FINAL-PROJECT

GROUP - 9

2023-05-02

Importing of recquired libraries for the visualization and interpretation.

```
library(dplyr)
library(tidyr)
library(tidyverse)
library(leaps)
library(knitr)
library(ggplot2)
library("reshape2")
library(caret)
library(psych)
library(tree)
library(rpart)
library(rattle)
library(randomForest)
library(lattice)
library(plotly)
library(corrplot)
library(car)
library(skimr)
library(ggsci)
library(moments)
library(randomForestSRC)
library(gridExtra)
library(class)
library(car)
```

Importing the dataset from the current directory.

```
Heart_disease = read.csv("/Users/vishaypaka/Documents/STAT-515/Datasets/heart_failure_clinical_records_d
```

PREPROCESSING OF THE DATASET

```
table(is.na(Heart_disease)) # no null values
```

```
##
## FALSE
##  3887
```

We can see that our dataset doesn't contain any null values. Now let's observe what datatype are our predictor variables and if there are any categorical or binary variables, let's factor them.

Structure of the dataset

```
#Exploring the dataset
str(Heart_disease) #checking the datatypes of variables present
```

```
## 'data.frame':    299 obs. of  13 variables:
##  $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
##  $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
##  $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
##  $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
##  $ platelets               : num  265000 263358 162000 210000 327000 ...
##  $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
##  $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
##  $ sex                     : int  1 1 1 1 0 1 1 1 0 1 ...
##  $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
##  $ time                    : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

Data pre-processing

```
table(Heart_disease$platelets)
```

```
##
##    25100    47000    51000    62000    70000    73000    75000    87000
##        1        1        1        1        1        1        1        1
##   105000   119000   122000   126000   127000   130000   132000   133000
##        1        1        1        1        2        1        1        2
##   136000   140000   141000   147000   149000   150000   151000   153000
##        1        2        1        2        3        1        1        3
##   155000   160000   162000   164000   166000   172000   173000   174000
##        1        1        2        1        2        2        2        1
##   176000   179000   181000   184000   185000   186000   188000   189000
##        1        1        1        1        2        1        1        3
##   192000   194000   196000   198000    2e+05   201000   203000   204000
##        1        3        2        1        1        1        3        2
##   208000   210000   211000   212000   213000   215000   216000   217000
##        1        3        1        1        1        2        2        1
##   218000   219000   220000   221000   222000   223000   224000   225000
##        2        2        3        4        2        3        1        1
##   226000   227000   228000   229000   231000   232000   233000   235000
##        4        1        4        1        2        1        1        4
##   236000   237000   241000   242000   243000   244000   246000   248000
##        1        4        1        2        1        3        1        1
##   249000   250000   252000   253000   254000   255000   257000   259000
##        3        1        1        2        3        4        1        1
##    260000    262000    263000 263358.03    264000    265000    266000    267000 
##         1         2         1        25         1         3         2         2 
##    268000    270000    271000    274000    275000    276000    277000    279000 
##         1         2         4         3         1         2         2         4 
##    281000    282000    283000    284000    286000    289000    290000    293000 
##         1         1         3         1         1         1         1         1 
##    294000    295000    297000    298000     3e+05    301000    302000    303000 
##         1         1         2         1         1         1         3         1 
##    304000    305000    306000    308000    309000    310000    314000    317000 
##         2         4         1         1         1         1         1         1 
##    318000    319000    321000    324000    325000    327000    328000    329000 
##         1         2         1         1         1         3         1         2 
##    330000    334000    336000    337000    338000    348000    350000    351000 
##         1         2         1         1         1         1         1         2 
##    358000    360000    362000    365000    368000    371000    374000    377000 
##         1         1         3         2         2         1         1         1 
##    382000    385000    388000    389000    390000    395000    404000    406000 
##         1         1         1         2         2         2         1         2 
##    418000    422000    427000    448000    451000    454000    461000    481000 
##         1         1         1         1         2         1         1         1 
##    497000    504000    507000    533000    543000    621000    742000    850000 
##         1         1         1         1         1         1         1         1 
```

```
Heart_disease$platelets <- round(Heart_disease$platelets)

#Formatting the scientific notations for the platelets variable.
Heart_disease$platelets <- format(Heart_disease$platelets, scientific = FALSE)

table(Heart_disease$platelets)
```

```
## 
##  25100  47000  51000  62000  70000  73000  75000  87000 105000 119000 122000 
##      1      1      1      1      1      1      1      1      1      1      1 
## 126000 127000 130000 132000 133000 136000 140000 141000 147000 149000 150000 
##      1      2      1      1      2      1      2      1      2      3      1 
## 151000 153000 155000 160000 162000 164000 166000 172000 173000 174000 176000 
##      1      3      1      1      2      1      2      2      2      1      1 
## 179000 181000 184000 185000 186000 188000 189000 192000 194000 196000 198000 
##      1      1      1      2      1      1      3      1      3      2      1 
## 200000 201000 203000 204000 208000 210000 211000 212000 213000 215000 216000 
##      1      1      3      2      1      3      1      1      1      2      2 
## 217000 218000 219000 220000 221000 222000 223000 224000 225000 226000 227000 
##      1      2      2      3      4      2      3      1      1      4      1 
## 228000 229000 231000 232000 233000 235000 236000 237000 241000 242000 243000 
##      4      1      2      1      1      4      1      4      1      2      1 
## 244000 246000 248000 249000 250000 252000 253000 254000 255000 257000 259000 
##      3      1      1      3      1      1      2      3      4      1      1 
## 260000 262000 263000 263358 264000 265000 266000 267000 268000 270000 271000 
##      1      2      1     25      1      3      2      2      1      2      4 
## 274000 275000 276000 277000 279000 281000 282000 283000 284000 286000 289000 
##      3      1      2      2      4      1      1      3      1      1      1 
## 290000 293000 294000 295000 297000 298000 300000 301000 302000 303000 304000 
##      1      1      1      1      2      1      1      1      3      1      2 
## 305000 306000 308000 309000 310000 314000 317000 318000 319000 321000 324000 
##      4      1      1      1      1      1      1      1      2      1      1
## 325000 327000 328000 329000 330000 334000 336000 337000 338000 348000 350000
##      1      3      1      2      1      2      1      1      1      1      1
## 351000 358000 360000 362000 365000 368000 371000 374000 377000 382000 385000
##      2      1      1      3      2      2      1      1      1      1      1
## 388000 389000 390000 395000 404000 406000 418000 422000 427000 448000 451000
##      1      2      2      2      1      2      1      1      1      1      2
## 454000 461000 481000 497000 504000 507000 533000 543000 621000 742000 850000
##      1      1      1      1      1      1      1      1      1      1      1
```

```
#rounding off the value of an age variable.
table(Heart_disease$age)
```

```
## 
##     40     41     42     43     44     45     46     47     48     49     50 
##      7      1      7      1      2     19      3      1      2      4     27 
##     51     52     53     54     55     56     57     58     59     60 60.667 
##      4      5     10      2     17      1      2     10      4     33      2 
##     61     62     63     64     65     66     67     68     69     70     72 
##      4      5      8      3     26      2      2      5      3     25      7 
##     73     75     77     78     79     80     81     82     85     86     87 
##      4     11      2      2      1      7      1      3      6      1      1 
##     90     94     95 
##      3      1      2
```

```
Heart_disease$age <- round(Heart_disease$age)

table(Heart_disease$age)
```

```
## 
## 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 
##  7  1  7  1  2 19  3  1  2  4 27  4  5 10  2 17  1  2 10  4 33  6  5  8  3 26 
## 66 67 68 69 70 72 73 75 77 78 79 80 81 82 85 86 87 90 94 95 
##  2  2  5  3 25  7  4 11  2  2  1  7  1  3  6  1  1  3  1  2
```

Here when we observe the column of age one observation as age 66.67, which stands out of the rest, so rounding off as we have less observations for prediction. Also platelets column has scientific notation for two records. So, formatting it into numeric as an pre-processing step.

Let's factorize the binary variables and convert continuous to numeric.

```
Heart_disease$anaemia = as.factor(Heart_disease$anaemia)
Heart_disease$diabetes = as.factor(Heart_disease$diabetes)
Heart_disease$high_blood_pressure = as.factor(Heart_disease$high_blood_pressure)
Heart_disease$sex = as.factor(Heart_disease$sex)
Heart_disease$smoking = as.factor(Heart_disease$smoking)
Heart_disease$DEATH_EVENT = as.factor(Heart_disease$DEATH_EVENT)
Heart_disease$creatinine_phosphokinase = as.numeric(Heart_disease$creatinine_phosphokinase)
Heart_disease$ejection_fraction = as.numeric(Heart_disease$ejection_fraction)
Heart_disease$serum_sodium = as.numeric(Heart_disease$ejection_fraction)
Heart_disease$platelets = as.numeric(Heart_disease$platelets)
str(Heart_disease)
```

```
## 'data.frame':    299 obs. of  13 variables:
##  $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ anaemia                 : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 2 1 2 ...
##  $ creatinine_phosphokinase: num  582 7861 146 111 160 ...
##  $ diabetes                : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
##  $ ejection_fraction       : num  20 38 20 20 20 40 15 60 65 35 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 1 1 2 ...
##  $ platelets               : num  265000 263358 162000 210000 327000 ...
##  $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
##  $ serum_sodium            : num  20 38 20 20 20 40 15 60 65 35 ...
##  $ sex                     : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 1 2 ...
##  $ smoking                 : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 1 2 ...
##  $ time                    : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT             : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Producing some numerical and graphical summaries of the data set.

Statistical Analysis

```
summary(Heart_disease) #summary of the dataset
```

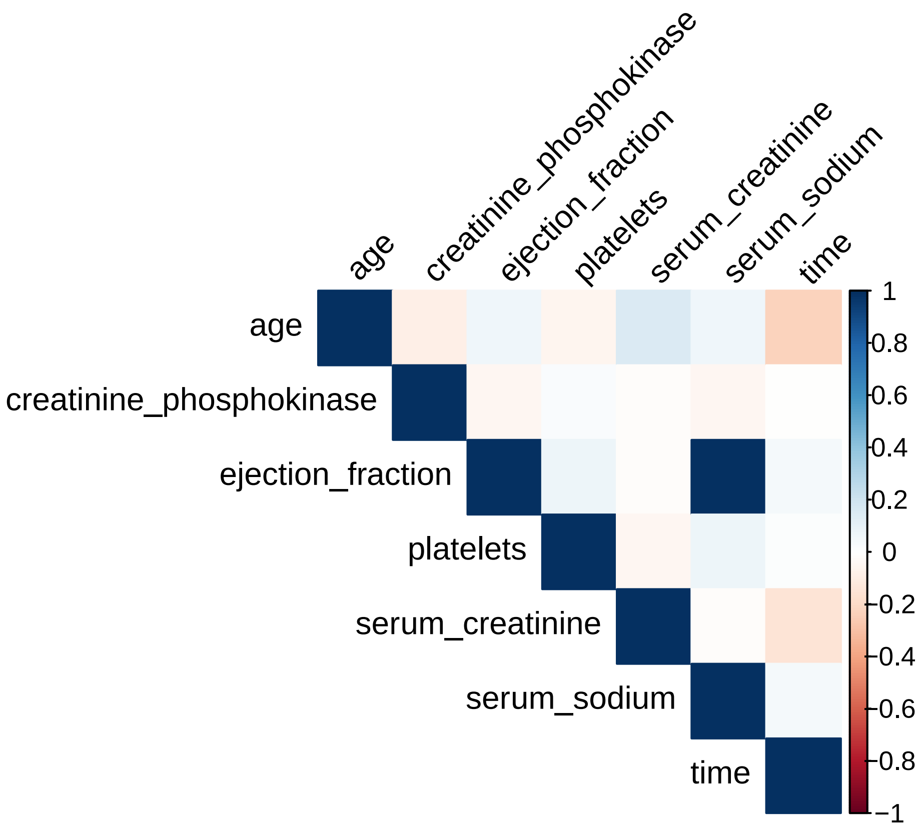
```
##       age        anaemia creatinine_phosphokinase diabetes ejection_fraction
##  Min.   :40.00   0:170   Min.   :  23.0           0:174    Min.   :14.00    
##  1st Qu.:51.00   1:129   1st Qu.: 116.5           1:125    1st Qu.:30.00    
##  Median :60.00           Median : 250.0                    Median :38.00    
##  Mean   :60.84           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.   :14.00   0:105  
##  1:105               1st Qu.:212500   1st Qu.:0.900    1st Qu.:30.00   1:194  
##                      Median :262000   Median :1.100    Median :38.00          
##                      Mean   :263358   Mean   :1.394    Mean   :38.08          
##                      3rd Qu.:303500   3rd Qu.:1.400    3rd Qu.:45.00          
##                      Max.   :850000   Max.   :9.400    Max.   :80.00          
##  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              
```

By summary of the datset,we can observe all the variable's mean, median, min and max. We can see there are patients having anaemia are 129 and does not have anaemia are 170. 125 patients have diabetes and 174 doesn't have diabetes 194 patients dose not have high bp and 105 patients have high bp. out of all patients 203 dose not smoke and the rest of them i.e; 96 people will smoke.In our dataset we have 105 patients recorded as female and 194 as of male.The total death events taken place during the follow up period are 96 and the patients who survived are 203.

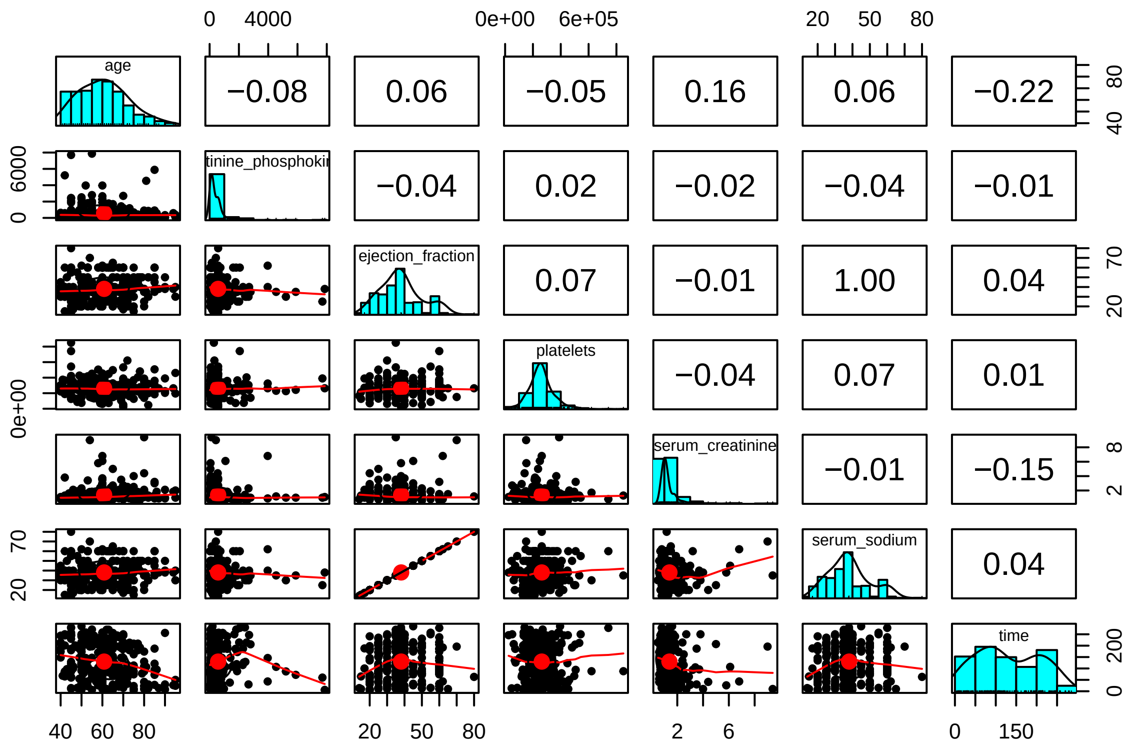
By observing all these statistical summaries of the predictors and response variables. We can conclude that the data set is slightly imbalanced as there are more observation of the patients who survived compare to the people who died because of the heart failure but have less difference of records with patients having anaemia or not, diabetes or not, high blood pressure or not, smoking or not. In general out of 7.2 billion population only less proportionate of people will die with heart failure condition.

Correlation analysis

```
numeric_cols <- Heart_disease %>% select_if(is.numeric)
cor_mat <- cor(numeric_cols)
corrplot(cor_mat, method = "color", type = "upper", tl.col = "black", tl.srt = 45)
```



```
Heart_disease1 = subset(Heart_disease, select = -c(anaemia,high_blood_pressure,sex,smoking,DEATH_EVENT,
pairs.panels(Heart_disease1)
```

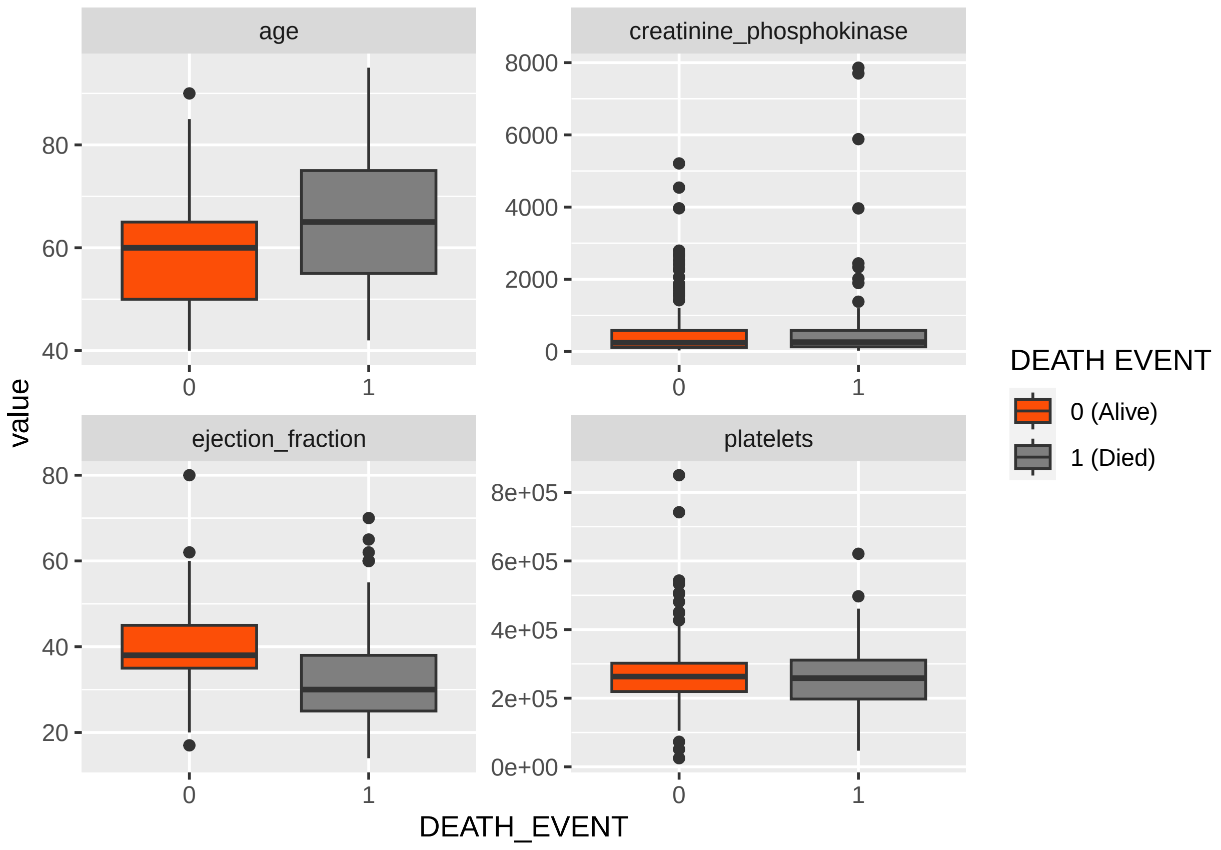


The correlation between serum sodium and ejection fraction is 1.0, which indicates that they are perfectly linearly related and increase or decrease together. Having correlation 1 means they are perfect positive and from the chart above we can see a linear line between serum sodium and ejection fraction. Let's see how it affects our models when fitted later.

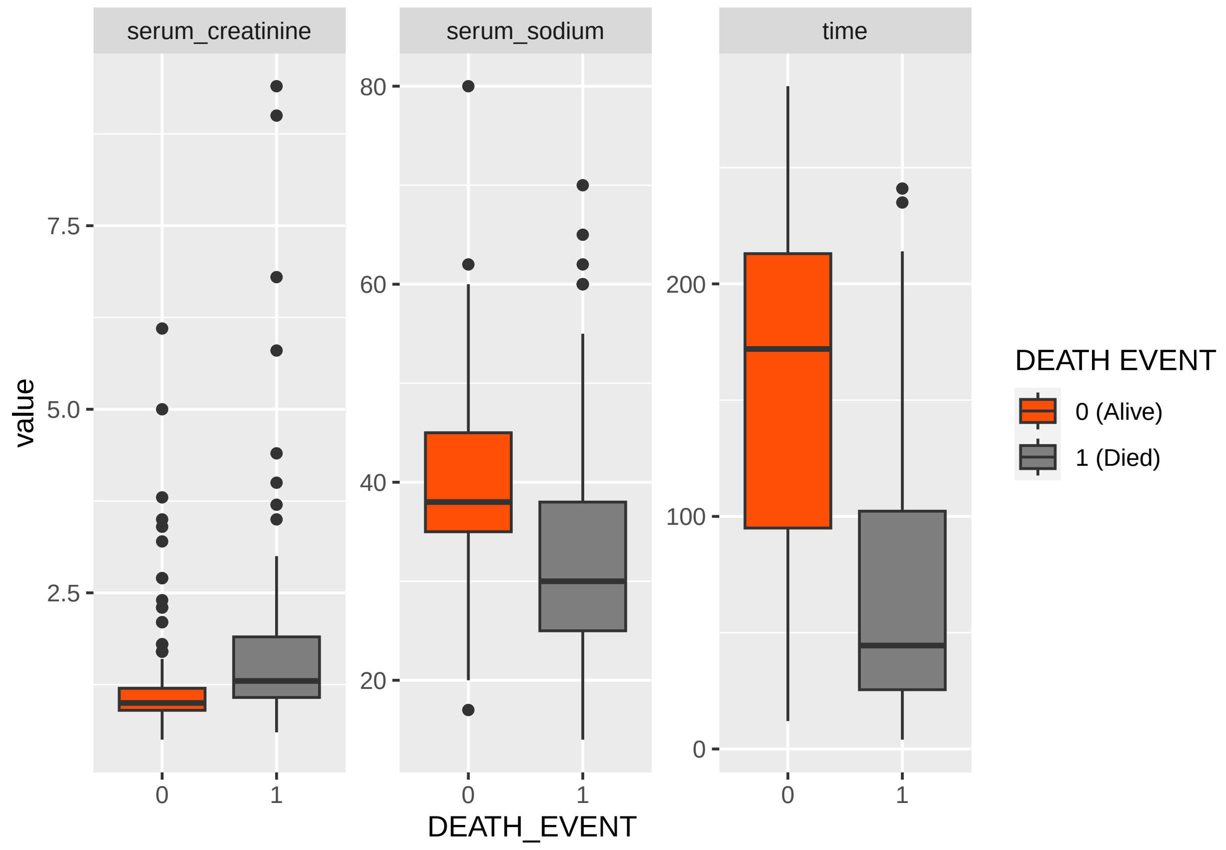
Distribution analysis

RELATIONSHIP OF PREDICTOR VARIABLES WITH DEATH EVENT COLUMN

```r
# Box plot of age, creatinine phosphokinase, ejection fraction, platelets and death event
palette_ro <- c("yellow", "#FC4E07")
df1 <- melt(Heart_disease[,c(1,3,5,7,13)], id.var = "DEATH_EVENT")
ggplot(data = df1, aes(x=DEATH_EVENT, y=value)) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  geom_boxplot(aes(fill=DEATH_EVENT)) + facet_wrap(~variable, scales="free")
```



```
# Box plot of serum creatinine, serum sodium, time and death event
df2 <- melt(Heart_disease[,c(8,9,12,13)], id.var = "DEATH_EVENT")
ggplot(data = df2, aes(x=DEATH_EVENT, y=value)) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  geom_boxplot(aes(fill=DEATH_EVENT)) + facet_wrap(~variable, scales="free")
```



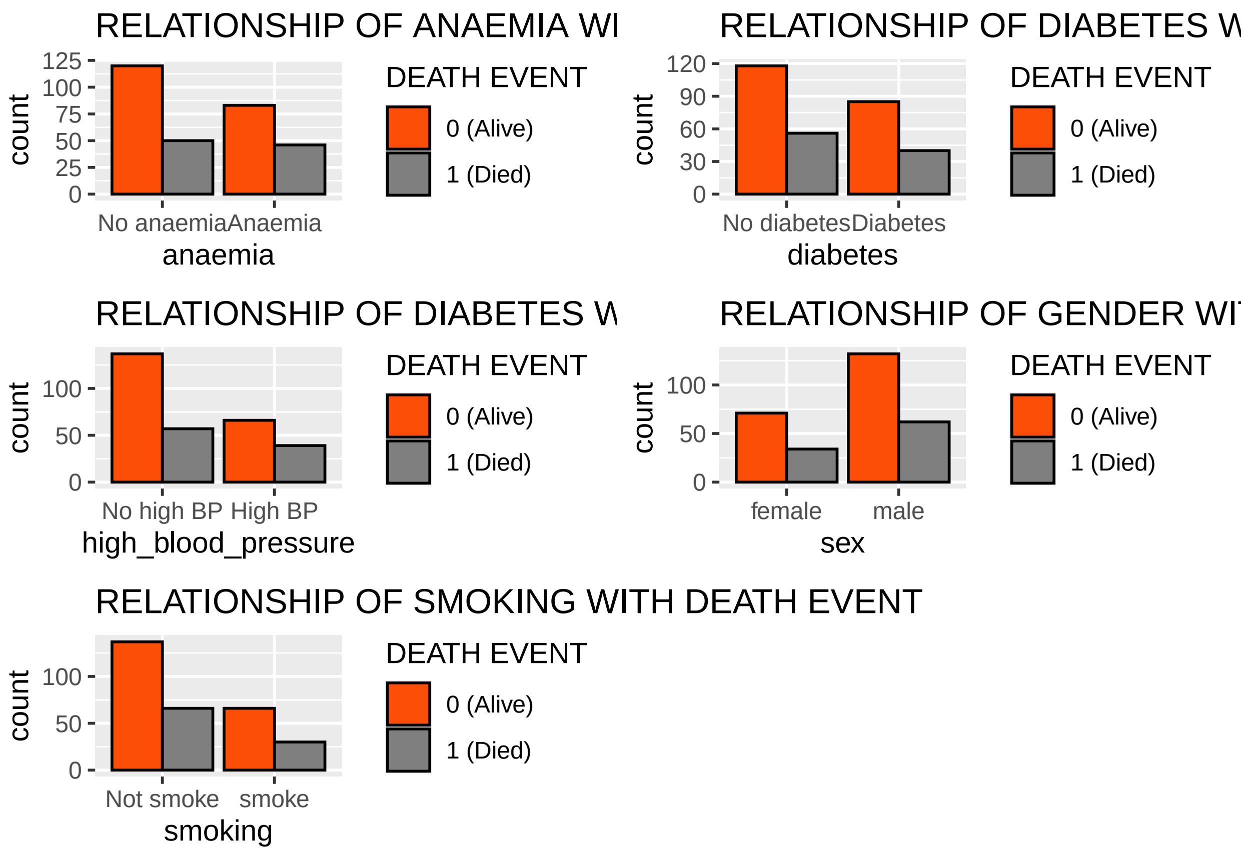
```
p11 = ggplot(data = Heart_disease, aes(x=anaemia)) +
  geom_bar(position = 'dodge', aes(fill=DEATH_EVENT), color = 'black') +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  labs(
    title = 'RELATIONSHIP OF ANAEMIA WITH DEATH EVENT'
  ) +
  scale_x_discrete(labels = c("No anaemia", "Anaemia"))

p22 = ggplot(data = Heart_disease, aes(x=diabetes)) +
  geom_bar(position = 'dodge', aes(fill=DEATH_EVENT), color = 'black') +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  labs(
    title = 'RELATIONSHIP OF DIABETES WITH DEATH EVENT'
  ) +
  scale_x_discrete(labels = c("No diabetes", "Diabetes"))

p33 = ggplot(data = Heart_disease, aes(x=high_blood_pressure)) +
  geom_bar(position = 'dodge', aes(fill=DEATH_EVENT), color = 'black') +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  labs(
    title = 'RELATIONSHIP OF DIABETES WITH DEATH EVENT'
  ) +
  scale_x_discrete(labels = c("No high BP", "High BP"))

p44 = ggplot(data = Heart_disease, aes(x=sex)) +
  geom_bar(position = 'dodge', aes(fill=DEATH_EVENT), color = 'black') +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  labs(
    title = 'RELATIONSHIP OF GENDER WITH DEATH EVENT'
  ) +
  scale_x_discrete(labels = c("female", "male"))

p55 = ggplot(data = Heart_disease, aes(x=smoking)) +
  geom_bar(position = 'dodge', aes(fill=DEATH_EVENT), color = 'black') +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  labs(
    title = 'RELATIONSHIP OF SMOKING WITH DEATH EVENT'
  ) +
  scale_x_discrete(labels = c("Not smoke", "smoke"))
grid.arrange(p11, p22, p33, p44, p55, nrow = 3, ncol = 2)
```



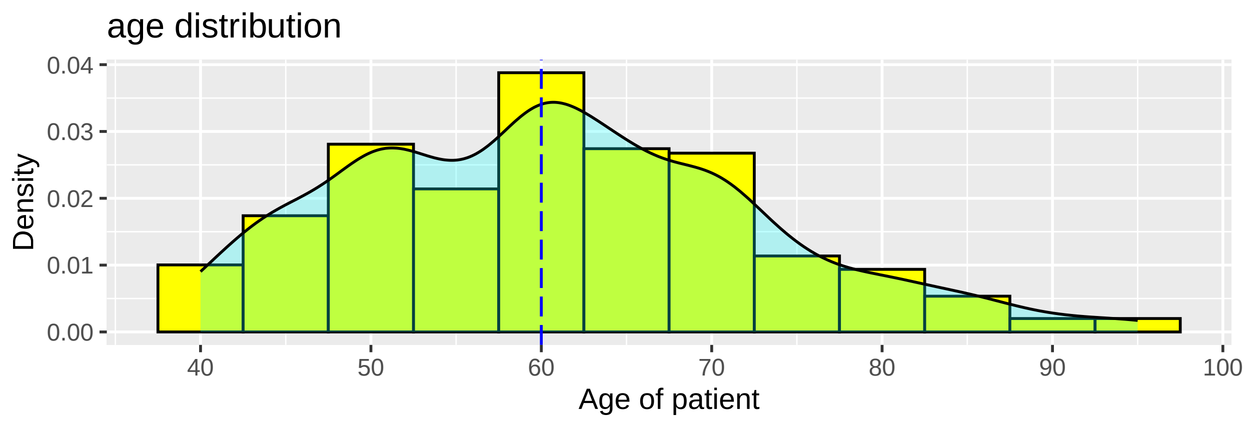
We can observe many As we see in the boxplots creatinine phosphokinase and serum creatinine has high variance and positively skewed. So we should carefully observe them in the histogram and density plots and also calculate the skewness and if necessary scale them.

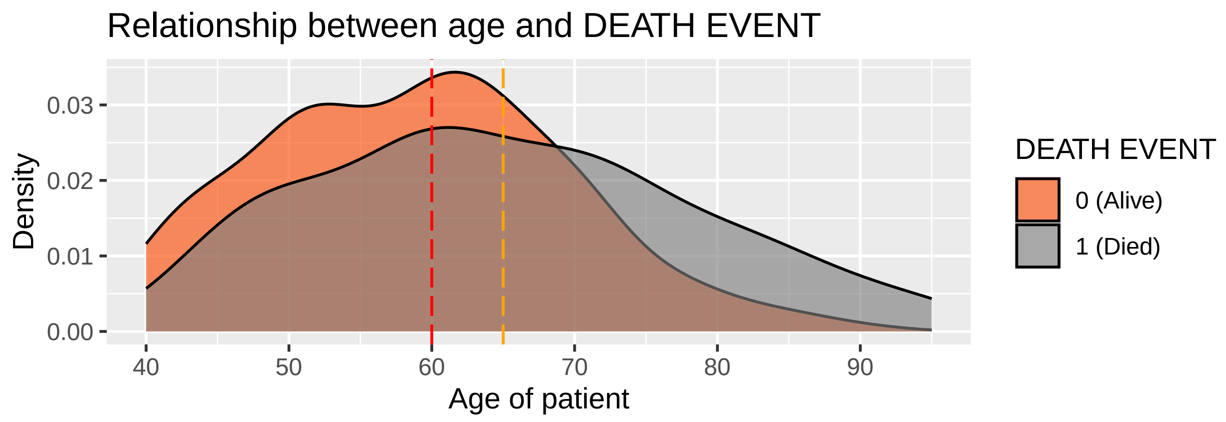
Age distribution

```r
#Histogram - age
palette_ro <- c("yellow", "#FC4E07")
p1 = ggplot(Heart_disease,aes(x=age)) +
  geom_histogram(aes(y = ..density..), binwidth = 5, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.25) +
  scale_x_continuous(breaks = seq(40, 100, 10)) +
  geom_vline(xintercept = median(Heart_disease$age), linetype="longdash", colour = "blue") +
  labs(x="Age of patient",
       y="Density",
       title="age distribution")

p2 = ggplot(Heart_disease, aes(x = age, fill = DEATH_EVENT)) +
  geom_density(aes(age,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  scale_x_continuous(breaks = seq(40, 100, 10)) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$age),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$age),
             linetype="longdash", colour = "orange") +
  labs(x="Age of patient",
       y="Density",
       title="Relationship between age and DEATH EVENT")

grid.arrange(p1, p2, nrow=2)
```





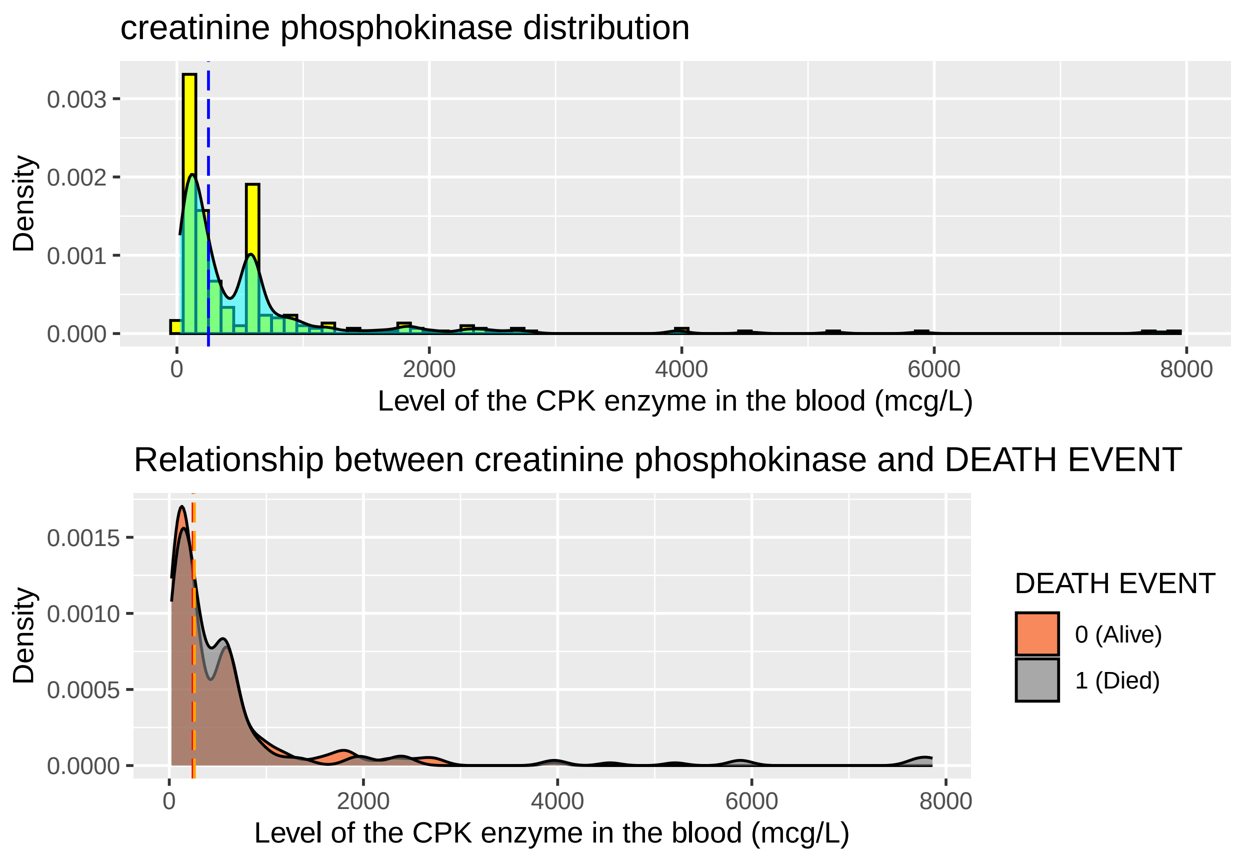
The insights we get from above histogram and density for age distribution are: * The age of the patients was highest around 60 years and you can observe that, younger the age, the density plot of survival is high and as the age increases density plot of death is more. After the age 70 the density plot is reversed.

creatinine phosphokinase distribution

```
#Histogram - creatinine phosphokinase
palette_ro <- c("yellow", "#FC4E07")
p3 = ggplot(Heart_disease,aes(x=creatinine_phosphokinase)) +
  geom_histogram(aes(y = ..density..), binwidth = 100, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.5) +
  geom_vline(xintercept = median(Heart_disease$creatinine_phosphokinase),
             linetype="longdash", colour = "blue") +
  labs(x="Level of the CPK enzyme in the blood (mcg/L)",
       y="Density",
       title="creatinine phosphokinase distribution")

p4 = ggplot(Heart_disease, aes(x = creatinine_phosphokinase, fill = DEATH_EVENT)) +
  geom_density(aes(creatinine_phosphokinase,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$creatinine_phosphokinase),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$creatinine_phosphokinase),
             linetype="longdash", colour = "orange") +
  labs(x="Level of the CPK enzyme in the blood (mcg/L)",
       y="Density",
       title="Relationship between creatinine phosphokinase and DEATH EVENT")

grid.arrange(p3, p4, nrow=2)
```



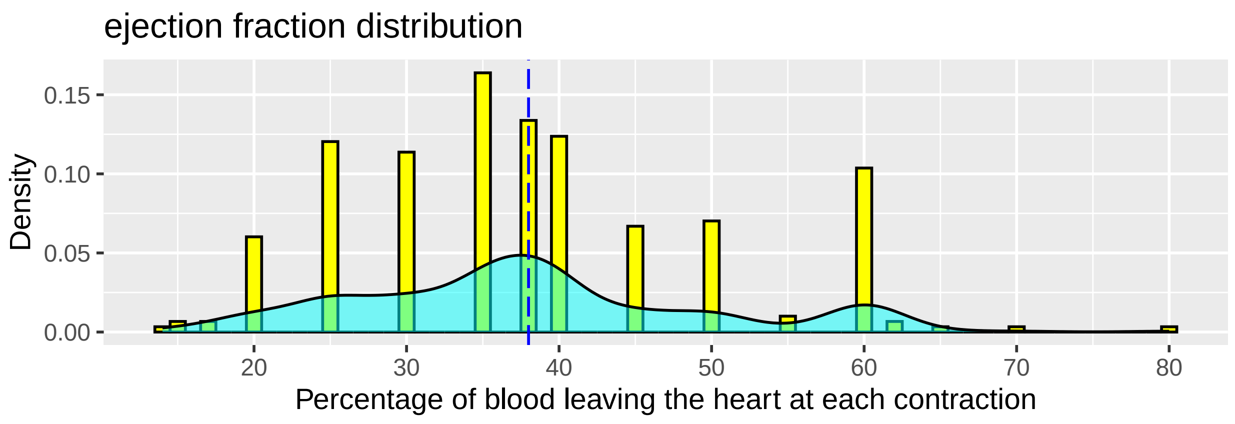
The insights we get from above histogram and density for creatinine phosphokinase distribution are: * The distribution is heavily skewed to one side and we need to calculate the skewness and scale it accordingly.

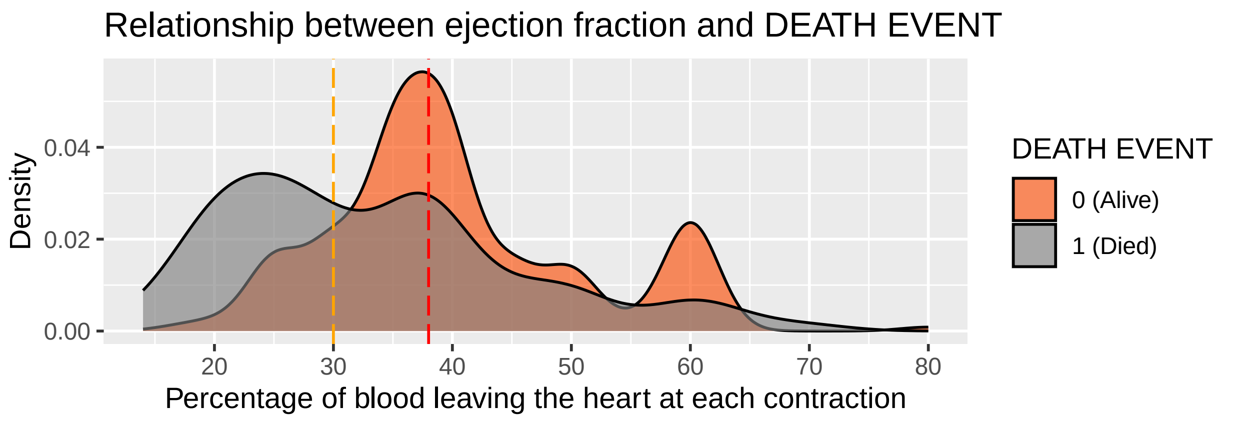
ejection fraction distribution

```
#Histogram - ejection fraction
palette_ro <- c("yellow", "#FC4E07")
p5 = ggplot(Heart_disease,aes(x = ejection_fraction)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.5) +
  geom_vline(xintercept = median(Heart_disease$ejection_fraction), linetype="longdash",
             colour = "blue") +
  scale_x_continuous(breaks = seq(10, 80, 10)) +
  labs(x="Percentage of blood leaving the heart at each contraction",
       y="Density",
       title="ejection fraction distribution")
```

```
p6 = ggplot(Heart_disease, aes(x = ejection_fraction, fill = DEATH_EVENT)) +
  geom_density(aes(ejection_fraction,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$ejection_fraction),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$ejection_fraction),
             linetype="longdash", colour = "orange") +
  scale_x_continuous(breaks = seq(10, 80, 10)) +
  labs(x="Percentage of blood leaving the heart at each contraction",
      y="Density",
      title="Relationship between ejection fraction and DEATH EVENT")

grid.arrange(p5, p6, nrow=2)
```





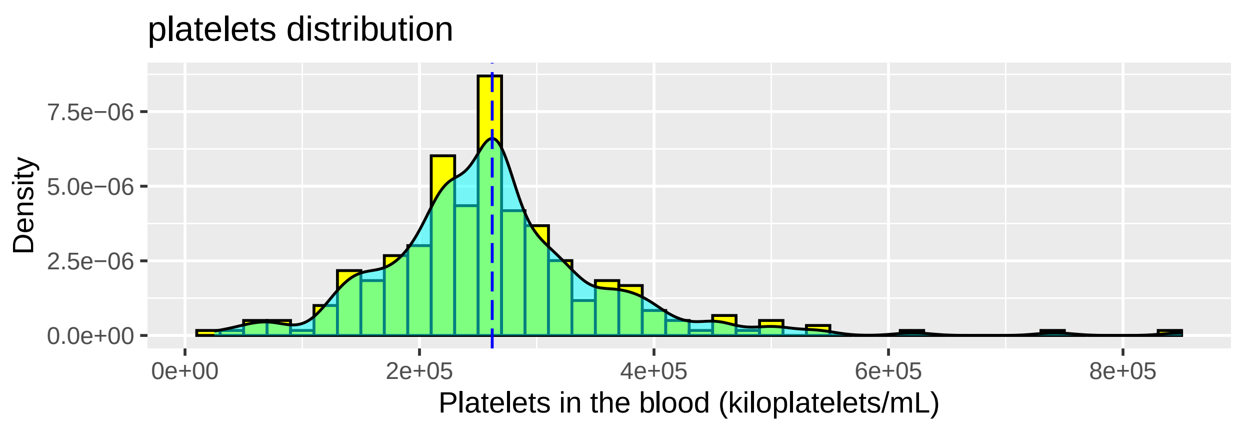
The insights we get from above histogram and density for creatinine phosphokinase distribution are: * The distribution is discrete. The ejection fraction for an normal person will be around 50-60 and if it is higher also, there will be no problem.But, if the ejection fraction falls to 40 and below there is a higher chances of heart failure. The same trend we can observe from the graph.

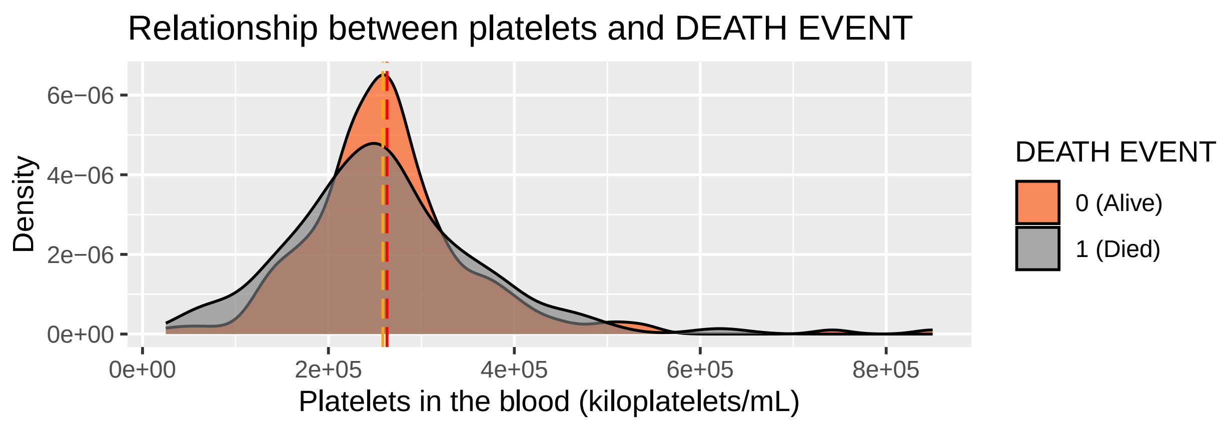
platelets distribution

```
#Histogram - platelets
palette_ro <- c("yellow", "#FC4E07")
p7 = ggplot(Heart_disease,aes(x = platelets)) +
  geom_histogram(aes(y = ..density..), binwidth = 20000, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.5) +
  geom_vline(xintercept = median(Heart_disease$platelets), linetype="longdash", colour = "blue") +
  labs(x="Platelets in the blood (kiloplatelets/mL)",
       y="Density",
       title="platelets distribution")

p8 = ggplot(Heart_disease, aes(x = platelets, fill = DEATH_EVENT)) +
  geom_density(aes(platelets,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$platelets),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$platelets),
             linetype="longdash", colour = "orange") +
  labs(x="Platelets in the blood (kiloplatelets/mL)",
       y="Density",
       title="Relationship between platelets and DEATH EVENT")

grid.arrange(p7, p8, nrow=2)
```





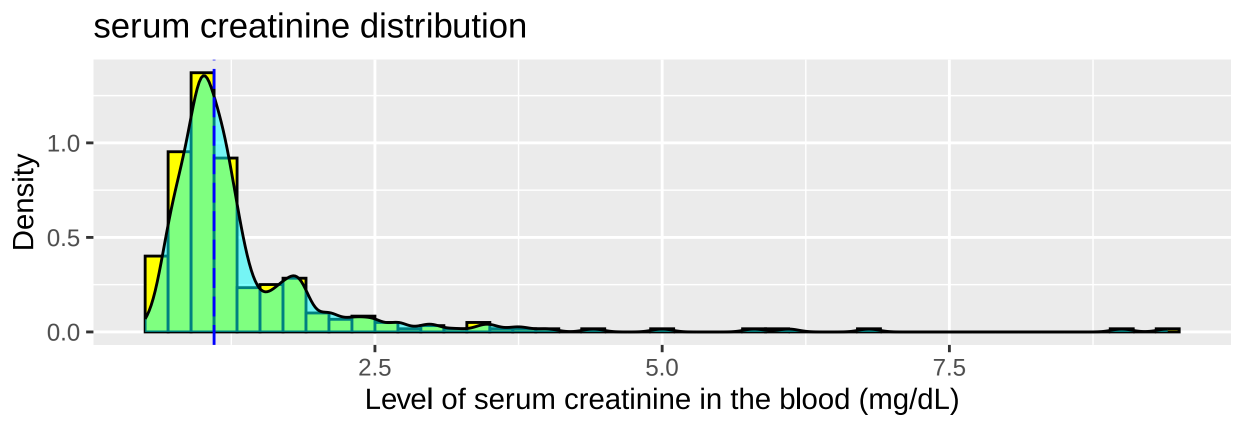
The insights we get from above histogram and density for platelets distribution are: * The distribution is symmetric , survivals have the highest platelets.

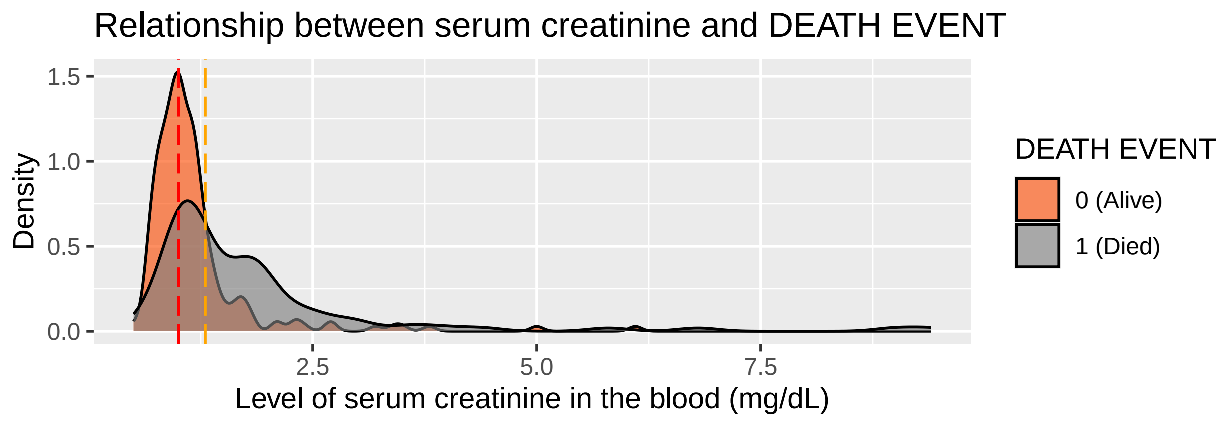
serum creatinine distribution

```r
#Histogram - serum creatinine
palette_ro <- c("yellow", "#FC4E07")
p9 = ggplot(Heart_disease,aes(x = serum_creatinine )) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.5) +
  geom_vline(xintercept = median(Heart_disease$serum_creatinine), linetype="longdash",
             colour = "blue") +
  labs(x="Level of serum creatinine in the blood (mg/dL)",
       y="Density",
       title="serum creatinine distribution")

p10 = ggplot(Heart_disease, aes(x = serum_creatinine, fill = DEATH_EVENT)) +
  geom_density(aes(serum_creatinine,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$serum_creatinine),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$serum_creatinine),
             linetype="longdash", colour = "orange") +
  labs(x="Level of serum creatinine in the blood (mg/dL)",
       y="Density",
       title="Relationship between serum creatinine and DEATH EVENT")

grid.arrange(p9, p10, nrow=2)
```





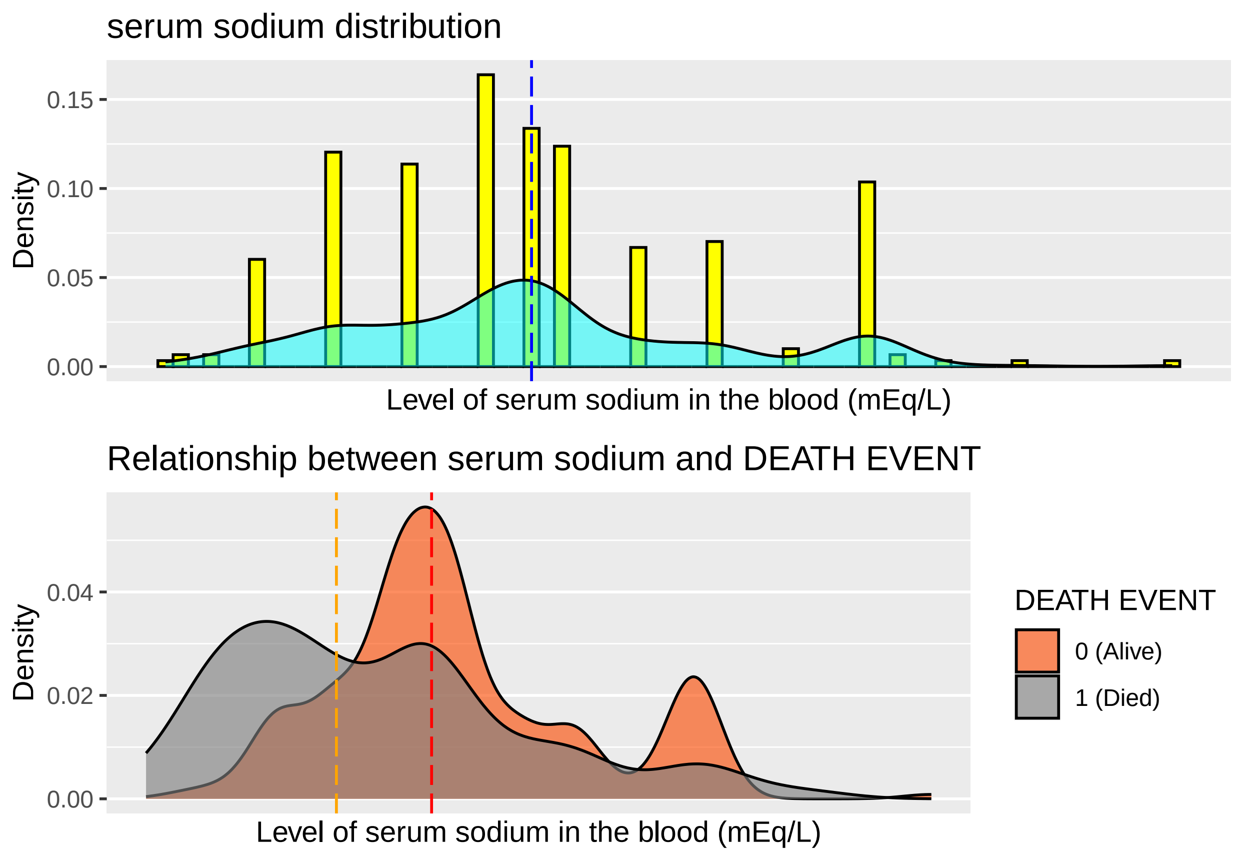
The insights we get from above histogram and density for serum creatinine distribution are: * The distribution is highly skewed and will be scaled later in this research, for survivals the value is around the median and the patients who have 1.5mg/dL have high risk of heart failure.

serum sodium distribution

```
#Histogram - serum sodium
palette_ro <- c("yellow", "#FC4E07")
p11 = ggplot(Heart_disease,aes(x = serum_sodium)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.5) +
  scale_x_continuous(breaks = seq(100, 150, 10)) +
  geom_vline(xintercept = median(Heart_disease$serum_sodium), linetype="longdash",
             colour = "blue") +
  labs(x="Level of serum sodium in the blood (mEq/L)",
       y="Density",
       title="serum sodium distribution")

p12 = ggplot(Heart_disease, aes(x = serum_sodium, fill = DEATH_EVENT)) +
  geom_density(aes(serum_sodium,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  scale_x_continuous(breaks = seq(100, 150, 10)) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$serum_sodium),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$serum_sodium),
             linetype="longdash", colour = "orange") +
  labs(x="Level of serum sodium in the blood (mEq/L)",
       y="Density",
       title="Relationship between serum sodium and DEATH EVENT")

grid.arrange(p11, p12, nrow=2)
```



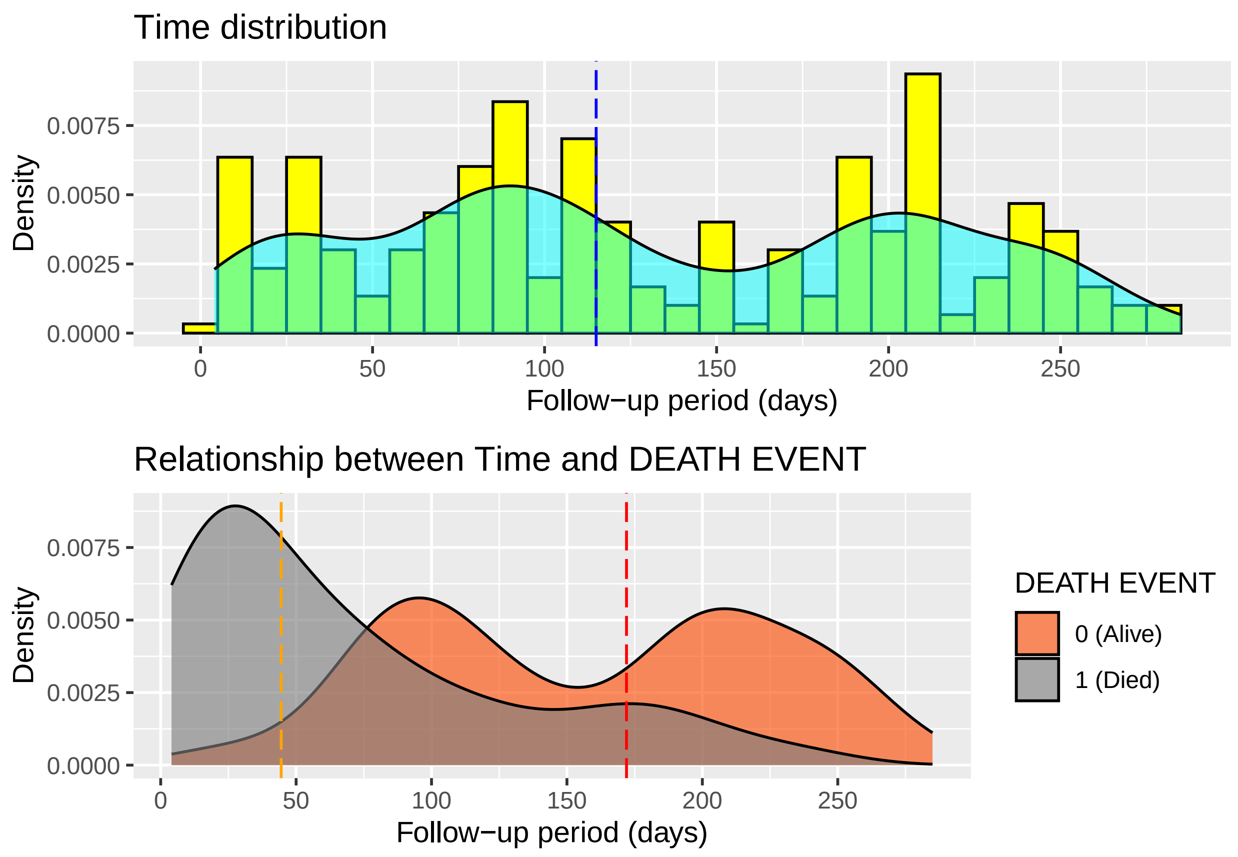
The insights we get from above histogram and density for serum sodium distribution are: * The survival of patients is more around the median and the value of deaths get's lower when the level of serum sodium increases.

Time distribution

```
#Histogram - Time
palette_ro <- c("yellow", "#FC4E07")
p13 = ggplot(Heart_disease,aes(x = time)) +
  geom_histogram(aes(y = ..density..), binwidth = 10, fill = palette_ro[1],color="black") +
    geom_density(adjust=.8, fill="cyan",color="black", alpha=0.5) +
  scale_x_continuous(breaks = seq(0, 300, 50)) +
  geom_vline(xintercept = median(Heart_disease$time), linetype="longdash",
             colour = "blue") +
  labs(x="Follow-up period (days)",
       y="Density",
       title="Time distribution")

p14 = ggplot(Heart_disease, aes(x = time, fill = DEATH_EVENT)) +
  geom_density(aes(time,fill=DEATH_EVENT),alpha=0.64) +
  scale_fill_manual(values = c(palette_ro[2], palette_ro[7]),
                    name = "DEATH EVENT",
                    labels = c("0 (Alive)", "1 (Died)")) +
  scale_x_continuous(breaks = seq(0, 300, 50)) +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 0)$time),
             linetype="longdash", colour = "red") +
  geom_vline(xintercept = median(filter(Heart_disease, DEATH_EVENT == 1)$time),
             linetype="longdash", colour = "orange") +
  labs(x="Follow-up period (days)",
       y="Density",
       title="Relationship between Time and DEATH EVENT")

grid.arrange(p13, p14, nrow=2)
```



The insights we get from above histogram and density for time distribution are: * The patients who have more follow-up days are likely to have higher chances of survival and the patients who have less than 60 follow-up days have higher chances of death. So, the more you follow-up on you regular health check-up's the more you have probability to survive.

From the above density and histogram plots, we can observe the data is moderately skewed, it may violate the assumptions of the model, leading to biased results or inaccurate predictions.Now let's calculate the skewness of the dataset and observe which has zero, negative and positive skewness and transform them.

skewness of dataset (All numeric variables)

```
#using sapply function to find skewness of all numeric variables
numeric_cols <- Heart_disease %>% select_if(is.numeric)
skewness_values <- sapply(numeric_cols, skewness)
skewness_df <- data.frame(variable = names(skewness_values), skewness = skewness_values)
skewness_df$index <- row.names(skewness_df)
rownames(skewness_df) <- NULL  # Remove row names
skewness_df <- skewness_df[, c( "variable", "skewness")]  # Rearrange columns
skewness_df
```

```
##                   variable  skewness
## 1                      age 0.4203739
## 2 creatinine_phosphokinase 4.4406886
## 3        ejection_fraction 0.5525927
## 4                platelets 1.4549746
## 5         serum_creatinine 4.4336102
## 6             serum_sodium 0.5525927
## 7                     time 0.1271606
```

From the above dataframe we can observe that the variables creatinine_phosphokinase, serum_creatinine and possibly platelets have a high degree of skewness, which could potentially affect the performance of some statistical models. The results might improve if the values are scaled. Let's compare both scaled and un-scaled data below by fitting into different models.

Scaling the data (Data Normalization)

```
scaled_heart_disease <- Heart_disease
scaled_heart_disease[, c("age", "creatinine_phosphokinase", "ejection_fraction",
                         "platelets", "serum_creatinine",
                         "serum_sodium","time")] <-
  scale(scaled_heart_disease[, c("age", "creatinine_phosphokinase", "ejection_fraction",
                                 "platelets", "serum_creatinine",
                                 "serum_sodium","time")])

summary(scaled_heart_disease)
```

```
##       age            anaemia creatinine_phosphokinase diabetes
##  Min.   :-1.75170   0:170   Min.   :-0.575952        0:174
##  1st Qu.:-0.82693   1:129   1st Qu.:-0.479589        1:125
##  Median :-0.07029           Median :-0.342001
##  Mean   : 0.00000           Mean   : 0.000000
##  3rd Qu.: 0.77041           3rd Qu.: 0.000165
##  Max.   : 2.87217           Max.   : 7.502063
##  ejection_fraction   high_blood_pressure   platelets        serum_creatinine
##  Min.   :-2.034976   0:194               Min.   :-2.43607   Min.   :-0.864061
##  1st Qu.:-0.683035   1:105               1st Qu.:-0.52000   1st Qu.:-0.477404
##  Median :-0.007065                       Median :-0.01388   Median :-0.284076
##  Mean   : 0.000000                       Mean   : 0.00000   Mean   : 0.000000
##  3rd Qu.: 0.584409                       3rd Qu.: 0.41043   3rd Qu.: 0.005916
##  Max.   : 3.541779                       Max.   : 5.99812   Max.   : 7.739045
##   serum_sodium       sex     smoking      time         DEATH_EVENT
##  Min.   :-2.034976   0:105   0:203   Min.   :-1.6268   0:203
##  1st Qu.:-0.683035   1:194   1: 96   1st Qu.:-0.7378   1: 96
##  Median :-0.007065                   Median :-0.1966
##  Mean   : 0.000000                   Mean   : 0.0000
##  3rd Qu.: 0.584409                   3rd Qu.: 0.9372
##  Max.   : 3.541779                   Max.   : 1.9937
```

```
Heart_disease1 = subset(Heart_disease, select = -c(serum_sodium))

scaled_heart_disease11 <- Heart_disease1
scaled_heart_disease11[, c("age", "creatinine_phosphokinase", "ejection_fraction",
                           "platelets", "serum_creatinine",
                     "time")] <-
  scale(scaled_heart_disease11[, c("age", "creatinine_phosphokinase", "ejection_fraction",
                                   "platelets", "serum_creatinine",
                     "time")])
```

Now you can observe that the scaled data has a mean of zero and a standard deviation of one.

Splitting the dataset into training and testing into 70% and 30%

```
set.seed(1)
train <- sample(nrow(Heart_disease), 0.7 * nrow(Heart_disease))
training_set_unscaled <- Heart_disease[train, ]
testing_set_unscaled <- Heart_disease[-train, ]

train_2 <- sample(nrow(scaled_heart_disease), 0.7 * nrow(scaled_heart_disease))
training_set_scaled <- scaled_heart_disease[train_2, ]
testing_set_scaled <- scaled_heart_disease[-train_2, ]

Heart_disease1 = subset(Heart_disease, select = -c(serum_sodium))

set.seed(1)
train11 <- sample(nrow(Heart_disease1), 0.7 * nrow(Heart_disease1))
training_set_unscaled11 <- Heart_disease1[train11, ]
testing_set_unscaled11 <- Heart_disease1[-train11, ]

train_22 <- sample(nrow(scaled_heart_disease11), 0.7 * nrow(scaled_heart_disease11))
training_set_scaled11 <- scaled_heart_disease11[train_22, ]
testing_set_scaled11 <- scaled_heart_disease11[-train_22, ]
```

Now let's fit the dataset to different classification models with scaled and unscaled data and compare the results of each.

MODEL 1 - LOGISTIC REGRESSION

MODEL 1.a -> LOGISTIC REGRESSION - ALL VARIABLES - UNSCALED

```r
#fitting the un-scaled data on logistic regression model
glm.all_unscaled = glm(DEATH_EVENT~.,data=training_set_unscaled,family="binomial")

#predicting on the testing un-scaled data
predict_test_all_unscaled <- factor(ifelse(predict(glm.all_unscaled, testing_set_unscaled ,
                                                   type ="response") > 0.5, "1", "0"))
summary(glm.all_unscaled)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ ., family = "binomial", data = training_set_unscaled)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0887  -0.6121  -0.2442   0.4879   2.6104
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               5.038e-01  1.528e+00   0.330 0.741634
## age                       5.829e-02  1.724e-02   3.381 0.000722 ***
## anaemia1                 -3.303e-01  4.289e-01  -0.770 0.441311
## creatinine_phosphokinase  1.326e-04  1.713e-04   0.774 0.438792
## diabetes1                 2.549e-01  4.066e-01   0.627 0.530734
## ejection_fraction        -7.143e-02  1.924e-02  -3.713 0.000205 ***
## high_blood_pressure1     -4.397e-02  4.156e-01  -0.106 0.915729
## platelets                -2.195e-06  2.069e-06  -1.061 0.288643
## serum_creatinine          5.823e-01  1.872e-01   3.110 0.001870 **
## serum_sodium                     NA         NA      NA       NA
## sex1                     -4.960e-01  4.679e-01  -1.060 0.289138
## smoking1                 -3.080e-02  4.890e-01  -0.063 0.949783
## time                     -1.925e-02  3.447e-03  -5.584 2.35e-08 ***
## ---
## 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: 162.08  on 197  degrees of freedom
## AIC: 186.08
##
## Number of Fisher Scoring iterations: 5
```

The logistic regression has fitted to whole dataset with death event as a response variable. We got an accuracy of 88.9% when logistic regression is done on the un-scaled data and a true positive rate (recall) of 93.44%. From Fig. 15 it can be noted the coefficients (age, ejection fraction, serum creatinine, time) have asterisks next to their p-values, indicating that they are statistically significant at certain significance levels ($p < 0.05$), while others are not. The coefficients with the variables which are insignificant have the p value greater than 0.05 meaning they have weak evidence against the null hypothesis, and we do not reject null hypothesis. The lower p-values are considered more statistically significant and are typically interpreted as having a stronger association with the response variable (Death Event). The others with no asterisk are insignificant variables and we can remove them for further models.

We can also see serum sodium has NA values in the estimate column, standard error column, and p-value column. Which means the variable is not included in the model and the variable was likely dropped during the selection process. We observed that it is perfect positive with ejection fraction predictor. The main reason for the model to not select serum sodium is, this variable is highly correlated with the other predictor which is already included in the model.

Confusion matrix for full unscaled model

```
confusionMatrix(predict_test_all_unscaled,testing_set_unscaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57  6
##          1  4 23
##
##                Accuracy : 0.8889
##                  95% CI : (0.8051, 0.9454)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 2.826e-06
##
##                   Kappa : 0.7409
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9344
##             Specificity : 0.7931
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.8519
##              Prevalence : 0.6778
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8638
##
##        'Positive' Class : 0
##
```

We got an accuracy of 88.9% when logistic regression is done on the un-scaled data.

The confusion matrix obtained for un-scaled data is as follows: True Positives : 57 False Positives: 6 True Negatives : 23 False Negatives: 4

The balanced accuracy got by the logistic regression on the given dataset is 86.38%. It is the mean of sensitivity and Specificity.

Balanced Accuracy = Sensitivity + Specificity / 2 => 0.9344 + 0.7931 / 2 => 0.8638

Specificity value is 79.31% (True Negative Rate) Sensitivity value is 93.44% (True Positive Rate)

MODEL 1.b -> LOGISTIC REGRESSION - ALL VARIABLES - SCALED

```
#fitting the scaled data on logistic regression model
glm.all_scaled = glm(DEATH_EVENT~.,data=training_set_scaled,family="binomial")

#predicting on the testing scaled data
predict_test_all_scaled <- factor(ifelse(predict(glm.all_scaled, testing_set_scaled , type =
                                          "response") > 0.5, "1", "0"))
summary(glm.all_scaled)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ ., family = "binomial", data = training_set_scaled)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2618  -0.4682  -0.2058   0.3803   2.4580
##
## Coefficients: (1 not defined because of singularities)
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.02355    0.54624  -1.874 0.060954 .
## age                       0.66366    0.24400   2.720 0.006530 **
## anaemia1                 -0.35631    0.45968  -0.775 0.438272
## creatinine_phosphokinase  0.36738    0.23927   1.535 0.124689
## diabetes1                 0.37462    0.43834   0.855 0.392750
## ejection_fraction        -0.89478    0.24387  -3.669 0.000243 ***
## high_blood_pressure1     -0.01415    0.44318  -0.032 0.974523
## platelets                -0.13437    0.24610  -0.546 0.585079
## serum_creatinine          0.87110    0.21995   3.961 7.48e-05 ***
## serum_sodium                   NA         NA      NA       NA
## sex1                     -0.67105    0.51033  -1.315 0.188531
## smoking1                  0.11796    0.51915   0.227 0.820253
## time                     -1.84400    0.30653  -6.016 1.79e-09 ***
## ---
## 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: 142.07  on 197  degrees of freedom
## AIC: 166.07
##
## Number of Fisher Scoring iterations: 6
```

Confusion matrix for full scaled model

```
confusionMatrix(predict_test_all_scaled,testing_set_scaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  0  1
##          0 56 10
##          1  5 19
##
##                Accuracy : 0.8333
##                  95% CI : (0.74, 0.9036)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 0.0006812
##
##                   Kappa : 0.6004
##
##  Mcnemar's Test P-Value : 0.3016996
##
##             Sensitivity : 0.9180
##             Specificity : 0.6552
##          Pos Pred Value : 0.8485
##          Neg Pred Value : 0.7917
##              Prevalence : 0.6778
##          Detection Rate : 0.6222
##    Detection Prevalence : 0.7333
##       Balanced Accuracy : 0.7866
##
##        'Positive' Class : 0
##
```

We got an accuracy of 83.3% when logistic regression is done on the scaled data.

The confusion matrix obtained for scaled data is as follows: True Positives : 56 False Positives: 10 True Negatives : 19 False Negatives: 5

The balanced accuracy got by the logistic regression on the given dataset is 78.66%. It is the mean of sensitivity and Specificity.

Balanced Accuracy = Sensitivity + Specificity / 2 =>0.9180 + 0.6552 / 2 => 0.7866

Specificity value is 65.52% (True Negative Rate) Sensitivity value is 91.80% (True Positive Rate)

As we can see that the un-scaled data when fitted for the logistic model we got an accuracy of 88.9% and when fitted with the scaled data we got an accuracy of 83.3%. It is possible that scaling the data may have reduced the accuracy of the logistic regression model compared to the un-scaled data. This could be due to a few reasons:

- Outliers: Scaling the data can sometimes amplify the effect of outliers, which can negatively impact the accuracy of the model.
- Non-linear relationships: If there are non-linear relationships between the predictors and the response, scaling the data may reduce the ability of the model to capture these relationships.

  In general, It is good to experiment both scaled and un-scaled data and choose the method that has good accuracy. Here in our case we choose un-scaled data fitted for logistic model.

Since, We had perfect positive correlation between serum sodium, ejection fraction. Lets try removing one variable and check the accuracy.Now let's remove the serum sodium and observe whether it effects the model in any way for both scaled and unscaled data.

MODEL 1.c -> LOGISTIC REGRESSION - REMOVING SERUM SODIUM VARIABLE - UN-SCALED

```
#fitting the un-scaled data on logistic regression model (removing serum sodium)
glm.removing_serum_sodium_unscaled = glm(DEATH_EVENT~.-serum_sodium,data=training_set_unscaled,
                                         family="binomial")

#predicting on the testing un-scaled data (removing serum sodium)
predict_test_unscaled <- factor(ifelse(predict(glm.removing_serum_sodium_unscaled, testing_set_unscaled
                                              type ="response") > 0.5, "1", "0"))
summary(glm.removing_serum_sodium_unscaled)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ . - serum_sodium, family = "binomial",
##     data = training_set_unscaled)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0887  -0.6121  -0.2442   0.4879   2.6104
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               5.038e-01  1.528e+00   0.330 0.741634
## age                       5.829e-02  1.724e-02   3.381 0.000722 ***
## anaemia1                 -3.303e-01  4.289e-01  -0.770 0.441311
## creatinine_phosphokinase  1.326e-04  1.713e-04   0.774 0.438792
## diabetes1                 2.549e-01  4.066e-01   0.627 0.530734
## ejection_fraction        -7.143e-02  1.924e-02  -3.713 0.000205 ***
## high_blood_pressure1     -4.397e-02  4.156e-01  -0.106 0.915729
## platelets                -2.195e-06  2.069e-06  -1.061 0.288643
## serum_creatinine          5.823e-01  1.872e-01   3.110 0.001870 **
## sex1                     -4.960e-01  4.679e-01  -1.060 0.289138
## smoking1                 -3.080e-02  4.890e-01  -0.063 0.949783
## time                     -1.925e-02  3.447e-03  -5.584 2.35e-08 ***
## ---
## 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: 162.08  on 197  degrees of freedom
## AIC: 186.08
##
## Number of Fisher Scoring iterations: 5
```

Confusion matrix for removing serum sodium predictor - un-scaled

```
confusionMatrix(predict_test_unscaled,testing_set_unscaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  0  1
##          0 57  6
##          1  4 23
##
##                Accuracy : 0.8889
##                  95% CI : (0.8051, 0.9454)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 2.826e-06
##
##                   Kappa : 0.7409
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9344
##             Specificity : 0.7931
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.8519
##              Prevalence : 0.6778
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8638
##
##        'Positive' Class : 0
##
```

MODEL 1.d -> LOGISTIC REGRESSION - REMOVING SERUM SODIUM VARIABLE - SCALED

```r
#fitting the scaled data on logistic regression model (removing serum sodium)
glm.removing_serum_sodium_scaled = glm(DEATH_EVENT~.-serum_sodium,data=training_set_scaled,
                                       family="binomial")

#predicting on the testing scaled data (removing serum sodium)
predict_test_scaled <- factor(ifelse(predict(glm.removing_serum_sodium_scaled, testing_set_scaled ,
                                             type ="response") > 0.5, "1", "0"))
summary(glm.removing_serum_sodium_scaled)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ . - serum_sodium, family = "binomial",
##     data = training_set_scaled)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2618  -0.4682  -0.2058   0.3803   2.4580
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.02355    0.54624  -1.874 0.060954 .
## age                       0.66366    0.24400   2.720 0.006530 **
## anaemia1                 -0.35631    0.45968  -0.775 0.438272
## creatinine_phosphokinase  0.36738    0.23927   1.535 0.124689
## diabetes1                 0.37462    0.43834   0.855 0.392750
## ejection_fraction        -0.89478    0.24387  -3.669 0.000243 ***
## high_blood_pressure1     -0.01415    0.44318  -0.032 0.974523
## platelets                -0.13437    0.24610  -0.546 0.585079
## serum_creatinine          0.87110    0.21995   3.961 7.48e-05 ***
## sex1                     -0.67105    0.51033  -1.315 0.188531
## smoking1                  0.11796    0.51915   0.227 0.820253
## time                     -1.84400    0.30653  -6.016 1.79e-09 ***
## ---
## 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: 142.07  on 197  degrees of freedom
## AIC: 166.07
##
## Number of Fisher Scoring iterations: 6
```

Confusion matrix for removing serum sodium predictor - scaled

```
confusionMatrix(predict_test_scaled,testing_set_scaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 10
##          1  5 19
##
##                Accuracy : 0.8333
##                  95% CI : (0.74, 0.9036)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 0.0006812
##
##                   Kappa : 0.6004
##
##  Mcnemar's Test P-Value : 0.3016996
##
##             Sensitivity : 0.9180
##             Specificity : 0.6552
##          Pos Pred Value : 0.8485
##          Neg Pred Value : 0.7917
##              Prevalence : 0.6778
##          Detection Rate : 0.6222
##    Detection Prevalence : 0.7333
##       Balanced Accuracy : 0.7866
##
##        'Positive' Class : 0
##
```

We got the same confusion matrix for the full model and when serum sodium is removed from both scaled and un-scaled data.We observe no difference when the predictor serum sodium is removed from both scaled and un-scaled data. The accuraccies remains the same i.e; 83.3% and 88.89% respectively.

Now let's find the significant predictors and fit the model and compare the accuracies for both scaled and unscaled data and also compare the model with the above models.

FORWARD STEPWISE SELECTION

```
Heart_disease2 = subset(Heart_disease, select = -c(serum_sodium))
regfit.bwd=regsubsets(DEATH_EVENT~.,data=Heart_disease2,nvmax=12, method ="forward")
summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(DEATH_EVENT ~ ., data = Heart_disease2, nvmax = 12,
##     method = "forward")
## 11 Variables  (and intercept)
##                          Forced in Forced out
## age                          FALSE      FALSE
## anaemia1                     FALSE      FALSE
## creatinine_phosphokinase     FALSE      FALSE
## diabetes1                    FALSE      FALSE
## ejection_fraction            FALSE      FALSE
## high_blood_pressure1         FALSE      FALSE
## platelets                    FALSE      FALSE
## serum_creatinine             FALSE      FALSE
## sex1                         FALSE      FALSE
## smoking1                     FALSE      FALSE
## time                         FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: forward
##           age anaemia1 creatinine_phosphokinase diabetes1 ejection_fraction
## 1  ( 1 )  " " " "      " "                      " "       " "
## 2  ( 1 )  " " " "      " "                      " "       "*"
## 3  ( 1 )  " " " "      " "                      " "       "*"
## 4  ( 1 )  "*" " "      " "                      " "       "*"
## 5  ( 1 )  "*" " "      "*"                      " "       "*"
## 6  ( 1 )  "*" " "      "*"                      " "       "*"
## 7  ( 1 )  "*" " "      "*"                      "*"       "*"
## 8  ( 1 )  "*" " "      "*"                      "*"       "*"
## 9  ( 1 )  "*" " "      "*"                      "*"       "*"
## 10  ( 1 ) "*" "*"      "*"                      "*"       "*"
## 11  ( 1 ) "*" "*"      "*"                      "*"       "*"
##           high_blood_pressure1 platelets serum_creatinine sex1 smoking1 time
## 1  ( 1 )  " "                  " "       " "              " "  " "      "*"
## 2  ( 1 )  " "                  " "       " "              " "  " "      "*"
## 3  ( 1 )  " "                  " "       "*"              " "  " "      "*"
## 4  ( 1 )  " "                  " "       "*"              " "  " "      "*"
## 5  ( 1 )  " "                  " "       "*"              " "  " "      "*"
## 6  ( 1 )  " "                  " "       "*"              "*"  " "      "*"
## 7  ( 1 )  " "                  " "       "*"              "*"  " "      "*"
## 8  ( 1 )  " "                  "*"       "*"              "*"  " "      "*"
## 9  ( 1 )  "*"                 "*"       "*"             "*"  " "     "*"
## 10  ( 1 ) "*"                 "*"       "*"             "*"  " "     "*"
## 11  ( 1 ) "*"                 "*"       "*"             "*"  "*"     "*"
```

BACKWARD STEPWISE SELECTION

```
Heart_disease2 = subset(Heart_disease, select = -c(serum_sodium))
regfit.bwd=regsubsets(DEATH_EVENT~.,data=Heart_disease2,nvmax=12, method ="backward")
summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(DEATH_EVENT ~ ., data = Heart_disease2, nvmax = 12,
##     method = "backward")
## 11 Variables  (and intercept)
##                            Forced in Forced out
## age                            FALSE      FALSE
## anaemia1                       FALSE      FALSE
## creatinine_phosphokinase       FALSE      FALSE
## diabetes1                      FALSE      FALSE
## ejection_fraction              FALSE      FALSE
## high_blood_pressure1           FALSE      FALSE
## platelets                      FALSE      FALSE
## serum_creatinine               FALSE      FALSE
## sex1                           FALSE      FALSE
## smoking1                       FALSE      FALSE
## time                           FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: backward
##           age anaemia1 creatinine_phosphokinase diabetes1 ejection_fraction
## 1  ( 1 )  " " " "      " "                      " "       " "
## 2  ( 1 )  " " " "      " "                      " "       "*"
## 3  ( 1 )  " " " "      " "                      " "       "*"
## 4  ( 1 )  "*" " "      " "                      " "       "*"
## 5  ( 1 )  "*" " "      "*"                      " "       "*"
## 6  ( 1 )  "*" " "      "*"                      " "       "*"
## 7  ( 1 )  "*" " "      "*"                      "*"       "*"
## 8  ( 1 )  "*" " "      "*"                      "*"       "*"
## 9  ( 1 )  "*" " "      "*"                      "*"       "*"
## 10  ( 1 ) "*" "*"      "*"                      "*"       "*"
## 11  ( 1 ) "*" "*"      "*"                      "*"       "*"
##           high_blood_pressure1 platelets serum_creatinine sex1 smoking1 time
## 1  ( 1 )  " "                  " "       " "              " "  " "      "*"
## 2  ( 1 )  " "                  " "       " "              " "  " "      "*"
## 3  ( 1 )  " "                  " "       "*"              " "  " "      "*"
## 4  ( 1 )  " "                  " "       "*"              " "  " "      "*"
## 5  ( 1 )  " "                  " "       "*"              " "  " "      "*"
## 6  ( 1 )  " "                  " "       "*"              "*"  " "      "*"
## 7  ( 1 )  " "                  " "       "*"              "*"  " "      "*"
## 8  ( 1 )  " "                  "*"       "*"              "*"  " "      "*"
## 9  ( 1 )  "*"                  "*"       "*"              "*"  " "      "*"
## 10  ( 1 ) "*"                  "*"       "*"              "*"  " "      "*"
## 11  ( 1 ) "*"                  "*"       "*"              "*"  "*"      "*"
```

We see that using backward stepwise selection, the best one-variable model contains only time, and the best two-variable model additionally includes ejection_fraction. After observing the summary of the full model with the p-values of the predictors and the variable selection technique (backward), we can conclude that time,ejection_fraction,serum_creatinine and age are taken has the significant variables.

MODEL 1.e -> LOGISTIC REGRESSION - SIGNIFICANT VARIABLES - UN-SCALED

```
#fitting the un-scaled data on logistic regression model (only significant variables)
glm.significant_unscaled =
      glm(DEATH_EVENT~time+ejection_fraction+serum_creatinine+age,data=training_set_unscaled,
          family="binomial")

#predicting on the testing un-scaled data
predict_test_unscaled1 <- factor(ifelse(predict(glm.significant_unscaled, testing_set_unscaled,
                                                 type ="response") > 0.5, "1", "0"))
summary(glm.significant_unscaled)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ time + ejection_fraction + serum_creatinine +
##     age, family = "binomial", data = training_set_unscaled)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9636  -0.6385  -0.2597   0.4775   2.7971
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.131529   1.183781  -0.111 0.911530
## time              -0.019215   0.003244  -5.923 3.16e-09 ***
## ejection_fraction -0.069250   0.018456  -3.752 0.000175 ***
## serum_creatinine   0.599730   0.180621   3.320 0.000899 ***
## age                0.052946   0.016381   3.232 0.001229 **
## ---
## 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: 165.55  on 204  degrees of freedom
## AIC: 175.55
##
## Number of Fisher Scoring iterations: 5
```

Confusion matrix for significant predictors - un-scaled

```
confusionMatrix(predict_test_unscaled1,testing_set_unscaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  0  1
##          0 57  6
##          1  4 23
##
##                Accuracy : 0.8889
##                  95% CI : (0.8051, 0.9454)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 2.826e-06
##
##                   Kappa : 0.7409
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9344
##             Specificity : 0.7931
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.8519
##              Prevalence : 0.6778
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8638
##
##        'Positive' Class : 0
##
```

MODEL 1.f -> LOGISTIC REGRESSION - SIGNIFICANT VARIABLES - SCALED

```r
#fitting the scaled data on logistic regression model (only significant variables)
glm.significant_scaled =
      glm(DEATH_EVENT~age+ejection_fraction+serum_creatinine+time,data=training_set_scaled,
          family="binomial")

#predicting on the testing scaled data
predict_test_scaled1 <- factor(ifelse(predict(glm.significant_scaled, testing_set_scaled ,
                                              type ="response") > 0.5, "1", "0"))
summary(glm.significant_scaled)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine +
##     time, family = "binomial", data = training_set_scaled)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2894  -0.4975  -0.2151   0.4442   2.3156
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.3340     0.2465  -5.411 6.27e-08 ***
## age                 0.5351     0.2249   2.380 0.017327 *
## ejection_fraction  -0.8526     0.2293  -3.718 0.000201 ***
## serum_creatinine    0.8528     0.2120   4.023 5.74e-05 ***
## time               -1.7865     0.2854  -6.259 3.87e-10 ***
## ---
## 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: 148.29  on 204  degrees of freedom
## AIC: 158.29
##
## Number of Fisher Scoring iterations: 6
```

Confusion matrix for significant predictors -scaled

```
confusionMatrix(predict_test_scaled1,testing_set_scaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 54  9
##          1  7 20
##
##                Accuracy : 0.8222
##                  95% CI : (0.7274, 0.8948)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 0.001599
##
##                   Kappa : 0.5855
##
##  Mcnemar's Test P-Value : 0.802587
##
##             Sensitivity : 0.8852
##             Specificity : 0.6897
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.7407
##              Prevalence : 0.6778
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.7875
##
##        'Positive' Class : 0
##
```

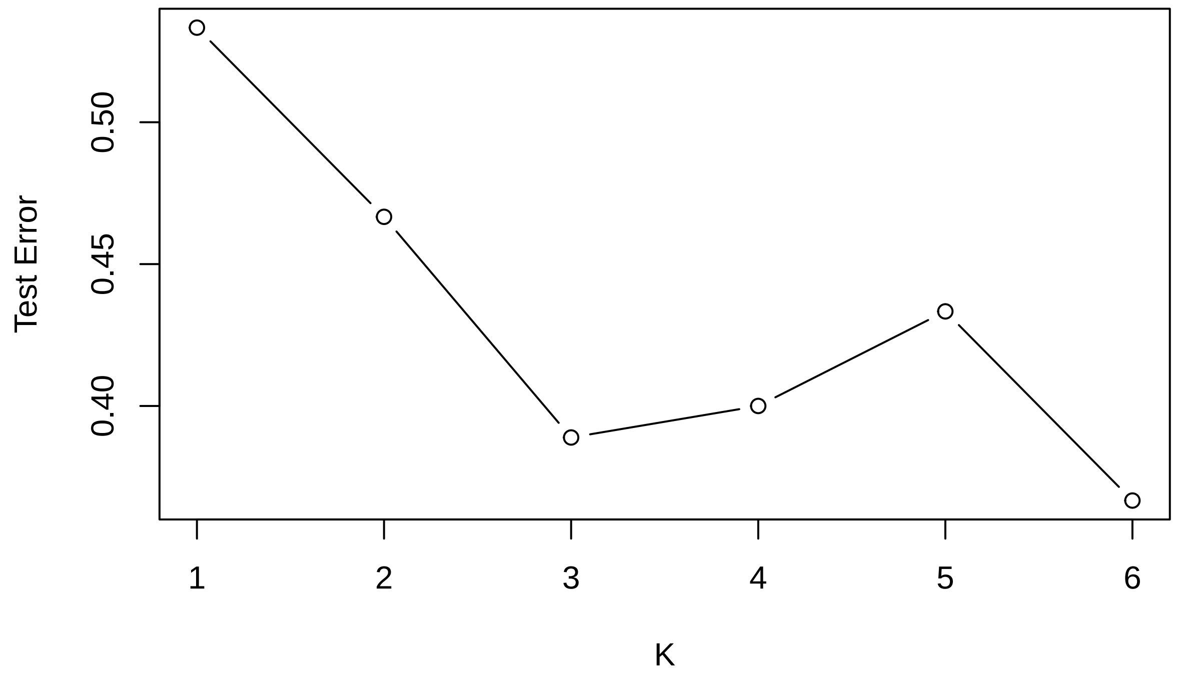
We got the same confusion matrix and the calculations of the model's performance. So,the model with 4 significant variables (age, ejection_fraction, serum_creatinine and time) is selected as the best model with an accuracy of 88.89% and recall/ true positive rate/ sensitivity of 93.44%.

MODEL 2 - K-NEAREST NEIGHBORS

MODEL 2.a - K-NEAREST NEIGHBORS - ALL VARIABLES - UNSCALED

```
error<-rep(NA,6) # Placeholder
training_set_unscaled_knn = subset(training_set_unscaled, select =
                                     c(age,creatinine_phosphokinase,ejection_fraction,
                                       platelets,serum_creatinine,time))
testing_set_unscaled_knn = subset(testing_set_unscaled, select =
                                    c(age,creatinine_phosphokinase,ejection_fraction,platelets,
                                      serum_creatinine,time))

for(i in 1:6)
{
  knn.pred = knn(training_set_unscaled_knn,testing_set_unscaled_knn,
                 training_set_unscaled$DEATH_EVENT,k=i)
  error[i]=mean(knn.pred!=testing_set_unscaled$DEATH_EVENT)
}
plot(error,type="b",xlab="K",ylab="Test Error")
```



Finding best k-value

```r
k = (loc=which.min(error))
k
```

```
## [1] 6
```

Confusion matrix for KNN - un-scaled (All variables)

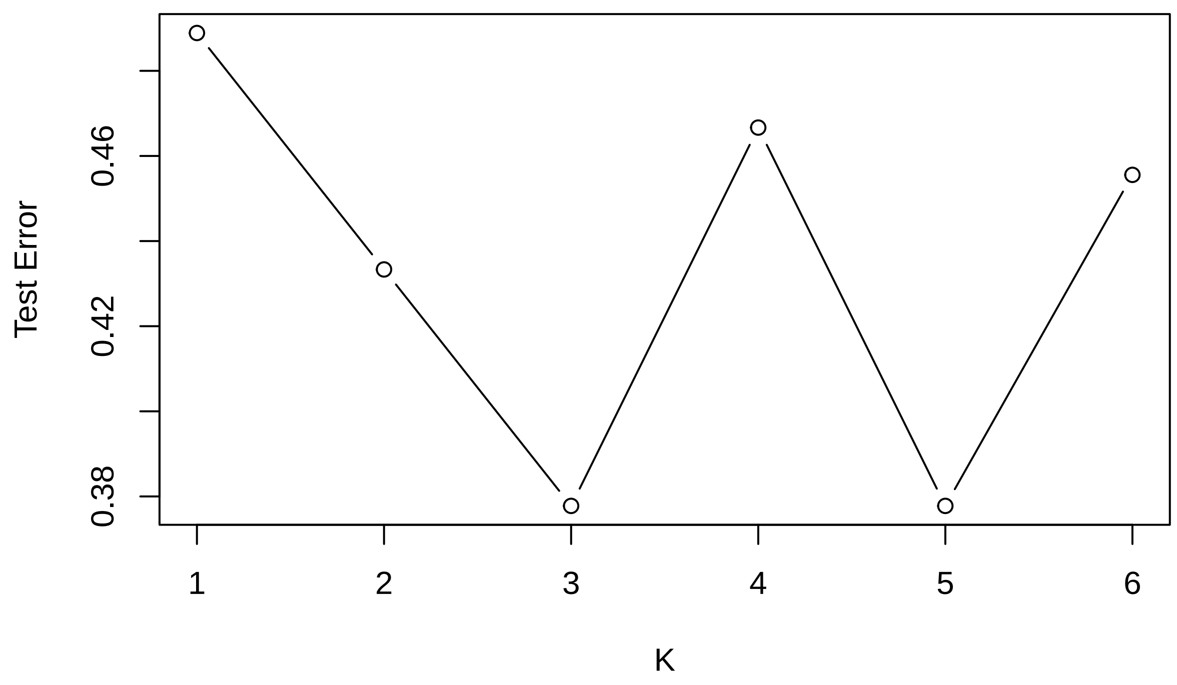
```r
knn.pred_unscaled=knn(train = training_set_unscaled_knn,test = testing_set_unscaled_knn,cl =
                        training_set_unscaled$DEATH_EVENT,k = k)
confusionMatrix(testing_set_unscaled$DEATH_EVENT, knn.pred_unscaled)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48 13
##          1 25  4
##
##                Accuracy : 0.5778
##                  95% CI : (0.4691, 0.6812)
##     No Information Rate : 0.8111
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : -0.0843
##
##  Mcnemar's Test P-Value : 0.07435
##
##             Sensitivity : 0.6575
##             Specificity : 0.2353
##          Pos Pred Value : 0.7869
##          Neg Pred Value : 0.1379
##              Prevalence : 0.8111
##          Detection Rate : 0.5333
##    Detection Prevalence : 0.6778
##       Balanced Accuracy : 0.4464
##
##        'Positive' Class : 0
##
```

From the un-scaled model fitted to KNN, the obtained accuracy is 65.56%for full variable (continuous) model.

MODEL 2.b - K-NEAREST NEIGHBORS - ALL VARIABLES - SCALED

```r
error<-rep(NA,6) # Placeholder
training_set_scaled_knn = subset(training_set_unscaled, select =
                                      c(age,creatinine_phosphokinase,ejection_fraction,
                                        platelets,serum_creatinine,time))
testing_set_scaled_knn = subset(testing_set_unscaled, select =
                                            c(age,creatinine_phosphokinase,ejection_fraction,
                                              platelets,serum_creatinine,time))
for(i in 1:6)
{
  knn.pred=knn(training_set_scaled_knn,testing_set_scaled_knn,training_set_scaled$DEATH_EVENT,
               k=i)
  error[i]=mean(knn.pred!=testing_set_scaled$DEATH_EVENT)
}
plot(error,type="b",xlab="K",ylab="Test Error")
```



Finding best k-value

```
k = (loc=which.min(error))
k
```

```
## [1] 3
```

Confusion matrix for KNN - scaled (All variables)

```
knn.pred_scaled=knn(train = training_set_scaled_knn,test = testing_set_scaled_knn,
                    cl = training_set_scaled$DEATH_EVENT,k = k)
confusionMatrix(testing_set_scaled$DEATH_EVENT, knn.pred_scaled)
```

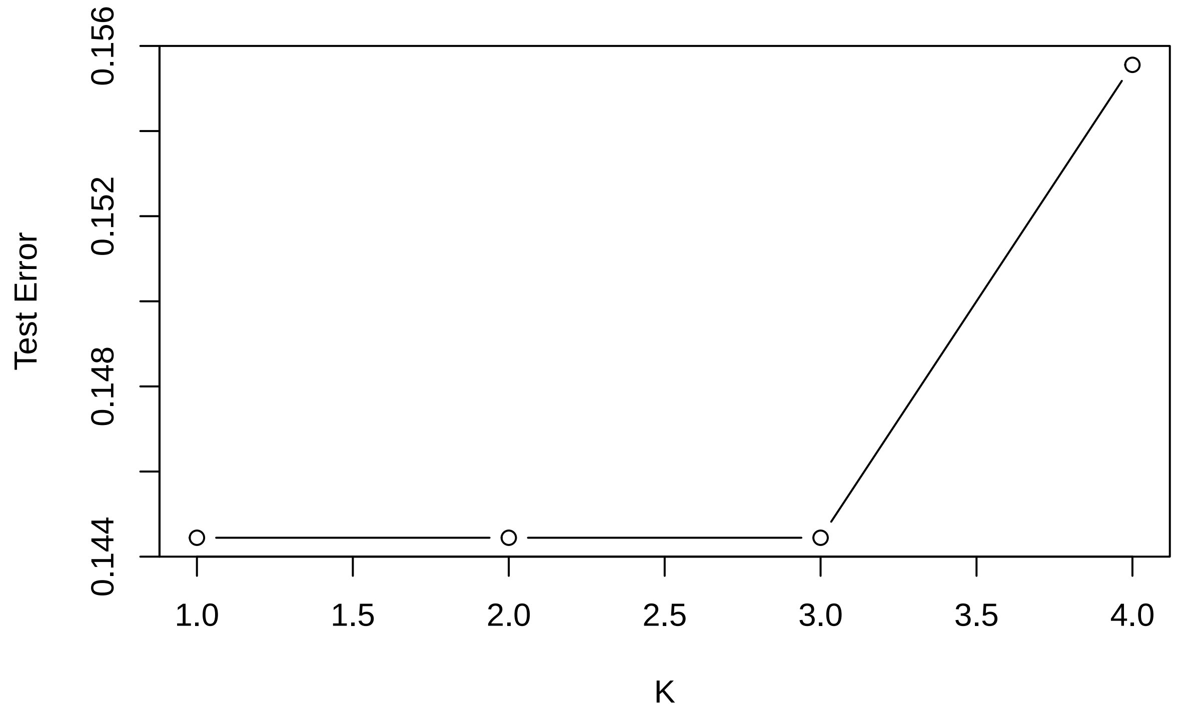
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 51 10
##          1 24  5
##
##                Accuracy : 0.6222
##                  95% CI : (0.5138, 0.7223)
##     No Information Rate : 0.8333
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.0097
##
##  Mcnemar's Test P-Value : 0.02578
##
##             Sensitivity : 0.6800
##             Specificity : 0.3333
##          Pos Pred Value : 0.8361
##          Neg Pred Value : 0.1724
##              Prevalence : 0.8333
##          Detection Rate : 0.5667
##    Detection Prevalence : 0.6778
##       Balanced Accuracy : 0.5067
##
##        'Positive' Class : 0
##
```

From the scaled model fitted to KNN, the obtained accuracy is 87.78% for full variable (continuous) model.

MODEL 2.c - K-NEAREST NEIGHBORS - SIGNIFICANT VARIABLES - UN-SCALED

```
error<-rep(NA,4) # Placeholder
training_set_unscaled_knn1 = subset(training_set_unscaled, select =
                                     c(age,ejection_fraction,
                                       serum_creatinine,time))
testing_set_unscaled_knn1 = subset(testing_set_unscaled, select =
                                    c(age,ejection_fraction,
                                      serum_creatinine,time))

for(i in 1:4)
{
  knn.pred = knn(training_set_unscaled_knn1,testing_set_unscaled_knn1,
                 training_set_unscaled$DEATH_EVENT,k=i)
  error[i]=mean(knn.pred!=testing_set_unscaled$DEATH_EVENT)
}
plot(error,type="b",xlab="K",ylab="Test Error")
```



Finding best k-value

```
k = (loc=which.min(error))
k
```

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

Confusion matrix for KNN - un-scaled (significant variables)

```
knn.pred_unscaled1 = knn(train = training_set_unscaled_knn1,test = testing_set_unscaled_knn1,
                         cl = training_set_unscaled$DEATH_EVENT,k = k)
confusionMatrix(testing_set_unscaled$DEATH_EVENT, knn.pred_unscaled1)
```

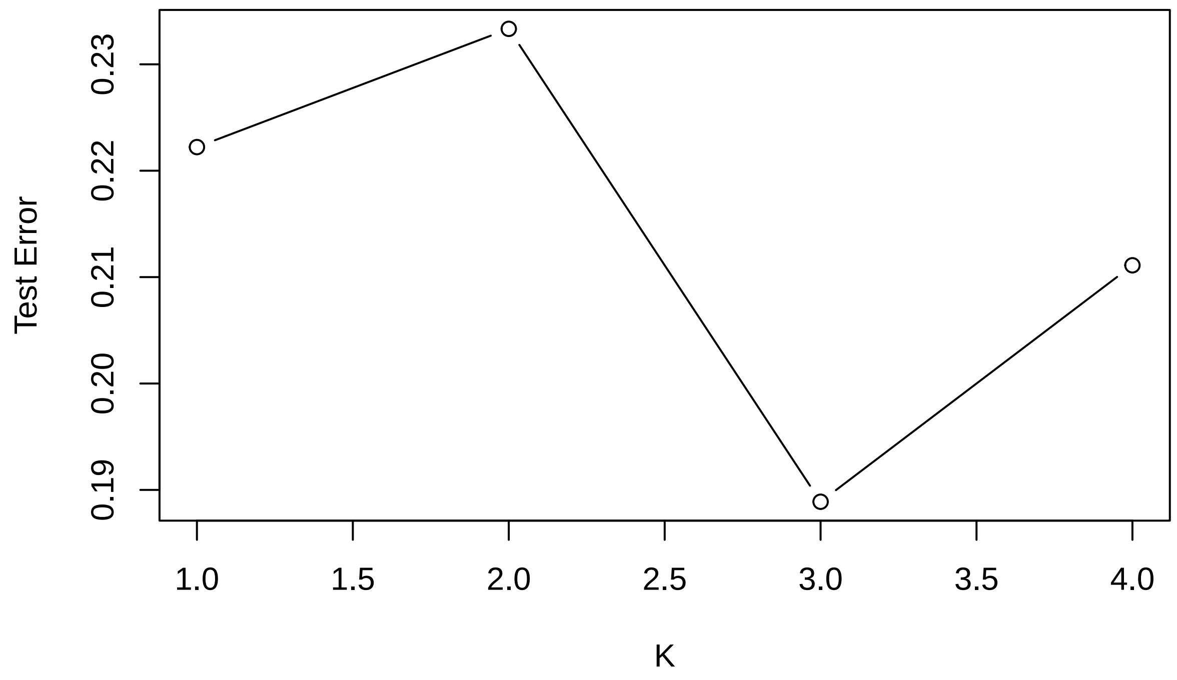
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57  4
##          1  9 20
##
##                Accuracy : 0.8556
##                  95% CI : (0.7657, 0.9208)
##     No Information Rate : 0.7333
##     P-Value [Acc > NIR] : 0.004229
##
##                   Kappa : 0.6536
##
##  Mcnemar's Test P-Value : 0.267257
##
##             Sensitivity : 0.8636
##             Specificity : 0.8333
##          Pos Pred Value : 0.9344
##          Neg Pred Value : 0.6897
##              Prevalence : 0.7333
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.6778
##       Balanced Accuracy : 0.8485
##
##        'Positive' Class : 0
##
```

From the un-scaled model fitted to KNN, the obtained accuracy is 85.56% for significant variables.

MODEL 2.d - K-NEAREST NEIGHBORS - SIGNIFICANT VARIABLES - SCALED

```r
error<-rep(NA,4) # Placeholder
training_set_scaled_knn1 = subset(training_set_scaled, select =
                                     c(age,ejection_fraction,
                                       serum_creatinine,time))
testing_set_scaled_knn1 = subset(testing_set_scaled, select =
                                    c(age,ejection_fraction,
                                      serum_creatinine,time))

for(i in 1:4)
{
  knn.pred = knn(training_set_scaled_knn1,testing_set_scaled_knn1,training_set_scaled$DEATH_EVENT,
               k=i)
  error[i]=mean(knn.pred!=testing_set_scaled$DEATH_EVENT)
}
plot(error,type="b",xlab="K",ylab="Test Error")
```



Finding best k-value

```
k = (loc=which.min(error))
k
```

```
## [1] 3
```

Confusion matrix for KNN - scaled (significant variables)

```
knn.pred_scaled1 = knn(train = training_set_scaled_knn1,test = testing_set_scaled_knn1,cl = training_se
confusionMatrix(testing_set_scaled$DEATH_EVENT, knn.pred_scaled1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52  9
##          1  8 21
##
##                Accuracy : 0.8111
##                  95% CI : (0.7149, 0.8859)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.001794
##
##                   Kappa : 0.5714
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8667
##             Specificity : 0.7000
##          Pos Pred Value : 0.8525
##          Neg Pred Value : 0.7241
##              Prevalence : 0.6667
##          Detection Rate : 0.5778
##    Detection Prevalence : 0.6778
##       Balanced Accuracy : 0.7833
##
##        'Positive' Class : 0
##
```

From the scaled model fitted to KNN, the obtained accuracy is 81.11% for significant variables.

We can observe the KNN algorithm performs well on un-scaled data compared to scaled data. So, the model with un-scaled data with significant variables is selected as best model in KNN with 85.56 accuracy and will compare it with other models later in this research.

MODEL 3 - DECISION TREES

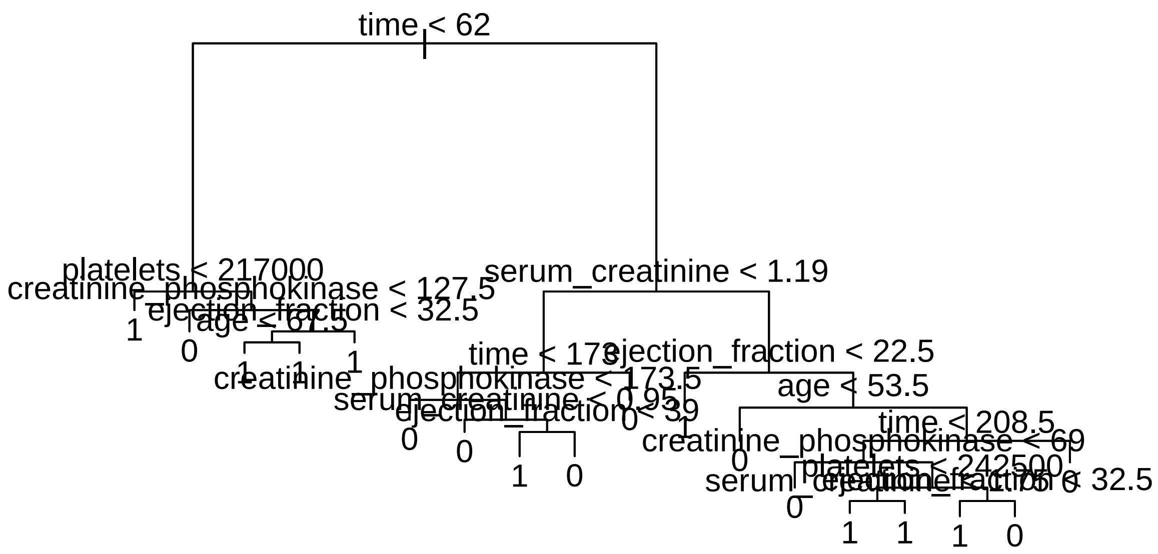
MODEL 3.a - DECISION TREES - ALL VARIABLES - UN-SCALED

```
tree.Heart_disease_unscaled = tree(DEATH_EVENT~.,training_set_unscaled)
summary(tree.Heart_disease_unscaled)
```

```
##
## Classification tree:
## tree(formula = DEATH_EVENT ~ ., data = training_set_unscaled)
## Variables actually used in tree construction:
## [1] "time"                     "platelets"
## [3] "creatinine_phosphokinase" "ejection_fraction"
## [5] "age"                      "serum_creatinine"
## Number of terminal nodes:  18
## Residual mean deviance:  0.3502 = 66.89 / 191
## Misclassification error rate: 0.08134 = 17 / 209
```

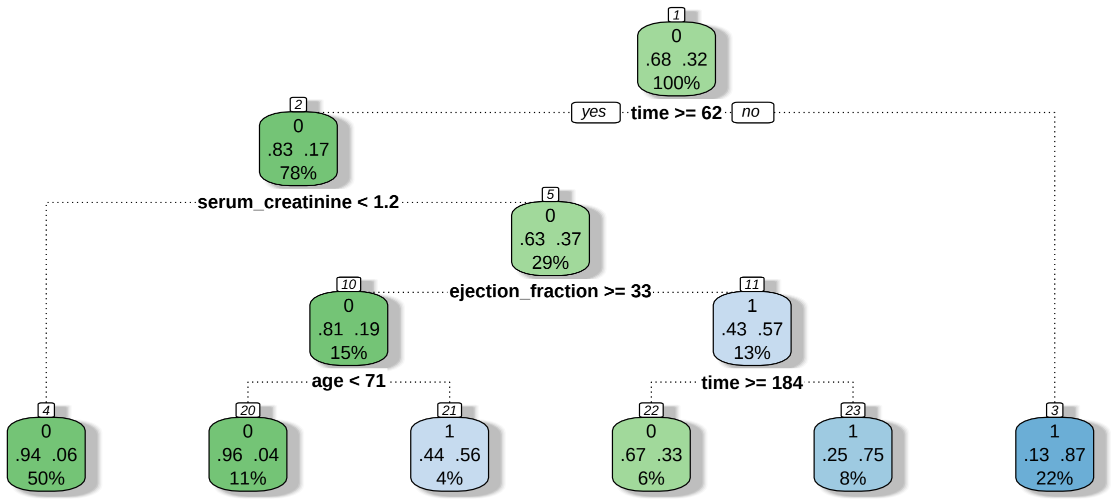
The summary() indicates that 6 of the variables are used in constructing the tree.

```
plot(tree.Heart_disease_unscaled)
text(tree.Heart_disease_unscaled,pretty=0)
```



Plotting the tree

```
#Better tree visualization using rpart package
tree.Heart_disease_unscaled = rpart(DEATH_EVENT~., training_set_unscaled)
fancyRpartPlot(tree.Heart_disease_unscaled)
```



Rattle 2023−May−16 13:31:19 vishaypaka

The type="class" argument specifies that the predicted values should be the class labels rather than probabilities.

We can come up with research question based on the tree: If a patient of age 78 admitted in hospital and stayed more than 70 days and the level of serum creatinine in his blood is 1.5(mg/dL). The percentage of the blood leaving his heart at each contraction is 42. Will the patient be alive or dead?

Confusion matrix for Random forest - un-scaled

```
# Make predictions on the un-scaled test set
predictions <- predict(tree.Heart_disease_unscaled, newdata=testing_set_unscaled,
                       type="class")

confusionMatrix(testing_set_unscaled$DEATH_EVENT, predictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 54  7
##          1  7 22
##
##                Accuracy : 0.8444
##                  95% CI : (0.7528, 0.9123)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 0.0002694
##
##                   Kappa : 0.6439
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.8852
##             Specificity : 0.7586
##          Pos Pred Value : 0.8852
##          Neg Pred Value : 0.7586
##              Prevalence : 0.6778
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6778
##       Balanced Accuracy : 0.8219
##
##        'Positive' Class : 0
##
```

When the un-scaled data is fitted to decision tree model, we obtained the accuracy of 75.5%

MODEL 3 - DECISION TREES

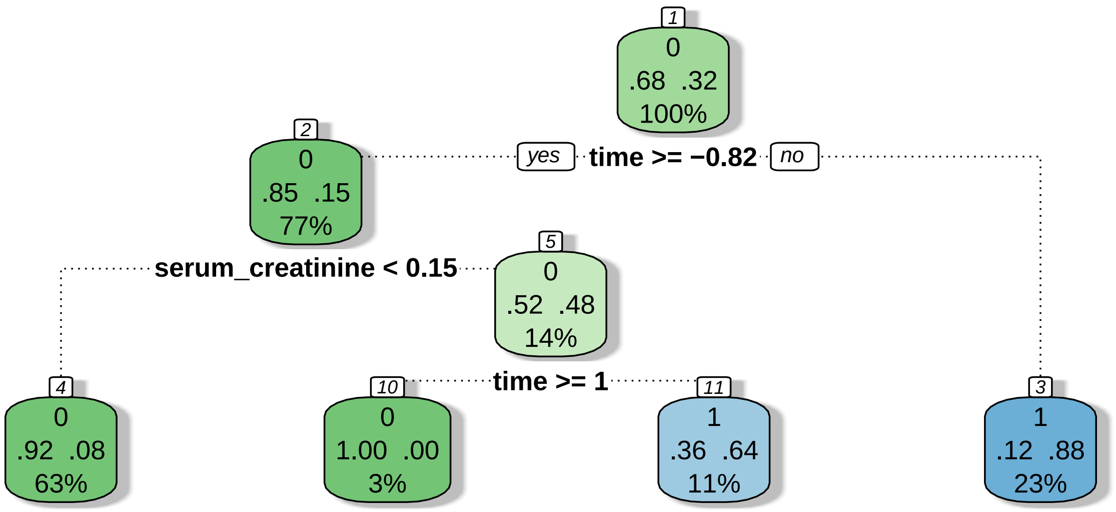
MODEL 3.b - DECISION TREES - ALL VARIABLES - SCALED

```
tree.Heart_disease_scaled = tree(DEATH_EVENT~.,training_set_scaled)
summary(tree.Heart_disease_scaled)
```

```
##
## Classification tree:
## tree(formula = DEATH_EVENT ~ ., data = training_set_scaled)
## Variables actually used in tree construction:
## [1] "time"                     "serum_creatinine"
## [3] "sex"                      "ejection_fraction"
## [5] "age"                      "creatinine_phosphokinase"
## [7] "high_blood_pressure"      "platelets"
## Number of terminal nodes:  14
## Residual mean deviance:  0.3906 = 76.16 / 195
## Misclassification error rate: 0.09569 = 20 / 209
```

Plotting the tree

```
#Better tree visualization using rpart package
tree.Heart_disease_scaled = rpart(DEATH_EVENT~., training_set_scaled)
fancyRpartPlot(tree.Heart_disease_scaled)
```



Rattle 2023−May−16 13:31:19 vishaypaka

### Confusion matrix for Decision tree - scaled

```
# Make predictions on the scaled test set
predictions1 <- predict(tree.Heart_disease_scaled, newdata=testing_set_scaled,
                        type="class")

confusionMatrix(testing_set_scaled$DEATH_EVENT, predictions1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52  9
##          1  8 21
##
##                Accuracy : 0.8111
##                  95% CI : (0.7149, 0.8859)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.001794
##
##                   Kappa : 0.5714
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8667
##             Specificity : 0.7000
##          Pos Pred Value : 0.8525
##          Neg Pred Value : 0.7241
##              Prevalence : 0.6667
##          Detection Rate : 0.5778
##    Detection Prevalence : 0.6778
##       Balanced Accuracy : 0.7833
##
##        'Positive' Class : 0
##
```

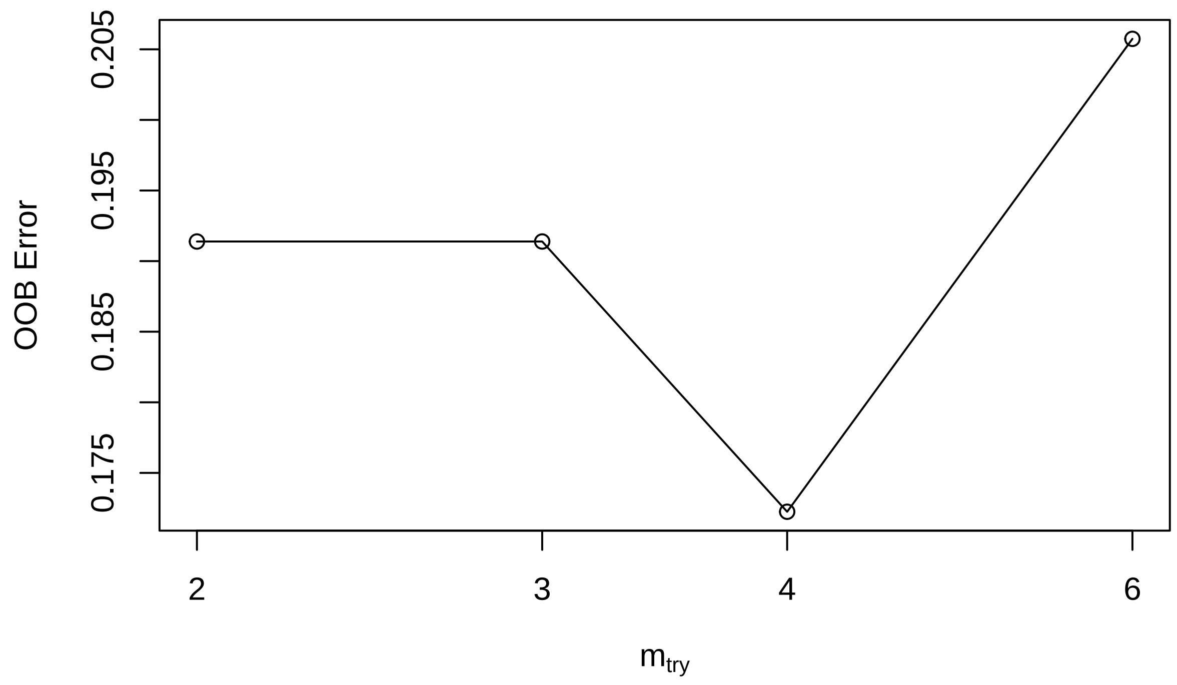
When the scaled data is fitted to decision tree model, we obtained the accuracy of 81.1% We can observe the Decision tree algorithm performs well on un-scaled data compared to scaled data. So, the model with un-scaled data is selected as best model in Decision tree with 84.4 accuracy and will compare it with other models later in this research.

MODEL 4 - RANDOM FOREST ALGORITHM

MODEL 4.a -> RANDOM FOREST ALGORITHM - ALL VARIABLES - UNSCALED

```r
#tuning the mtry parameter using the tuneRF function from the randomForest package.
set.seed(100)
tune.rf <- tuneRF(training_set_unscaled[, -13], training_set_unscaled$DEATH_EVENT,
                  ntreeTry=100,stepFactor=1.5, plot=TRUE, dobest=TRUE)
```

```
## mtry = 3  OOB error = 19.14%
## Searching left ...
## mtry = 2     OOB error = 19.14%
## 0 0.05
## Searching right ...
## mtry = 4     OOB error = 17.22%
## 0.1 0.05
## mtry = 6     OOB error = 20.57%
## -0.1944444 0.05
```



The lower the out-of-bag error rate the best the model gives the accuracy.So, from above chart analysis we can set mtry = 4.

```
set.seed(0)
rf.Heart_disease_unscaled = randomForest(DEATH_EVENT~.,data=training_set_unscaled, mtry=4,
                                         importance=TRUE)

predicted_values_rf = predict(rf.Heart_disease_unscaled,newdata = testing_set_unscaled)
```

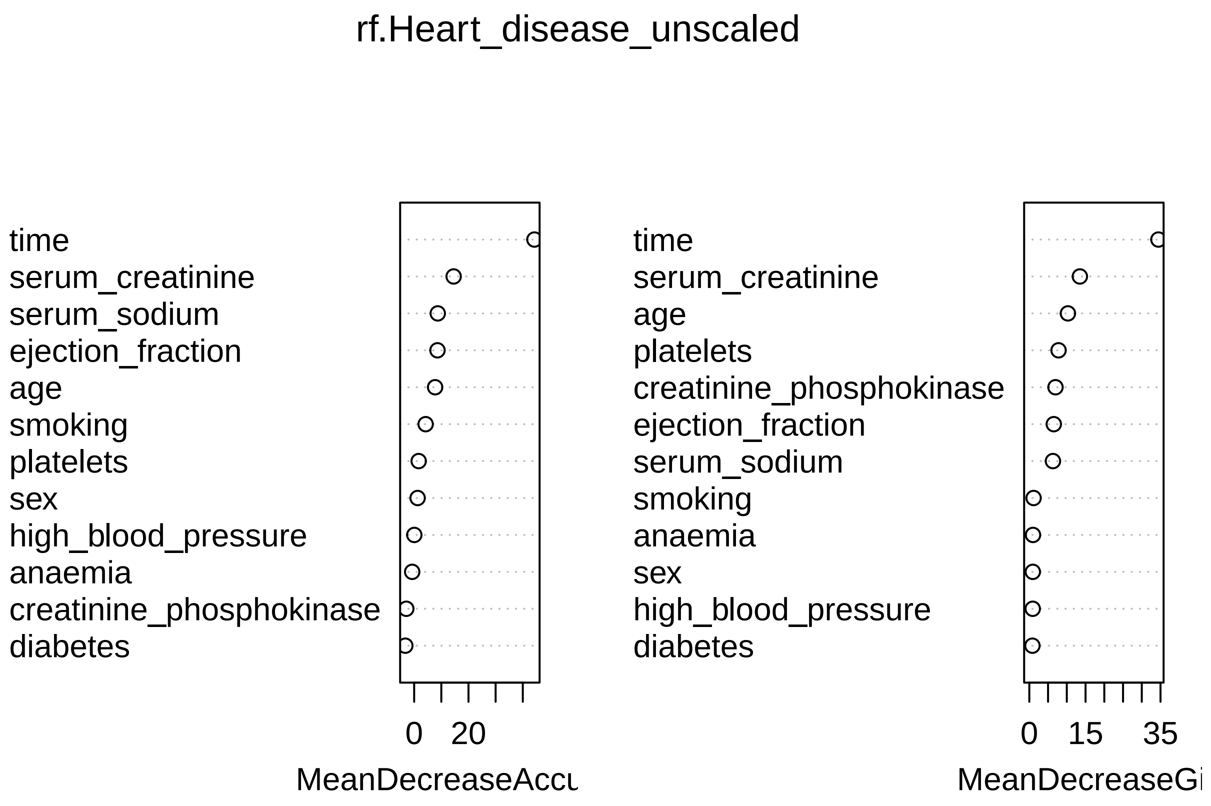
Confusion matrix for Random forest - un-scaled

```
#Confusion Matrix - UNSCALED MODEL
confusionMatrix(predicted_values_rf, testing_set_unscaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57  6
##          1  4 23
##
##                Accuracy : 0.8889
##                  95% CI : (0.8051, 0.9454)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 2.826e-06
##
##                   Kappa : 0.7409
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9344
##             Specificity : 0.7931
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.8519
##              Prevalence : 0.6778
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8638
##
##        'Positive' Class : 0
##
```

When the un-scaled data is fitted to random forest model, we obtained the accuracy of 88.89%

```
# Plot variable importance to determine which variables are most important.
varImpPlot(rf.Heart_disease_unscaled)
```



MODEL 4.b -> RANDOM FOREST ALGORITHM - ALL VARIABLES - SCALED

```r
set.seed(100)
rf.Heart_disease_scaled = randomForest(DEATH_EVENT~.,data=training_set_scaled, mtry=4,
                                       importance=TRUE)

predicted_values_rf1 = predict(rf.Heart_disease_scaled,newdata = testing_set_scaled)
```

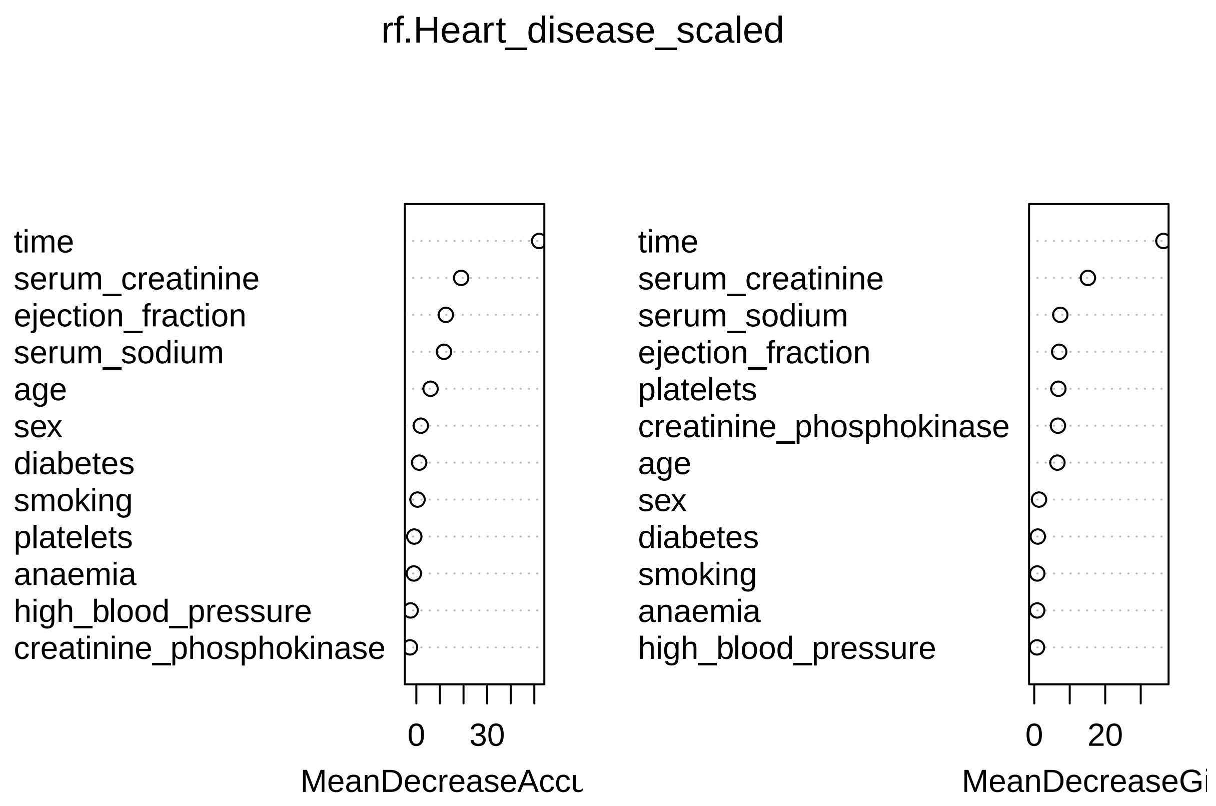
Confusion matrix for Random forest - scaled

```r
#Confusion Matrix - SCALED MODEL
confusionMatrix(predicted_values_rf1, testing_set_scaled$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 50  8
##          1 11 21
##
##                Accuracy : 0.7889
##                  95% CI : (0.6901, 0.8679)
##     No Information Rate : 0.6778
##     P-Value [Acc > NIR] : 0.01376
##
##                   Kappa : 0.5294
##
##  Mcnemar's Test P-Value : 0.64636
##
##             Sensitivity : 0.8197
##             Specificity : 0.7241
##          Pos Pred Value : 0.8621
##          Neg Pred Value : 0.6562
##              Prevalence : 0.6778
##          Detection Rate : 0.5556
##    Detection Prevalence : 0.6444
##       Balanced Accuracy : 0.7719
##
##        'Positive' Class : 0
##
```

When the scaled data is fitted to random forest model, we obtained the accuracy of 78.89% We can observe the random forest algorithm performs well on un-scaled data compared to scaled data. So, the model with un-scaled data is selected as best model in random forest with 88.89 accuracy and will compare it with other models below.

```r
# Plot variable importance to determine which variables are most important.
varImpPlot(rf.Heart_disease_scaled)
```



By comparing both the plots of importance variables from scaled and un-scaled data, we can tell that the predictors : time, serum creatinine, serum sodium,ejection fraction and age are the most important features for the prediction and are most predictive.

COMPARING ALL THE MODELS.

```
#Collecting all the best accuracy and sensitivity(recall) metrics from out all the models(scaled/unscal
acc_lr = 88.89
acc_knn = 85.56
acc_dtree = 84.44
acc_rf = 88.89
tpr_lr = 93.44
tpr_knn = 87.50
tpr_dtree = 88.52
tpr_rf = 93.44
str(Heart_disease)
```

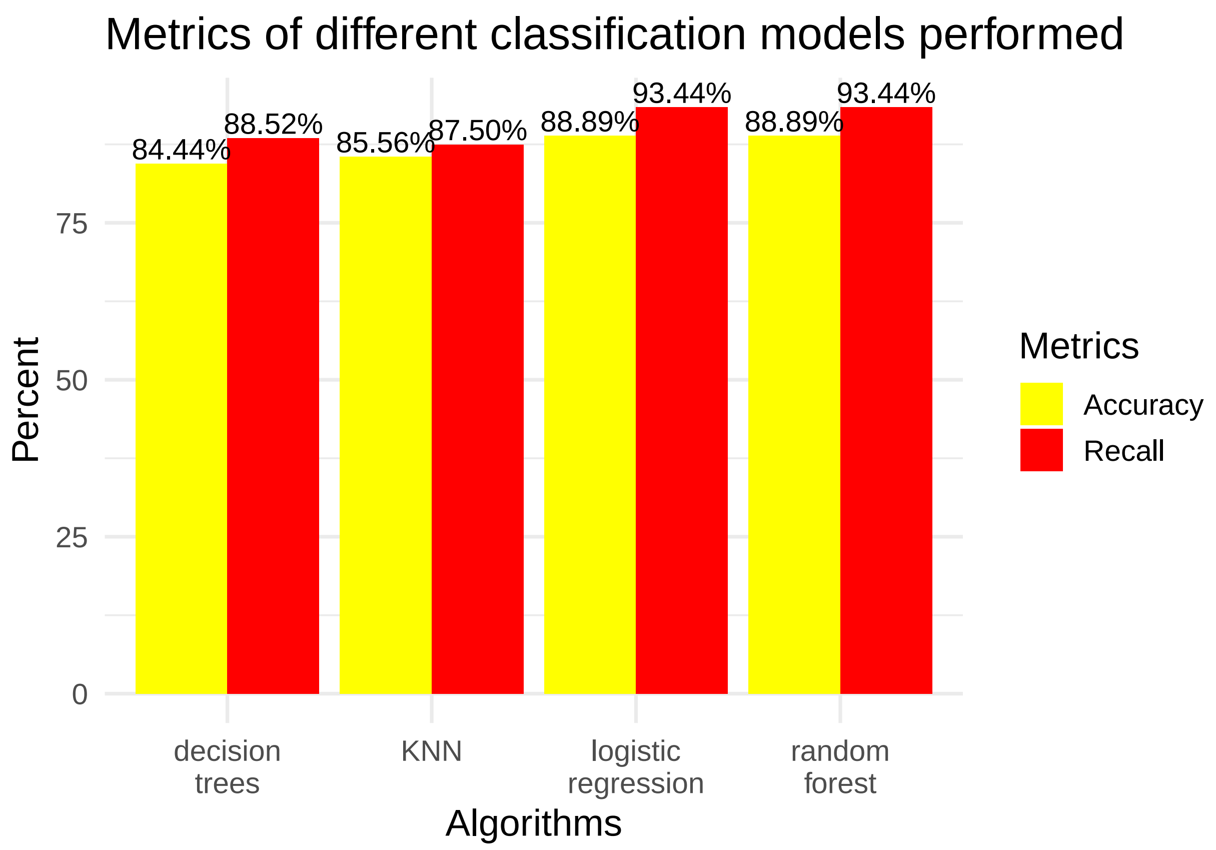
```
## 'data.frame':    299 obs. of  13 variables:
##  $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ anaemia                 : Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 2 1 2 ...
##  $ creatinine_phosphokinase: num  582 7861 146 111 160 ...
##  $ diabetes                : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
##  $ ejection_fraction       : num  20 38 20 20 20 40 15 60 65 35 ...
##  $ high_blood_pressure     : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 1 1 2 ...
##  $ platelets                : num  265000 263358 162000 210000 327000 ...
##  $ serum_creatinine         : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
##  $ serum_sodium             : num  20 38 20 20 20 40 15 60 65 35 ...
##  $ sex                      : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 1 2 ...
##  $ smoking                  : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 1 2 ...
##  $ time                     : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT              : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

PLOTTING THE METRICS OF ALL DIFFERENT CLASSIFICATION MODELS(BEST ONE)

```r
# create a data frame with algorithm names, accuracy, and recall
df <- tibble(
  Algorithm = c("logistic\nregression", "KNN", "decision\ntrees", "random\nforest"),
  Accuracy = c(acc_lr, acc_knn, acc_dtree, acc_rf),
  Recall = c(tpr_lr, tpr_knn, tpr_dtree, tpr_rf)
)

# reshape the data frame into a longer format
df_long <- df %>%
  pivot_longer(cols = -Algorithm, names_to = "Metrics", values_to = "Percent")

# plot the data using ggplot2
 ggplot(df_long,aes(x = reorder(Algorithm, X = Percent),
            y = Percent,
            fill = Metrics)) +
    geom_bar(stat = "identity",
            position = "dodge",
            alpha=1.0) +
    geom_text(aes(group = Metrics, label = str_c(sprintf("%2.2f", Percent), "%")),
             position = position_dodge(width = 0.9), vjust = -0.2) +
    scale_fill_manual(values = c("yellow", "red")) +
    labs(x = "Algorithms", title = "Metrics of different classification models performed") +
    theme_minimal(base_size = 14)
```



We can observe the accuracy and recall(sensitivity) of all the models which are plotted above the bars.

From the above plot we can say that logistic regression is the "best" model in terms of recall, while KNN is the "best" model in terms of accuracy.

Now let's try to do simulation and add some data.

SIMULATION OF THE DATA BASED ON EXISTING DATA

Using random normal distributions with mean and standard deviation values taken from the original Heart_disease dataset. we will create a new dataset of 1000 observations with the same columns as the original, but with different values generated by the random distributions. The new dataset can be used for testing or training machine learning models without having to collect new data. We are creating 1000 observations becuase the simulated dataset should not impact more than the original dataset.

```
set.seed(1400)

N = 1400
simulated_heart_disease = data.frame(
  age = round(rnorm(N,mean(Heart_disease$age),sd(Heart_disease$age))),
  anaemia = sample(c(0, 1), N, replace = TRUE),
  creatinine_phosphokinase = round(rnorm(N,mean(Heart_disease$creatinine_phosphokinase),
                                        sd(Heart_disease$creatinine_phosphokinase))),
  diabetes = sample(c(0, 1), N, replace = TRUE),
  ejection_fraction = round(rnorm(N,mean(Heart_disease$ejection_fraction),
                                 sd(Heart_disease$ejection_fraction))),
  high_blood_pressure = sample(c(0, 1), N, replace = TRUE),
  platelets = round(rnorm(N,mean(Heart_disease$platelets),sd(Heart_disease$platelets))),
  serum_creatinine = round(rnorm(N,mean(Heart_disease$serum_creatinine),
                               sd(Heart_disease$serum_creatinine))),
  serum_sodium = round(rnorm(N,mean(Heart_disease$serum_sodium),
                           sd(Heart_disease$serum_sodium))),
  sex = sample(c(0, 1), N, replace = TRUE),
  smoking = sample(c(0, 1), N, replace = TRUE),
  time = round(rnorm(N,mean(Heart_disease$time),sd(Heart_disease$time)))
)

simulated_heart_disease$anaemia <- factor(simulated_heart_disease$anaemia, levels = c(0,1))
simulated_heart_disease$diabetes <- factor(simulated_heart_disease$diabetes, levels = c(0,1))
simulated_heart_disease$high_blood_pressure <- factor(simulated_heart_disease$high_blood_pressure,
                                                      levels = c(0,1))
simulated_heart_disease$sex <- factor(simulated_heart_disease$sex, levels = c(0,1))
simulated_heart_disease$smoking <- factor(simulated_heart_disease$smoking, levels = c(0,1))


#Fitting the LOGISTIC MODEL
#predicting on the un-scaled testing data
simulated_prediction <- predict(rf.Heart_disease_unscaled, newdata=simulated_heart_disease)
simulated_data <- cbind(simulated_heart_disease, simulated_prediction)
```

Combing heart disease dataset with the obtained simulation dataset

```
colnames(simulated_data)[13] <- "DEATH_EVENT"
heart_disease_simulated = rbind(training_set_unscaled, testing_set_unscaled, simulated_data)
```

SPLITTING INTO TRAINING AND TESTING

```
set.seed(1001)
train_simulated <- sample(nrow(heart_disease_simulated), 0.8 * nrow(heart_disease_simulated))
training_set_simulated <- heart_disease_simulated[train_simulated, ]
testing_set_simulated <- heart_disease_simulated[-train_simulated, ]
```

FITTING THE RANDOM FOREST MODEL AND PREDICTING RESULTS FOR SIMULATED DATA

```
#fitting the un-scaled data on random forest model
set.seed(100)
rf.Heart_disease_simulated_unscaled = randomForest(DEATH_EVENT~.,data=training_set_simulated,
                                                   mtry=6, importance=TRUE)

predict_test_all_simulated = predict(rf.Heart_disease_simulated_unscaled,
                                     newdata = testing_set_simulated)
```

Confusion matrix for full simulated model on random forest.

```
confusionMatrix(predict_test_all_simulated,testing_set_simulated$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 254  13
##          1   8  65
##
##                Accuracy : 0.9382
##                  95% CI : (0.9071, 0.9614)
##     No Information Rate : 0.7706
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8213
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.9695
##             Specificity : 0.8333
##          Pos Pred Value : 0.9513
##          Neg Pred Value : 0.8904
##              Prevalence : 0.7706
##          Detection Rate : 0.7471
##    Detection Prevalence : 0.7853
##       Balanced Accuracy : 0.9014
##
##        'Positive' Class : 0
##
```

From the added simulated data to our original dataset we got an accuracy of 93.82% and and an sensitivity (recall) of 96.95%. Which is very good than all the models we performed above.

FITTING THE LOGISTIC MODEL AND PREDICTING RESULTS

```
#fitting the un-scaled data on logistic regression model
glm.all_simulated = glm(DEATH_EVENT~.,data=training_set_simulated,family="binomial")

#predicting on the testing un-scaled data
predict_test_all_simulated1 <- factor(ifelse(predict(glm.all_simulated,
                                                     testing_set_simulated ,
                                                     type ="response") > 0.5, "1", "0"))
summary(glm.all_simulated)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ ., family = "binomial", data = training_set_simulated)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4635  -0.4706  -0.1486   0.1656   3.4745
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               7.467e-01  7.116e-01   1.049   0.2940
## age                       5.613e-02  8.283e-03   6.776 1.23e-11 ***
## anaemia1                  4.378e-01  1.811e-01   2.418   0.0156 *
## creatinine_phosphokinase  4.395e-04  9.849e-05   4.463 8.10e-06 ***
## diabetes1                -1.795e-01  1.803e-01  -0.996   0.3194
## ejection_fraction        -3.168e-02  7.907e-03  -4.006 6.17e-05 ***
## high_blood_pressure1      3.541e-01  1.805e-01   1.962   0.0498 *
## platelets                -4.996e-07  9.071e-07  -0.551   0.5818
## serum_creatinine          6.232e-01  9.241e-02   6.744 1.54e-11 ***
## serum_sodium             -4.654e-02  7.991e-03  -5.824 5.73e-09 ***
## sex1                     -4.090e-01  1.811e-01  -2.259   0.0239 *
## smoking1                  1.819e-01  1.798e-01   1.011   0.3119
## time                     -3.358e-02  2.092e-03 -16.050  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1579.44  on 1358  degrees of freedom
## Residual deviance:  802.85  on 1346  degrees of freedom
## AIC: 828.85
##
## Number of Fisher Scoring iterations: 6
```

For the added simulated data to original dataset, we can see there are added significant variables to our full model, which are age, sex, ejection fraction, serum creatinine, time and, creatinine phosphokinase.

Confusion matrix for full simulated model

```
confusionMatrix(predict_test_all_simulated1,testing_set_simulated$DEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245  21
##          1  17  57
##
##                Accuracy : 0.8882
##                  95% CI : (0.8498, 0.9197)
##     No Information Rate : 0.7706
##     P-Value [Acc > NIR] : 1.994e-08
##
##                   Kappa : 0.6781
##
##  Mcnemar's Test P-Value : 0.6265
```

```
##
##             Sensitivity : 0.9351
##             Specificity : 0.7308
##          Pos Pred Value : 0.9211
##          Neg Pred Value : 0.7703
##              Prevalence : 0.7706
##          Detection Rate : 0.7206
##    Detection Prevalence : 0.7824
##       Balanced Accuracy : 0.8329
##
##        'Positive' Class : 0
##
```

From the added simulated data to our original dataset we got an accuracy of 88.82% and and an sensitivity (recall) of 93.51% when fitted with logistic model.

So, the Randome forest model fitted with simulated data is very good than logistic model we performed above with an accuracy of 93.82% when fitted with random forest algorithm.

Tabulating the results of all models.

```
Classification_models = c('Logistic Model - All Variables(unscaled)',
                          'Logistic Model - All Variables(scaled)',
                          'Logistic Model - Removing serum_sodium Variable(unscaled)',
                          'Logistic Model - Removing serum_sodium Variable(scaled)',
                          'Logistic Model - Significant Variables(unscaled)',
                          'Logistic Model - Significant Variables(scaled)',
                          'KNN Model - All continuous variables',
                          'KNN Model - significant Continuous variables',
                          'Decision tree Model (unscaled)','Decision tree Model (scaled)',
                          'Random-forest Model (unscaled)','Random-forest Model (scaled)',
                          'Random-forest Model - All Variables (After addidng simulated data)',
                          'Logistic Model - All Variables (After addidng simulated data)')
Accuracy_rates = c('88.89%','83.33%','88.89%','83.33%','88.89%',
                   '82.22%','61.11%','85.56%','84.44%','81.11%','88.89%%','78.89%','93.82%','88.82%')

All_models_comparision = data.frame(Classification_models, Accuracy_rates)
All_models_comparision
```

```
##                                                Classification_models
## 1                           Logistic Model - All Variables(unscaled)
## 2                             Logistic Model - All Variables(scaled)
## 3          Logistic Model - Removing serum_sodium Variable(unscaled)
## 4            Logistic Model - Removing serum_sodium Variable(scaled)
## 5                   Logistic Model - Significant Variables(unscaled)
## 6                     Logistic Model - Significant Variables(scaled)
## 7                               KNN Model - All continuous variables
## 8                       KNN Model - significant Continuous variables
## 9                                     Decision tree Model (unscaled)
## 10                                      Decision tree Model (scaled)
## 11                                    Random-forest Model (unscaled)
## 12                                      Random-forest Model (scaled)
## 13 Random-forest Model - All Variables (After addidng simulated data)
## 14      Logistic Model - All Variables (After addidng simulated data)
##    Accuracy_rates
## 1          88.89%
## 2          83.33%
## 3          88.89%
## 4          83.33%
## 5          88.89%
## 6          82.22%
## 7          61.11%
## 8          85.56%
## 9          84.44%
## 10         81.11%
## 11        88.89%%
## 12         78.89%
## 13         93.82%
## 14         88.82%
```

Performing chi-square test.

```
# Create a contingency table between smoking and death event
smoking_death_table <- table(Heart_disease$smoking, Heart_disease$DEATH_EVENT)

# Print the contingency table
smoking_death_table
```

```
##
##       0   1
##   0 137  66
##   1  66  30
```

```
# Perform the chi-square test
chisq.test(smoking_death_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  smoking_death_table
## X-squared = 0.0073315, df = 1, p-value = 0.9318
```

```
# Create a contingency table between sex and death event
sex_death_table <- table(Heart_disease$sex, Heart_disease$DEATH_EVENT)

# Print the contingency table
sex_death_table
```

```
##
##       0   1
##   0  71  34
##   1 132  62
```

```r
# Perform the chi-square test
chisq.test(sex_death_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  sex_death_table
## X-squared = 0, df = 1, p-value = 1
```

```r
# Create a contingency table between high blood pressure and death event
high_blood_pressure_death_table <- table(Heart_disease$high_blood_pressure,
                                         Heart_disease$DEATH_EVENT)

# Print the contingency table
high_blood_pressure_death_table
```

```
##
##       0   1
##   0 137  57
##   1  66  39
```

```r
# Perform the chi-square test
chisq.test(high_blood_pressure_death_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  high_blood_pressure_death_table
## X-squared = 1.5435, df = 1, p-value = 0.2141
```

```r
# Create a contingency table between diabetes and death event
diabetes_death_table <- table(Heart_disease$diabetes, Heart_disease$DEATH_EVENT)

# Print the contingency table
diabetes_death_table
```

```
##
##       0   1
##   0 118  56
##   1  85  40
```

```r
# Perform the chi-square test
chisq.test(diabetes_death_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  diabetes_death_table
## X-squared = 2.1617e-30, df = 1, p-value = 1
```

```r
# Create a contingency table between anaemia and death event
anaemia_death_table <- table(Heart_disease$anaemia, Heart_disease$DEATH_EVENT)

# Print the contingency table
anaemia_death_table
```

```
##
##       0   1
##   0 120  50
##   1  83  46
```

```r
# Perform the chi-square test
chisq.test(anaemia_death_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  anaemia_death_table
## X-squared = 1.0422, df = 1, p-value = 0.3073
```

Analysing the Chi-square results with death event and making conclusion

```r
Predictor_variable = c("anaemia","diabetes","High blood pressure","sex","smoking")
xsquared = c('0.0073315','0','1.5435','2.1617e-30','1.0422')
pvalue = c('0.9318','1','0.2141','1','0.3073')

Chi_square_df = data.frame(Predictor_variable,xsquared,pvalue)

Chi_square_df
```

```
##    Predictor_variable   xsquared pvalue
## 1             anaemia  0.0073315 0.9318
## 2            diabetes          0      1
## 3 High blood pressure     1.5435 0.2141
## 4                 sex 2.1617e-30      1
## 5             smoking     1.0422 0.3073
```

The above table shows the results of chi-squared tests performed to test the association between various predictor variables (anaemia, diabetes, high blood pressure, sex, smoking) and the death event in the heart disease dataset.

The x-squared value indicates the degree of association between the predictor variable and the outcome variable (death event). A higher value of x-squared indicates a stronger association between the predictor variable and the outcome variable.

Based on the above table, we can conclude that the smoking and diabetes are weakly associated with the death event, as indicated by their relatively low x-squared values and high p-values.

Performing multiple linear regression on time(follow-up days)

To predict the effect of age, ejection fraction, serum creatinine level, and other clinical features on the patient's follow-up period or time to death:

```
# Fit the multiple linear regression model
model <- lm(time ~ age + ejection_fraction + serum_creatinine + anaemia + diabetes +
              high_blood_pressure + sex + smoking +
            +platelets + creatinine_phosphokinase, data = Heart_disease)

# Print the summary of the model
summary(model)
```

```
##
## Call:
## lm(formula = time ~ age + ejection_fraction + serum_creatinine +
##     anaemia + diabetes + high_blood_pressure + sex + smoking +
##     +platelets + creatinine_phosphokinase, data = Heart_disease)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -154.204  -58.029   -6.694   66.368  154.128
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              2.272e+02  3.062e+01   7.420 1.32e-12 ***
## age                     -1.204e+00  3.732e-01  -3.225  0.00141 **
## ejection_fraction        3.558e-01  3.692e-01   0.964  0.33600
## serum_creatinine        -8.681e+00  4.219e+00  -2.057  0.04056 *
## anaemia1                -2.040e+01  8.929e+00  -2.285  0.02306 *
## diabetes1               -1.673e-01  8.925e+00  -0.019  0.98506
## high_blood_pressure1    -2.971e+01  9.114e+00  -3.260  0.00125 **
## sex1                    -5.096e-01  1.040e+01  -0.049  0.96094
## smoking1                -6.917e+00  1.041e+01  -0.664  0.50694
## platelets               -1.585e-06  4.489e-05  -0.035  0.97186
## creatinine_phosphokinase -4.905e-03  4.550e-03  -1.078  0.28192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 74.2 on 288 degrees of freedom
## Multiple R-squared:  0.1168, Adjusted R-squared:  0.0861
## F-statistic: 3.807 on 10 and 288 DF,  p-value: 7.627e-05
```

We can observe from the above summary that age, serum creatinine, anaemia, high blood pressure are statistically significant predictors of the dependent variable. This suggests that older patients, patients with higher serum creatinine levels, anaemic patients, and patients with high blood pressure are more likely to have shorter follow-up periods or higher risk of death. But the adjusted R-squared value is only 0.08615 suggests that the model explains only 8.6% of the variance in the dependent variable. The low adjusted R-squared value suggests that the model may not be a good fit for the data and may not provide accurate predictions.