and generalize well to unseen data, and provide clear interpretation and explanation of the predictions made by the model.

## 2. Data and Exploratory Analysis

The dataset "heart_failure_clinical_records_dataset.csv" has 13 features, such as "age", "anaemia", "creatinine_phosphokinase", "diabetes", "high_blood_pressure", "platelets", "serum_sodium", "sex", "smoking", "time", "ejection_fraction", "serum_creatinine", and "DEATH_EVENT". Each instance is a patient, and each feature value was recorded from 299 patients from age 40 to 95, of which the "age" feature is a numerical feature. There were 194 male patients and 105 female patients; the "sex" feature is a categorical feature (boolean): male = 1 and female = 0. The "time" feature is a numerical feature, indicating the length of a person's health monitored over time; it is also known as the "follow-up period". The "smoking" feature is a categorical feature (boolean), indicating 1 as a smoker and 0 as a non-smoker. The features "diabetes", "high_blood_pressure", and "anaemia" are categorical features (boolean), indicating 0 as not having them and 1 as having them. The "creatine_phosphokinase" feature is a numerical feature that indicates the level of creatine phosphokinase enzyme (protein that helps to elicit chemical changes in your body found in your heart, brain, and skeletal muscles) in the blood in the units of mcg/L. The "ejection_fraction" feature is a numerical feature, indicating the percentage of blood leaving the heart at each contraction. The "serum_sodium" feature is a numerical feature, indicating the level of serum sodium in the blood in the units of mEq/L. The "serum_creatinine" feature is a numerical feature, indicating the level of serum creatinine in the blood (bad kidney means high level of serum creatinine in the blood) in the units of mg/dL. Finally, the "DEATH_EVENT" feature is a categorical feature (boolean), indicating 1 as death has occurred and 0 as no death has occurred or alive for a patient that had a heart failure.

Using the “readr” library, we loaded the dataset into a dataframe called “heart_failure”. We used the R code below to see the characteristics of the dataframe with the ‘summary()’ command, and check if there were any ‘NA’ values. The dataset did not contain any ‘NA’ values, so we did not have to replace any ‘NA’ values.

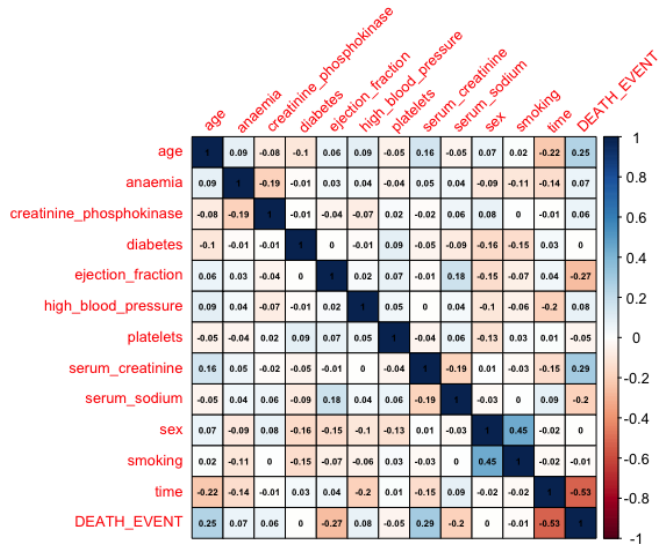
```
# load data
library(readr)
heart_failure <- read_csv("heart_failure_clinical_records_dataset.csv")
# characteristics of data frame
summary(heart_failure)
# verify NA values
na_counts <- colSums(is.na(heart_failure))
print(na_counts)
```

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

```
              age                anaemia creatinine_phosphokinase                 diabetes
                0                      0                         0                        0
ejection_fraction      high_blood_pressure                 platelets         serum_creatinine
                0                      0                         0                        0
     serum_sodium                    sex                    smoking                     time
                0                      0                         0                        0
      DEATH_EVENT
                0
```

Next, we first made a correlation matrix to find the most correlated attributes with “DEATH_EVENT”. We found correlation values for “age” = 0.25, “ejection_fraction” = -0.27, “serum_creatinine” = 0.29, and “time” = -0.53 with respect to “DEATH_EVENT”.

```
correlation_matrix <- cor(heart_failure)
print(correlation_matrix)
installed.packages("corrplot")
library(corrplot)
corrplot(correlation_matrix, method = "color", type = "full", tl.srt = 45,
         addCoef.col = "black", number.cex = 0.5, tl.cex = 0.8,
         outline = TRUE)
```

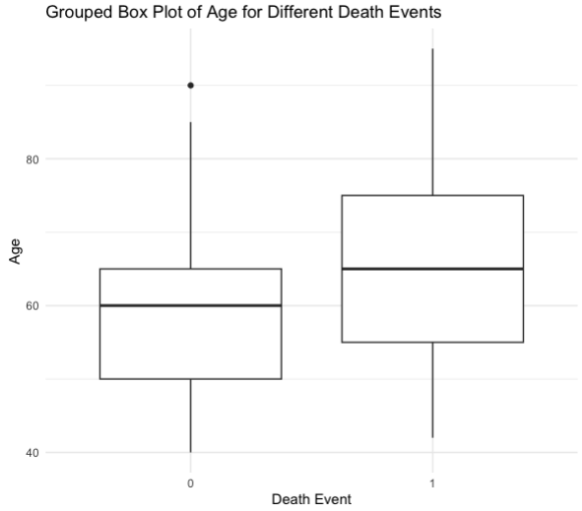



Based on the result, we will continue exploring these 4 features, because our model needs to accurately predict feature values that cause a “DEATH_EVENT”.

Next, we loaded the library “ggplot2” to create a plot visualization of each feature with respect to “DEATH_EVENT” to explore the data and features.

For “age”, we used R code to look at the summary and created a grouped box plot.

```
summary(heart_failure$age)
summary(heart_failure$age[heart_failure$DEATH_EVENT == 0])
summary(heart_failure$age[heart_failure$DEATH_EVENT == 1])
# Grouped box plot of "age" for different death events
library(ggplot2)
heart_failure$DEATH_EVENT <- as.factor(heart_failure$DEATH_EVENT)
ggplot(heart_failure, aes(x = DEATH_EVENT, y = age)) +
  geom_boxplot() +
  labs(title = "Grouped Box Plot of Age for Different Death Events",
       x = "Death Event",
       y = "Age") +
  theme_minimal()
```

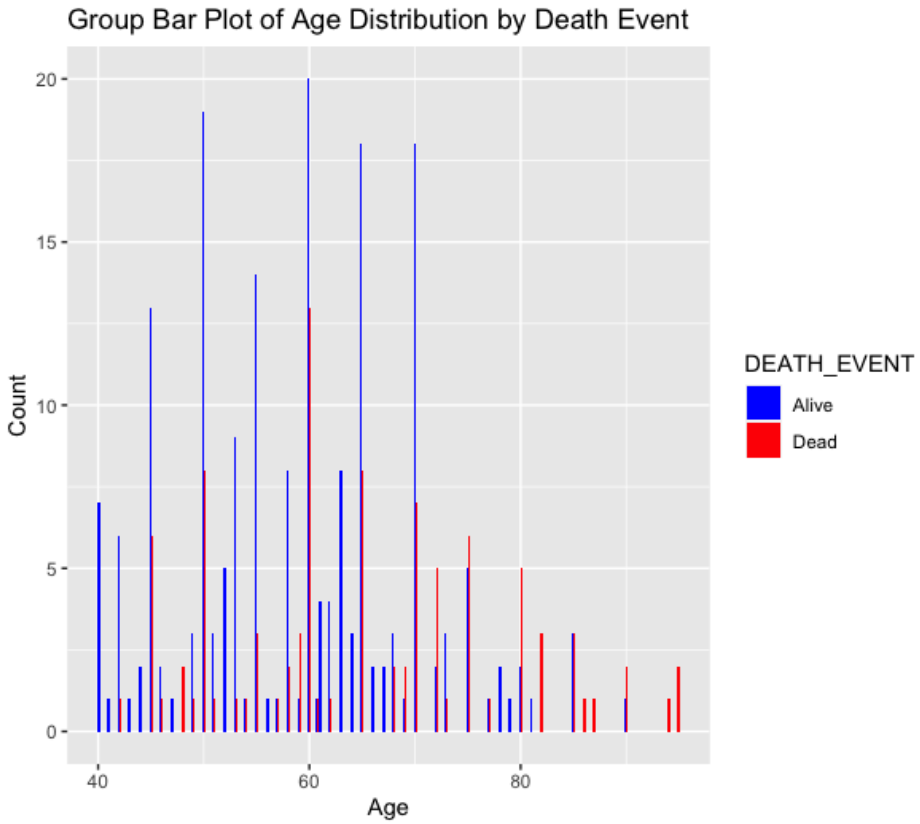



```
summary(heart_failure$age)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
40.00   51.00   60.00   60.83   70.00   95.00
summary(heart_failure$age[heart_failure$DEATH_EVENT == 0])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
40.00   50.00   60.00   58.76   65.00   90.00
summary(heart_failure$age[heart_failure$DEATH_EVENT == 1])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
42.00   55.00   65.00   65.22   75.00   95.00
```

Based on the grouped box plot and summary, alive patients indicate a lower median age compared to dead patients, suggesting that individuals who experienced the death event tend to be older. There is one outlier for an alive patient of age 90. However, we decided to keep the outlier since it does not seem to significantly affect the accuracy of the predictions made by the models we plan to create. However, we will keep an eye on it as we build our models.

In addition, we created a grouped bar plot for “age” with respect to “DEATH_EVENT”.

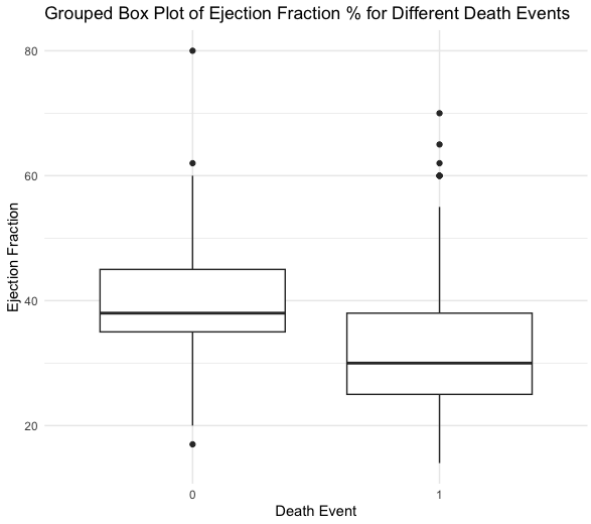
```
# Grouped bar plot of "age" for different death events
heart_failure$DEATH_EVENT <- factor(heart_failure$DEATH_EVENT, labels = c("Alive", "Dead"))
heart_failure |>
  ggplot(aes(x = age)) +
  geom_bar(aes(fill = DEATH_EVENT), position = "dodge") +
  scale_fill_manual(values = c("blue", "red")) +
  labs(title = "Group Bar Plot of Age Distribution by Death Event", x = "Age", y = "Count")
```




Based on the grouped bar plot, there is a gap in ages; the lowest age is 40. We typically see deaths in ages 60 and above with age 60 having the most deaths. The graph seems to be right skewed with a tail.

For “ejection_fraction”, we used R code to look at the summary and created a grouped box plot.

```r
summary(heart_failure$ejection_fraction)
summary(heart_failure$ejection_fraction[heart_failure$DEATH_EVENT == 0])
summary(heart_failure$ejection_fraction[heart_failure$DEATH_EVENT == 1])
# Grouped box plot of "ejection_fraction" for different death events
library(ggplot2)
heart_failure$DEATH_EVENT <- as.factor(heart_failure$DEATH_EVENT)
ggplot(heart_failure, aes(x = DEATH_EVENT, y = ejection_fraction)) +
  geom_boxplot() +
  labs(title = "Grouped Box Plot of Ejection Fraction % for Different Death Events",
       x = "Death Event",
       y = "Ejection Fraction") +
  theme_minimal()
```

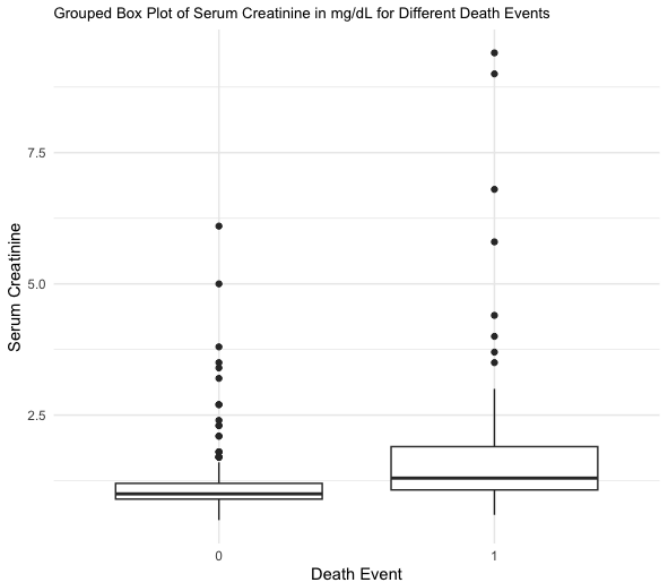



```r
summary(heart_failure$ejection_fraction)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
14.00   30.00   38.00   38.08   45.00   80.00
summary(heart_failure$ejection_fraction[heart_failure$DEATH_EVENT == 0])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
17.00   35.00   38.00   40.27   45.00   80.00
summary(heart_failure$ejection_fraction[heart_failure$DEATH_EVENT == 1])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
14.00   25.00   30.00   33.47   38.00   70.00
```

Based on the grouped box plot and summary, alive patients indicate a higher median range compared to dead patients, suggesting that patients who stayed alive tend to have higher ejection fraction rate. There are 3 outliers for alive patients but one alive patient with 80% ejection fraction rate stands out as a notable outlier, which we will consider when we are building our models. There are 4 outliers for dead patients above the third quartile, but it does not seem too significant to remove or impute. Therefore, we decided to keep all of the outliers since these are true observations, and do not seem to affect the accuracy of the predictions until we build our model. However, we will keep an eye on the notable and discovered outliers we have discovered when building our models.

For “serum_creatinine”, we used R code to look at the summary and created a grouped box plot.

```r
summary(heart_failure$serum_creatinine)
summary(heart_failure$serum_creatinine[heart_failure$DEATH_EVENT == 0])
summary(heart_failure$serum_creatinine[heart_failure$DEATH_EVENT == 1])
# Grouped box plot of "serum_creatinine" for different death events
library(ggplot2)
heart_failure$DEATH_EVENT <- as.factor(heart_failure$DEATH_EVENT)
ggplot(heart_failure, aes(x = DEATH_EVENT, y = serum_creatinine)) +
  geom_boxplot() +
  labs(title = "Grouped Box Plot of Serum Creatinine in mg/dL for Different Death Events",
       x = "Death Event",
       y = "Serum Creatinine") +
  theme_minimal() +
  theme(plot.title = element_text(size = 10))
```

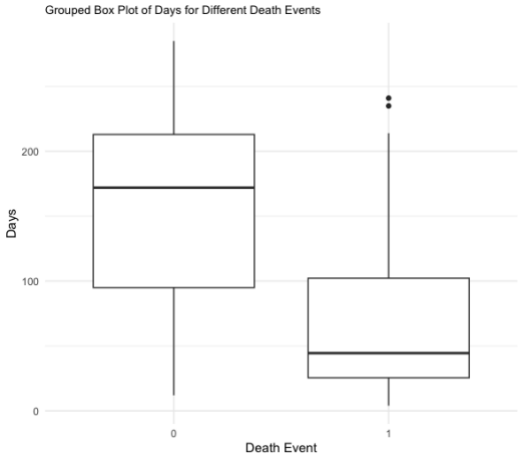



```
summary(heart_failure$serum_creatinine)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.500   0.900   1.100   1.394   1.400   9.400
summary(heart_failure$serum_creatinine[heart_failure$DEATH_EVENT == 0])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.500   0.900   1.000   1.185   1.200   6.100
summary(heart_failure$serum_creatinine[heart_failure$DEATH_EVENT == 1])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.600   1.075   1.300   1.836   1.900   9.400
```

Based on the grouped box plot and summary, we can see that alive patients had overall lower serum creatinine levels with lower range of variance. There were several outliers in Serum Creatinine levels for both Death Events. We will consider either removing the outliers or replacing the outlier values with the mean when we develop our model, if it is significantly affecting the accuracy of the predictions made by the model. For now, we will keep the outliers, as it is a true observation but will keep an eye on those outliers as we progress in building our model.

For “time”, we used R code to look at the summary and created a grouped box plot.

```
summary(heart_failure$time)
summary(heart_failure$time[heart_failure$DEATH_EVENT == 0])
summary(heart_failure$time[heart_failure$DEATH_EVENT == 1])
# Grouped box plot of "time" for different death events
library(ggplot2)
heart_failure$DEATH_EVENT <- as.factor(heart_failure$DEATH_EVENT)
ggplot(heart_failure, aes(x = DEATH_EVENT, y = time)) +
  geom_boxplot() +
  labs(title = "Grouped Box Plot of Days for Different Death Events",
       x = "Death Event",
       y = "Days") +
  theme_minimal() +
  theme(plot.title = element_text(size = 10))
```




```
summary(heart_failure$time)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.0    73.0   115.0   130.3   203.0   285.0
summary(heart_failure$time[heart_failure$DEATH_EVENT == 0])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 12.0    95.0   172.0   158.3   213.0   285.0
summary(heart_failure$time[heart_failure$DEATH_EVENT == 1])
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.00   25.50   44.50   70.89  102.25  241.00
```

Based on the grouped box plot and summary, alive patients indicate a higher median days, suggesting that individuals who experienced death had shorter follow up periods. There are two outliers for two dead patients who had a longer follow up period than usual for dead patients, but it does not seem like a significant factor that will reduce the accuracy of prediction generated by our ideal model. Therefore, we will keep the outlier for now.

Finally, here is a data dictionary:

```
age: Age of the patient (years).
anaemia: Binary variable indicating if the patient has anemia (0 = No, 1 = Yes).
creatinine_phosphokinase: Level of the creatinine phosphokinase enzyme in the blood (mcg/L).
diabetes: Binary variable indicating if the patient has diabetes (0 = No, 1 = Yes).
ejection_fraction: Percentage of blood leaving the heart at each contraction (percentage).
high_blood_pressure: Binary variable indicating if the patient has high blood pressure (0 = No, 1 = Yes).
platelets: Platelets in the blood (kiloplatelets/mL).
serum_creatinine: Level of serum creatinine in the blood (mg/dL).
serum_sodium: Level of serum sodium in the blood (mEq/L).
sex: Gender of the patient (0 = Female, 1 = Male).
smoking: Binary variable indicating if the patient smokes (0 = No, 1 = Yes).
time: Follow-up period (days).
DEATH_EVENT: Binary variable indicating if the patient deceased during the follow-up period (0 = No, 1 = Yes).
```

## 3. Methods

The methods we are exploring are random forest and logistic regression, because they are used in classifying instances. Our problem is to find a way to predict if someone is going to die from a heart failure. To better prepare doctors to potentially save lives of the people with heart failure, it would be best to classify the instances into alive or dead. We found that time might lead to data leakage because if the doctors follow up period was only for a few days, then the patient was most likely dead. So, we decided to remove time from our models.

We considered k-nearest neighbors, but we felt like it would not help in this case, because there are outliers that we chose to keep since there were people who survived who should not normally survive and vice versa. It was also the lowest accurate scoring model. We evaluated the accuracy by dividing the sum of the y_test by the length. It was 72% accurate.

```
> set.seed(42)  # for reproducibility
> train_indices <- sample(1:nrow(heart_failure), 0.7 * nrow(heart_failure))  # 70% for training
> train_data <- heart_failure[train_indices, ]
> test_data <- heart_failure[-train_indices, ]
>
> # Features (X) and target variable (y) for training
> X_train <- train_data[, c("age", "ejection_fraction", "serum_creatinine")]  # Adjust feature names
> y_train <- train_data$DEATH_EVENT
>
> # Features (X) and target variable (y) for testing
> X_test <- test_data[, c("age", "ejection_fraction", "serum_creatinine")]  # Adjust feature names
> y_test <- test_data$DEATH_EVENT
>
> # Choose the number of neighbors (k)
> k <- 5
>
> # Train the KNN model
> knn_model <- knn(train = X_train, test = X_test, cl = y_train, k = k)
>
> # Evaluate the model
> accuracy <- sum(knn_model == y_test) / length(y_test)
> conf_matrix <- table(knn_model, y_test)
>
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.722222222222222"
> print("Confusion Matrix:")
[1] "Confusion Matrix:"
> print(conf_matrix)
         y_test
knn_model  0  1
        0 52 13
        1 12 13
```

We chose random forest because of its ability to look at the importance of all variables that are used in prediction. The random forest model is usually more accurate than other models which was also a reason

we chose this, in hopes of being able to have a more accurate model in general. Another reason for choosing the random forest model is because it is good at reducing the chances of overfitting since the model will average out multiple trees, leading to a better and more accurate score. Along with that, randomly selected features from the data set are considered where the best options get chosen, because of this, the trees have very low correlation, once again usually giving a higher accuracy score.

Logistic regression on the other hand is easier to train and implement. We also thought that the data, considering it is to predict a death or not, could be linearly separable. It is also less inclined to overfitting, and it works well on smaller data sets. Lastly, it not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative) which we thought would be helpful in determining what could cause a heart failure.

## 4. Tools

We utilized R's “rsample” package for data splitting, which allowed us to partition the dataset into training and test sets with a 70-30 split. This was chosen to ensure a substantial amount of data for model training while still retaining enough for an unbiased evaluation.

We employed logistic regression for its interpretability and effectiveness in binary classification problems. Visualization tools, like "ggplot2", were integral for exploring relationships between variables and the outcome, which helped in feature selection.

The "pROC" package was used to construct ROC curves and calculate the AUC, providing a robust measure of model performance. While logistic regression offers insights into individual predictors' effects, it may have limitations in capturing complex nonlinear relationships.

We also experimented with a random forest model using the "randomForest" and “caret”package due to its ability to handle non-linear data and provide variable importance scores. However, random forests are less interpretable than logistic regression models.

To create our baseline model, we used the library “e1071”. It allowed us to make a model with the Naive Bayes function.

Lastly, we explored a KNN classifier for its simplicity and non-parametric nature. The class package was utilized for KNN, which can be effective for smaller, cleaner datasets but might not scale well or handle noisy data effectively.

Each tool and method was selected to balance between the interpretability of the results and the predictive performance. While logistic regression and random forest were integral to our analysis, KNN was ultimately not adopted due to its lower performance in preliminary assessments. Future improvements could include more sophisticated feature engineering, hyperparameter tuning, and exploring ensemble methods.

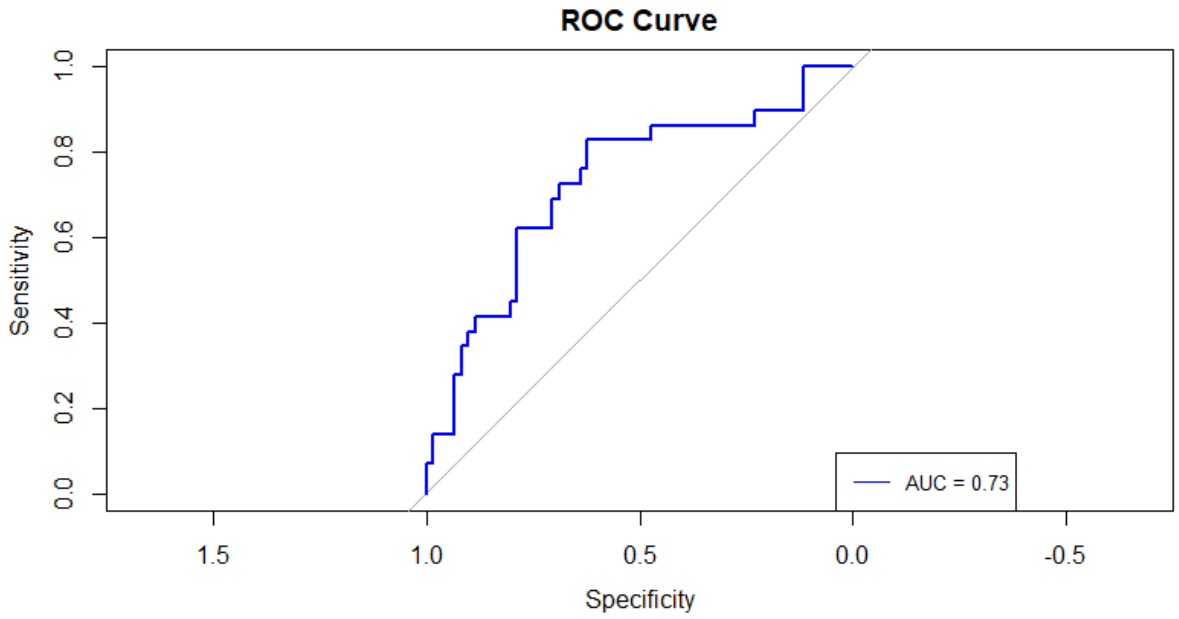
## 5. Results

Logistic Regression Results and Performance Measures:

For logistic regression, we split the original dataset, “heart_failure”, into “train_data” and “test_data”. The “train_data” contains 70% of the original dataset, and the “test_data” contains 30% of the original dataset. Using R code, we took the “heart_failure” data frame and took a look at the summary of the graphed the observations for each feature: “age”, “ejection_fraction”, and “serum_creatinine” for the x-axis and probability of death on the y-axis as Death = 1 (dead) and Death = 0 (alive). It found all three variables to be significant with ejection_fraction and serum_creatinine the most significant. We did try different combinations of different variables, and we found that these three variables produced the lowest AIC score and highest accuracy (AUC). This model was 73% accurate.

```
logisticModel <- glm(DEATH_EVENT ~ age + ejection_fraction + serum_creatinine, data = train_data, family = "binomial")
summary(logisticModel)
```

```
predictions <- predict(logisticModel, newdata = test_data, type = "response")
install.packages("pROC")
library(pROC)
roc_curve <- roc(test_data$DEATH_EVENT, predictions)
plot(roc_curve, main = "ROC Curve", col = "blue")
auc_result <- auc(roc_curve)
legend("bottomright", legend = paste("AUC =", round(auc_result, 2)), col = "blue", lty = 1, cex = 0.8)
```



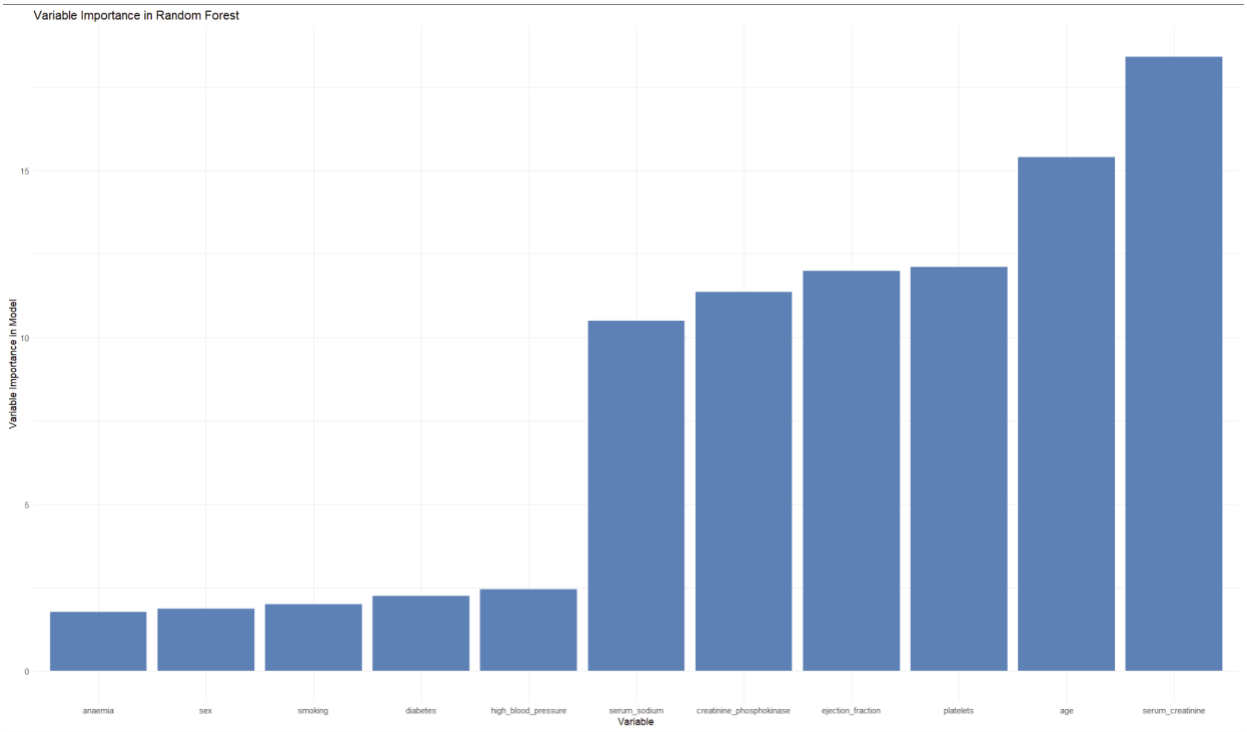

Random Forest Results and Performance Measures:

For the random forest model, we started by creating a subset of the original data frame to exclude time, since time can be seen as a data leak in the model. Once completed, we set the seed for the random

forest, as well as split the data into training and testing sets giving 209 objects for the training, and 90 for the testing, being a 70% split of the data. Afterwards, we created the random forest model and the random forest prediction on the training data, then we created a confusion matrix to get a look into the testing data a bit more. After that, we began to run the model on the test data, once again creating another confusion matrix, however with respect to the testing data now, and then finally created a variable to calculate the accuracy on the model and the confusion matrix. Based on the calculations, the model was accurate 74% of the time. Finally, we plotted the variable importance that was within the model. From the plot, we can see that the random forest found serum_creatine, age, and platelets to have the top 3 highest importance, with ejection_fraction, creatinine_phosphokinase, and serum_sodium following shortly behind, and then the rest of the variables were significantly less impactful on the model.

```
# random forest
rfDataFrame <- subset(df, select = -c(time)) # creates a new df without time
rfDataFrame$DEATH_EVENT <- as.factor(rfDataFrame$DEATH_EVENT)

set.seed(1)
train <- sample(1:nrow(rfDataFrame), 209) # separates the data for training and testing
train_data <- rfDataFrame[train,] # gets 209 objects for training data
test_data <- rfDataFrame[-train,] # gets the remaining 90 objects in the data frame

rf <- randomForest(DEATH_EVENT ~., data = train_data)
rfPrediction <- predict(rf, train_data)
cm <- confusionMatrix(rfPrediction, train_data$DEATH_EVENT)

rfTest <- predict(rf, test_data)
rfConfMat <- confusionMatrix(rfTest, test_data$DEATH_EVENT)
rfAccuracy <- sum(diag(rfConfMat$table)) / sum(rfConfMat$table) # calculates accuracy based on the confusion matrix
rfAccuracy
```

```
> rf

Call:
 randomForest(formula = DEATH_EVENT ~ ., data = train_data)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 26.32%
Confusion matrix:
    0  1 class.error
0 125 17   0.1197183
1  38 29   0.5671642
> cm
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 142   0
         1   0  67

               Accuracy : 1
                 95% CI : (0.9825, 1)
    No Information Rate : 0.6794
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.6794
         Detection Rate : 0.6794
   Detection Prevalence : 0.6794
      Balanced Accuracy : 1.0000

       'Positive' Class : 0

> rfAccuracy
[1] 0.7444444
```




Therefore, our primary model will be the random forest classifier because it did get on average a higher accuracy. As for comparing to our baseline model, we chose to make a Naive Bayes model. The model got between 70-78% accuracy. So our model is on the higher end of the accuracy range. Though, random forest models are typically more complex than the Naive Bayes Model while the Naive Bayes model is easier to interpret than the random forest. The Random forest also can handle diverse data and complex relationships better than the Naive Bayes model which is what is happening in our data set. There is also imbalance data as we have more older people than younger, etc. and random forests can handle imbalanced data better. Random Forest was also able to provide the features it found important, but Naive Bayes can not.

```
> (tab1 <- table(p1, train_data$DEATH_EVENT))

p1    0   1
  0 131  43
  1  13  22
> 1 - sum(diag(tab1)) / sum(tab1)
[1] 0.2679426
> #Test data
> p2 <- predict(model, test_data)
> (tab2 <- table(p2, test_data$DEATH_EVENT))

p2   0  1
  0 59 24
  1  0  7
> 1 - sum(diag(tab2)) / sum(tab2)
[1] 0.2666667
> set.seed(123)
> health_split <- initial_split(heart_failure,prop = 0.70)
> train_data <- training(health_split)
> test_data <- testing(health_split)
> model <- naiveBayes(DEATH_EVENT ~ age + ejection_fraction + serum_creatinine, data = heart_failure, usekernel = T)
> #train data
> p <- predict(model, train_data, type = 'class')
> p1 <- predict(model, train_data)
> (tab1 <- table(p1, train_data$DEATH_EVENT))

p1    0   1
  0 134  45
  1   8  22
> 1 - sum(diag(tab1)) / sum(tab1)
[1] 0.2535885
> #Test data
> p2 <- predict(model, test_data)
> (tab2 <- table(p2, test_data$DEATH_EVENT))

p2   0  1
  0 56 22
  1  5  7
> 1 - sum(diag(tab2)) / sum(tab2)
[1] 0.3
>
```

For an informal success measure, we used stratified k-fold cross-validation (cv) for logistic regression and random forest, because the dataset is imbalanced where DEATH_EVENT = 1 and DEATH_EVENT = 0 in the dataset is not 50:50, and the dataset is small enough (only 299 instances).

Here is the R code for stratified k-fold cross-validation (cv) for logistic regression:

```
# Stratified k-fold Cross-Validation on Logistic Regression

# Set working directory that suits you
# setwd("/Users/Brian/Desktop/ds-project/Heart-Failure-Prediction")

# Load data
library(readr)
hfdata <- read_csv("heart_failure_clinical_records_dataset.csv")

# install.packages("caret") # run once
library(ggplot2)
library(caret)

# we need DEATH_EVENT to be a factor
hfdata$DEATH_EVENT <- as.factor(hfdata$DEATH_EVENT)

# change DEATH_EVENT (factor) to 0 or 1 to valid R variable name
levels(hfdata$DEATH_EVENT) <- c("ALIVE", "DEAD")

# 10-fold cross-validation repeated 3 times
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 3,
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary)
set.seed(123) # for reproducibility
# build the Stratified k-fold Cross-Validation model for Logistic Regression of whole data
cv_model_logReg <- train(DEATH_EVENT ~ age + serum_creatinine + ejection_fraction,
                data = hfdata,
                method = "glm", # logistic regression
                family = "binomial",
                trControl = control,
                preProcess = c("center", "scale"),
                metric = "ROC")
print(cv_model_logReg) # print the result
```

Here is the result of the R code for stratified k-fold cross-validation (cv) for logistic regression:

```
Generalized Linear Model

299 samples
  3 predictor
  2 classes: 'ALIVE', 'DEAD'

Pre-processing: centered (3), scaled (3)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 268, 269, 269, 270, 270, 269, ...
Resampling results:

  ROC        Sens   Spec
  0.7767059  0.895  0.4811111
```

The 3 predictors are “age”, “serum_creatine”, and “ejection_fraction”. The outcome variable “DEATH_EVENT” has two classes, ‘ALIVE’ or ‘DEAD’. Resampling method is 10-fold cross-validation, and the process was repeated 3 times for robustness. The summary of sample sizes indicates the number of samples in each fold of the cross validation, and numbers vary slightly due to division of the dataset into folds. The “ROC '' is the area under the Receiver Operating Characteristic curve, which about 0.777 suggests a good level of predictive ability for this model. The “Sens” (sensitivity) or (true positive rate) or

(recall) of 0.895 measures the proportion of actual positives (DEAD) correctly identified by the model. In other words, the model correctly identifies about 89.5% of patients who are classified as 'DEAD'. The "Spec" (specificity) or (true negative rate) or (precision) of 0.48 measures the proportion of actual positives (ALIVE) correctly identified. In other words, the model correctly identifies about 48% of the patients who are classified as 'ALIVE'.

Here is the R code for stratified k-fold cross-validation (cv) for random forest:

```
# Stratified k-fold Cross-Validation on Random Forest

# set working directory that suits you
# setwd("/Users/Brian/Desktop/ds-project/Heart-Failure-Prediction")

# Load data
library(readr)
hfdata <- read_csv("heart_failure_clinical_records_dataset.csv")

# install.packages("caret") # run once
# install.packages("randomForest") # run once
# install.packages("ggplot2") # run once
library(ggplot2)
library(caret)
library(randomForest)

# we need DEATH_EVENT to be a factor
hfdata$DEATH_EVENT <- as.factor(hfdata$DEATH_EVENT)

# change DEATH_EVENT (factor) to 0 or 1 to valid R variable name
levels(hfdata$DEATH_EVENT) <- c("ALIVE", "DEAD")

# 10-fold cross-validation repeated 3 times
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 3,
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary)
set.seed(123)  # for reproducibility
# build the Stratified k-fold Cross-Validation model for Random Forest of whole data
cv_model_rf <- train(DEATH_EVENT ~ age + serum_creatinine + ejection_fraction,
                     data = hfdata,
                     method = "rf", # random forest
                     trControl = control,
                     metric = "ROC")
print(cv_model_rf) # print the result
```

Here is the result of the R code for stratified k-fold cross-validation (cv) for random forest:

```
Random Forest

299 samples
  3 predictor
  2 classes: 'ALIVE', 'DEAD'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 268, 269, 269, 270, 270, 269, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
  2     0.8211318  0.8392857  0.5818519
  3     0.8129766  0.8276984  0.5888889

ROC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

The 3 predictors are “age”, “serum_creatine”, and “ejection_fraction”. The outcome variable “DEATH_EVENT” has two classes, ‘ALIVE’ or ‘DEAD’. Resampling method is 10-fold cross-validation, and the process was repeated 3 times for robustness. The summary of sample sizes indicates the number of samples in each fold of the cross validation, and numbers vary slightly due to division of the dataset into folds. The “mtry” is the number of variables randomly sampled as candidates at each split in the tree. The “ROC” is the area under the Receiver Operating Characteristic curve for each “mtry” value. The higher values are better, indicating a better model. For “mtry” = 2, ROC is 0.8211318, and for “mtry” = 3, it's 0.8129766. The “Sens” (sensitivity) or (true positive rate) or (recall) for “mtry” = 2, the sensitivity is 0.8392857, and for “mtry” = 3, it's 0.8276984. This metric indicates the proportion of actual positives correctly identified. The “Spec” (specificity) or (true negative rate) or (precision) for “mtry” = 2, the specificity is 0.5818519, and for “mtry” = 3, it's 0.5888889. This metric indicates the proportion of actual negatives correctly identified.

**6. Summary and Conclusions**

The random forest model proved the most effective model out of the three we had tried. From the results, we found the model is accurate about 74% of the time, with our baseline model being about 70-78% accurate. The model showed that serum_creatinine, age, and platelets were the top 3 influencers in the prediction with ejection_fraction following shortly behind platelets. This was surprising as we thought ejection_fraction would be higher up on the scale of importance, but according to the model this was not the case. Another surprising result that was found from the model was that serum sodium was also a pretty important factor when it came to predicting failure. When serum sodium levels are low, it usually always indicates a high chance of heart failure, since serum sodium links to an electrolyte disorder called hyponatremia. This is surprising since we had not considered serum sodium to be a big predictor when it came to heart failure, however, after using the model we discovered otherwise, and it is also a link to a disorder where heart failure is very common.

High-level summary of results for logistic regression:

Logistic regression was only 73% accurate which is only 1 percent less accurate than the random forest. We built the model off multiple combinations of the attributes, but we found that the three: age, ejection_fraction, and serum_creatinine were the highest accurate ones and they also had the lowest AIC score. The coefficients for the model lines up with the significance values from the summary of the

```
Call:
glm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine,
    family = "binomial", data = train_data)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -2.16264    1.01049  -2.140   0.0323 *
age                0.05672    0.01525   3.719   0.0002 ***
ejection_fraction -0.09029    0.01909  -4.728 2.26e-06 ***
serum_creatinine   0.79863    0.19100   4.181 2.90e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 262.21  on 208  degrees of freedom
Residual deviance: 201.45  on 205  degrees of freedom
AIC: 209.45

Number of Fisher Scoring iterations: 5

> coefficients(logisticModel)
      (Intercept)               age ejection_fraction
      -2.16264197        0.05672099       -0.09028553
 serum_creatinine
       0.79862689
>
```

model. The null deviance is substantially higher than the residual deviance which might suggest that there is some predictive power. We also had 5 Fisher Scoring iterations which means our data needed several steps to find the maximum likelihood of the estimates. It quickly found the parameter estimates that satisfied the optimization criteria. The random forest is able to handle non linearity in the data, feature interactions, and has overfitting control which might contribute to its 1% performance increase since there is not a linear relationship between every attribute.

High-level summary of results for stratified k-fold cross validation for logistic regression:

The ROC values indicate that the model has a good ability to distinguish between the two classes: “ALIVE” and “DEAD”. The high sensitivity suggests that the model is quite effective at identifying patients who are at risk of death (label “DEAD”). However, the relatively lower specificity indicates that the model is less effective at correctly identifying patients who are not at risk of death (label “ALIVE”). These results suggest that while the model is good at identifying high-risk patients, it might also have a higher rate of false alarms (wrongly predicting death). This kind of insight is crucial for making informed decisions on how to use the model, especially in a critical field like healthcare.

High-level summary of results for stratified k-fold cross validation for random forest:

The model with "mtry" = 2 is considered the best among the ones tested, based on the ROC metric. The ROC value of approximately 0.821 suggests good predictive ability, superior to the logistic regression model which had an ROC of 0.780851.The sensitivity and specificity values give insights into the model's ability to correctly identify positive (DEAD) and negative (ALIVE) cases, respectively. While the model has

a reasonably good sensitivity (around 84%), its specificity is somewhat lower (around 58%), indicating it's better at identifying positive cases than negative ones.

## 7. Appendix

https://github.com/kavang02/Heart-Failure-Prediction

https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data