HarvardX: PH125.9x Capstone Heart Failure Prediction

Suhaimi William Chan

9/22/2020

1. Introduction/Overview

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs and this dataset contains 12 features that can be used to predict mortality by heart failure.[3]

Most cardiovascular diseases can be prevented by addressing behavioural risk factors such as tobacco use, unhealthy diet and obesity, physical inactivity and harmful use of alcohol using population-wide strategies.[3]

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.[3]

The primary goal of heart failure prediction is to create a model to assess the likelihood of a death by heart failure event. This can be used to help hospitals in assessing the severity of patients with cardiovascular diseases.

In this document, we create a heart failure prediction using the heart failure clinical record dataset[5] and applying the courses/lessons learned during the HarvardX's Data Science Professional Certificate program.

This document is structured as follows:
Chapter 1 describes the dataset and summarizes the goal of the project and key steps that were performed.
Chapter 2 We explain the method, process and techniques used, such as data cleaning, data exploration and visualization, any insights gained.
Chapter 3 We present the modeling approach and analysis.
Chapter 4 We discuss the results and model performance.
Chapter 5 We conclude with a brief summary of the report, its limitations and future work.

We are going to start with the following library:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
```

Heart Failure Clinical Record Dataset is downloaded directly from UCI machine learning repository (https://archive.ics.uci.edu/ml/machine-learning-databases/00519/heart_failure_clinical_records_dataset.csv).[7]

Ultimately, a machine learning algorithm is evaluated on how it performs in the real world with completely new datasets. However, when developing an algorithm, we usually have a dataset for which we know the outcomes. Therefore, to mimic the ultimate evaluation process, we typically split the data into two parts and act as if we don't know the outcome for one of these. We stop pretending we don't know the outcome

to evaluate the algorithm, but only after we are done constructing it. We refer to the group for which we know the outcome, and use to develop the algorithm, as the training set. We refer to the group for which we pretend we don't know the outcome as the test set and/or validation set.[1]

The csv file data were directly downloaded into our local variable that we named it "dl".
Then we partitioned dl dataset into edx dataset and validation dataset with 80%/20% split, respectively.
We will convert the edx dataset into train_set data frame.
We split the data into 80/20 because we need large enough population to train our models.
Since we only have 299 records, splitting into 80/20 would result with 239 records to train our models, which is good enough.

We will now develop an algorithm using only the training set. Once we are done developing the algorithm, we will freeze it and evaluate it using the test set. The simplest way to evaluate the algorithm when the outcomes are categorical is by simply reporting the proportion of cases that were correctly predicted in the test set/validation set. This metric is usually referred to as overall accuracy.[1]

Validation dataset will be split 50%/50% into test_set data frame and validation_set data frame.
Test_set and validation_set, each will have 30 records, which are usually the minimum to have a reliable accuracy.

We are going to train our models using train_set and test our models using test_set to confirm the performance of our models.

Once we confirm the performance/accuracy of our models are good enough, we will do a final test/validation with our validation_set.

Let's get started.
First, we are going to glimpse the dataset that we imported into dl variable.

```
## Rows: 299
## Columns: 13
## $ age                       <dbl> 75, 55, 65, 50, 65, 90, 75, 60, 65, 80, 75...
## $ anaemia                   <dbl> 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, ...
## $ creatinine_phosphokinase <dbl> 582, 7861, 146, 111, 160, 47, 246, 315, 15...
## $ diabetes                  <dbl> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...
## $ ejection_fraction         <dbl> 20, 38, 20, 20, 20, 40, 15, 60, 65, 35, 38...
## $ high_blood_pressure       <dbl> 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, ...
## $ platelets                 <dbl> 265000, 263358, 162000, 210000, 327000, 20...
## $ serum_creatinine          <dbl> 1.90, 1.10, 1.30, 1.90, 2.70, 2.10, 1.20, ...
## $ serum_sodium              <dbl> 130, 136, 129, 137, 116, 132, 137, 131, 13...
## $ sex                       <dbl> 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, ...
## $ smoking                   <dbl> 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, ...
## $ time                      <dbl> 4, 6, 7, 7, 8, 8, 10, 10, 10, 10, 10, 10, ...
## $ DEATH_EVENT               <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
```

It shows that we have 299 rows and 13 columns.

Second, we are going to check and count any missing values (NA or NULL)

```
## # A tibble: 1 x 13
##   NA_age NA_anaemia NA_cp NA_diabetes NA_ef NA_hbp NA_platelets NA_sc NA_ss
##    <int>      <int> <int>       <int> <int>  <int>        <int> <int> <int>
## 1      0          0     0           0     0      0            0     0     0
## # ... with 4 more variables: NA_sex <int>, NA_smoking <int>, NA_time <int>,
## #   NA_DEATH_EVENT <int>
```

It shows that our data are clean, and we have no missing values.

Third, we are going to create our first data partition from dl dataset into edx dataset and validation dataset with 80%/20% ratio, respectively.

Then we are going to create our second data partition from validation dataset into test_set and validation_set with 50%/50% ratio, respectively.

After partitioning our datasets, let's check the dimensions of each dataset.

Here is the dimension of our edx dataset

```
## [1] 239  13
```

Here is the dimension of our validation dataset

```
## [1] 60 13
```

2. Exploratory Data Analysis

Now we are doing our data exploration using edx dataset, to see a more complete data set, instead of using train_set. We can see some examples of our edx dataset with available columns

```
## # A tibble: 6 x 13
##     age anaemia creatinine_phos~ diabetes ejection_fracti~ high_blood_pres~
##   <dbl>   <dbl>            <dbl>    <dbl>            <dbl>            <dbl>
## 1    75       0              582        0               20                1
## 2    55       0             7861        0               38                0
## 3    65       0              146        0               20                0
## 4    65       1              160        1               20                0
## 5    75       1              246        0               15                0
## 6    60       1              315        1               60                0
## # ... with 7 more variables: platelets <dbl>, serum_creatinine <dbl>,
## #   serum_sodium <dbl>, sex <dbl>, smoking <dbl>, time <dbl>, DEATH_EVENT <dbl>
```

We can see classes and dimension of our edx dataset

```
## tibble [239 x 13] (S3: tbl_df/tbl/data.frame)
##  $ age                     : num [1:239] 75 55 65 65 75 60 65 80 75 62 ...
##  $ anaemia                 : num [1:239] 0 0 0 1 1 1 0 1 1 0 ...
##  $ creatinine_phosphokinase: num [1:239] 582 7861 146 160 246 ...
##  $ diabetes                : num [1:239] 0 0 0 1 0 1 0 0 0 0 ...
##  $ ejection_fraction       : num [1:239] 20 38 20 20 15 60 65 35 38 25 ...
##  $ high_blood_pressure     : num [1:239] 1 0 0 0 0 0 0 1 1 1 ...
##  $ platelets               : num [1:239] 265000 263358 162000 327000 127000 ...
##  $ serum_creatinine        : num [1:239] 1.9 1.1 1.3 2.7 1.2 1.1 1.5 9.4 4 0.9 ...
##  $ serum_sodium            : num [1:239] 130 136 129 116 137 131 138 133 131 140 ...
##  $ sex                     : num [1:239] 1 1 1 0 1 1 0 1 1 1 ...
##  $ smoking                 : num [1:239] 0 0 1 0 0 1 0 1 1 1 ...
##  $ time                    : num [1:239] 4 6 7 8 10 10 10 10 10 10 ...
##  $ DEATH_EVENT             : num [1:239] 1 1 1 1 1 1 1 1 1 1 ...
```

Attribute Information [6]:
Thirteen (13) clinical features:
- 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)
- [target] death event: if the patient deceased during the follow-up period (boolean)

Based on the clinical features above, here is a list of our boolean data type with what its values mean:
- Age - Age of patient
- Sex - Gender of patient 0 = Female, 1 = Male
- Diabetes - 0 = No, 1 = Yes

- Anaemia - 0 = No, 1 = Yes
- High_blood_pressure - 0 = No, 1 = Yes
- Smoking - 0 = No, 1 = Yes
- DEATH_EVENT - 0 = No, 1 = Yes

We are going to convert all the boolean data types to factor class into train_set data frame for better plotting of our exploratory data analysis.

Let's check the class and dimension of our new data frame train_set:

```
## 'data.frame':    239 obs. of  13 variables:
##  $ age                     : num  75 55 65 65 75 60 65 80 75 62 ...
##  $ anaemia                 : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 1 2 2 1 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 2 2 2 ...
##  $ creatinine_phosphokinase: num  582 7861 146 160 246 ...
##  $ diabetes                : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 1 1 ...
##  $ ejection_fraction       : num  20 38 20 20 15 60 65 35 38 25 ...
##  $ platelets               : num  265000 263358 162000 327000 127000 ...
##  $ sex                     : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 1 2 2 2 ...
##  $ serum_creatinine        : num  1.9 1.1 1.3 2.7 1.2 1.1 1.5 9.4 4 0.9 ...
##  $ serum_sodium            : num  130 136 129 116 137 131 138 133 131 140 ...
##  $ smoking                 : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 2 2 ...
##  $ time                    : num  4 6 7 8 10 10 10 10 10 10 ...
##  $ DEATH_EVENT             : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Let's check the class and dimension of our new data frame test_set:

```
## 'data.frame':    30 obs. of  13 variables:
##  $ age                     : num  90 45 65 57 68 60 60 72 63 45 ...
##  $ anaemia                 : Factor w/ 2 levels "0","1": 2 1 1 2 2 1 1 2 1 1 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 2 1 1 ...
##  $ creatinine_phosphokinase: num  47 582 94 129 577 897 53 328 936 292 ...
##  $ diabetes                : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 1 1 2 ...
##  $ ejection_fraction       : num  40 14 50 30 25 45 50 30 38 35 ...
##  $ platelets               : num  204000 166000 188000 395000 166000 297000 286000 621000 304000 850
##  $ sex                     : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 1 1 2 2 ...
##  $ serum_creatinine        : num  2.1 0.8 1 1 1 1 2.3 1.7 1.1 1.3 ...
##  $ serum_sodium            : num  132 127 140 140 138 133 143 138 133 142 ...
##  $ smoking                 : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 2 2 2 ...
##  $ time                    : num  8 14 29 42 43 80 87 88 88 88 ...
##  $ DEATH_EVENT             : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 1 2 1 1 ...
```

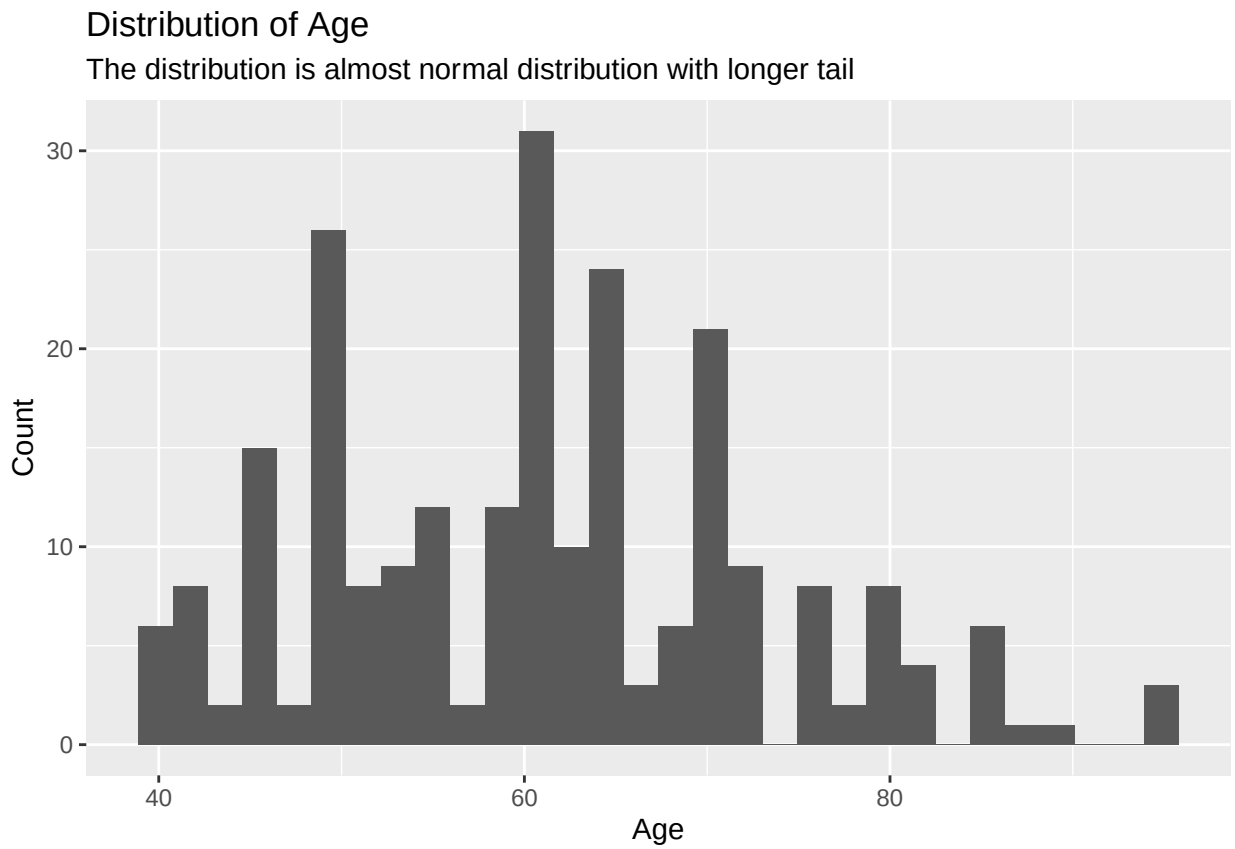
Let's check the class and dimension of our new data frame validation_set:

```
## 'data.frame':    30 obs. of  13 variables:
##  $ age                     : num  50 75 58 49 72 45 65 70 55 70 ...
##  $ anaemia                 : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 1 1 1 2 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "0","1": 1 2 1 2 2 2 1 2 2 1 ...
##  $ creatinine_phosphokinase: num  111 582 60 789 364 ...
##  $ diabetes                : Factor w/ 2 levels "0","1": 1 2 1 1 2 2 2 1 1 1 ...
##  $ ejection_fraction       : num  20 30 38 20 20 25 50 60 45 60 ...
##  $ platelets               : num  210000 263358 153000 319000 254000 ...
##  $ sex                     : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 1 1 1 ...
##  $ serum_creatinine        : num  1.9 1.83 5.8 1.1 1.3 1 1.3 0.8 0.9 1.1 ...
```

```
##  $ serum_sodium            : num  137 134 134 136 136 139 137 140 140 136 ...
##  $ smoking                 : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 2 1 1 ...
##  $ time                    : num  7 23 26 55 59 60 72 74 74 85 ...
##  $ DEATH_EVENT             : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 1 1 1 ...
```

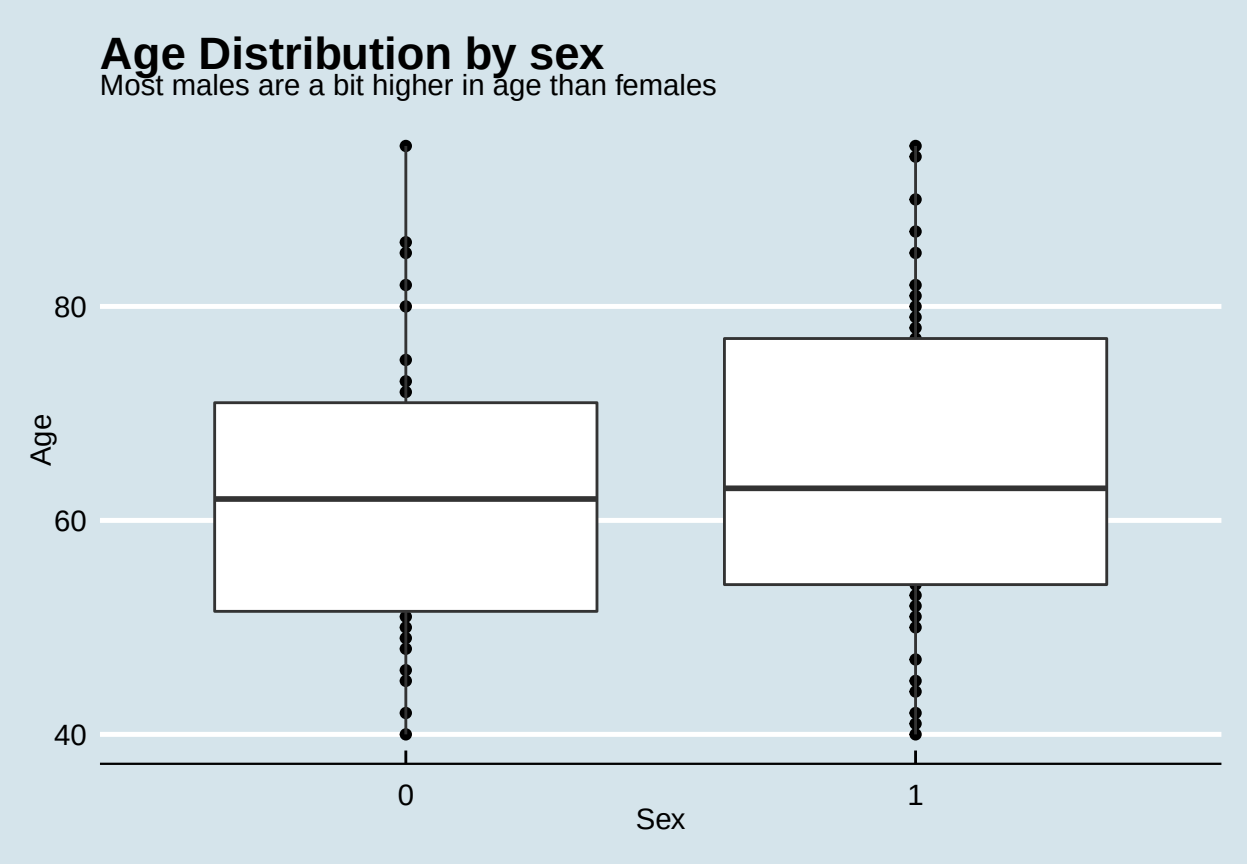
We are going to add the following library for plotting the charts:

```
# Install library for themes and scales
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "http://cran.us.r-project.org")
library(ggthemes)
library(scales)
options(pillar.sigfig = 4)
```



[Chart 1] We can visually see the Age Distribution of our edx dataset

The Age distribution is almost normal distribution with a little longer tail. Majority ages are between 40 and 80.



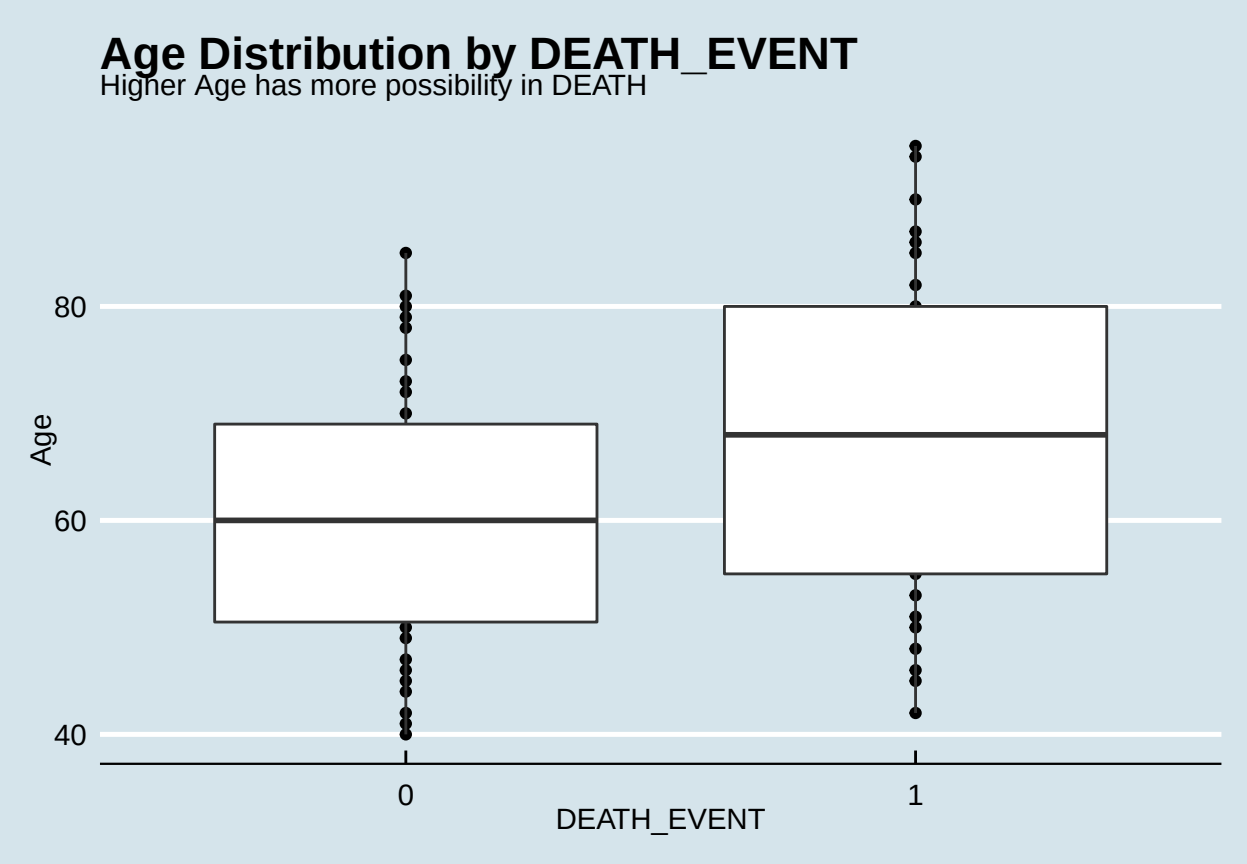
[Chart 2] We can also visually see the Distribution of Age by sex in a box plot.

According to Age Distribution by sex, most males are a bit higher in age than females.

A question that is worth to investigage:

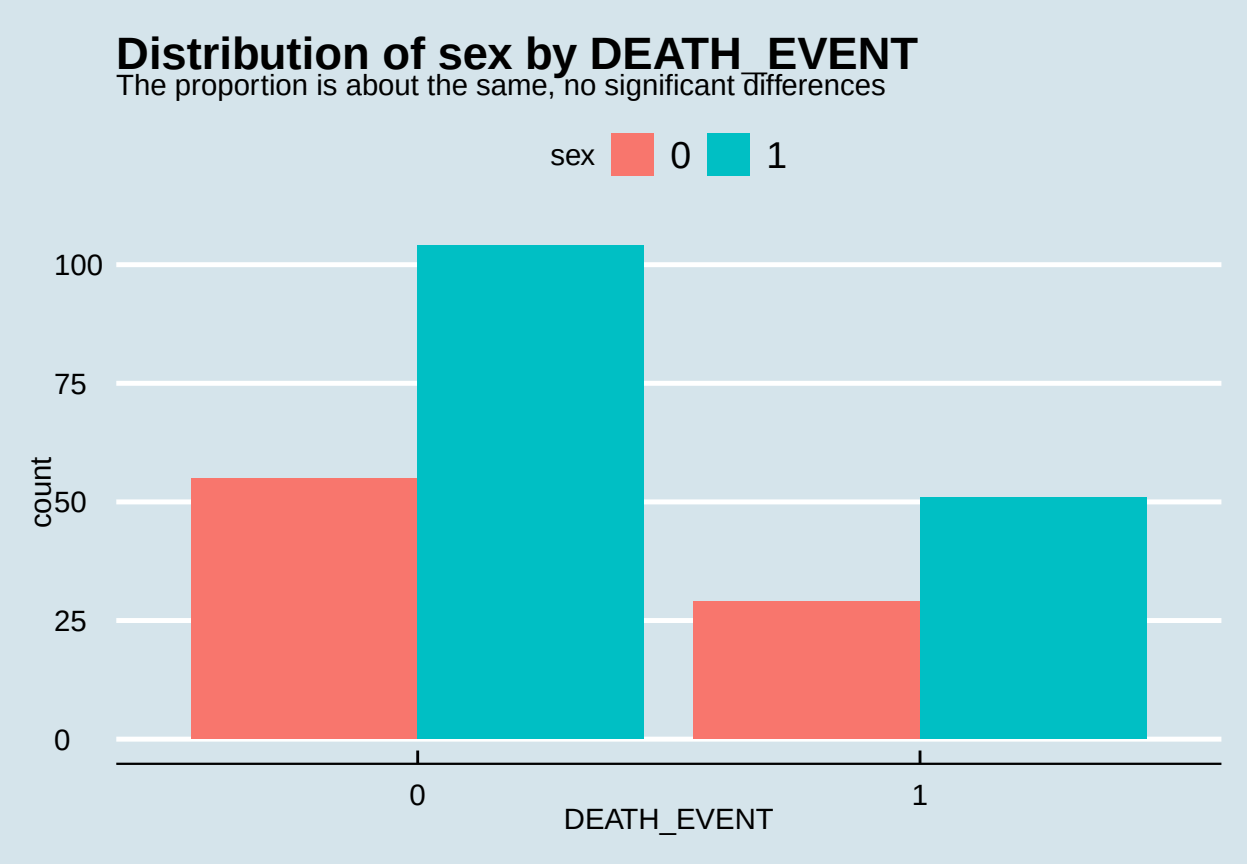
Is age or sex an indicator for death event?

Let's exploit the relationship of age or sex to Death_Event.



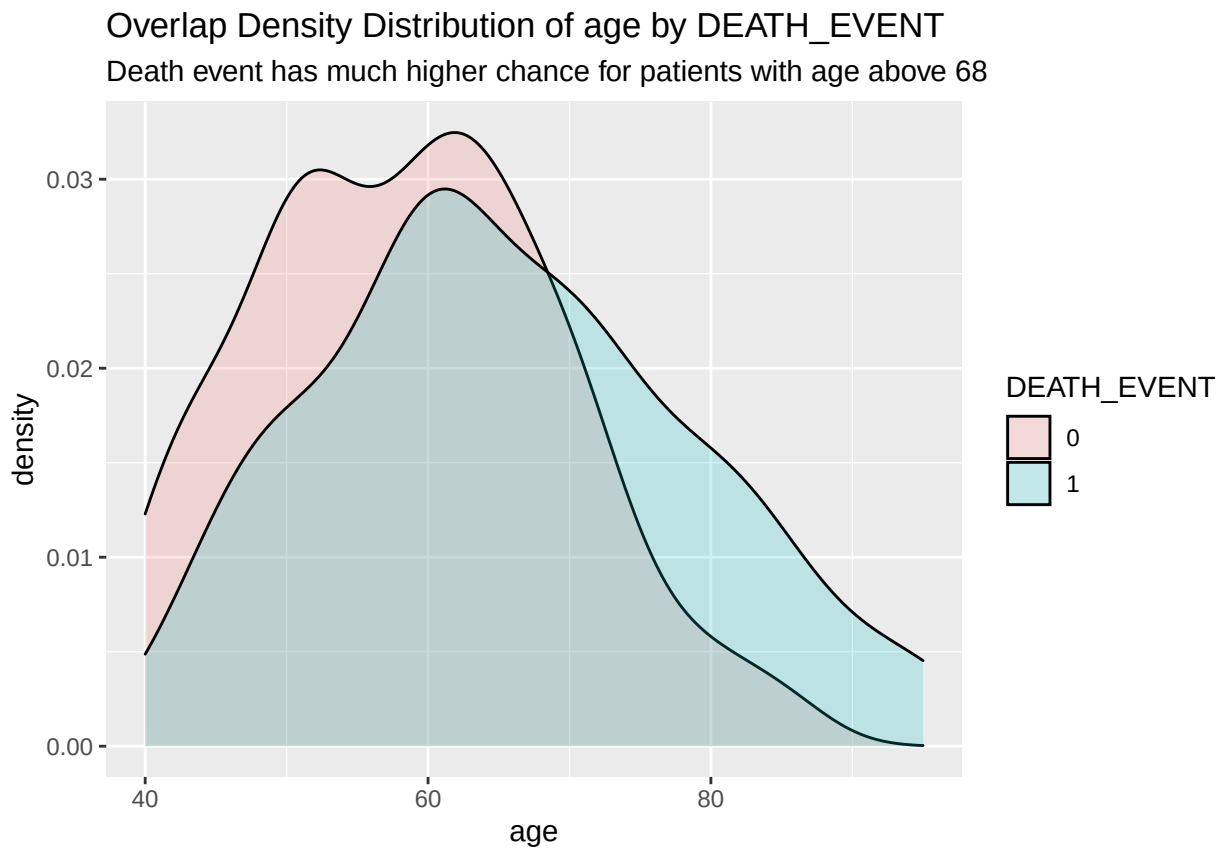
[chart 3] We can also visually see the distribution of age by Death Event.

According to Age Distribution by DEATH_EVENT, higher age has more possibility in death event.



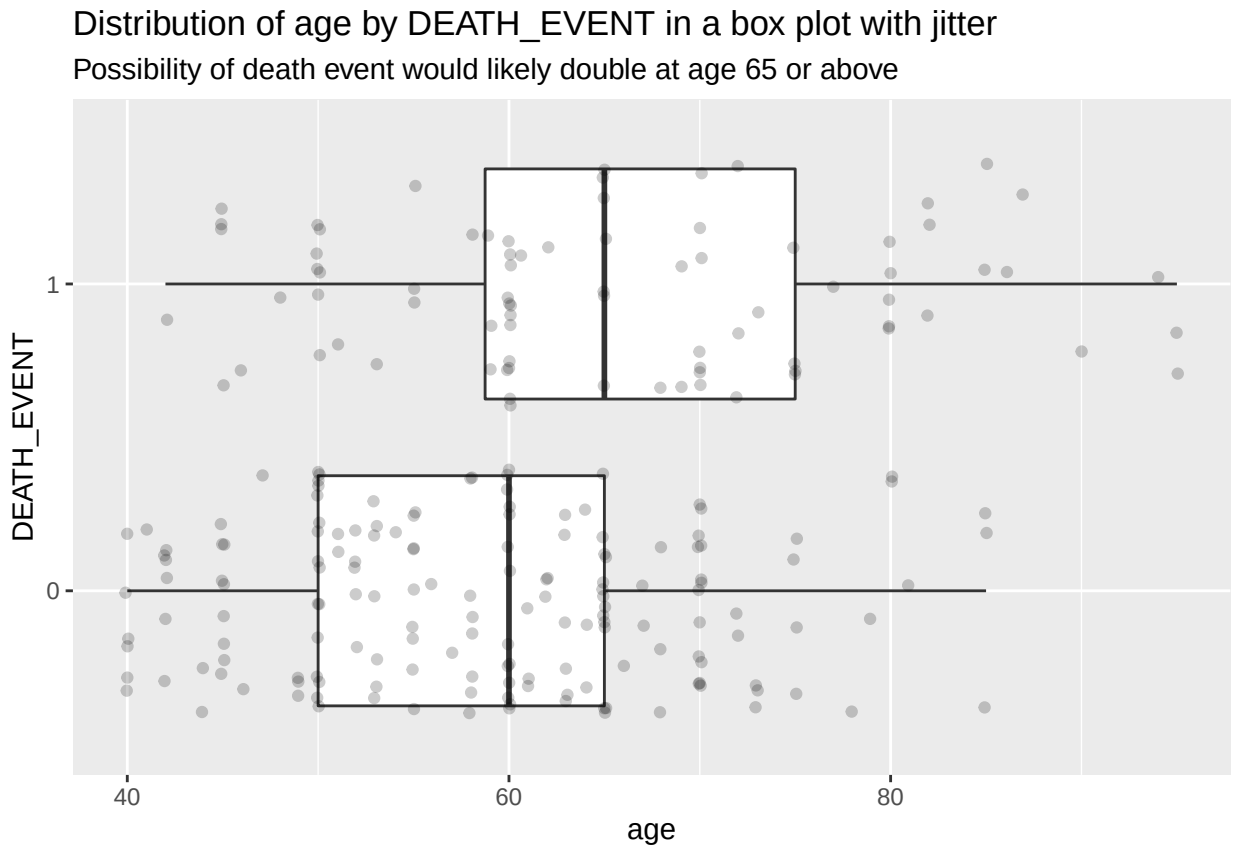
[chart 4] We can also visually see the distribution of sex by Death Event

According to the Distribution of sex by DEATH_EVENT, the proportion is about the same, no significant differences.



[chart 5] Age by DEATH_EVENT based on our findings on chart 3.

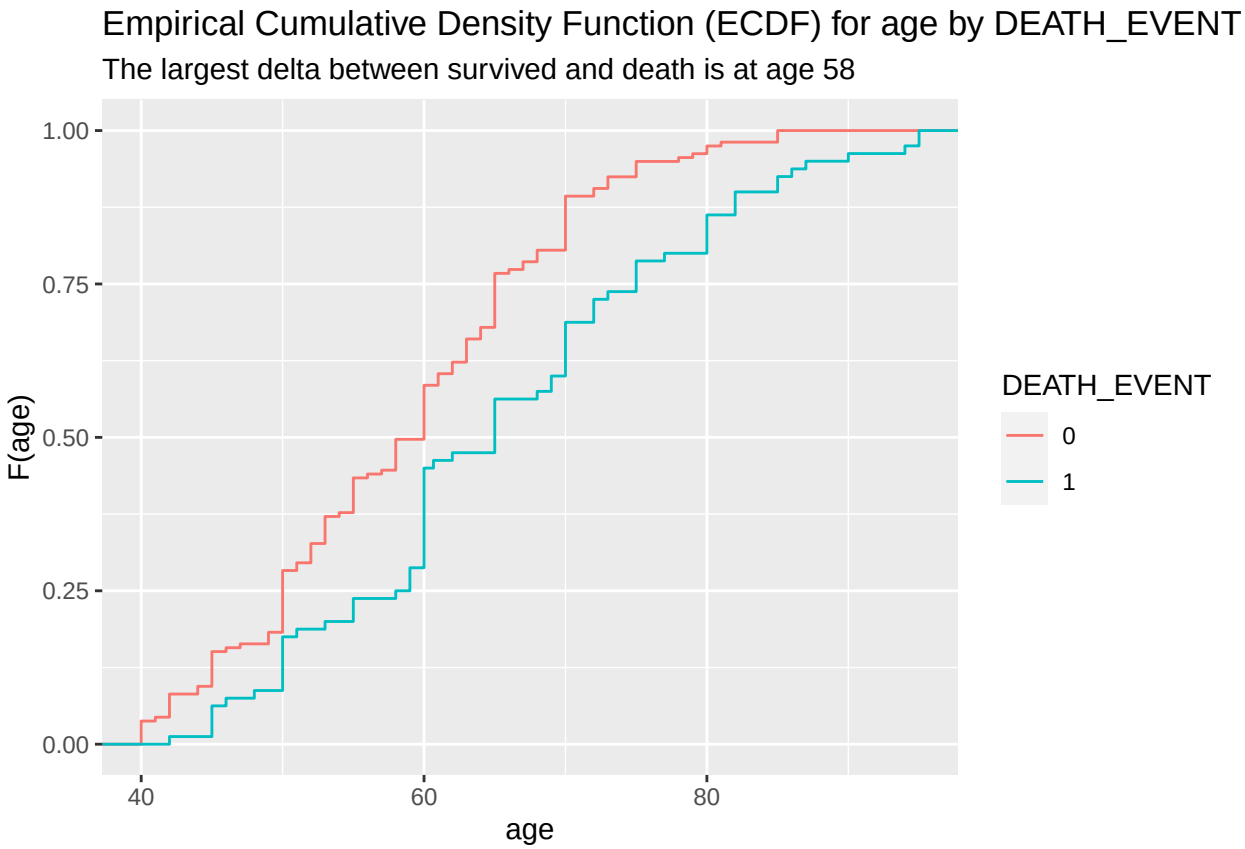
From the chart above, we can see that death event has much higher possibility/proportion for the patients with age above 68.



[chart 6] Age by DEATH_EVENT in a box plot with jitter based on our findings on chart 5.

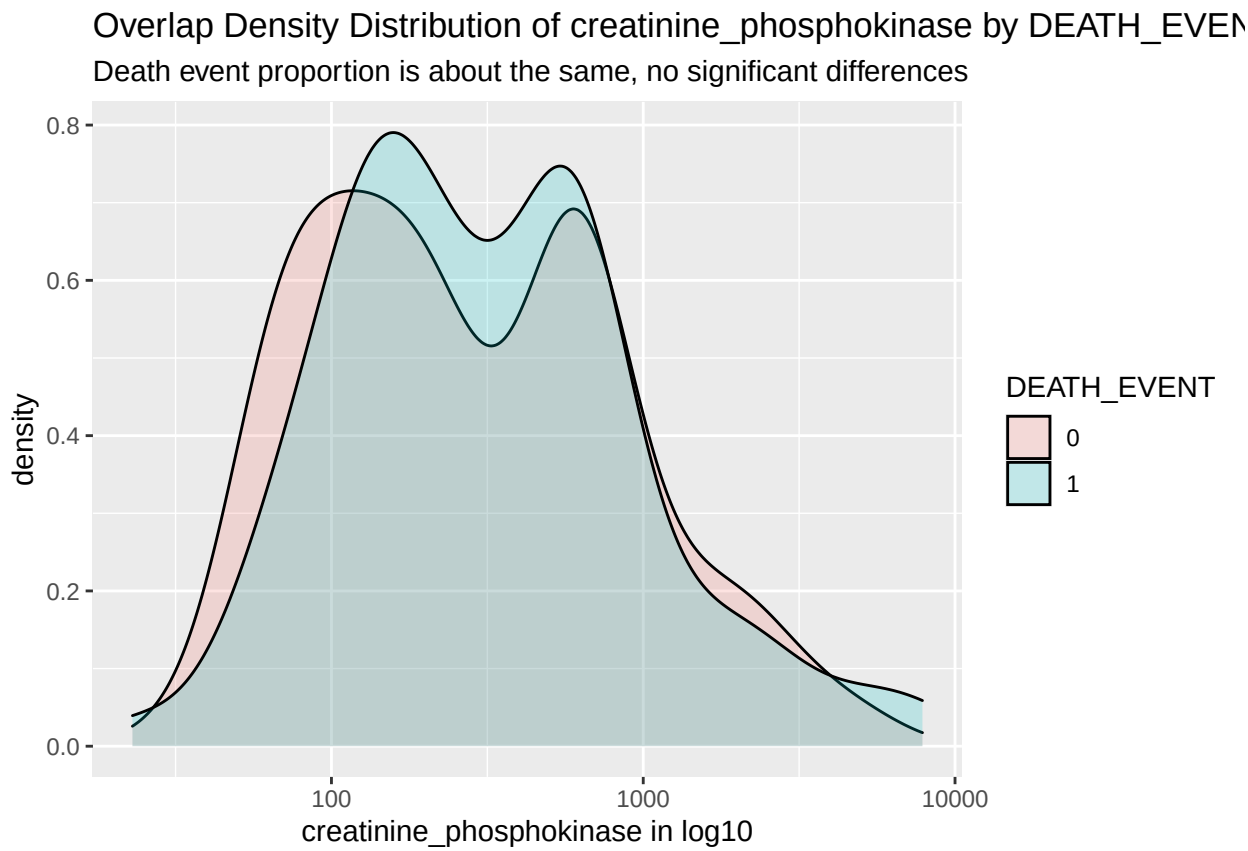
The chart above shows us that the possibility of death event would likely double at age 65 or above.

It also shows us that the chance of survival would likely almost double around age 58 or less.



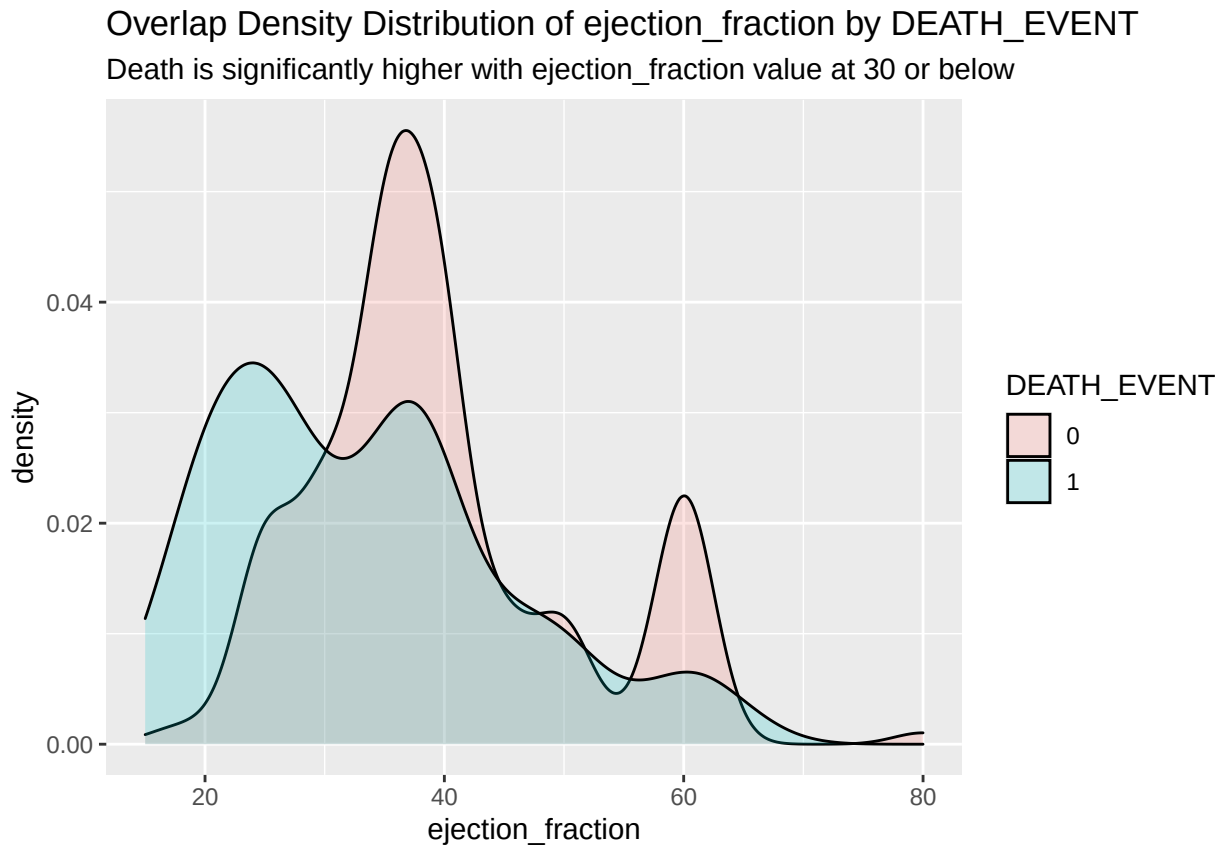
[chart 7] Age by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).

According to ECDF chart, there are big variance between survived and death at age 55, 58, 65, and 70, with the largest delta between survived and death is at age 58.



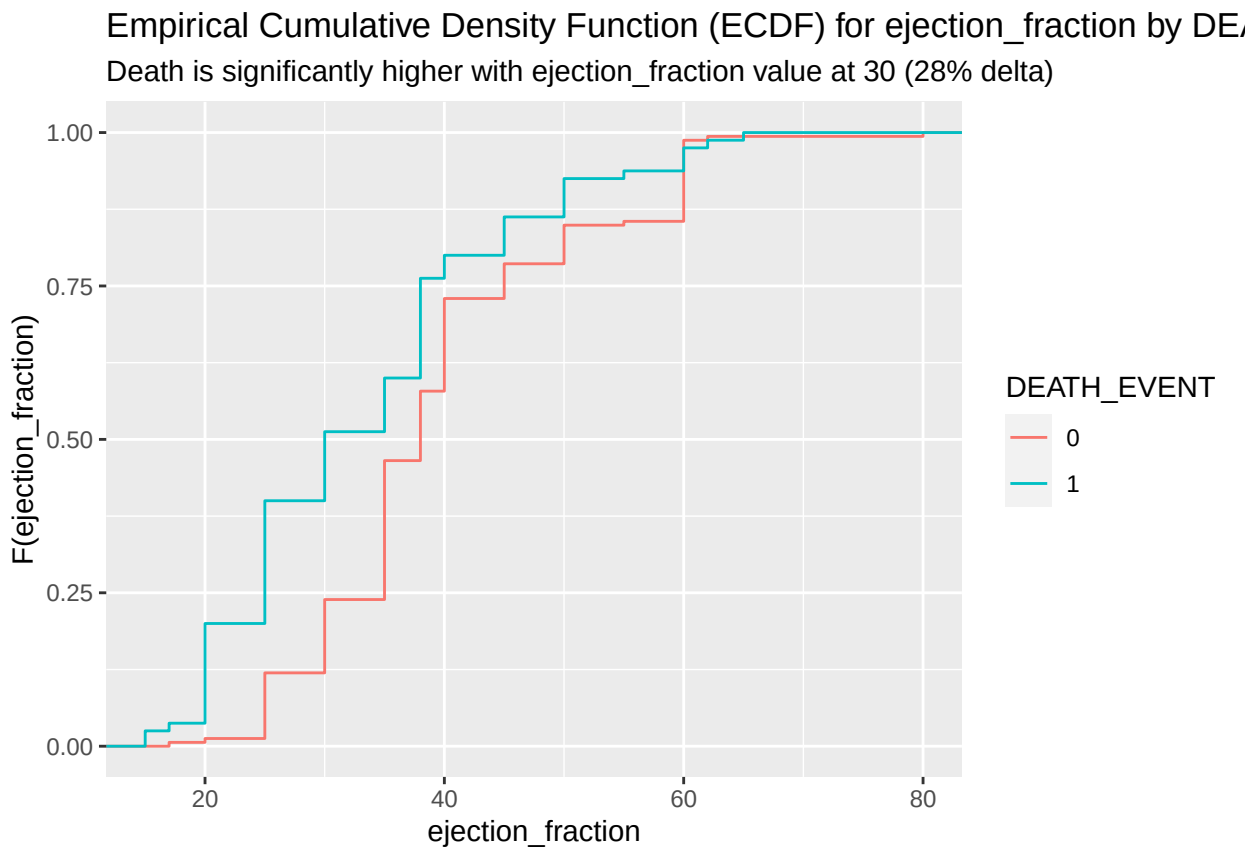
[chart 8] Distribution of creatinine_phosphokinase by Death Event

From the chart above, we can see that death event proportion is about the same, no significant differences.



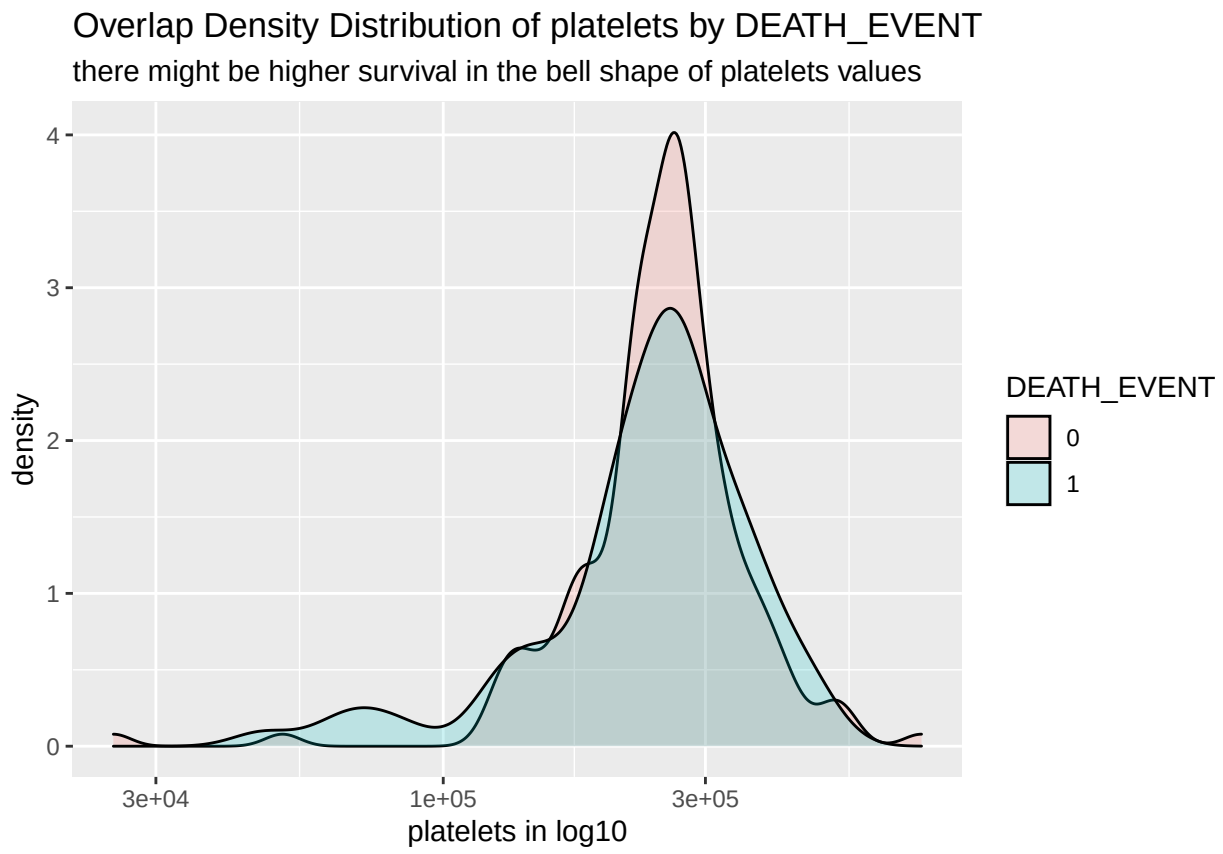
[chart 9] Distribution of ejection_fraction by Death Event

From the chart above, Death is significantly higher with ejection_fraction value at 30 or below. It also shows that survival is higher with ejection_fraction between 30 and low 40s.



[chart 10] Ejection_fraction by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).

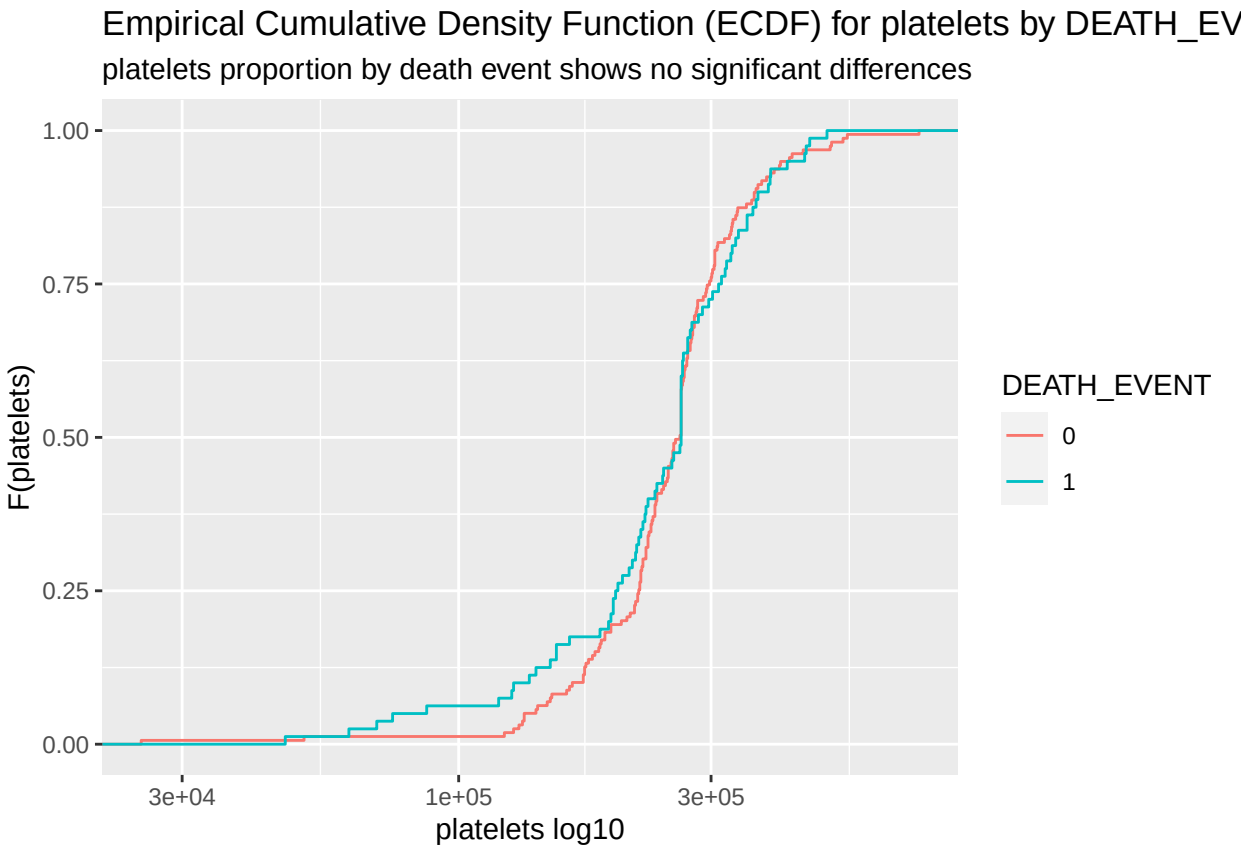
From the ECDF chart above, it shows 28% delta with 52% Death and 24% survival for ejection_fraction value at 30.



[chart 11] Distribution of platelets by Death Event.

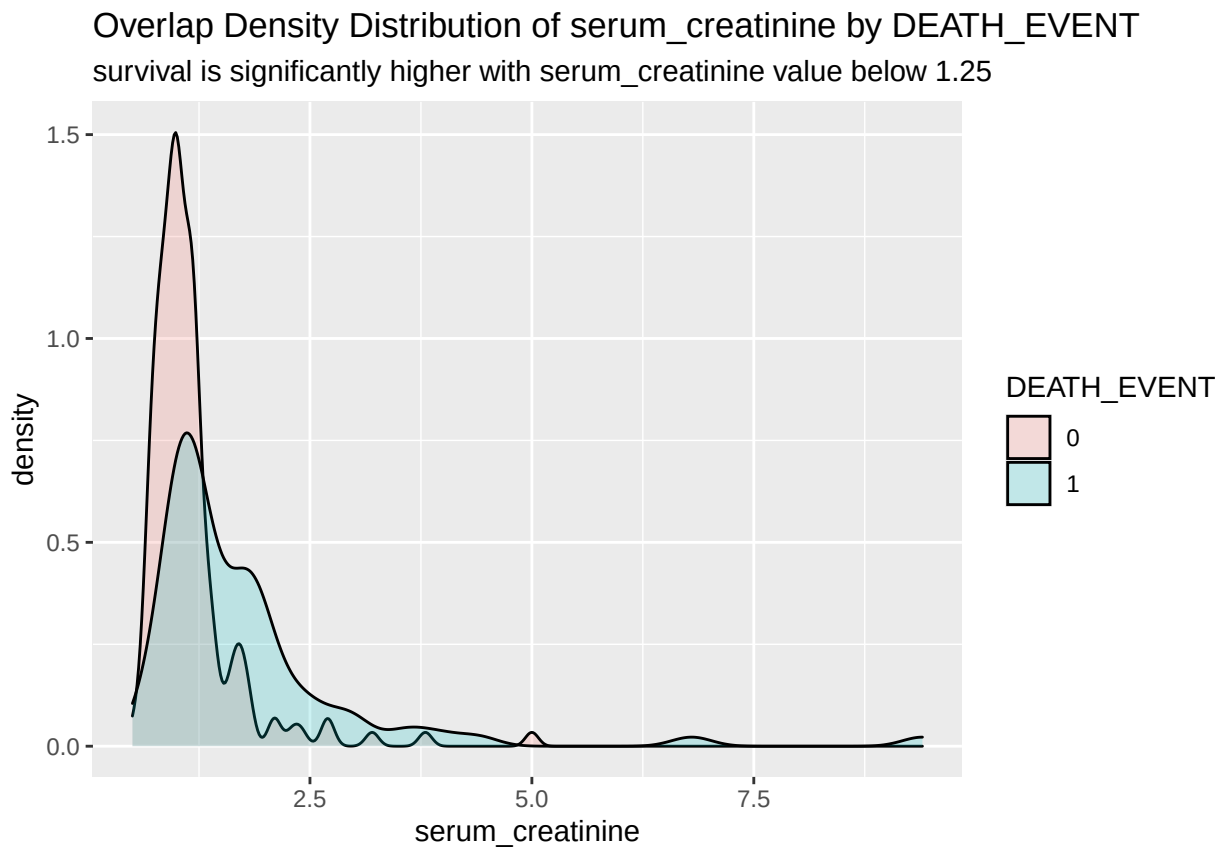
There might be higher survival in the bell shape of platelets values.

Let's explore further details on platelets by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).



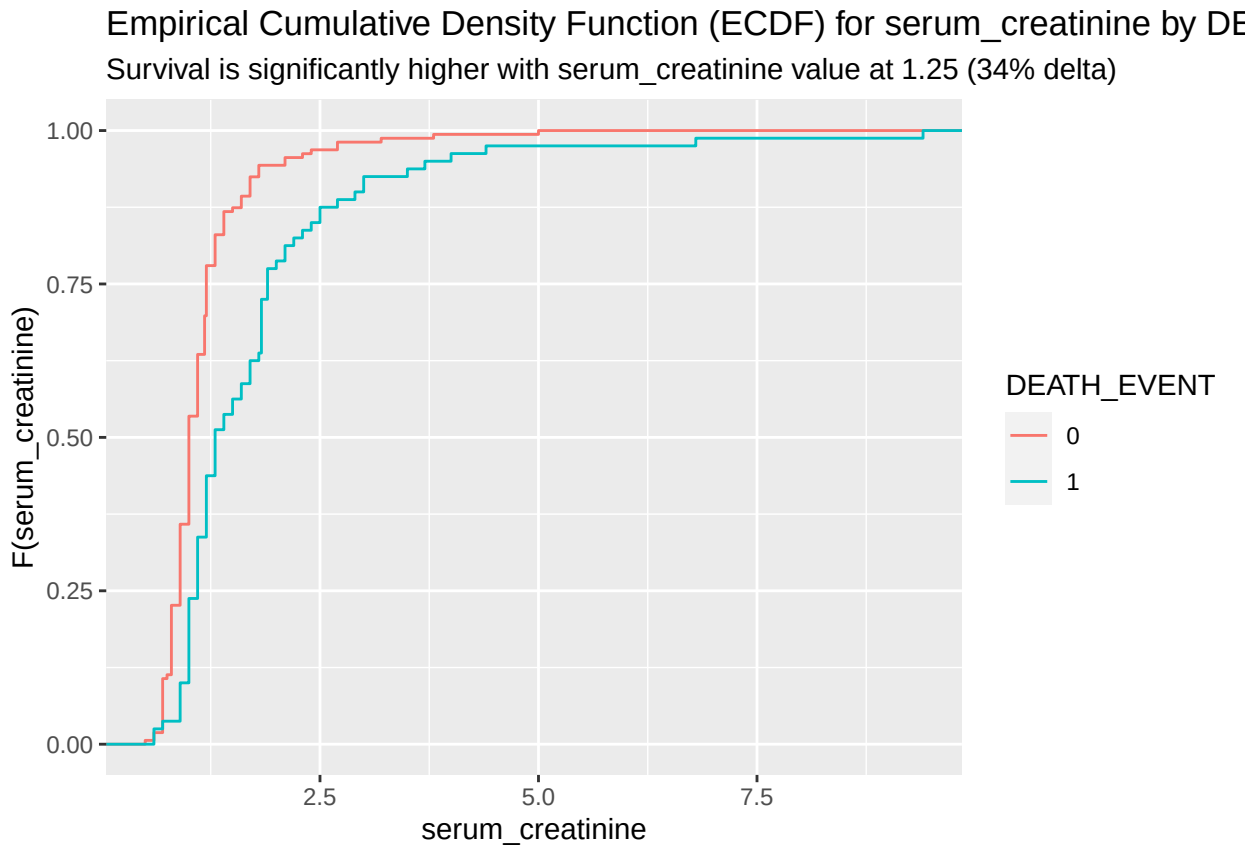
[chart 12] Platelets by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).

From the ECDF chart above, we can see that platelets proportion by death event shows no significant differences.



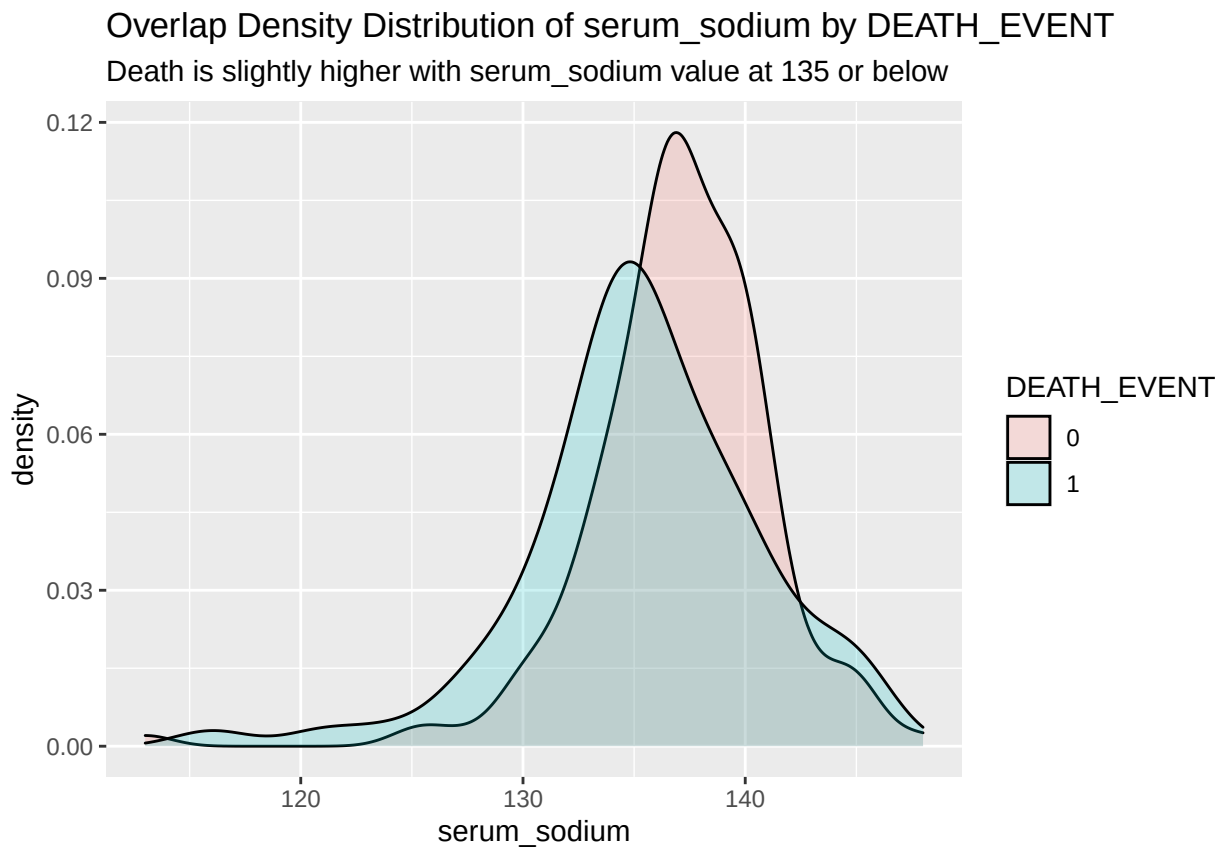
[chart 13] Distribution of serum_creatinine by Death Event

From the chart above, survival is significantly higher with serum_creatinine value at 1.25 or below.



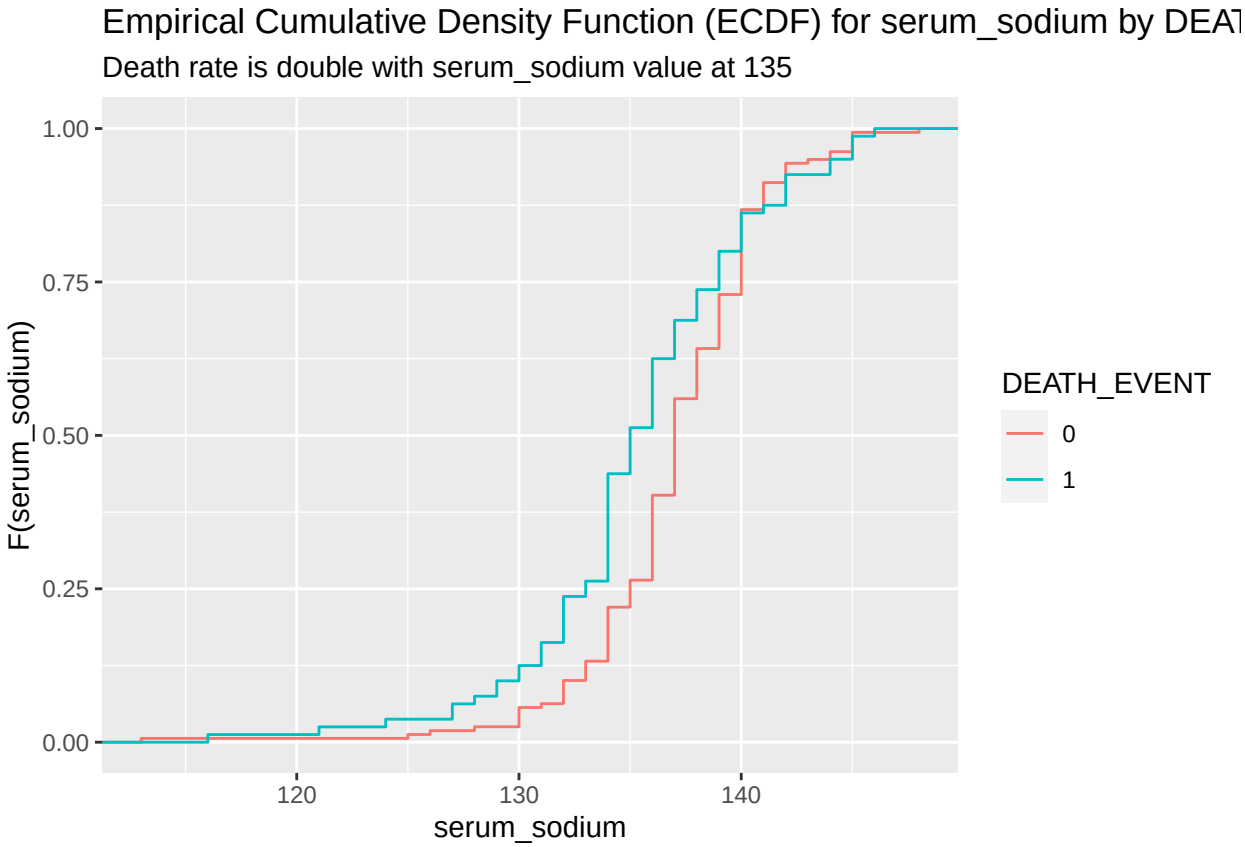
[chart 14] Serum_creatinine by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).

From the ECDF chart above, the survival rate is almost double at 34% delta with 44% Death and 78% survival for serum_creatinine value at 1.25.



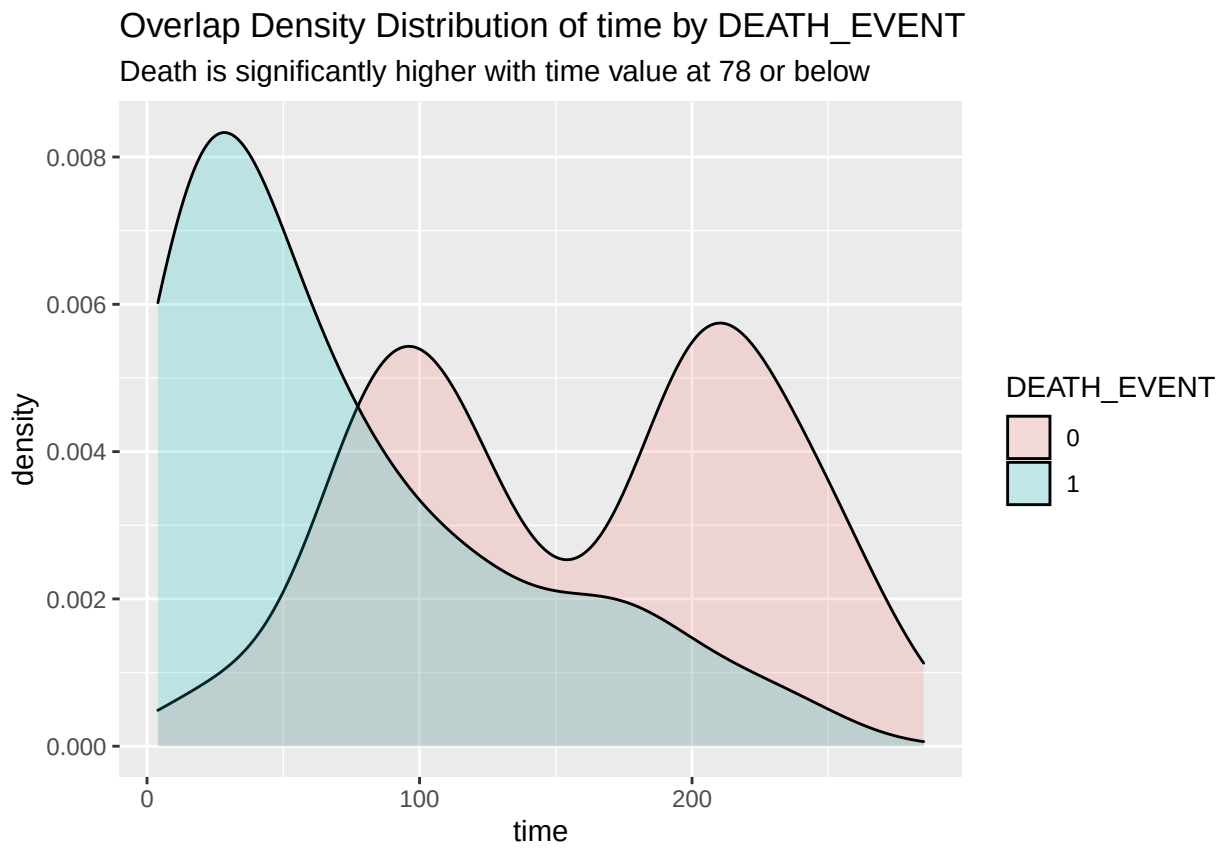
[chart 15] Distribution of serum_sodium by Death Event

From the chart above, survival is significantly higher with serum_sodium value at 135 or below.



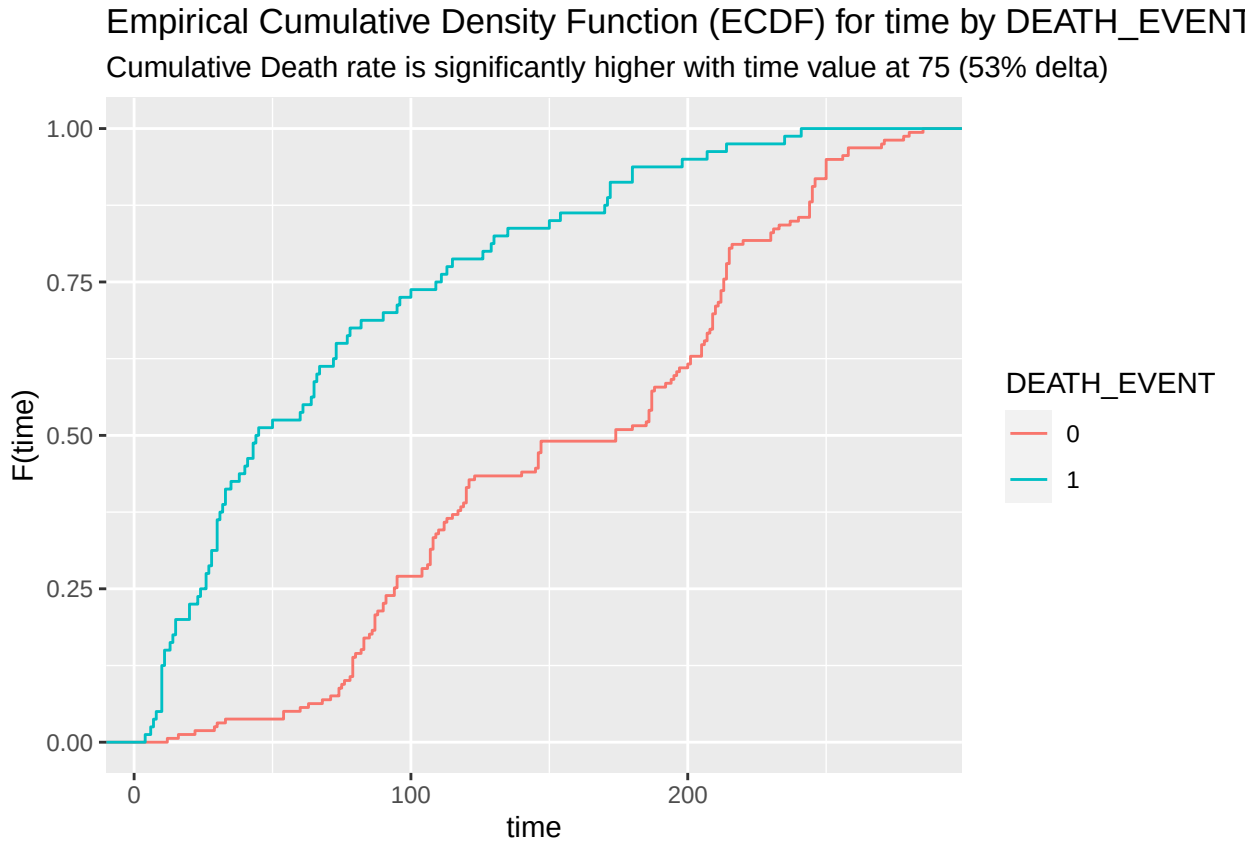
[chart 16] Serum_sodium by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).

From the ECDF chart above, the cumulative death rate is double at 25% delta with 50% Death and 25% survival for serum_sodium value at 135.



[chart 17] Distribution of time by Death Event

From the chart above, Death is significantly higher with time value at 78 or below.



[chart 18] Time by DEATH_EVENT using Empirical Cumulative Density Function (ECDF).

From the ECDF chart above, the cumulative death rate is over 6.8 times at 53% delta with 61% Death and 8% survival for time value at 78.

It also means that with time value above 78, survival rate is more than double with 53% delta for 39% death and 92% survival.

3. Method/Analysis

We are going to train our train_set using the following methods/algorithms as these models are the most appropriate and most popular to be used for classifier models:

1. Logistic Regression / Generalized Linear Model (glm)

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences. For example, the Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patients, was originally developed by Boyd et al. using logistic regression. Many other medical scales used to assess severity of a patient have been developed using logistic regression. Logistic regression may be used to predict the risk of developing a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.). [9]

2. Linear Discriminant Analysis (lda)

Linear discriminant analysis (LDA), normal discriminant analysis (NDA), or discriminant function analysis is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. LDA is closely related to analysis of variance (ANOVA) and regression analysis, which also attempt to express one dependent variable as a linear combination of other features or measurements. LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. [10]

3. Quadratic Discriminant Analysis (qda)

Quadratic discriminant analysis (QDA) is closely related to linear discriminant analysis (LDA), where it is assumed that the measurements from each class are normally distributed.[1] Unlike LDA however, in QDA there is no assumption that the covariance of each of the classes is identical.[2] When the normality assumption is true, the best possible test for the hypothesis that a given measurement is from a given class is the likelihood ratio test. [11]

4. Random Forest (rf)

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance. [12]

5. Decision Tree (rpart)

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning. [13]

6. Support Vector Machine (svmLinear2)

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting).

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. [14]

7. Ensemble

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives. [15]

Our Ensemble model is constructed using all the seven models above that are used in this heart failure prediction analysis.

First, we are going to train all our classifier models using our train_set dataset.
Then, we are going to test all our trained models using our test_set.
And finally, we are going to cross-validate our trained models using our validation_set as our final results.

Let's start with adding the following library for our models

```
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")

library(randomForest)
library(e1071)
```

Let's start with our first model

Logistic Regression / Generalized Linear Model (glm)

For this Generalized Linear Model, we get an accuracy of

```
## [1] 0.8333333
```

Let's check our confusion matrix detail for our logistic regression / generalized linear model:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 19  3
##          1  2  6
##
```

```
##                Accuracy : 0.8333
##                  95% CI : (0.6528, 0.9436)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.07659
##
##                   Kappa : 0.5902
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.9048
##             Specificity : 0.6667
##          Pos Pred Value : 0.8636
##          Neg Pred Value : 0.7500
##              Prevalence : 0.7000
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7333
##       Balanced Accuracy : 0.7857
##
##        'Positive' Class : 0
##
```

To create our classifier models more efficiently, we are going to use caret package with lapply and sapply functions.

Here is a list of accuracy of our each classifier model:

```
##        glm        lda        qda         rf      rpart svmLinear2
##  0.8333333  0.8000000  0.7000000  0.8666667  0.8666667  0.8333333
```

Based on the results of testing our trained classifier models on the test_set, it looks quite promising as we have our test accuracy within acceptable range.

Let's look at the average of our model accuracy:

```
## [1] 0.8166667
```

Let's calculate our ensemble model accuracy:

```
## [1] 0.8
```

Since the accuracy of our classifier models on the test_set are quite promising and acceptable, we will go ahead to cross-validate our trained classifier models with our validation set as our final results.

4. Results

Here is the accuracy results of our each classifier model cross-validated using validation_set:

```
##        glm        lda        qda         rf      rpart svmLinear2
##  0.9666667  0.9666667  0.9000000  0.9666667  0.9333333  0.9666667
```

Based on the accuracy results of our trained classifier models on the validation_set, it looks very promising.

Let's look at the average of our classifier model accuracy on the validation_set:

```
## [1] 0.95
```

Let's calculate our final ensemble model:

```
## [1] 0.9666667
```

Here is the accuracy summary of our final classifier models:

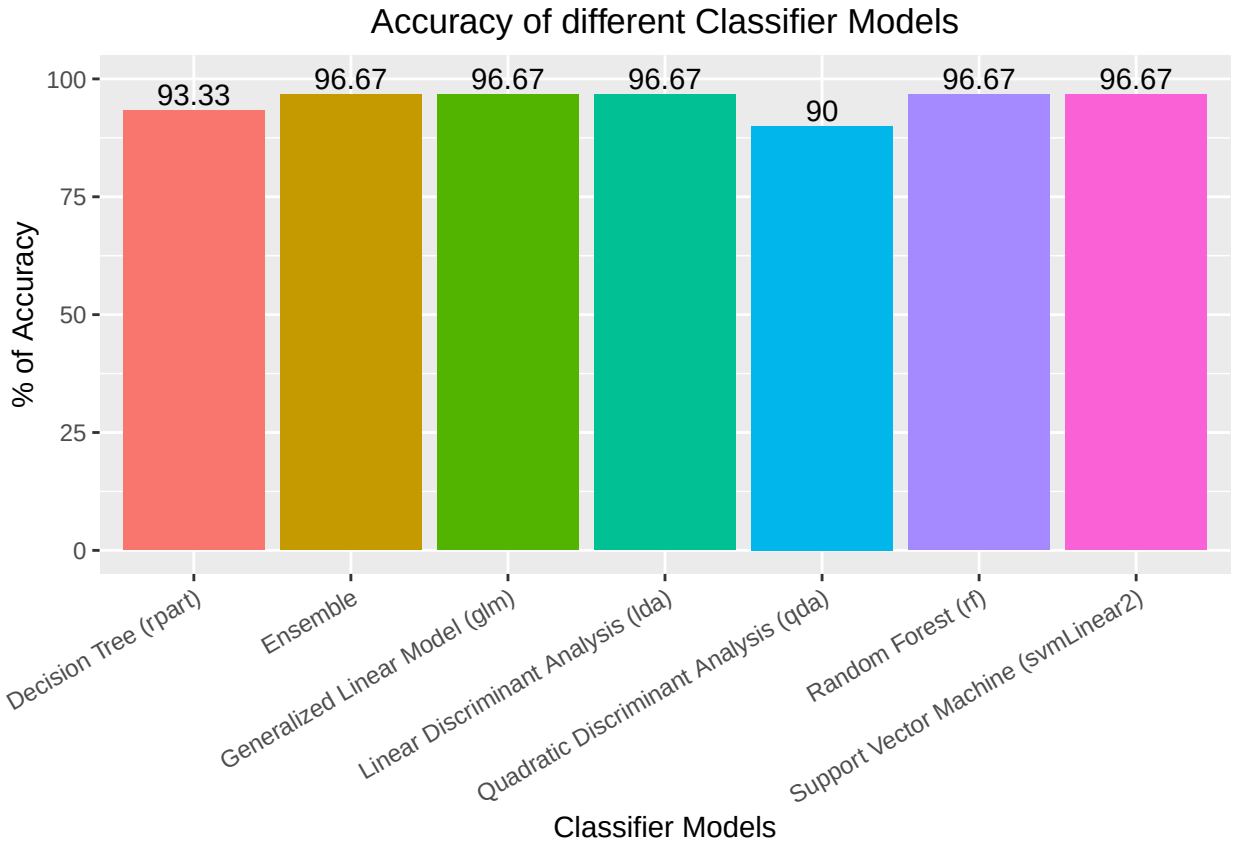
```
## # A tibble: 7 x 2
##   method                                accuracy
##   <chr>                                    <dbl>
## 1 Decision Tree (rpart)                   0.9333
## 2 Ensemble                                0.9667
## 3 Generalized Linear Model (glm)          0.9667
## 4 Linear Discriminant Analysis (lda)      0.9667
## 5 Quadratic Discriminant Analysis (qda)   0.9
## 6 Random Forest (rf)                      0.9667
## 7 Support Vector Machine (svmLinear2)     0.9667
```

Based on the accuracy percentage above, our best final classifier models are
Logistic Regression / Generalized Linear Model (glm)
Linear Discriminant Analysis (lda)
Random Forest (rf)
Support Vector Machine (svmLinear2)
Ensemble

Based on the accuracy percentage above, our worst final classifier model is
Quadratic Discriminant Analysis (qda)

5. Conclusion

Here is the accuracy chart of our final classifier models:



Here is the accuracy summary of our final classifier models:

```
## # A tibble: 7 x 2
##   method                                accuracy
##   <chr>                                    <dbl>
## 1 Decision Tree (rpart)                   0.9333
## 2 Ensemble                                0.9667
## 3 Generalized Linear Model (glm)          0.9667
## 4 Linear Discriminant Analysis (lda)      0.9667
## 5 Quadratic Discriminant Analysis (qda)   0.9
## 6 Random Forest (rf)                      0.9667
## 7 Support Vector Machine (svmLinear2)     0.9667
```

Based on our analysis and results of our final classifier models, we can conclude that our predictive models have very high accuracy to assess the likelihood of a death by heart failure event.
This can be used to help hospitals in assessing the severity of patients with cardiovascular diseases.

Limitations

1. We were only given a small sample of dataset (299 rows and 13 columns, including the result, DEATH_EVENT) to perform our analysis. Out of 299 rows, we split it up to 80% for training set, and 10% for each test_set and validation_set.

A larger dataset would have permitted us to obtain more reliable results. Additional information about the physical features of the patients (height, weight, body mass index, etc.) and their occupational history would have been useful to detect additional risk factors for cardiovascular health diseases. Also, if an additional external dataset with the same features from a different geographical region had been available, we would have used it as a validation cohort to verify our findings.[4]

2. Some machine learning algorithms are computationally expensive to run in a commodity laptop. The required amount of memory to compute far exceeded the available memory in a commodity laptop, even with increased virtual memory.
3. Only twelve predictors are used, not considering other criteria. Modern classifier models may use many predictors.
4. The model works only for existing patients, so the algorithm must run every time a new patient is included. This is not an issue for small client base, but may become a concern for large data sets. The model should consider these changes and update the predictions as information changes.

Future Work

This report briefly describes most appropriate and popular classifier models that predict DEATH_EVENT. There are many more widely adopted approaches not discussed here, such as: k-Nearest Neighbors (knn), SVM with radial/polynomial kernel, Chi-Square Test, Gradient Descent, Neural Network, Bagged models, Bayesian models, Boosting models, Fuzzy models, Gaussian models, Multi-layer models, Quantile models, Penalized models, Robust models, Regularized models, etc.

Regarding future developments, we can apply our machine learning approach to alternative datasets of cardiovascular heart diseases and other illnesses such as chronic lower respiratory diseases, stroke, alzheimer's disease, diabetes, kidney disease, pneumonia and influenza, as well as many different kind of cancers such as carcinoma, sarcoma, melanoma, lymphoma, leukemia, cervical cancer, breast cancer, lung cancer, skin cancer, etc.

Besides the methods/models used in this report, there are other packages/methods/models available in The Comprehensive R Archive Network (CRAN) website and/or caret package.

References

[1] Rafael A. Irizarry (2020), Introduction to Data Science: Data Analysis and Prediction Algorithms with R

[2] https://www.edx.org/professional-certificate/harvardx-data-science

[3] https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

[4] 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). [Web Link] (https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5)

[5] The current version of the dataset was elaborated by Davide Chicco (Krembil Research Institute, Toronto, Canada) and donated to the University of California Irvine Machine Learning Repository under the same Attribution 4.0 International (CC BY 4.0) copyright in January 2020. Davide Chicco can be reached at <davidechicco '@' davidechicco.it>

[6] https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records

[7] https://archive.ics.uci.edu/ml/machine-learning-databases/00519/heart_failure_clinical_records_dataset.csv

[8] https://cran.r-project.org/web/packages/available_packages_by_name.html

[9] https://en.wikipedia.org/wiki/Logistic_regression

[10] https://en.wikipedia.org/wiki/Linear_discriminant_analysis

[11] https://en.wikipedia.org/wiki/Quadratic_classifier#Quadratic_discriminant_analysis

[12] https://en.wikipedia.org/wiki/Random_forest

[13] https://en.wikipedia.org/wiki/Decision_tree

[14] https://en.wikipedia.org/wiki/Support_vector_machine

[15] https://en.wikipedia.org/wiki/Ensemble_learning