HarvardX PH125.9x Data Science Capstone Project: Capstone Rmd Heart Failure Survival Prediction

Kingsley Sam

10 April 2021

Content

1.Introduction
2.Method
2.1. Tidying the Dataset
2.2. Statistical quantitative description of the variables
2.3. Data Exploration
2.4. Distributions of each variable
2.5. Selection of Meaningful Features
3.Modeling
4.Result and Discussion
5.Conclusion
6.Reference

1.Introduction

Heart failure is affecting more than 26 million people worldwide and is increasing the prevalence[1]. When heart failure progresses, the heart becomes less effective to pump blood to the aorta. With the reduced cardiac output, perfusion to end-organs becomes insufficient, which eventually leads to fatal outcome.

New York Heart Association(NYHA) functional classification is used to categorizing patients from Class I to Class IV according to the clinical signs and symptoms to indicate the disease severity. Kaplan Meier plot is used to studying the general pattern of survival from censoring data over a period of time for different patient groups in the study. By applying supervised machine learning methodologies to the dataset obtained from electronic health records, we will be able to develop algorithms to predict survival of patients based on variables in the dataset.

In this project, the Heart Failure Clinical Records Data Set will be used to provide a survival predict model of patient with heart failure. The data set can be found at https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records. The original dataset version was collected by Government College University in Pakistan.The current version of the dataset was elaborated by Krembil Research Institute in Toronto and donated to the University of California Irvine Machine Learning Repository in 2020.

2. Methods and Analysis

2.1 Tidying the Dataset

The Heart failure clinical records Data Set was downloaded from UCI repository for machine learning. This data set is tidy without missing data when we use "view" functiont to check the data visually. Since the dataset is relatively small in size, visual check is feasible and quick. The column names have been defined when we load the dataset in local environment. This dataset contains the medical records of 299 patients who had heart failure, collected during their follow-up period, where each patient profile has 13 clinical features in the following.

```
# Download dateset from UCI repository for machine learning
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00519/heart_failure_clinical_r
```

```
              "heart_failure_clinical_records_dataset.csv")

# Load the dataset with named columns
data_columns <- c("age", "anaemia", "hbp", "CPK", "diabetes", "ef", "platelets",
                  "sex","serum_creatinine", "serum_Na", "smoking",
                  "time_fu_period", "death")
data<- read.csv ("heart_failure_clinical_records_dataset.csv", sep="," ,
                 header = TRUE)

#Visual check the dataset
view(data)

#2.2 Statistical quantitative description of the variables
str(data)
```

```
## '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 ...
```

```
dim(data)
```

```
## [1] 299  13
```

```
head(data)
```

```
##   age anaemia creatinine_phosphokinase diabetes ejection_fraction
## 1  75       0                     582        0                20
## 2  55       0                    7861        0                38
## 3  65       0                     146        0                20
## 4  50       1                     111        0                20
## 5  65       1                     160        1                20
## 6  90       1                      47        0                40
##   high_blood_pressure platelets serum_creatinine serum_sodium sex smoking time
## 1                   1    265000             1.9          130   1       0    4
## 2                   0    263358             1.1          136   1       0    6
## 3                   0    162000             1.3          129   1       1    7
## 4                   0    210000             1.9          137   1       0    7
## 5                   0    327000             2.7          116   0       0    8
## 6                   1    204000             2.1          132   1       1    8
##   DEATH_EVENT
## 1           1
## 2           1
## 3           1
## 4           1
```

```
## 5           1
## 6           1
```

```r
summary(data)
```

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

2.3 Data Exploration

To explore the distribution pattern of each independent variables against the death event, which implied the potential influence on prediction model. Afterward we will consider which columns of the dataset will be kept for the analysis. The dependent variable is death event which is a categorical variables and the target to be predicted by the 12 variables in dataset. We applied ggplot function to plot the distribution of each variable in regards to death event.

Density plots are used for the 7 continuous independent variables, which included age, creatinine phosphokinase, ejection fraction, platelets, serum creatinine, serum sodium and time.

```r
# Vizualize the density distributions for Death cases against the variables
# the available continuous features
# Create a function for the density plots
density_plot <- function(column, param_name){
  ggplot(data, aes(x=column, fill=DEATH_EVENT, color=DEATH_EVENT)) +
    geom_density(alpha=0.2) +
    theme(legend.position="bottom") +
    scale_x_continuous(name=param_name) +
    scale_fill_discrete(name='DEATH EVENT',labels=c("No", "Yes")) +
    scale_color_discrete(name='DEATH EVENT',labels=c("No", "Yes"))
}
```

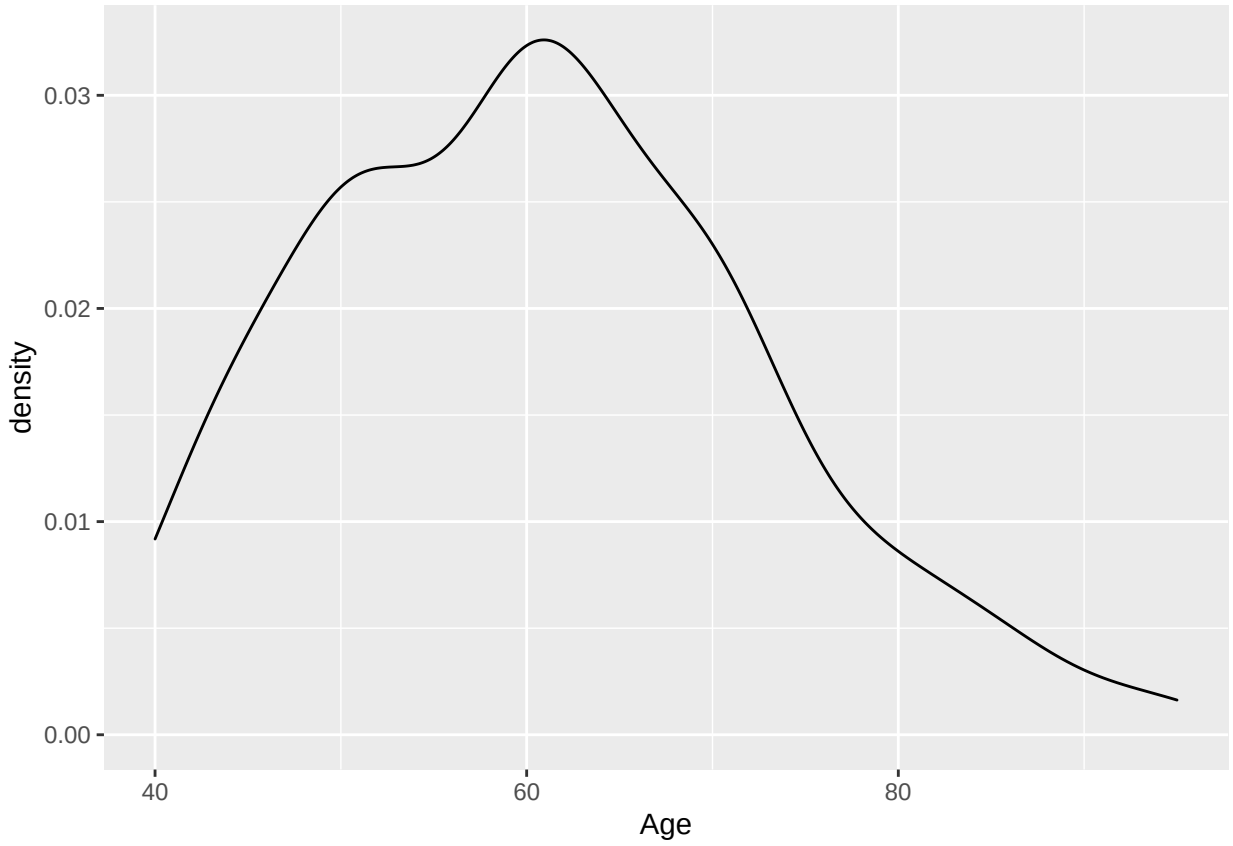
Stacked barplots are used for the 5 categorical independent variables, which included anaemia, high blood pressure, diabetes, sex, and smoking.

```
# Vizualize the density distributions for Death cases against the variables
# the available categorical features
# Create a function for the barplots
format_barplot <- function(gc, columngroup, param_name, labelling){
  ggplot(gc, aes(x=columngroup, y=n, fill=DEATH_EVENT))+
    geom_bar( stat="identity") +
    scale_x_discrete(name=param_name, labels=labelling) +
    scale_fill_discrete(name='DEATH EVENT', labels=c("No", "Yes")) +
    scale_color_discrete(name='DEATH EVENT', labels=c("No", "Yes")) +
    theme(legend.position="top")
}
```

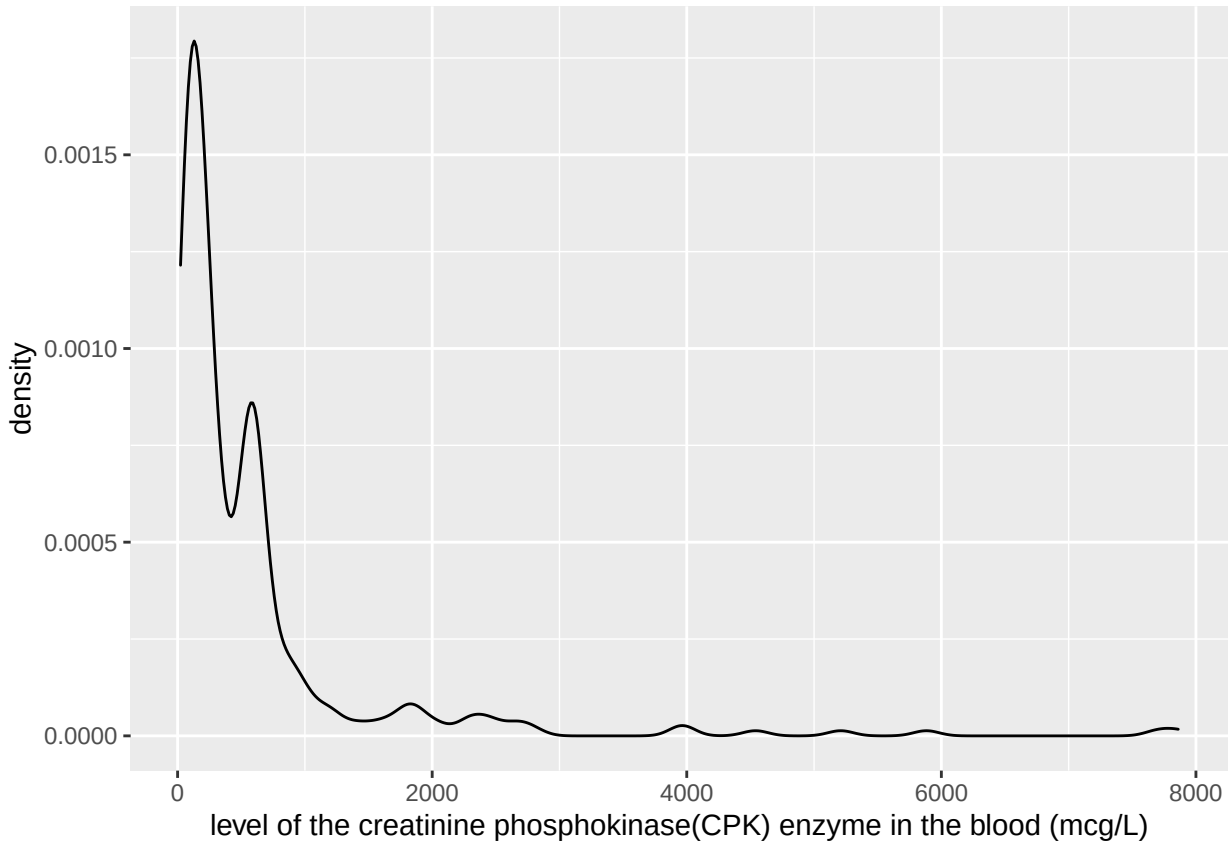
2.4 Distributions of each variable

The second function is applied to categorical variables to convert from numeric to factor and create stacked barplots.

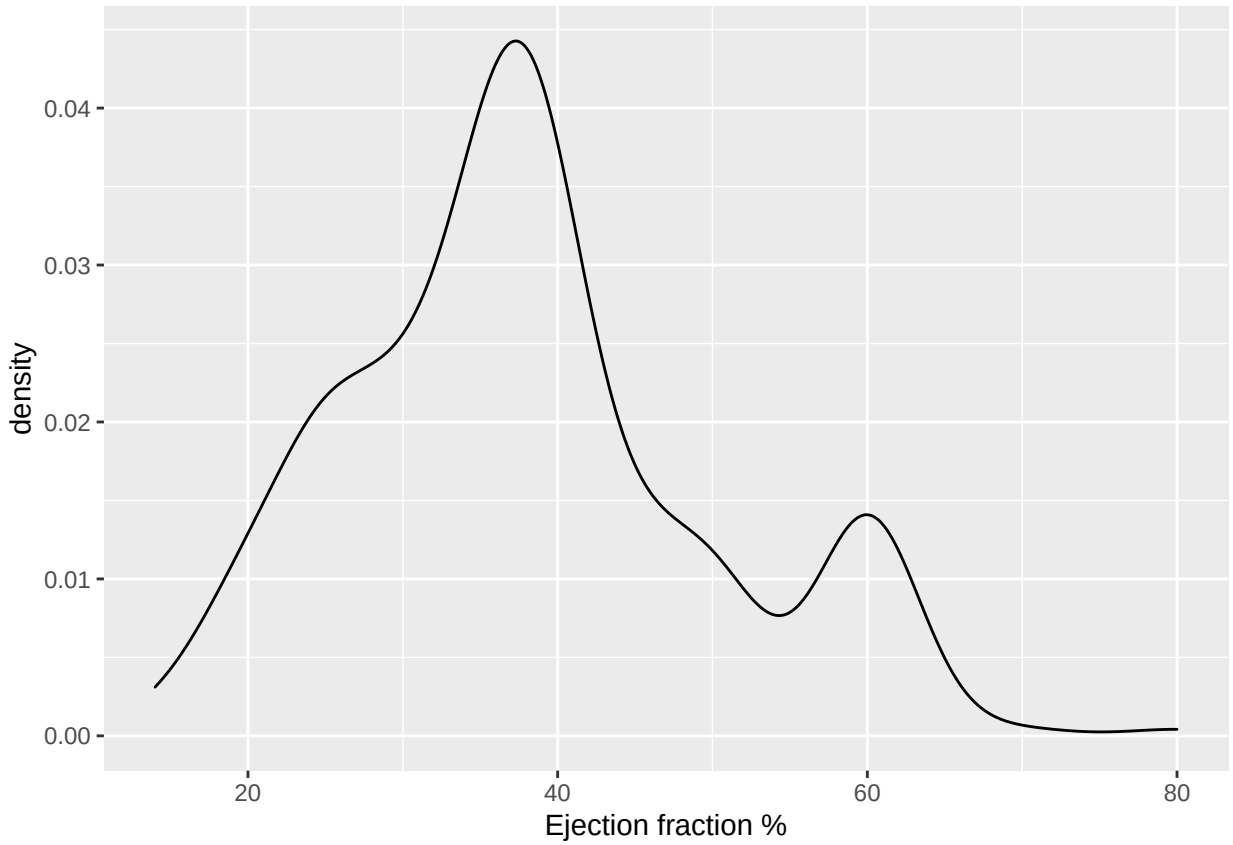
```
# Plot for all continuous variables
plotAge <- density_plot(data$age, "Age")
plotAge
```



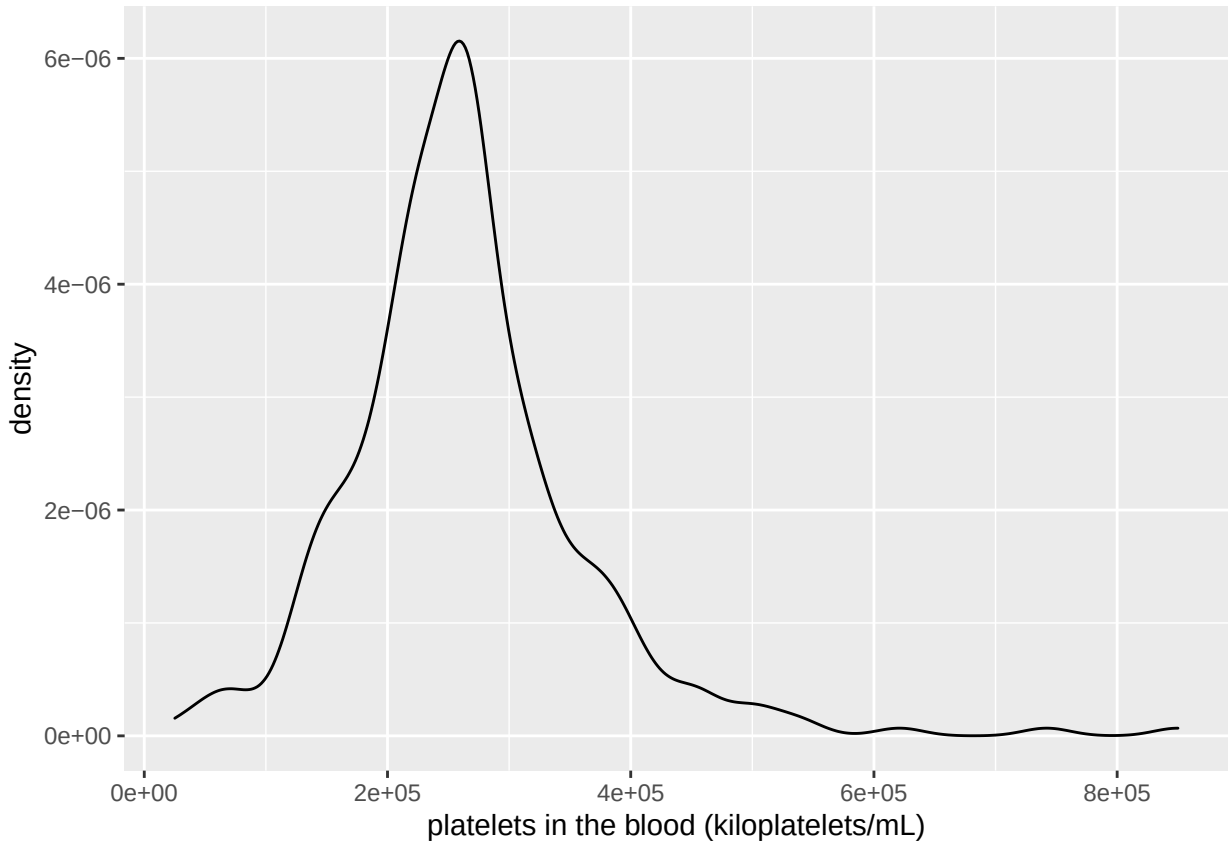

```
plotCreatinine_phosphokinase<-
  density_plot(data$creatinine_phosphokinase,
               "level of the creatinine phosphokinase(CPK) enzyme in the blood (mcg/L)")
plotCreatinine_phosphokinase
```



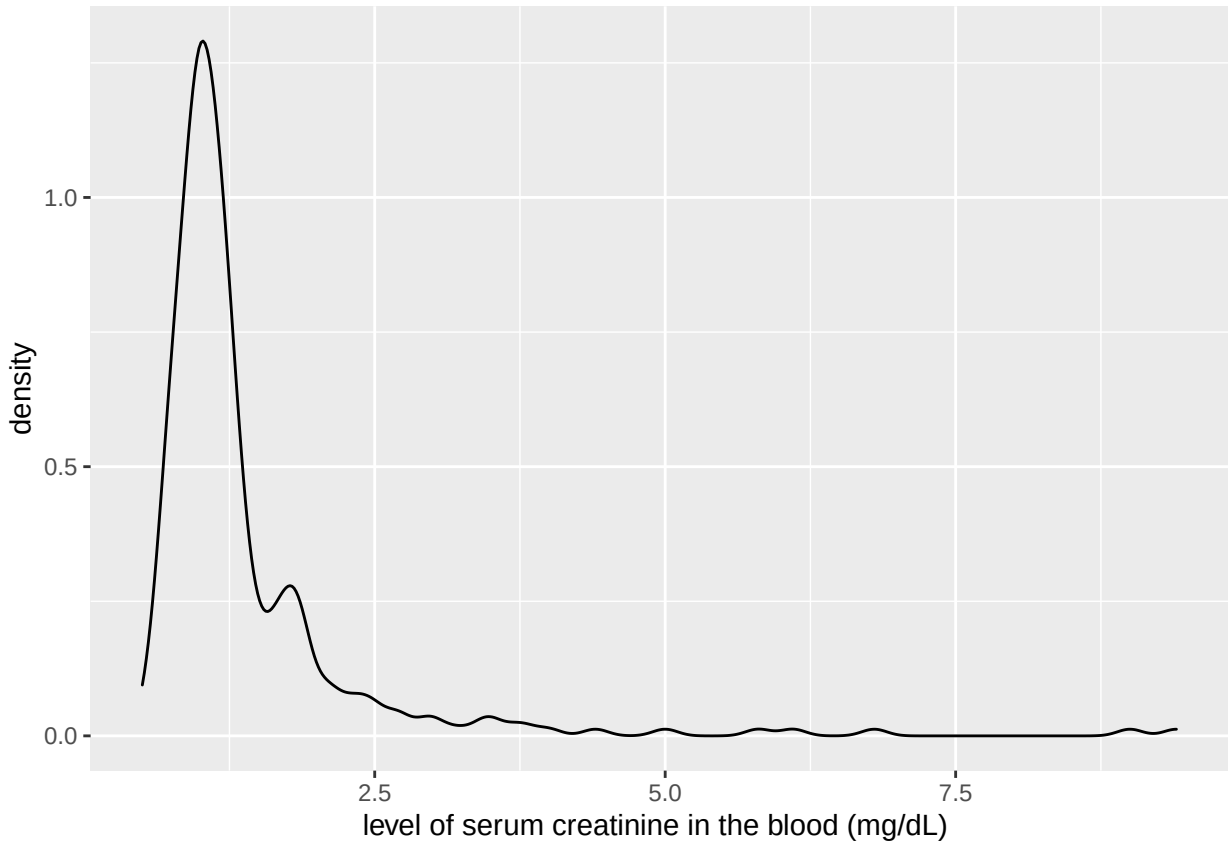

```
plotEjection_fraction<-density_plot(data$ejection_fraction, "Ejection fraction %")
plotEjection_fraction
```



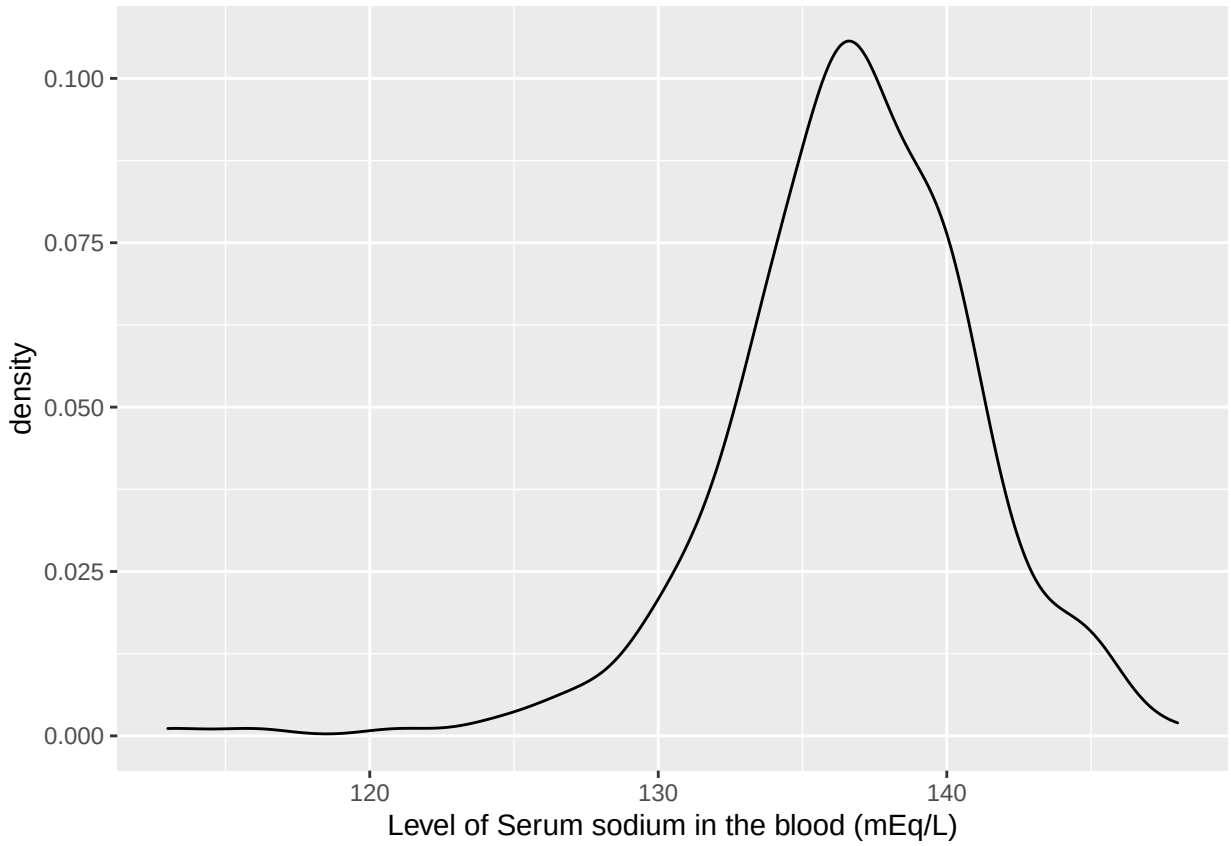

```
plotPlatelets <- density_plot(data$platelets,
                              "platelets in the blood (kiloplatelets/mL)")
plotPlatelets
```



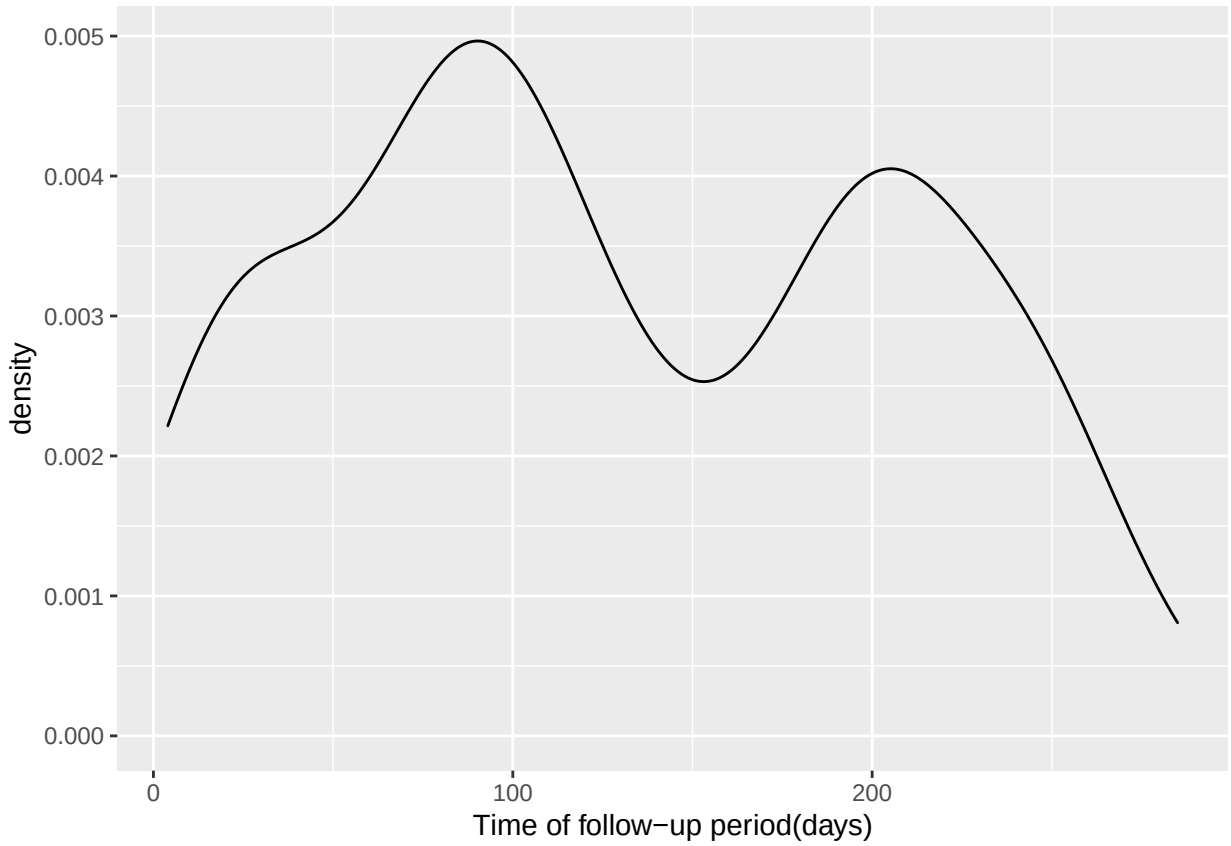

```
plotSerum_creatinine <- density_plot(data$serum_creatinine,
                                     "level of serum creatinine in the blood (mg/dL) ")
plotSerum_creatinine
```



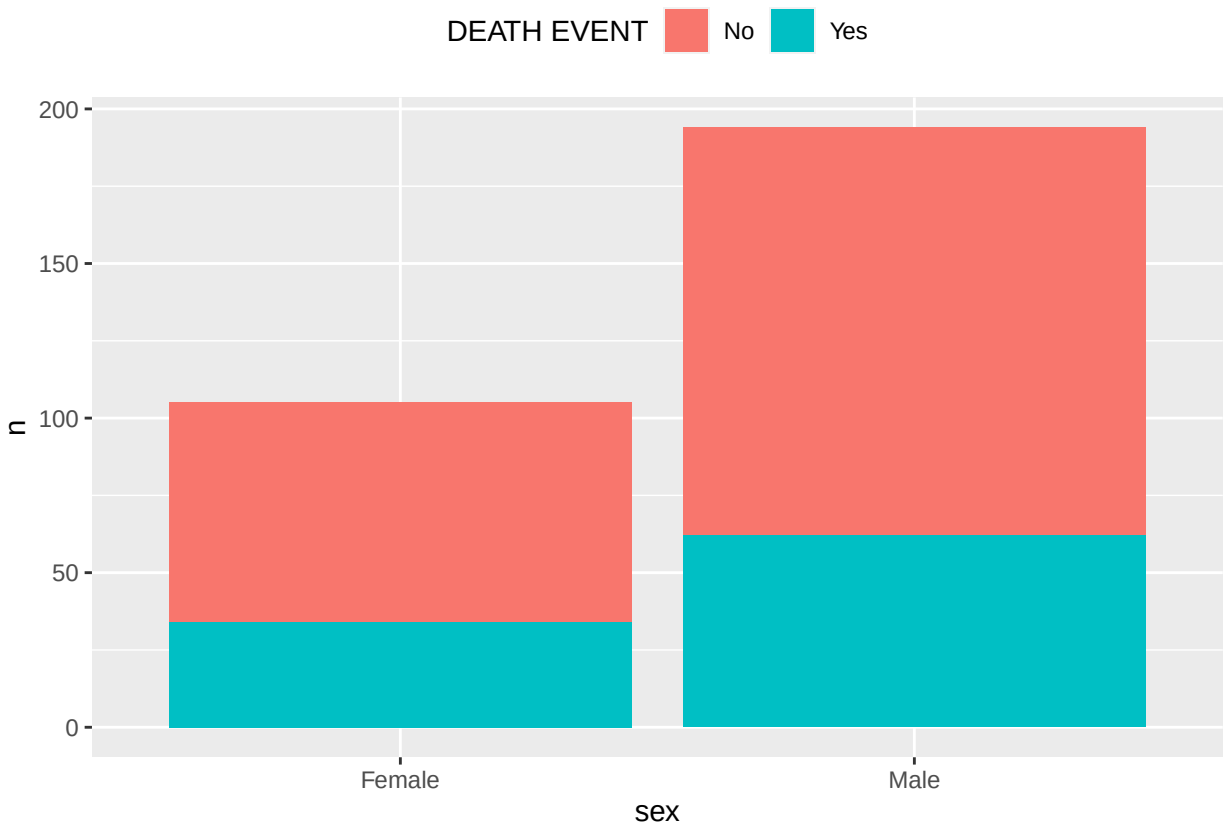

```
plotSerum_sodium <- density_plot(data$serum_sodium,
                                 "Level of Serum sodium in the blood (mEq/L)")
plotSerum_sodium
```



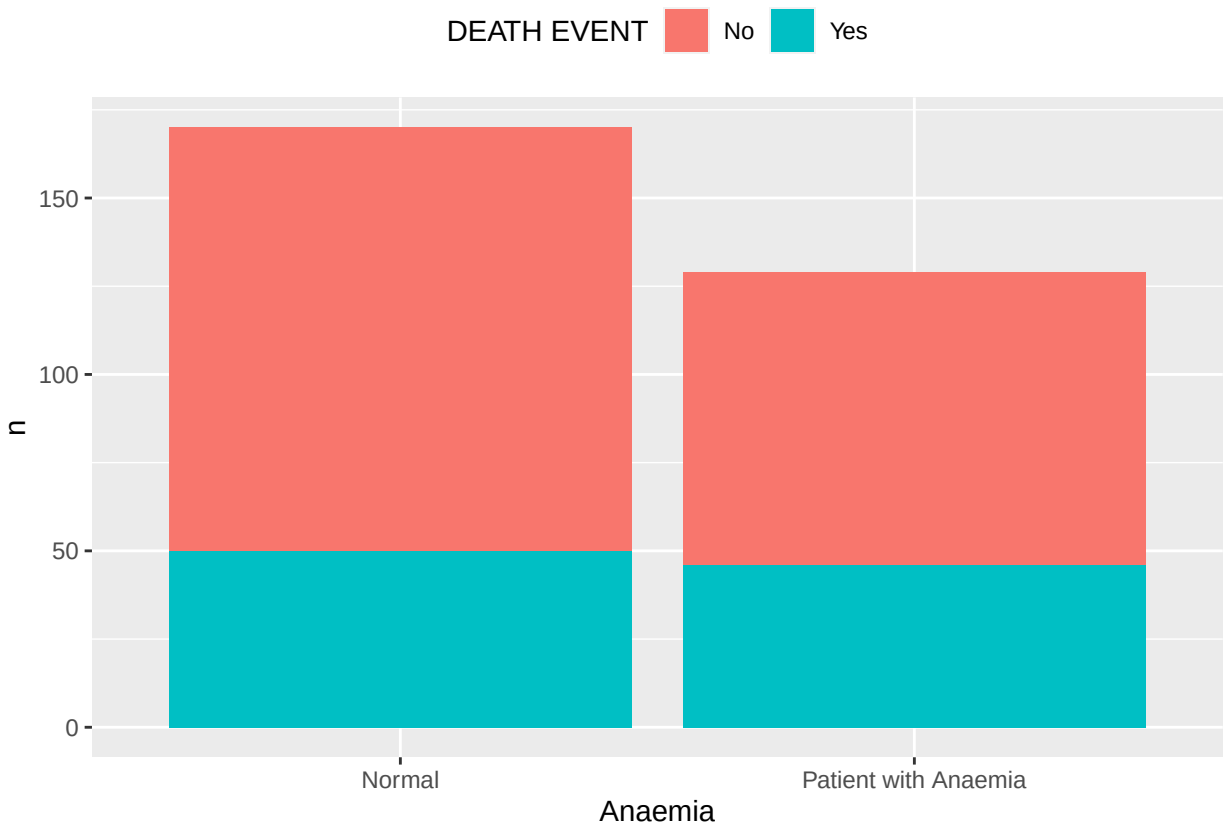

```
plotTime <- density_plot(data$time, "Time of follow-up period(days)")
plotTime
```



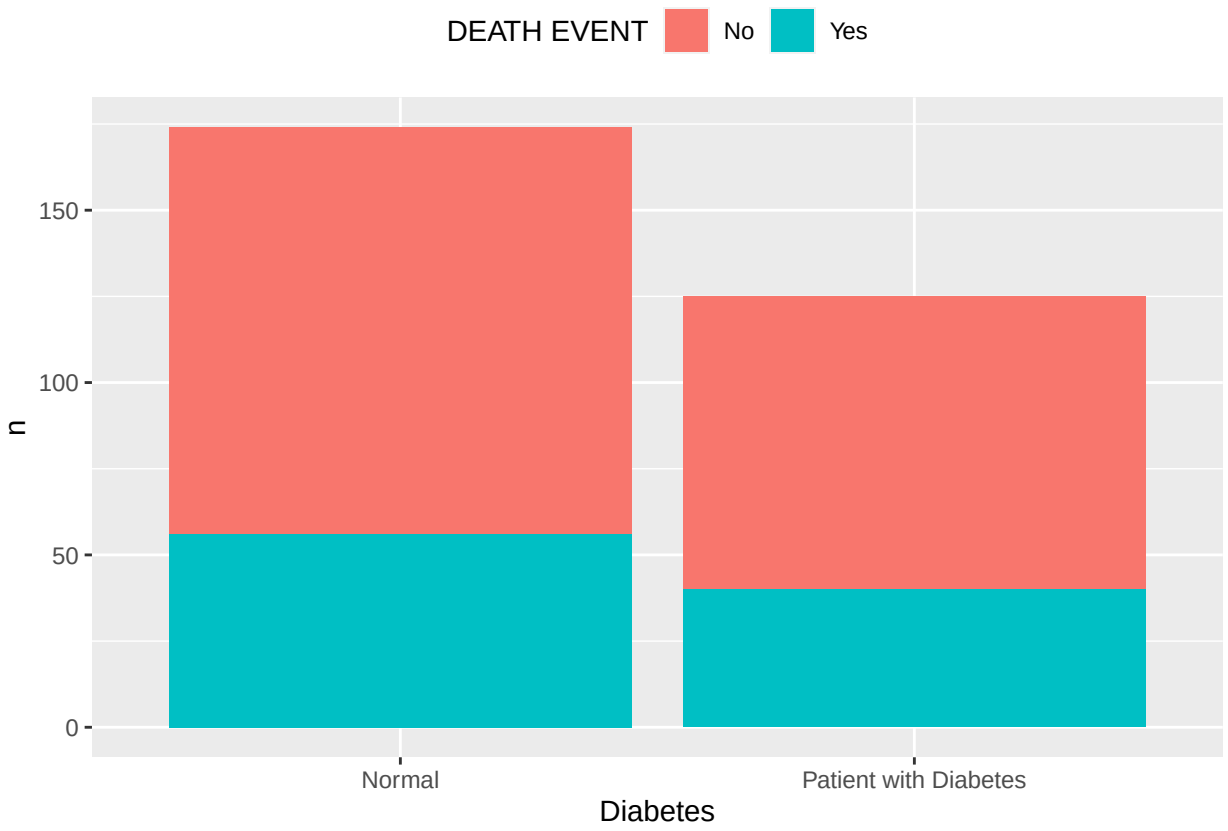

```r
# Convert categorical variables from numeric to factor.
cols <- c("anaemia", "diabetes", "high_blood_pressure", "sex", "smoking",
          "DEATH_EVENT")
data[cols] <- lapply(data[cols], factor)


# Plot for all categorical features
groupby_sex <- data %>% group_by(sex) %>% count(DEATH_EVENT) %>% as.data.frame()
plotSex <- format_barplot(groupby_sex, groupby_sex$sex, "sex",
                          c("Female", "Male"))
plotSex
```



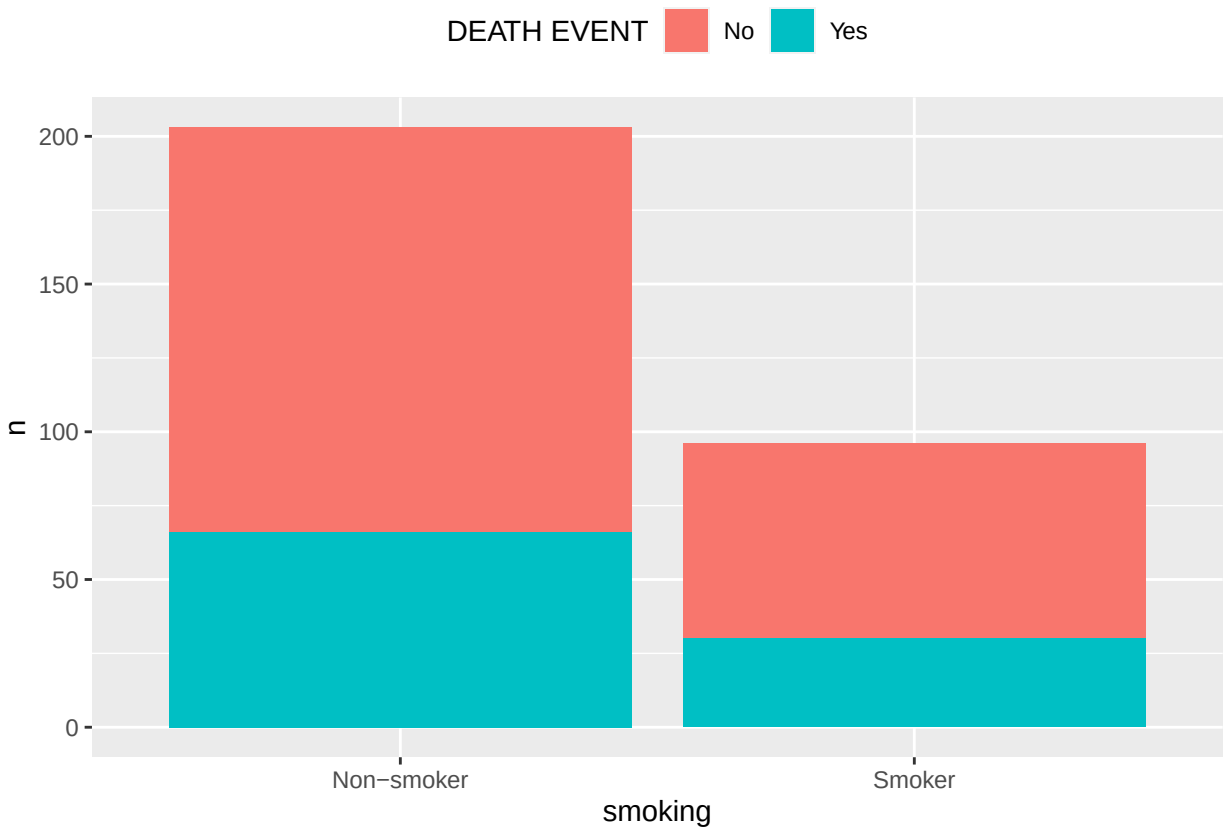

```
groupby_anaemia <- data %>% group_by(anaemia) %>%
  count(DEATH_EVENT) %>% as.data.frame()
plotAnaemia <- format_barplot(groupby_anaemia, groupby_anaemia$anaemia,
                              "Anaemia", c("Normal", "Patient with Anaemia"))
plotAnaemia
```



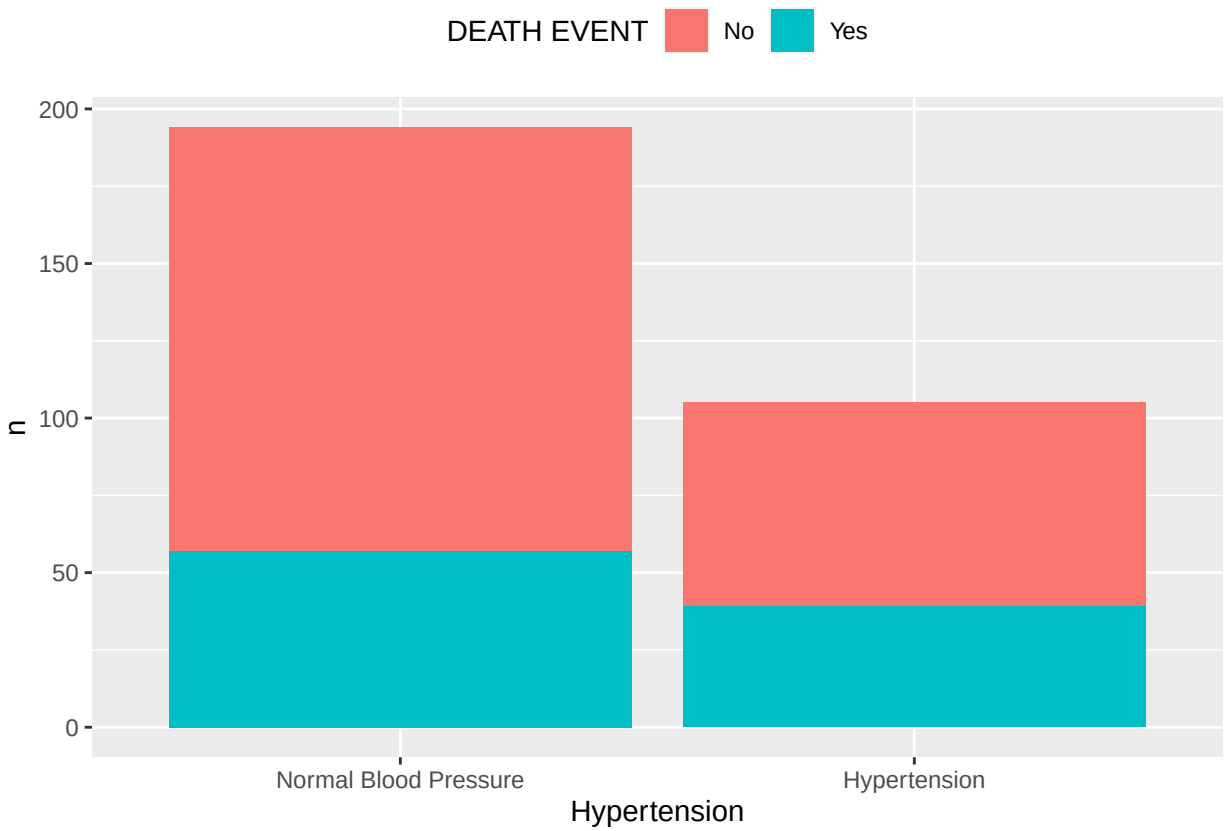

```
groupby_diabetes <- data %>% group_by(diabetes) %>% count(DEATH_EVENT) %>%
  as.data.frame()
plotDiabetes <- format_barplot(groupby_diabetes, groupby_diabetes$diabetes,
                               "Diabetes", c("Normal", "Patient with Diabetes"))
plotDiabetes
```




```
groupby_smoking <- data %>% group_by(smoking) %>% count(DEATH_EVENT) %>%
  as.data.frame()
plotSmoking <- format_barplot(groupby_smoking, groupby_smoking$smoking,
                              "smoking", c("Non-smoker", "Smoker"))
plotSmoking
```




```
groupby_hbp <- data %>% group_by(high_blood_pressure) %>% count(DEATH_EVENT) %>%
  as.data.frame()
plotHBP <- format_barplot(groupby_hbp, groupby_hbp$high_blood_pressure,
                          "Hypertension", c("Normal Blood Pressure",
                                            "Hypertension"))
plotHBP
```



Chi-Square Test for each independent categorical variable to show their association with the dependent variable death event.

```
# Chi-Square Test for each independent categorical variable
# sex
table(data$DEATH_EVENT, data$sex)
```

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

```
chisq.test(data$DEATH_EVENT, data$sex, correct=FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$DEATH_EVENT and data$sex
## X-squared = 0.0055707, df = 1, p-value = 0.9405
```

```
# anaemia
table(data$DEATH_EVENT, data$anaemia)
```

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

```
chisq.test(data$DEATH_EVENT, data$anaemia, correct=FALSE)
```

```
##
```

```
##  Pearson's Chi-squared test
##
## data:  data$DEATH_EVENT and data$anaemia
## X-squared = 1.3131, df = 1, p-value = 0.2518
```

```
# diabetes
table(data$DEATH_EVENT, data$diabetes)
```

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

```
chisq.test(data$DEATH_EVENT, data$diabetes, correct=FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$DEATH_EVENT and data$diabetes
## X-squared = 0.0011287, df = 1, p-value = 0.9732
```

```
# smoking
table(data$DEATH_EVENT, data$smoking)
```

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

```
chisq.test(data$DEATH_EVENT, data$smoking, correct=FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$DEATH_EVENT and data$smoking
## X-squared = 0.047644, df = 1, p-value = 0.8272
```

```
# sex
table(data$DEATH_EVENT, data$high_blood_pressure)
```

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

```
chisq.test(data$DEATH_EVENT, data$high_blood_pressure, correct=FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$DEATH_EVENT and data$high_blood_pressure
## X-squared = 1.8827, df = 1, p-value = 0.17
```

```
#ejection_fraction
data$DEATH_EVENT <- as.numeric(data$DEATH_EVENT)
cor.test(data$ejection_fraction, data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
```

```
##
## data:  data$ejection_fraction and data$DEATH_EVENT
## t = -4.8056, df = 297, p-value = 2.453e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3707381 -0.1600493
## sample estimates:
##        cor
## -0.2686033
```

```
# high_blood_pressure
table(data$DEATH_EVENT, data$high_blood_pressure)
```

```
##
##       0   1
##   1 137  66
##   2  57  39
```

```
chisq.test(data$DEATH_EVENT, data$high_blood_pressure, correct=FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$DEATH_EVENT and data$high_blood_pressure
## X-squared = 1.8827, df = 1, p-value = 0.17
```

Pearson's correlation for each independent continuous variable to show their association with the dependent variable death event.

```
# Pearson's correlation for each independent continuous variable
#age
cor.test(data$age, data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$age and data$DEATH_EVENT
## t = 4.5206, df = 297, p-value = 8.917e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1444557 0.3568875
## sample estimates:
##       cor
## 0.2537285
```

```
#creatinine_phosphokinase
cor.test(data$creatinine_phosphokinase, data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$creatinine_phosphokinase and data$DEATH_EVENT
## t = 1.0832, df = 297, p-value = 0.2796
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.05106551  0.17491392
## sample estimates:
##        cor
```

```
## 0.06272816
```

```
#Platelets
cor.test(data$platelets , data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$platelets and data$DEATH_EVENT
## t = -0.84787, df = 297, p-value = 0.3972
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.16166808  0.06465181
## sample estimates:
##         cor
## -0.04913887
```

```
#Serum_creatinine
cor.test(data$serum_creatinine, data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$serum_creatinine and data$DEATH_EVENT
## t = 5.3065, df = 297, p-value = 2.19e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1870924 0.3945382
## sample estimates:
##       cor
## 0.2942776
```

```
#serum_sodium
cor.test(data$serum_sodium, data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$serum_sodium and data$DEATH_EVENT
## t = -3.4301, df = 297, p-value = 0.0006889
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3019482 -0.0836249
## sample estimates:
##        cor
## -0.1952036
```

```
#time
cor.test(data$time, data$DEATH_EVENT)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$time and data$DEATH_EVENT
## t = -10.686, df = 297, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
```

```
##  -0.6042744 -0.4398233
## sample estimates:
##        cor
## -0.5269638
```

2.5 Selection of Meaningful Features

The correlation between variable and death event is considered to be statistically significant when p-value <0.05 in Chi-square test or Pearson's correlation. P-value <0.05 was a commonly used threshold in biostatistic research. According to the findings in the following, there are 5 variables statistically significant in the correlation to the death event. The variables are age(p-value = 8.917e-06), ejection fraction(p-value = 2.453e-06), serum creatinine(p-value = 2.19e-07), serum sodium(p-value = 0.0006889) and time of follow up period(p-value < 2.2e-16).
Having identified these 5 variables, we can filter the dataset and prepare for analysis using this optimized, tidy version. The dataset contains 299 observations, each containing 5 independent variables as the predictors as well as the death event which is the outcome trying to predict.
We only keep the 5 statistically significant independent variables and the dependent variable in the a clean dataset. For the dependent variable DEATH_EVENT, is it required to be factor with 2 levels in the confusion matrix of machine learning models.

```
# Keep the statistically significant independent variables and the dependent
# variable in the a clean dataset
keep_columns <- c(1, 5, 8, 9, 12, 13)
data_clean <- data[, keep_columns]
dim(data_clean)
```

```
## [1] 299   6
```

```
str(data_clean)
```

```
## 'data.frame':    299 obs. of  6 variables:
##  $ age              : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ ejection_fraction: int  20 38 20 20 20 40 15 60 65 35 ...
##  $ 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 ...
##  $ time             : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT      : num  2 2 2 2 2 2 2 2 2 2 ...
```

```
view(data_clean)

# We are now ready to select a machine learning algorithm to create a prediction
# model for our datasets.
cols <- c(6)
data_clean[cols] <- lapply(data_clean[cols], factor)
str(data_clean)
```

```
## 'data.frame':    299 obs. of  6 variables:
##  $ age              : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ ejection_fraction: int  20 38 20 20 20 40 15 60 65 35 ...
##  $ 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 ...
##  $ time             : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT      : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 2 ...
```

3. Modeling

Creating the Training and Testing Sets
In order to predict heart disease in patients, we must separate the dataset into a training and a testing set,

each containing different observations. 20% of the dataset is thus assigned to the testing set.

```
# The testing set will be 20% of the orignal dataset.
set.seed(1)
index <- createDataPartition(y = data_clean$DEATH_EVENT, times = 1, p = 0.2,
                             list = FALSE)
trainingSet <- data_clean[-index,]
testingSet <- data_clean[index,]
```

We train a k-nearest neighbor algorithm with a tunegrid parameter to optimize for k

```
library(caret)
library(e1071)
set.seed(1000)
train_knn <- train(DEATH_EVENT ~ ., method = "knn",
                   data = trainingSet,
                   tuneGrid = data.frame(k = seq(2, 30, 2)))
train_knn$bestTune
```

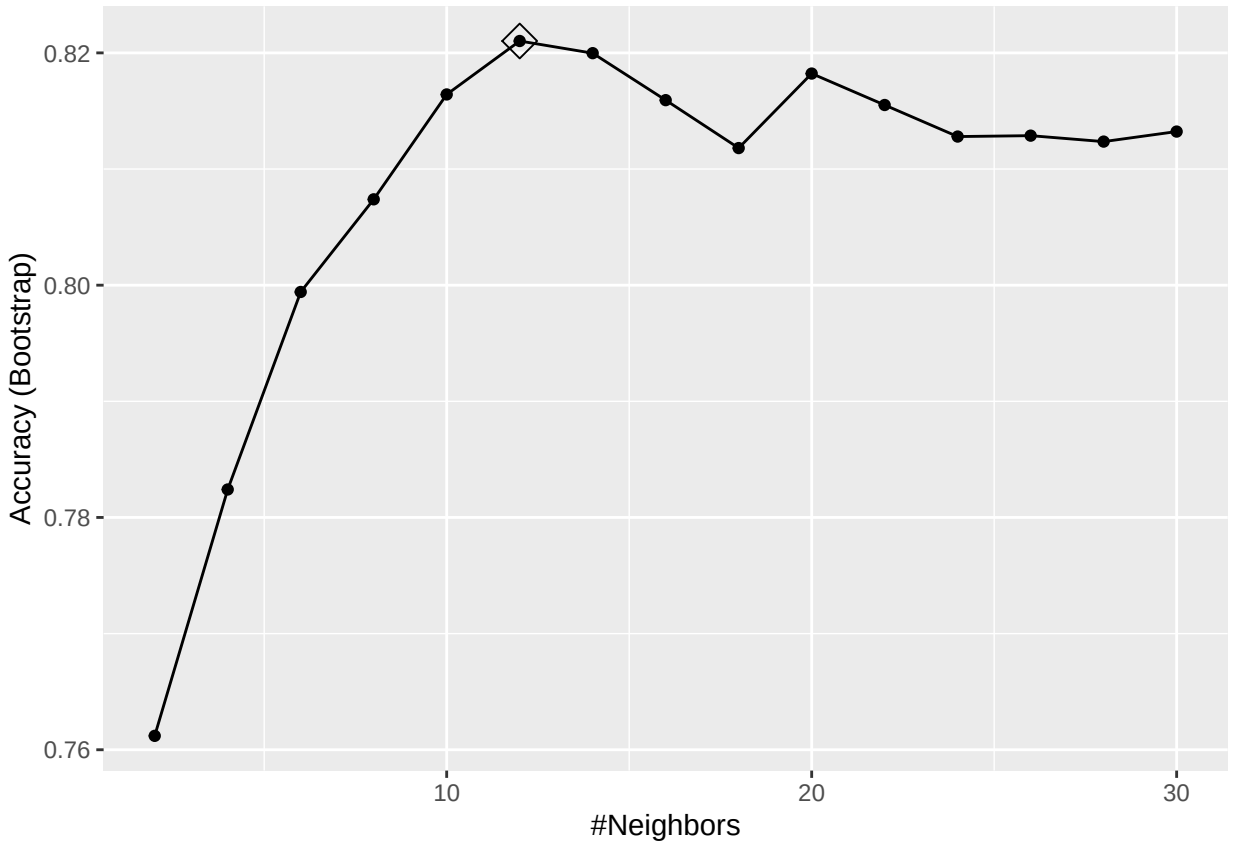
```
##    k
## 6 12
```

```
confusionMatrix(predict(train_knn, testingSet, type = "raw"),
                testingSet$DEATH_EVENT)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8852459
```

1st model: Knn

```
# 1st model: Knn
# Visualize and save the optimal value for k
k_plot <- ggplot(train_knn, highlight = TRUE)
k_plot
```




```
optim_k <- train_knn$bestTune[1, 1]
optim_k
```

```
## [1] 12
```

```
# Train and predict using k-nn with optimized k value
knn_fit <- knn3(DEATH_EVENT ~ ., data = trainingSet, k = optim_k)
y_hat_knn <- predict(knn_fit, testingSet, type = "class")
cm_knn <- confusionMatrix(data = y_hat_knn, reference = testingSet$DEATH_EVENT,
                          positive = NULL)
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 40  6
##          2  1 14
##
##                Accuracy : 0.8852
##                  95% CI : (0.7778, 0.9526)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 0.0001138
##
##                   Kappa : 0.7218
##
##  Mcnemar's Test P-Value : 0.1305700
##
```

```
##             Sensitivity : 0.9756
##             Specificity : 0.7000
##          Pos Pred Value : 0.8696
##          Neg Pred Value : 0.9333
##              Prevalence : 0.6721
##          Detection Rate : 0.6557
##    Detection Prevalence : 0.7541
##       Balanced Accuracy : 0.8378
##
##        'Positive' Class : 1
##
```

```
# Return optimized k value, Accuracy, Sensitivity and Specificity
Accuracy_knn <- cm_knn$overall["Accuracy"]
Sensitivity_knn <- cm_knn$byClass["Sensitivity"]
Specificity_knn <- cm_knn$byClass["Specificity"]
Accuracy_knn
```

```
##  Accuracy
## 0.8852459
```

```
Sensitivity_knn
```

```
## Sensitivity
##   0.9756098
```

```
Specificity_knn
```

```
## Specificity
##         0.7
```

2nd model: Naive Bayes

```
# 2nd model: Naive Bayes
# Look at correlation between features to verify independance
matrix_data <- matrix(as.numeric(unlist(data_clean)),nrow=nrow(data_clean))
correlations <- cor(matrix_data)
correlations
```

```
##             [,1]        [,2]        [,3]        [,4]        [,5]       [,6]
## [1,]  1.00000000  0.06009836  0.15918713 -0.04596584 -0.22406842  0.2537285
## [2,]  0.06009836  1.00000000 -0.01130247  0.17590228  0.04172924 -0.2686033
## [3,]  0.15918713 -0.01130247  1.00000000 -0.18909521 -0.14931542  0.2942776
## [4,] -0.04596584  0.17590228 -0.18909521  1.00000000  0.08764000 -0.1952036
## [5,] -0.22406842  0.04172924 -0.14931542  0.08764000  1.00000000 -0.5269638
## [6,]  0.25372854 -0.26860331  0.29427756 -0.19520360 -0.52696378  1.0000000
```

```
# Train and predict using Naive Bayes
train_nb <- train(DEATH_EVENT ~ ., method = "nb", data = trainingSet)
y_hat_nb <- predict(train_nb, testingSet)
cm_nb <- confusionMatrix(data = y_hat_nb, reference = testingSet$DEATH_EVENT,
                         positive = NULL)
cm_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 35  8
```

```
##          2  6 12
##
##                Accuracy : 0.7705
##                  95% CI : (0.645, 0.8685)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 0.06369
##
##                   Kappa : 0.4656
##
##  Mcnemar's Test P-Value : 0.78927
##
##             Sensitivity : 0.8537
##             Specificity : 0.6000
##          Pos Pred Value : 0.8140
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6721
##          Detection Rate : 0.5738
##    Detection Prevalence : 0.7049
##       Balanced Accuracy : 0.7268
##
##        'Positive' Class : 1
##
```

```
# Return Accuracy, Sensitivity and Specificity
Accuracy_nb <- cm_nb$overall["Accuracy"]
Sensitivity_nb <- cm_nb$byClass["Sensitivity"]
Specificity_nb <- cm_nb$byClass["Specificity"]
Accuracy_nb
```

```
##  Accuracy
## 0.7704918
```

```
Sensitivity_nb
```

```
## Sensitivity
##   0.8536585
```

```
Specificity_nb
```

```
## Specificity
##         0.6
```

3rd model: Generalized Linear Regression Model

```
# 3rd model: Generalized Linear Regression Model
# perform 10-fold cross validation
trCntl <- trainControl(method = "CV",number = 10)
# fit into the generalized linear regression model
glmModel <- train(DEATH_EVENT ~ .,data = trainingSet,trControl = trCntl,
                  method="glm",family = "binomial")
# print the model info
summary(glmModel)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.2073  -0.6139  -0.2359   0.5135   2.7312
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       10.989013   6.379684   1.723 0.084979 .
## age                0.044486   0.016839   2.642 0.008247 **
## ejection_fraction -0.072312   0.016954  -4.265 2.00e-05 ***
## serum_creatinine   0.689663   0.179559   3.841 0.000123 ***
## serum_sodium      -0.078365   0.045563  -1.720 0.085449 .
## time              -0.018698   0.003102  -6.027 1.67e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 298.15  on 237  degrees of freedom
## Residual deviance: 179.53  on 232  degrees of freedom
## AIC: 191.53
##
## Number of Fisher Scoring iterations: 6
```

```
glmModel
```

```
## Generalized Linear Model
##
## 238 samples
##   5 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 215, 213, 214, 214, 215, 214, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8104783  0.5488945
```

```
confusionMatrix(glmModel)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    1    2
##          1 60.5 11.3
##          2  7.6 20.6
##
##  Accuracy (average) : 0.8109
```

```
Accuracy_glm<-"0.8067"
Accuracy_glm
```

```
## [1] "0.8067"
```

4th model: Random Forest Model with K-Fold Cross-Validation

```r
# 4th model: Random Forest Model with K-Fold Cross-Validation
library(randomForest)
library(rsample)
# Define train control for k-fold (10-fold here) cross validation
set.seed(1984)
train_control <- trainControl(method="cv", number=10)
# Train and predict using Random Forest
set.seed(1989)
train_rf <- train(DEATH_EVENT ~ ., data = trainingSet,
                  method = "rf",
                  trControl = train_control)
y_hat_rf <- predict(train_rf, testingSet)
cm_rf <- confusionMatrix(data = y_hat_rf, reference = testingSet$DEATH_EVENT,
                         positive = NULL)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 39  5
##          2  2 15
##
##                Accuracy : 0.8852
##                  95% CI : (0.7778, 0.9526)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 0.0001138
##
##                   Kappa : 0.7292
##
##  Mcnemar's Test P-Value : 0.4496918
##
##             Sensitivity : 0.9512
##             Specificity : 0.7500
##          Pos Pred Value : 0.8864
##          Neg Pred Value : 0.8824
##              Prevalence : 0.6721
##          Detection Rate : 0.6393
##    Detection Prevalence : 0.7213
##       Balanced Accuracy : 0.8506
##
##        'Positive' Class : 1
##
```

```r
# Return Accuracy, Sensitivity and Specificity
Accuracy_rf <- cm_rf$overall["Accuracy"]
Sensitivity_rf <- cm_rf$byClass["Sensitivity"]
Specificity_rf <- cm_rf$byClass["Specificity"]
Accuracy_rf
```

```
##  Accuracy
## 0.8852459
```

```
Sensitivity_rf
```

```
## Sensitivity
##   0.9512195
```

```
Specificity_rf
```

```
## Specificity
##        0.75
```

5th model: Weighted Subspace Random Forest with K-Fold Cross-Validation

```
# 5th model: Weighted Subspace Random Forest with K-Fold Cross-Validation
set.seed(1989)
train_wsrf <- train(DEATH_EVENT ~ ., data = trainingSet,
                    method = "wsrf",
                    trControl = train_control)
y_hat_wsrf <- predict(train_wsrf, testingSet)
cm_wsrf <- confusionMatrix(data = y_hat_wsrf, reference = testingSet$DEATH_EVENT,
                           positive = NULL)
cm_wsrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 40  5
##          2  1 15
##
##                Accuracy : 0.9016
##                  95% CI : (0.7981, 0.963)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 2.824e-05
##
##                   Kappa : 0.7648
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.9756
##             Specificity : 0.7500
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.9375
##              Prevalence : 0.6721
##          Detection Rate : 0.6557
##    Detection Prevalence : 0.7377
##       Balanced Accuracy : 0.8628
##
##        'Positive' Class : 1
##
```

```
# Return Accuracy, Sensitivity and Specificity
Accuracy_wsrf <- cm_wsrf$overall["Accuracy"]
Sensitivity_wsrf <- cm_wsrf$byClass["Sensitivity"]
Specificity_wsrf <- cm_wsrf$byClass["Specificity"]
Accuracy_wsrf
```

```
##  Accuracy
```

```
## 0.9016393
```

```
Sensitivity_wsrf
```

```
## Sensitivity
##   0.9756098
```

```
Specificity_wsrf
```

```
## Specificity
##        0.75
```

Model 6: Adaptive Boosting

```
#Model 6: Adaptive Boosting
library(adabag)
train_ada <- train(DEATH_EVENT ~ ., method = "adaboost", data = trainingSet)
y_hat_ada <- predict(train_ada, testingSet)
cm_ada <- confusionMatrix(data = y_hat_ada, reference = testingSet$DEATH_EVENT,
                          positive = NULL)
cm_ada
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 37  5
##          2  4 15
##
##                Accuracy : 0.8525
##                  95% CI : (0.7383, 0.9302)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 0.001205
##
##                   Kappa : 0.6609
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9024
##             Specificity : 0.7500
##          Pos Pred Value : 0.8810
##          Neg Pred Value : 0.7895
##              Prevalence : 0.6721
##          Detection Rate : 0.6066
##    Detection Prevalence : 0.6885
##       Balanced Accuracy : 0.8262
##
##        'Positive' Class : 1
##
```

```
# Return Accuracy, Sensitivity and Specificity
Accuracy_ada <- cm_ada$overall["Accuracy"]
Sensitivity_ada <- cm_ada$byClass["Sensitivity"]
Specificity_ada <- cm_ada$byClass["Specificity"]
Accuracy_ada
```

```
## Accuracy
## 0.852459
```

```
Sensitivity_ada
```

```
## Sensitivity
##    0.902439
```

```
Specificity_ada
```

```
## Specificity
##        0.75
```

Model 7: Extreme Gradient Boosting

```
#Model 7: Extreme Gradient Boosting
library(readxl)
library(xgboost)
train_xgb <- train(DEATH_EVENT ~ ., method = "xgbTree", data = trainingSet)
y_hat_xgb <- predict(train_xgb, testingSet)
cm_xgb <- confusionMatrix(data = y_hat_xgb, reference = testingSet$DEATH_EVENT,
                          positive = NULL)
cm_xgb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 39  5
##          2  2 15
##
##                Accuracy : 0.8852
##                  95% CI : (0.7778, 0.9526)
##     No Information Rate : 0.6721
##     P-Value [Acc > NIR] : 0.0001138
##
##                   Kappa : 0.7292
##
##  Mcnemar's Test P-Value : 0.4496918
##
##             Sensitivity : 0.9512
##             Specificity : 0.7500
##          Pos Pred Value : 0.8864
##          Neg Pred Value : 0.8824
##              Prevalence : 0.6721
##          Detection Rate : 0.6393
##    Detection Prevalence : 0.7213
##       Balanced Accuracy : 0.8506
##
##        'Positive' Class : 1
##
```

```
# Return Accuracy, Sensitivity and Specificity
Accuracy_xgb <- cm_xgb$overall["Accuracy"]
Sensitivity_xgb <- cm_xgb$byClass["Sensitivity"]
Specificity_xgb <- cm_xgb$byClass["Specificity"]
Accuracy_xgb
```

```
##  Accuracy
## 0.8852459
```

```
Sensitivity_xgb
```

```
## Sensitivity
##   0.9512195
```

```
Specificity_xgb
```

```
## Specificity
##        0.75
```

4. Result and Discussion

In this report, I tried to apply 7 models into the prediction algorithm for the death event of patient with heart failure by using the 5 identified variables. Among these 7 models, both the Model 1: K-Nearest Neighbors and Model 7: Extreme Gradient Boosting yield the highest accuracy with same value 0.885245. Regarding the higher specificity of the Model 7, 0.75, the Model 7: Extreme Gradient Boosting is adopted as the final model to be used in prediction.

The model is potential to be a prediction tool for clinical triage and patient management of heart failure. With the algorithm prediction, the patient with end-stage heart failure may be told earlier about the disease prognosis for psychological preparation and healthcare professionals can arrange a better palliative care and psychosocial support to patients and their family. Meanwhile, clinicians may screen out the patients who are predicted to have better chance to survive, so that the healthcare system can concentrate the resources to save the patients' life.

In the dataset, there are only 299 observations from a country. The current model should be further improved by a large dataset obtained from different countries and institutes in order to improve the generalizability. Additional clinical parameters may also be considered to collect in future study regarding the known pathophysiological and pharmacological information, for example other heart failure clinical parameters, blood test result, drug usage and activities of daily living(ADL) score ,in order to further improve the accuracy and specificity of the prediction algorithm.

```
results <- data_frame(
Model=c("Model 1: K-Nearest Neighbors","Model 2: Naive Bayes",
        "Model 3: Generalized Linear Regression Model with K-Fold Cross-Validation",
        "Model 4: Random Forest and K-Fold Cross-Validation",
        "Model 5: Weighted Subspace Random Forest and K-Fold Cross-Validation",
        "Model 6: Adaptive Boosting","Model 7: Extreme Gradient Boosting"),
Accuracy=c(Accuracy_knn, Accuracy_nb, Accuracy_glm, Accuracy_rf, Accuracy_wsrf,
           Accuracy_ada, Accuracy_xgb),
Sensitivity=c(Sensitivity_knn, Sensitivity_nb, "-", Sensitivity_rf,
              Accuracy_wsrf, Sensitivity_ada, Sensitivity_xgb),
Specificity=c(Specificity_knn, Specificity_nb, "-", Specificity_rf,
              Specificity_wsrf, Specificity_ada, Specificity_xgb)
)
results
```

```
## # A tibble: 7 x 4
##   Model                                Accuracy      Sensitivity   Specificity
##   <chr>                                <chr>         <chr>         <chr>
## 1 Model 1: K-Nearest Neighbors         0.8852459016~ 0.9756097560~ 0.7
## 2 Model 2: Naive Bayes                 0.7704918032~ 0.8536585365~ 0.6
## 3 Model 3: Generalized Linear Regressio~ 0.8067      -             -
## 4 Model 4: Random Forest and K-Fold Cro~ 0.8852459016~ 0.9512195121~ 0.75
## 5 Model 5: Weighted Subspace Random For~ 0.9016393442~ 0.9016393442~ 0.75
## 6 Model 6: Adaptive Boosting           0.8524590163~ 0.9024390243~ 0.75
## 7 Model 7: Extreme Gradient Boosting   0.8852459016~ 0.9512195121~ 0.75
```

5. Conclusion

In this report, we present the process of analysis and building algorithms by using various data analysis and machine learning methodologies in R as part of course HarvardX: PH125.9 - Data Science: Capstone. We are able to build a model by using Extreme Gradient Boosting to predict the death event of patient with heart failure. The accuracy of the model is high and able to reach 88.5%. The Sensitivity and specificity are high and able to reach 95.1% and 75% respectively. The key predictors for death event are age, ejection fraction, serum creatinine, serum sodium and time of follow up period, which are selected to optimize the model. The model should be further improved upon to be applied in clinical settings.

6. Reference:

G. Savarese, and L.H Lund(2017). Global Public Health Burden of Heart Failure. Card Fail Rev. 2017 Apr; 3(1): 7–11.https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5494150/

D. Chicco and G. Jurman(2020). Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making (2020) 20:16 https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5

Ahmad T, Munir A, Bhatti SH, Aftab M, Raza MA (2017) Survival analysis of heart failure patients: A case study. PLoS ONE 12(7):e0181001. https://doi.org/10.1371/journal.pone.0181001

H. Zhao, G.J. Williams, J.Z.Huang(2017). Journal of Statistical , March 2017, Volume 77, Issue 3. Software. https://www.researchgate.net/publication/315908727_wsrf_An_R_Package_for_Classification_with_Scalable_Weighted_Sub space_Random_Forests/fulltext/58ed03a10f7e9bf619bb0614/wsrf-An-R-Package-for-Classification-with-Scalable-Weighted- Subspace-Random-Forests.pdf