School of Computer Science and Engineering

In-Hospital Mortality Prediction

CSE3020 – Data Visualisation Project Report

Submitted to:

Dr. Pattabiraman V

Submitted by:

Katya Pandey (19BCE1312)

In partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

April 2021

TABLE OF CONTENTS

In-Hospital Mortality Prediction

1. Abstract

Patients with major health complications or life-threatening diseases are frequently admitted to intensive care units (ICUs) for adequate treatment and care. When a patient is suffering from a medical condition, they may exhibit a variety of indications and symptoms. These include individual procedures, laboratory tests, internal health assessments, fluid balance, records, and measures of vital signs and symptoms. If any aspect is deteriorating, it is critical to seek immediate medical attention. It is critical to recognize the symptoms that are most likely to endanger the patient's life as soon as possible. If the patient's symptoms are identified in a timely manner, quick medical attention and care can help to lower the odds of mortality. However, monitoring these patterns becomes tedious when done manually. Technology-assisted disease diagnosis is now possible thanks to the recent advances in machine learning. In this project, five classification and regression techniques are implemented and compared for identifying patients nearing death. The algorithm having the most accuracy is implemented to classify the mortally endangered patients and the leading symptoms of death are also identified.

2. Introduction

The concept of health is significant. It refers to the ability of the human body to function normally in a proper fashion. The importance of health cannot be overstated, as it comes first, followed by all other factors. Likewise, a range of factors influence one's overall health. Everything, from the oxygen you breathe to the people you choose to spend your time with, is part of it. Many other facets of health are equally vital. Even if one of their organs is dysfunctional or deteriorating, a person cannot be completely healthy. As a result, it is critical to maintain physical, mental, and social well-being. Patients with serious medical issues or life-threatening illnesses are commonly admitted to intensive care units for appropriate treatment and care.

Heart failure is the final stage of all heart disorders. It is a sickness caused by a heart function abnormality. Heart failure has become an enormous hazard to human health and social development as a leading cause of cardiovascular illness and mortality. Despite recent advances in diagnosis and evidence-based therapy, heart failure results remain dismal. With gravely threatening diseases combined with advanced organ malfunction and other serious

conditions, a substantial majority of heart failure patients may require complex, high-tech, life-saving treatment that is only available in intensive care units, which are accompanied by significant staffing in terms of nurse and physician-to-patient ratios. Considering the substantially higher in-hospital death rate, ICU admitted heart failure patients may benefit more from precisely estimating prognosis and receiving intense treatment with tighter follow-up. Although there are various in-hospital mortality prediction models available, their accuracies are mediocre and they are not frequently used. Furthermore, there is a scarcity of data on prediction models for ICU-admitted heart failure patients.

Recognizing individuals at the greatest risk of poor consequences after hospital discharge can assist ICU-admitted heart failure patients have improved outcome. Machine-learning techniques can boost the effectiveness of identifying crucial predictors by automatically recreating correlations between variables and response values from huge data. Utilizing data from the Medical Information Mart for Intensive Care database, this analysis sought to create and evaluate a prediction model for in-hospital mortality among ICU-admitted heart failure patients.

3. Feasibility study

Heart failure appears to be a nightmare because it can drastically reduce one's quality of life. In the worst-case scenario, the patient may die. This has ramifications for countries, whose economy is predominantly built on human labour. As a result, early detection of death symptoms is a relevant study area since it could be effective in monitoring large groups of patients and thus automatically detect death symptoms as soon as they emerge. It's vital to keep an eye on patients for disease symptoms if you want to lower mortality rates. As a result, it's vital to develop a system for detecting patient mortality that's quick, automatic, low-cost, and accurate. The use of classification techniques can be efficiently used to accurately identify and classify patients nearing death. They could be used to create an expert system for doctors to detect dying patients early. Furthermore, it is possible to detect many symptoms on a broad scale. By increasing the volume of patient data and inputs to the training model, the algorithms can be enhanced. This technology could be a big benefit to the medical business if it is translated into a sophisticated interface in the form of a website or a mobile app.

4. About the dataset

Kaggle was used to get the dataset for the analysis. The MIMIC-III database is a publicly accessible critical care database that includes information on 46,520 patients and 58,976 admissions to the Beth Israel Deaconess Medical Center's intensive care unit (ICU). Amongst some of the data gathered are demographic trends (age, sex, ethnicity, weight, and height at the time of admission to the hospital), admission records, diagnostic procedures, laboratory investigations, medications, procedures, fluid levels, discharge reports, vital sign evaluations performed at the site (systolic blood pressure, diastolic blood pressure, average blood pressure, respiratory rate, body temperature, saturation pulse oxygen, urine output), caretaker remarks, radiology records, and survival statistics. During the first 24 hours of each admission, demographic details and vital signs were retrieved, and laboratory variables were measured throughout the ICU stay. ICD-9 codes were used to evaluate comorbidities. The estimated mean value was included in the analysis of variable data with numerous measurements. The study's primary outcome was in-hospital mortality, which was described as the critical status of survivors and non-survivors at the time of hospital release.

The dataset has the medical data of around 1157 patients. The various columns in the dataset are: group, ID, outcome, age, gendera, BMI, hypertensive, atrial fibrillation, CHD with no MI, diabetes, deficiency anemias, depression, Hyperlipemia, Renal failure, COPD, heart rate, Systolic blood pressure, Diastolic blood pressure, Respiratory rate, temperature, SP O2, Urine output, hematocrit, RBC, MCH, MCHC, MCV, RDW, Leucocyte, Platelets, Neutrophils, Basophils, Lymphocyte, PT, INR, NT proBNP, Creatine kinase, Creatinine, Urea nitrogen, glucose, Blood potassium, Blood sodium, Blood calcium, Chloride, Anion gap, Magnesium ion, PH, Bicarbonate, Lactic acid, PCO2, and EF. The dataset's target variable is the outcome variable, with 0 representing the alive status and 1 representing the death status. Figure 1 shows a few sample records taken from the patient dataset.

	group <int>	ID <int>	outcome <int>	age <int>	gendera <int>	BMI <dbl>	hypertensive <int>
1	1	125047	0	72	1	37.58818	0
2	1	139812	0	75	2	*NA*	0
3	1	109787	0	83	2	26.57263	0
4	1	130587	0	43	2	83.26463	0
5	1	138290	0	75	2	31.82484	1
6	1	154653	0	76	1	24.26229	1

6 rows | 1-8 of 51 columns

Figure (i)

atrialfibrillation <int>	CHD.with.no.MI <int>	diabetes <int>	deficiencyanemias <int>
0	0	1	1
0	0	0	1
0	0	0	1
0	0	0	0
0	0	0	1
1	0	0	1

6 rows | 9-12 of 51 columns

Figure (ii)

depression <int>	Hyperlipemia <int>	Rel.failure <int>	COPD <int>	heart.rate <dbl>
0	1	1	0	68.83784
0	0	0	1	101.37037
0	0	1	0	72.31818
0	0	0	0	94.50000
0	0	1	1	67.92000
0	1	1	1	74.18182

6 rows | 13-17 of 51 columns

Figure (iii)

Systolic.blood.pressure <dbl>	Diastolic.blood.pressure <dbl>	Respiratory.rate <dbl>
155.8667	68.33333	16.62162
140.0000	65.00000	20.85185
135.3333	61.37500	23.64000
126.4000	73.20000	21.85714
156.5600	58.12000	21.36000
118.1000	52.95000	20.54545

6 rows | 18-20 of 51 columns

Figure (iv)

temperature <dbl>	SP.O2 <dbl>	Urine.output <dbl>	hematocrit <dbl>	RBC <dbl>	MCH <dbl>
36.71429	98.39474	2155	26.27273	2.960000	28.25000
36.68254	96.92308	1425	30.78000	3.138000	31.06000
36.45370	95.29167	2425	27.70000	2.620000	34.32000
36.28704	93.84615	8760	36.63750	4.277500	26.06250
36.76190	99.28000	4455	29.93333	3.286667	30.66667
35.26667	96.81818	1840	27.33333	3.235000	26.56667

6 rows | 21-26 of 51 columns

Figure (v)

MCHC <dbl>	MCV <dbl>	RDW <dbl>	Leucocyte <dbl>	Platelets <dbl>	Neutrophils <dbl>
31.52000	89.900	16.22000	7.650000	305.100	74.65
31.66000	98.200	14.26000	12.740000	246.400	*NA*
31.30000	109.800	23.82000	5.480000	204.200	68.10
30.41250	85.625	17.03750	8.225000	216.375	81.80
33.66667	91.000	16.26667	8.833333	251.000	*NA*
31.48333	84.500	16.51667	9.516667	273.000	85.40

6 rows | 27-32 of 51 columns

Figure (vi)

Basophils <dbl>	Lymphocyte <dbl>	PT <dbl>	INR <dbl>	NT.proBNP <dbl>	Creatine.kise <dbl>
0.40	13.3	10.60000	1.000000	1956	148.0000
NA	*NA*	*NA*	*NA*	2384	60.6000
0.55	24.5	11.27500	0.950000	4081	16.0000
0.15	14.5	27.06667	2.666667	668	85.0000
NA	*NA*	*NA*	*NA*	30802	111.6667
0.30	9.3	18.78333	1.700000	34183	28.0000

6 rows | 33-38 of 51 columns

Figure (vii)

Creatinine <dbl>	Urea.nitrogen <dbl>	glucose <dbl>	Blood.potassium <dbl>	Blood.sodium <dbl>
1.9583333	50.00000	114.63636	4.816667	138.7500
1.1222222	20.33333	147.50000	4.450000	138.8889
1.8714286	33.85714	149.00000	5.825000	140.7143
0.5857143	15.28571	128.25000	4.386667	138.5000
1.9500000	43.00000	145.75000	4.783333	136.6667
1.6125000	26.62500	98.33333	4.075000	136.2500

6 rows | 39-43 of 51 columns

Figure (viii)

Blood.calcium <dbl>	Chloride <dbl>	Anion.gap <dbl>	Magnesium.ion <dbl>	PH <dbl>	Bicarbote <dbl>
7.463636	109.16667	13.16667	2.618182	7.230	21.16667
8.162500	98.44444	11.44444	1.887500	7.225	33.44444
8.266667	105.85714	10.00000	2.157143	7.268	30.57143
9.476923	92.07143	12.35714	1.942857	7.370	38.57143
8.733333	104.50000	15.16667	1.650000	7.250	22.00000
8.466667	96.75000	13.12500	1.771429	7.310	30.50000

6 rows | 44-49 of 51 columns

Figure (ix)

Anion.gap <dbl>	Magnesium.ion <dbl>	PH <dbl>	Bicarbote <dbl>	Lactic.acid <dbl>	PCO2 <dbl>	EF <int>
13.16667	2.618182	7.230	21.16667	0.5	40.0	55
11.44444	1.887500	7.225	33.44444	0.5	78.0	55
10.00000	2.157143	7.268	30.57143	0.5	71.5	35
12.35714	1.942857	7.370	38.57143	0.6	75.0	55
15.16667	1.650000	7.250	22.00000	0.6	50.0	55
13.12500	1.771429	7.310	30.50000	0.6	65.5	35

6 rows | 46-52 of 51 columns

Figure (x)

Figure 1. Sample records taken from the patient dataset

5. Design and flow of the project

Figure 3 shows the architecture and the flow for in-hospital mortality prediction.

Figure 5. Flow chart for in-hospital mortality prediction

6. Methodology

6.1. Data Extraction

The process of extracting data from data sources for subsequent processing or storage is known as data extraction. The patient dataset is retreived from Kaggle, whih in turn has acquired the dataset from the MIMIC-III database, which is a publicly accessible critical care database that includes information on 46,520 patients and 58,976 admissions to the Beth Israel Deaconess Medical Center's intensive care unit.

6.2. Data Preprocessing

The primary goal of data preprocessing is to remove unwanted data while also enhancing some key data features to make it suitable for training purpose.

1. A total of 1157 rows of data, will be given to the machine learning algorithms.
2. Since the dataset contains 1929 null values, it is important to take care of it. Removing the rows containing the null values may lead to the loss of vital information.
3. Instead of deleting the null values, we perform data imputation so that the data is preserved and no information is lost.
4. The dataset is split into training and testing datasets in a ratio of 75/25 respectively.

6.3. Exploratory Data Analysis

Exploratory Data Analysis is a method of data analysis through the use of visual techniques. With the use of statistical summaries and graphical representations, it is used to identify trends, patterns, and examine assumptions. Through this, the patients' data has been visualized. Each parameter is explored through different graphs and trends and patterns are identified easily.

6.4. Training

The model will be trained over five machine learning algorithms named- Naïve Bayes , Logistic Regression, Support Vector Machine, Linear Regression and Random Forest. The models will be validated using the 10 K- Fold Cross Validation technique to predict each model's accuracy.

6.5. Testing

1. The prediction models based on each algorithm are tested by implementing them onto the testing dataset.
2. The predicted values are analyzed and the accuracy for each algorithm is calculated independently by making use of confusion matrices.
3. The confusion matrices give the number of True Positive, True Negative, False Negative and False Positive values of predicted.

6.6 Comparison of models

1. The accuracy of all models is compared and the algorithm, having the maximum accuracy, is chosen as the best classifier for mortality prediction.
2. The best classifier algorithm is used to classify patients' mortality as well predict the leading symptoms of death in an ill patient quickly, so that treatment can be provided at the earliest.

7. Algorithms used

For the mortality prediction process, the following classifier algorithms were used and analyzed.

1. **Naïve Bayes:** Based on Bayes' theorem, this method is used to solve classification problems. It is primarily used in text classification tasks that require a large training dataset. It is a simple and effective classification algorithm that aids in the development of fast machine learning models capable of making quick predictions. It's a probabilistic classifier, which means it makes predictions based on the probabilities.
2. **Logistic Regression:** It's a method for predicting a categorical dependent variable from a set of independent variables. It forecasts a categorical dependent variable's output. As a result, the result must be a discrete or categorical value. It can be Yes or No, 0 or 1, true or false, and so on, but instead of giving exact values like 0 and 1, it gives probabilistic values that are somewhere between 0 and 1.
3. **Support Vector Machine:** Each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a specific coordinate in the SVM algorithm. Then classification is performed by locating the hyper-plane that clearly distinguishes the two classes.
4. **Random Forest:** It creates a "forest" out of an ensemble of decision trees, which are

usually trained using the "bagging" method. The bagging method's basic premise is that combining different learning models improves the overall result. Simply put, random forest combines multiple decision trees to produce a more accurate and stable prediction.

8. Risk Analysis

Because of numerous types of signs and symptoms in patients, achieving high efficiency in mortality diagnosis methods is a major challenge. To address this issue, five machine learning classification algorithms were proposed and their accuracy was compared. However, the techniques proposed are typically limited in scope and rely on ideal capture conditions to function properly. This apparent lack of significant progress could be explained in part by the subject's difficult challenges: fluctuating medical test results, incorrect test values and uncontrolled capture conditions like sudden deterioration in health may produce characteristics that make mortality analysis more difficult.

9. Implementation

Firstly, the necessary R libraries and packages are installed and imported into the program. Next, a total of 1157 rows of data containing patients' personal information, medical data, test results and vital signs and symptoms, is loaded into the Rmd file by specifying the path to the dataset.

```
## 1. Loading the dataset

```{r}
options(max.print=1000000)
# Reading the dataset
df <- read.csv("data01.csv", fileEncoding = "UTF-8", na.strings = "..")
head(df)
```
```

The overall structure and summary of the data is made visible so that it becomes easy to get an idea of the dataset we are dealing with along with its summarized values and data types.

```{r}
# The structure of the data
str(df)
```

```
'data.frame':   1177 obs. of  51 variables:
 $ group                    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ ID                       : int  125047 139812 109787 130587 138290 154653 194420 153461
113076 147252 ...
 $ outcome                  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ age                      : int  72 75 83 43 75 76 72 83 61 67 ...
 $ gendera                  : int  1 2 2 2 2 1 1 2 2 1 ...
 $ BMI                      : num  37.6 NA 26.6 83.3 31.8 ...
 $ hypertensive             : int  0 0 0 0 1 1 1 1 1 1 ...
 $ atrialfibrillation       : int  0 0 0 0 0 1 0 1 1 0 ...
 $ CHD.with.no.MI           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ diabetes                 : int  1 0 0 0 0 0 0 1 1 1 ...
 $ deficiencyanemias        : int  1 1 1 0 1 1 0 1 0 0 ...
 $ depression               : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Hyperlipemia             : int  1 0 0 0 0 1 1 0 0 0 ...
 $ Rel.failure              : int  1 0 1 0 1 1 1 0 1 0 ...
 $ COPD                     : int  0 1 0 0 1 1 1 0 0 0 ...
 $ heart.rate               : num  68.8 101.4 72.3 94.5 67.9 ...
 $ Systolic.blood.pressure  : num  156 140 135 126 157 ...
 $ Diastolic.blood.pressure: num  68.3 65 61.4 73.2 58.1 ...
 $ Respiratory.rate         : num  16.6 20.9 23.6 21.9 21.4 ...
 $ temperature              : num  36.7 36.7 36.5 36.3 36.8 ...
 $ SP.O2                    : num  98.4 96.9 95.3 93.8 99.3 ...
 $ Urine.output             : num  2155 1425 2425 8760 4455 ...
 $ hematocrit               : num  26.3 30.8 27.7 36.6 29.9 ...
 $ RBC                      : num  2.96 3.14 2.62 4.28 3.29 ...
 $ MCH                      : num  28.2 31.1 34.3 26.1 30.7 ...
 $ MCHC                     : num  31.5 31.7 31.3 30.4 33.7 ...
 $ MCV                      : num  89.9 98.2 109.8 85.6 91 ...
 $ RDW                      : num  16.2 14.3 23.8 17 16.3 ...
 $ Leucocyte                : num  7.65 12.74 5.48 8.22 8.83 ...
 $ Platelets                : num  305 246 204 216 251 ...
 $ Neutrophils              : num  74.7 NA 68.1 81.8 NA ...
 $ Basophils                : num  0.4 NA 0.55 0.15 NA 0.3 0.2 NA 0.55 NA ...
 $ Lymphocyte               : num  13.3 NA 24.5 14.5 NA ...
 $ PT                       : num  10.6 NA 11.3 27.1 NA ...
 $ INR                      : num  1 NA 0.95 2.67 NA ...
 $ NT.proBNP                : num  1956 2384 4081 668 30802 ...
 $ Creatine.kise            : num  148 60.6 16 85 111.7 ...
 $ Creatinine               : num  1.958 1.122 1.871 0.586 1.95 ...
 $ Urea.nitrogen            : num  50 20.3 33.9 15.3 43 ...
 $ glucose                  : num  115 148 149 128 146 ...
 $ Blood.potassium          : num  4.82 4.45 5.83 4.39 4.78 ...
 $ Blood.sodium             : num  139 139 141 138 137 ...
 $ Blood.calcium            : num  7.46 8.16 8.27 9.48 8.73 ...
 $ Chloride                 : num  109.2 98.4 105.9 92.1 104.5 ...
 $ Anion.gap                : num  13.2 11.4 10 12.4 15.2 ...
 $ Magnesium.ion            : num  2.62 1.89 2.16 1.94 1.65 ...
 $ PH                       : num  7.23 7.22 7.27 7.37 7.25 ...
 $ Bicarbote                : num  21.2 33.4 30.6 38.6 22 ...
 $ Lactic.acid              : num  0.5 0.5 0.5 0.6 0.6 ...
 $ PCO2                     : num  40 78 71.5 75 50 ...
 $ EF                       : int  55 55 35 55 55 35 55 75 50 55 ...
```

```r
# Summary of the dataset
summary(df)
```

```
     group             ID            outcome            age            gendera      
 Min.   :1.000   Min.   :100213   Min.   :0.0000   Min.   :19.00   Min.   :1.000  
 1st Qu.:1.000   1st Qu.:125603   1st Qu.:0.0000   1st Qu.:65.00   1st Qu.:1.000  
 Median :1.000   Median :151901   Median :0.0000   Median :77.00   Median :2.000  
 Mean   :1.299   Mean   :150778   Mean   :0.1352   Mean   :74.06   Mean   :1.525  
 3rd Qu.:2.000   3rd Qu.:176048   3rd Qu.:0.0000   3rd Qu.:85.00   3rd Qu.:2.000  
 Max.   :2.000   Max.   :199952   Max.   :1.0000   Max.   :99.00   Max.   :2.000  
                                  NA's   :1                                       
      BMI          hypertensive    atrialfibrillation CHD.with.no.MI       diabetes      
 Min.   : 13.35   Min.   :0.0000   Min.   :0.0000     Min.   :0.00000   Min.   :0.0000  
 1st Qu.: 24.33   1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.:0.00000   1st Qu.:0.0000  
 Median : 28.31   Median :1.0000   Median :0.0000     Median :0.00000   Median :0.0000  
 Mean   : 30.19   Mean   :0.7179   Mean   :0.4511     Mean   :0.08581   Mean   :0.4214  
 3rd Qu.: 33.63   3rd Qu.:1.0000   3rd Qu.:1.0000     3rd Qu.:0.00000   3rd Qu.:1.0000  
 Max.   :104.97   Max.   :1.0000   Max.   :1.0000     Max.   :1.00000   Max.   :1.0000  
 NA's   :215                                                                           
 deficiencyanemias   depression      Hyperlipemia     Rel.failure          COPD        
 Min.   :0.000     Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  
 1st Qu.:0.000     1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000  
 Median :0.000     Median :0.0000   Median :0.0000   Median :0.0000   Median :0.00000  
 Mean   :0.339     Mean   :0.1189   Mean   :0.3798   Mean   :0.3653   Mean   :0.07562  
 3rd Qu.:1.000     3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000  
 Max.   :1.000     Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000  
```

```
  heart.rate     Systolic.blood.pressure Diastolic.blood.pressure Respiratory.rate
 Min.   : 36.00   Min.   : 75.0           Min.   : 24.74           Min.   :11.14   
 1st Qu.: 72.37   1st Qu.:105.4           1st Qu.: 52.17           1st Qu.:17.93   
 Median : 83.61   Median :116.1           Median : 58.46           Median :20.37   
 Mean   : 84.58   Mean   :118.0           Mean   : 59.53           Mean   :20.80   
 3rd Qu.: 95.91   3rd Qu.:128.6           3rd Qu.: 65.46           3rd Qu.:23.39   
 Max.   :135.71   Max.   :203.0           Max.   :107.00           Max.   :40.90   
 NA's   :13       NA's   :16              NA's   :16               NA's   :13      
  temperature        SP.O2         Urine.output    hematocrit         RBC        
 Min.   :33.25   Min.   : 75.92   Min.   :   0   Min.   :20.31   Min.   :2.030  
 1st Qu.:36.29   1st Qu.: 95.00   1st Qu.: 980   1st Qu.:28.16   1st Qu.:3.120  
 Median :36.65   Median : 96.45   Median :1675   Median :30.80   Median :3.490  
 Mean   :36.68   Mean   : 96.27   Mean   :1899   Mean   :31.91   Mean   :3.575  
 3rd Qu.:37.02   3rd Qu.: 97.92   3rd Qu.:2500   3rd Qu.:35.01   3rd Qu.:3.900  
 Max.   :39.13   Max.   :100.00   Max.   :8820   Max.   :55.42   Max.   :6.575  
 NA's   :19      NA's   :13       NA's   :36                                    
      MCH             MCHC             MCV              RDW          Leucocyte      
 Min.   :18.12   Min.   :27.82   Min.   : 62.60   Min.   :12.09   Min.   : 0.10  
 1st Qu.:28.25   1st Qu.:32.01   1st Qu.: 86.25   1st Qu.:14.46   1st Qu.: 7.44  
 Median :29.75   Median :32.99   Median : 90.00   Median :15.51   Median : 9.68  
 Mean   :29.54   Mean   :32.86   Mean   : 89.90   Mean   :15.95   Mean   :10.71  
 3rd Qu.:31.24   3rd Qu.:33.83   3rd Qu.: 93.86   3rd Qu.:16.94   3rd Qu.:12.74  
 Max.   :40.31   Max.   :37.01   Max.   :116.71   Max.   :29.05   Max.   :64.75  

  Platelets         Neutrophils      Basophils        Lymphocyte           PT        
 Min.   :   9.571   Min.   : 5.00   Min.   :0.1000   Min.   : 0.9667   Min.   :10.10  
 1st Qu.: 168.909   1st Qu.:74.78   1st Qu.:0.2000   1st Qu.: 6.6500   1st Qu.:13.16  
 Median : 222.667   Median :82.47   Median :0.3000   Median :10.4750   Median :14.63  
 Mean   : 241.504   Mean   :80.11   Mean   :0.4056   Mean   :12.2330   Mean   :17.48  
 3rd Qu.: 304.250   3rd Qu.:87.45   3rd Qu.:0.5000   3rd Qu.:15.4625   3rd Qu.:18.80  
 Max.   :1028.200   Max.   :98.00   Max.   :8.8000   Max.   :83.5000   Max.   :71.27  
                    NA's   :144     NA's   :259      NA's   :145       NA's   :20     
```

```
      INR              NT.proBNP      Creatine.kise        Creatinine      Urea.nitrogen
 Min.   :0.8714   Min.   :    50   Min.   :    8.00   Min.   : 0.2667   Min.   :  5.357
 1st Qu.:1.1400   1st Qu.:  2251   1st Qu.:   46.00   1st Qu.: 0.9400   1st Qu.: 20.833
 Median :1.3000   Median :  5840   Median :   89.25   Median : 1.2875   Median : 30.667
 Mean   :1.6255   Mean   : 11014   Mean   :  246.78   Mean   : 1.6428   Mean   : 36.298
 3rd Qu.:1.7364   3rd Qu.: 14968   3rd Qu.:  185.19   3rd Qu.: 1.9000   3rd Qu.: 45.250
 Max.   :8.3429   Max.   :118928   Max.   :42987.50   Max.   :15.5273   Max.   :161.750
 NA's   :20                        NA's   :165
    glucose       Blood.potassium  Blood.sodium   Blood.calcium       Chloride
 Min.   : 66.67   Min.   :3.000   Min.   :114.7   Min.   : 6.700   Min.   : 80.27
 1st Qu.:113.94   1st Qu.:3.900   1st Qu.:136.7   1st Qu.: 8.149   1st Qu.: 99.00
 Median :136.40   Median :4.115   Median :139.2   Median : 8.500   Median :102.50
 Mean   :148.80   Mean   :4.177   Mean   :138.9   Mean   : 8.501   Mean   :102.28
 3rd Qu.:169.50   3rd Qu.:4.400   3rd Qu.:141.6   3rd Qu.: 8.869   3rd Qu.:105.57
 Max.   :414.10   Max.   :6.567   Max.   :154.7   Max.   :10.950   Max.   :122.53
 NA's   :18                                       NA's   :1
   Anion.gap      Magnesium.ion         PH          Bicarbote      Lactic.acid
 Min.   : 6.636   Min.   :1.400   Min.   :7.090   Min.   :12.86   Min.   :0.500
 1st Qu.:12.250   1st Qu.:1.956   1st Qu.:7.335   1st Qu.:23.45   1st Qu.:1.200
 Median :13.667   Median :2.092   Median :7.380   Median :26.50   Median :1.600
 Mean   :13.925   Mean   :2.120   Mean   :7.379   Mean   :26.91   Mean   :1.853
 3rd Qu.:15.417   3rd Qu.:2.242   3rd Qu.:7.430   3rd Qu.:29.88   3rd Qu.:2.200
 Max.   :25.500   Max.   :4.073   Max.   :7.580   Max.   :47.67   Max.   :8.333
                                  NA's   :292                     NA's   :229
      PCO2             EF
 Min.   :18.75   Min.   :15.00
 1st Qu.:37.04   1st Qu.:40.00
 Median :43.00   Median :55.00
 Mean   :45.54   Mean   :48.72
 3rd Qu.:50.59   3rd Qu.:55.00
 Max.   :98.60   Max.   :75.00
 NA's   :294
```

Data preprocessing

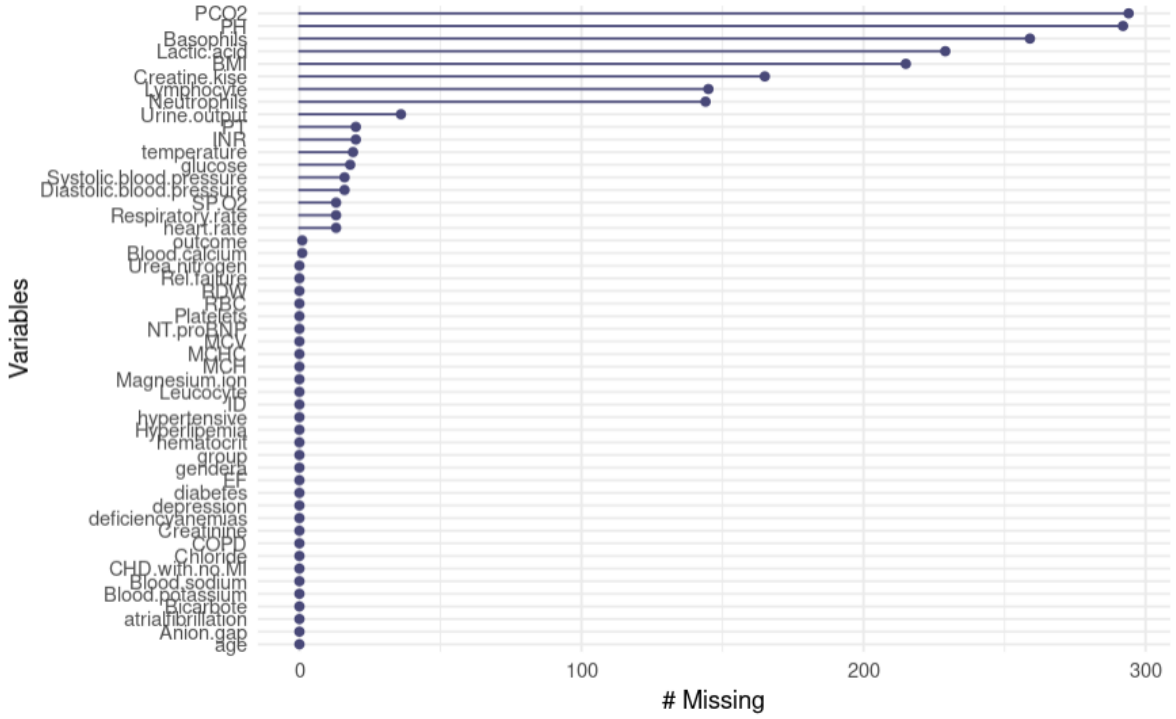
The aim of pre-processing is to enhance some data features relevant for further data processing and analysis. A function is used to check the number of null values present in the dataset.

```r
```{r}
# Check for null or missing values in the dataset
sum(is.na(df))
# Check for null or missing values in each column of the dataset
print("Column-wise presence of missing data: ")
colSums(is.na(df))
```
```

```
[1] 1929
[1] "Column-wise presence of missing data: "
                    group                        ID                   outcome
                        0                         0                         1
                      age                   gendera                       BMI
                        0                         0                       215
             hypertensive       atrialfibrillation           CHD.with.no.MI
                        0                         0                         0
                 diabetes        deficiencyanemias                depression
                        0                         0                         0
             Hyperlipemia              Rel.failure                      COPD
                        0                         0                         0
               heart.rate  Systolic.blood.pressure Diastolic.blood.pressure
                       13                        16                        16
        Respiratory.rate              temperature                     SP.O2
                       13                        19                        13
            Urine.output              hematocrit                       RBC
                       36                         0                         0
                      MCH                      MCHC                       MCV
                        0                         0                         0
                      RDW                 Leucocyte                 Platelets
                        0                         0                         0
```

Neutrophils	Basophils	Lymphocyte
144	259	145
PT	INR	NT.proBNP
20	20	0
Creatine.kise	Creatinine	Urea.nitrogen
165	0	0
glucose	Blood.potassium	Blood.sodium
18	0	0
Blood.calcium	Chloride	Anion.gap
1	0	0
Magnesium.ion	PH	Bicarbote
0	292	0
Lactic.acid	PCO2	EF
229	294	0

There is a prominent number of null values present in the dataset, ie, 1929 cell values.



The presence of so much null values affects the quality of the result. Thus, it becomes necessary to get rid of them. However, since there are too many null values present, we cannot remove all the rows containing null values. This may lead to the loss of vital information. Instead, we perform data imputation wherein the null values are replaced by the mean values of the dataset.

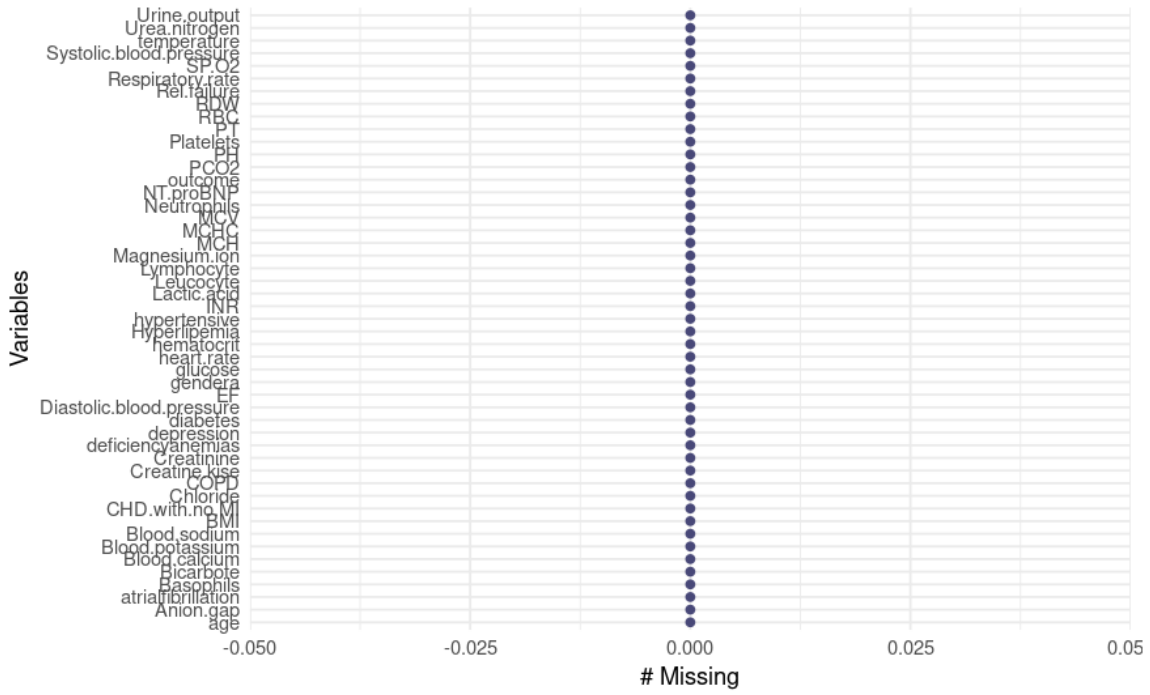
```{r}
#PCO2 and PH have many missing values. Analyzing the columns further by creating a linear
regression model using lm() function, we'll get the summary output using the summary() function.

summary(lm(PCO2~.,data=df))
summary(lm(PH~.,data=df))
```

```r
# Linear Regression Imputation
# Importing the necessary libraries
library(simputation)
df$PCO2<-as.numeric(df$PCO2)
imp_df <- impute_lm(df[,-c(1,2)],PCO2~as.numeric(outcome)+Platelets+PH)
imp_df <- impute_lm(imp_df,PH~gendera+temperature+Creatinine+Bicarbonate+PCO2)
```

```r
#Importing the necessary libraries
library(imputeTS)
imp_df$outcome<-as.factor(imp_df$outcome)
imp_df<-na_mean(imp_df[,-c(45,48)])
```

```r
imp_df <- impute_lm(df[,-c(1,2)],PCO2~as.numeric(outcome)+Platelets+PH)
imp_df <- impute_lm(imp_df,PH~gendera+temperature+Creatinine+Bicarbonate+PCO2)
```

```r
imp_df<-na.omit(imp_df)
# Check for null or missing values in the dataset after imputation
sum(is.na(imp_df))
# Plotting the missing dataset values if any
gg_miss_var(imp_df)
```



After performing the process of data imputation, we do not find any missing values left. All the null values in the dataset have been successfully replaced by the imputed data. Additionally, the columns group and ID have been removed since they have no relevance in relation to the patient's mortality status.

Next, the data is normalized to bring all the values between 0 and 1. This will be done by creating a normalize function. This will assist in the training of the dataset using different algorithms.

```r
normalize <- function(x) {
    return((x - min(x)) / (max(x) - min(x)))
}
imp_df$outcome<-as.numeric(imp_df$outcome)
imp_df<-apply(imp_df,2,normalize)
imp_df<-as.data.frame(imp_df)
```

```r
head(imp_df)
str(imp_df)
```

The following figure shows the sample of data after data imputation.

	outcome <dbl>	age <dbl>	gendera <dbl>	BMI <dbl>	hypertensive <dbl>
1	0	0.578125	0	0.34364433	0
3	0	0.750000	1	0.18535454	0
4	0	0.125000	1	1.00000000	0
6	0	0.640625	0	0.15215570	1
7	0	0.578125	0	0.37352243	1
9	0	0.406250	1	0.09079647	1

Figure 4. Plant leaf image in RGB format

Exploratory Data Analysis

```r
#Importing the necessary libraries
library("ggplot2")
library("ggpubr")
theme_set(theme_pubr())
```

```r
# Importing the necessary packages
library(funModeling)
library(tidyverse)
```

```{r}
# Correlation between mortality outcome and all other parameters
cor(imp_df$outcome,imp_df$age)
cor(imp_df$outcome,imp_df$gendera)
cor(imp_df$outcome,imp_df$BMI)
cor(imp_df$outcome,imp_df$hypertensive)
cor(imp_df$outcome,imp_df$atrialfibrillation)
cor(imp_df$outcome,imp_df$CHD.with.no.MI)
cor(imp_df$outcome,imp_df$diabetes)
cor(imp_df$outcome,imp_df$deficiencyanemias)
cor(imp_df$outcome,imp_df$depression)
cor(imp_df$outcome,imp_df$Hyperlipemia)
cor(imp_df$outcome,imp_df$Rel.failure)
cor(imp_df$outcome,imp_df$heart.rate)
cor(imp_df$outcome,imp_df$Systolic.blood.pressure)
cor(imp_df$outcome,imp_df$Diastolic.blood.pressure)
cor(imp_df$outcome,imp_df$Respiratory.rate)
cor(imp_df$outcome,imp_df$temperature)
cor(imp_df$outcome,imp_df$SP.O2)
cor(imp_df$outcome,imp_df$Urine.output)
cor(imp_df$outcome,imp_df$hematocrit)
cor(imp_df$outcome,imp_df$RBC)
cor(imp_df$outcome,imp_df$MCH)
cor(imp_df$outcome,imp_df$MCHC)
cor(imp_df$outcome,imp_df$MCV)
cor(imp_df$outcome,imp_df$RDW)
cor(imp_df$outcome,imp_df$Leucocyte)
cor(imp_df$outcome,imp_df$Platelets)
cor(imp_df$outcome,imp_df$Neutrophils)
cor(imp_df$outcome,imp_df$Basophils)
cor(imp_df$outcome,imp_df$Lymphocyte)
cor(imp_df$outcome,imp_df$PT)
```

```{r}
cor(imp_df$outcome,imp_df$INR)
cor(imp_df$outcome,imp_df$NT.proBNP)
cor(imp_df$outcome,imp_df$Creatine.kise)
cor(imp_df$outcome,imp_df$Creatinine)
cor(imp_df$outcome,imp_df$Urea.nitrogen)
cor(imp_df$outcome,imp_df$glucose)
cor(imp_df$outcome,imp_df$Blood.potassium)
cor(imp_df$outcome,imp_df$Blood.sodium)
cor(imp_df$outcome,imp_df$Blood.calcium)
cor(imp_df$outcome,imp_df$Chloride)
cor(imp_df$outcome,imp_df$BMI)
cor(imp_df$outcome,imp_df$Anion.gap)
cor(imp_df$outcome,imp_df$Magnesium.ion)
cor(imp_df$outcome,imp_df$PH)
cor(imp_df$outcome,imp_df$Bicarbote)
cor(imp_df$outcome,imp_df$Lactic.acid)
cor(imp_df$outcome,imp_df$PCO2)
cor(imp_df$outcome,imp_df$EF)
```

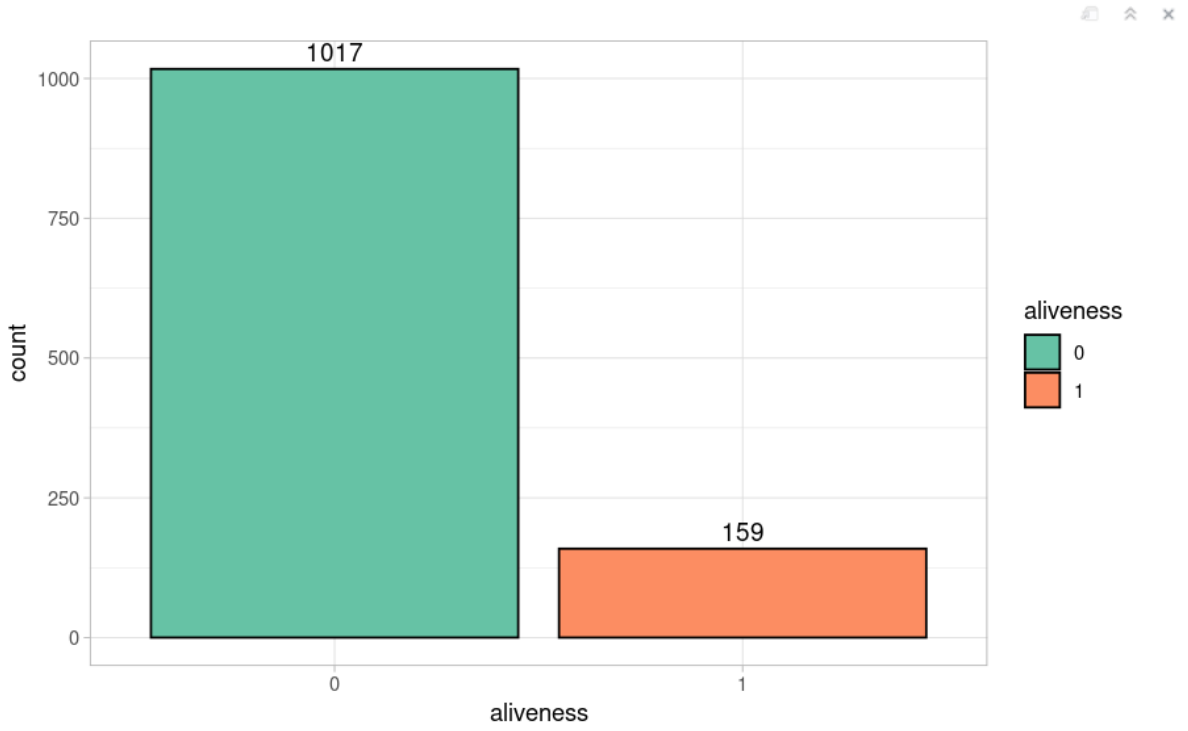
```
[1] 0.1405766
[1] 0.0325957
[1] -0.1014818
[1] -0.02771204
[1] 0.06937784
[1] 0.01367672
[1] -0.05028431
[1] -0.1443279
[1] -0.1046085
[1] 0.047873
[1] -0.1628312
[1] 0.151978
[1] -0.1236475
[1] -0.1032774
[1] 0.1305617
[1] -0.1423999
[1] -0.03425699
[1] -0.1672132
[1] 0.003655336
[1] -0.01140242
[1] 0.02832917
[1] -0.01484627
[1] 0.04449833
[1] 0.09881948
[1] 0.2656676
[1] -0.1771206
[1] 0.1289318
[1] -0.08797684
[1] -0.1898058
[1] 0.2148562
[1] 0.2140523

[1] 0.1673489
[1] 0.1125789
[1] 0.05218842
[1] 0.2466415
[1] 0.1193366
[1] 0.1429006
[1] -0.02856525
[1] -0.2409433
[1] 0.1080335
[1] -0.1014818
[1] 0.3063565
[1] 0.08056279
[1] -0.1774157
[1] -0.2530703
[1] 0.2964591
[1] -0.05940958
[1] -0.04615095
```

The maximum correlation is found between outcome and BMI of the dataset.

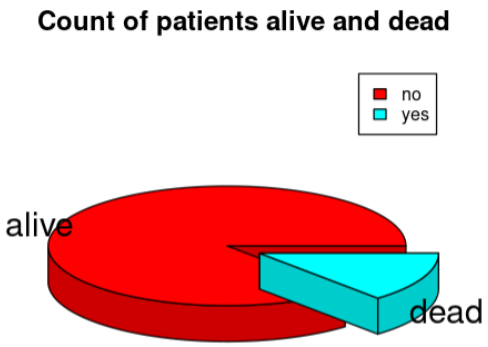
Next, the patient attributes are visualized for better understanding.

```
###### Count of patients who are alive and dead
```{r}
t <- table(df$outcome)
t <- as.data.frame(t)
colnames(t) <- c("aliveness","count")
ggplot(t, aes(x=aliveness, y=count, fill=aliveness)) +
geom_bar(stat="identity", color="black") +
theme_light() +
geom_text(aes(label=count), vjust=-0.4, size=4) +
scale_fill_brewer(palette="Set2")
```
```

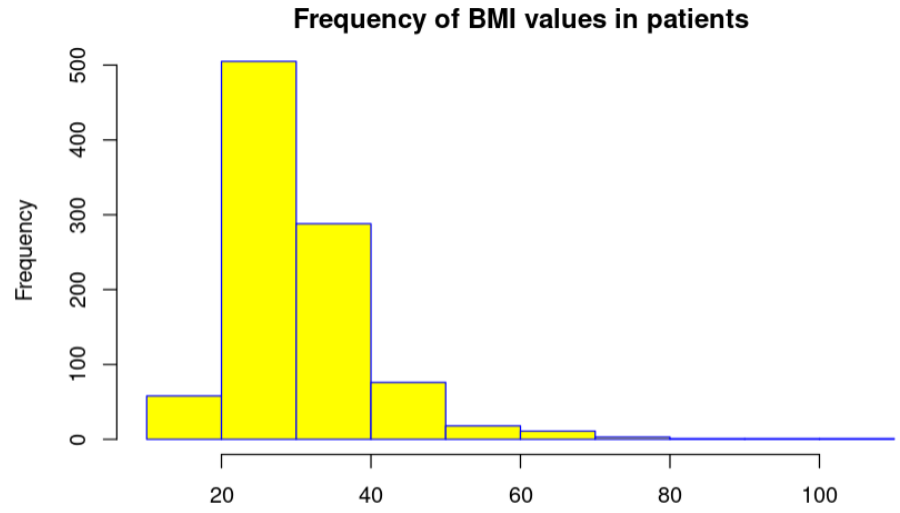


159 patients died while they were admitted to the ICU.

```{r}
library(plotrix)
x <- table(df$outcome)
labels <-  c("alive","dead")
piepercent<- round(100*x/sum(x), 1)
pie3D(x,labels = labels, explode = 0.1, main = "Count of patients alive and dead",col = rainbow(length(x)))
legend("topright", c("no","yes"), cex = 0.8,
   fill = rainbow(length(x)))
```



```r
hist(df$BMI, xlab = "BMI",col = "yellow",border = "blue", main="Frequency of BMI values in patients")
```



Training the models

Firstly, we begin by importing the necessary packages for implementing the classification algorithms.

```r
## 4. Implementation of the prediction models
```

```r
# Importing the necessary libraries
library(caret)
library(e1071)
library(caTools)
library(kernlab)
library(stats19)
library(dplyr)
library(randomForest)
```

The algorithms- Random Forests, Naive Bayes, Logistic Regression, and Support Vector Machine are used in our machine learning models. The K-Fold Cross Validation, a model-validation technique is used to predict each model's accuracy.

Firstly, the data is split into training and testing data in the ratio of 75/25 respectively.

```r
# Splitting the data into training and testing data
# Split ratio is taken as 0.75
sample <- sample.split(imp_df,SplitRatio = 0.75)
training_data <- subset(imp_df,sample==TRUE)
testing_data <- subset(imp_df,sample==FALSE)
```

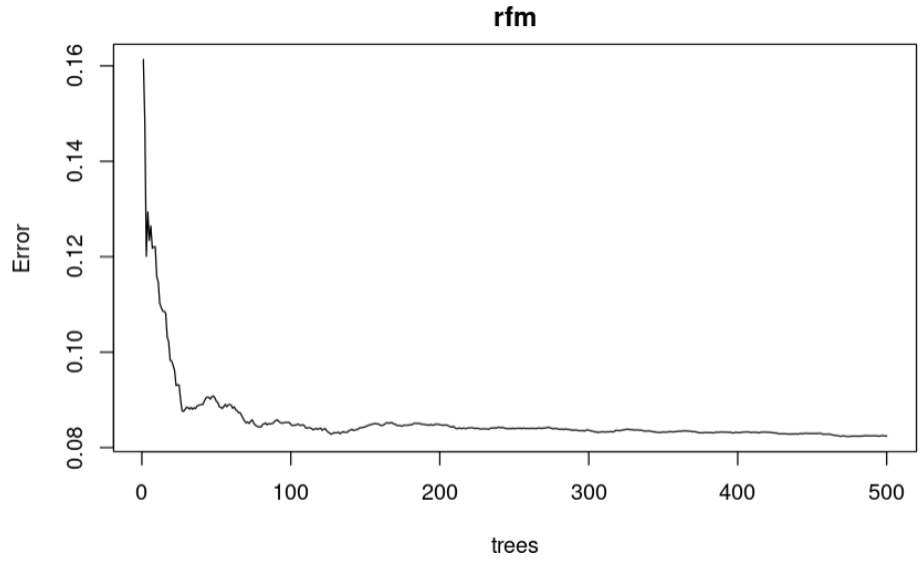
The Random Forest model is implemented at first. The model is plotted and it is seen that the number of errors decreases with the increase in number of trees used in the model.

Model 1: Random Forest

```{r}
# imp_df$outcome[imp_df$outcome == 0] <- "alive"
# imp_df$outcome[imp_df$outcome == 1] <- "dead"
# Training the model
rfm <- randomForest(outcome~.,data=training_data, importance=T, proximity=T)
rfm
```

```
Warning in randomForest.default(m, y, ...) :
  The response has five or fewer unique values.  Are you sure you want to do regression?

Call:
 randomForest(formula = outcome ~ ., data = training_data, importance = T,      proximity = T)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 16

          Mean of squared residuals: 0.0824616
                    % Var explained: 27.49
```

```
plot(rfm)
```




```{r}
pred <- predict(rfm,testing_data)
pred <- ifelse(pred >0.5, 1, 0)
pred
```

```
  1   3  18  22  40  46  47  49  66  67  77  89 107 108 109 130 131 140 143 144
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
145 156 160 168 183 213 216 219 235 237 259 263 264 269 278 281 290 299 332 335
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
338 363 366 376 379 381 384 397 398 405 418 448 450 451 462 463 479 482 486 488
  0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
497 498 512 528 553 554 555 571 572 579 587 593 595 606 607 617 627 647 649 650
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   0
663 664 829 832 835 836 847 850 857 878 895 899 900 920 923 934 937 939 940 952
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
959 965 982 1011 1012 1013 1026 1028 1046 1051 1053 1055 1064 1065 1072 1084
  0   0   0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```{r}
confusionMatrix(cm)
```

```
Confusion Matrix and Statistics

   pred
     0  1
  0 93  0
  1 20  3

               Accuracy : 0.8276
                 95% CI : (0.7464, 0.8914)
    No Information Rate : 0.9741
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1939

 Mcnemar's Test P-Value : 2.152e-05

            Sensitivity : 0.8230
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.1304
             Prevalence : 0.9741
         Detection Rate : 0.8017
   Detection Prevalence : 0.8017
      Balanced Accuracy : 0.9115

       'Positive' Class : 0
```

The Random Forest gives an accuracy rate of 82.76%. Next, the Naive Bayes model is implemented, which predicts data based on probabilities.

Model 2: Naive Bayes Model

```{r}
set.seed(234)
# Training the model
model <- naiveBayes(outcome~., data=training_data)
model
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.8691589 0.1308411

   PCO2
Y        [,1]      [,2]
  0 0.3321511 0.1590255
  1 0.3125131 0.1878149

   EF
Y       [,1]      [,2]
  0 0.562724 0.2119256
  1 0.531746 0.2523282
```

```{r}
# Predicting on test data
pred <- predict(model, testing_data)
pred
```

```
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
 [49] 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 [97] 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0
Levels: 0 1
```

```
confusionMatrix(cm)
```

```
Confusion Matrix and Statistics

   pred
     0  1
  0 83 10
  1 12 11

               Accuracy : 0.8103
                 95% CI : (0.7271, 0.8772)
    No Information Rate : 0.819
    P-Value [Acc > NIR] : 0.6494

                  Kappa : 0.3833

 Mcnemar's Test P-Value : 0.8312

            Sensitivity : 0.8737
            Specificity : 0.5238
         Pos Pred Value : 0.8925
         Neg Pred Value : 0.4783
             Prevalence : 0.8190
         Detection Rate : 0.7155
   Detection Prevalence : 0.8017
      Balanced Accuracy : 0.6987

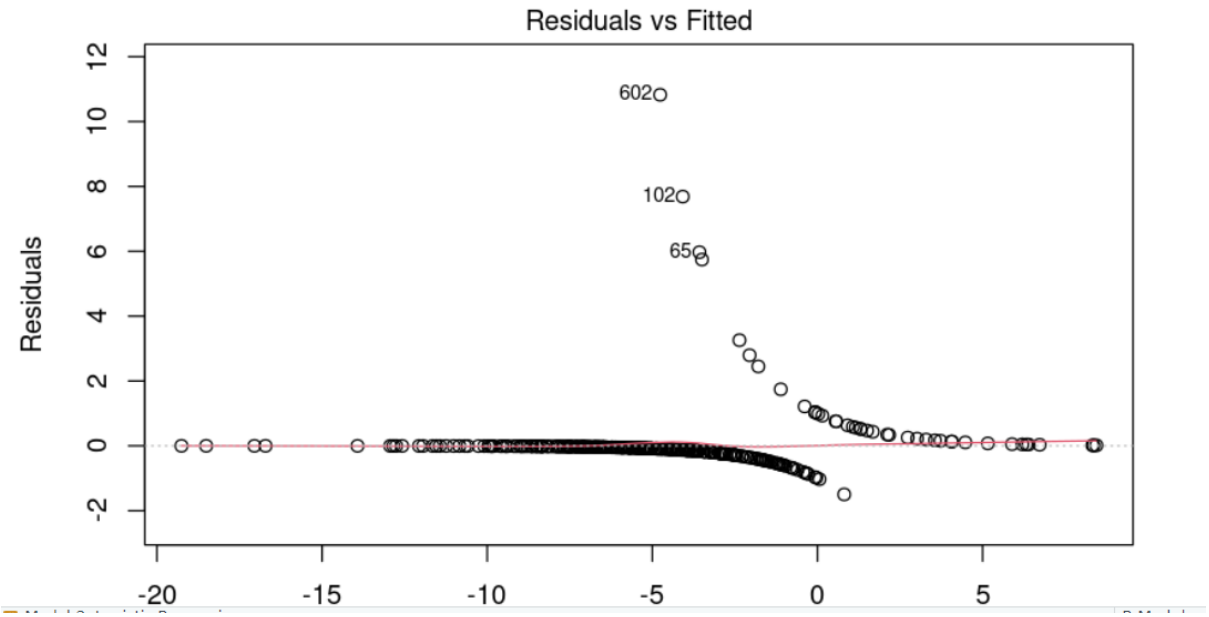
       'Positive' Class : 0
```

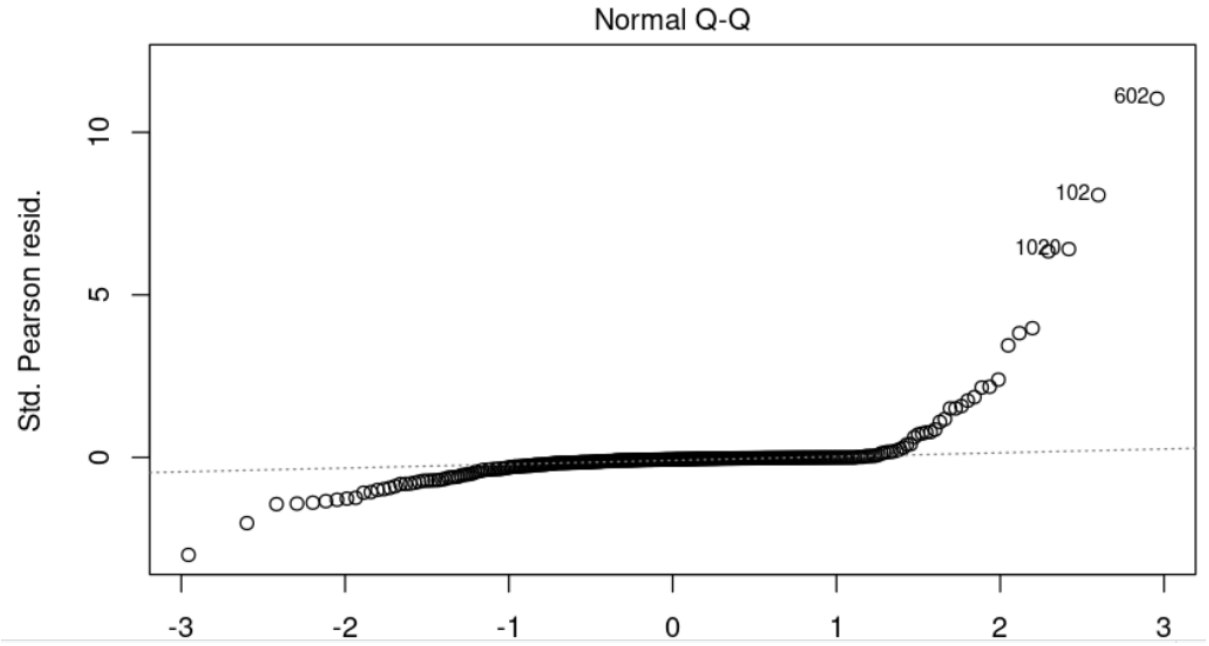
The Naive Bayes technique gives an accuracy of 81.03%. Next, we implemented the Logistic Regression model.

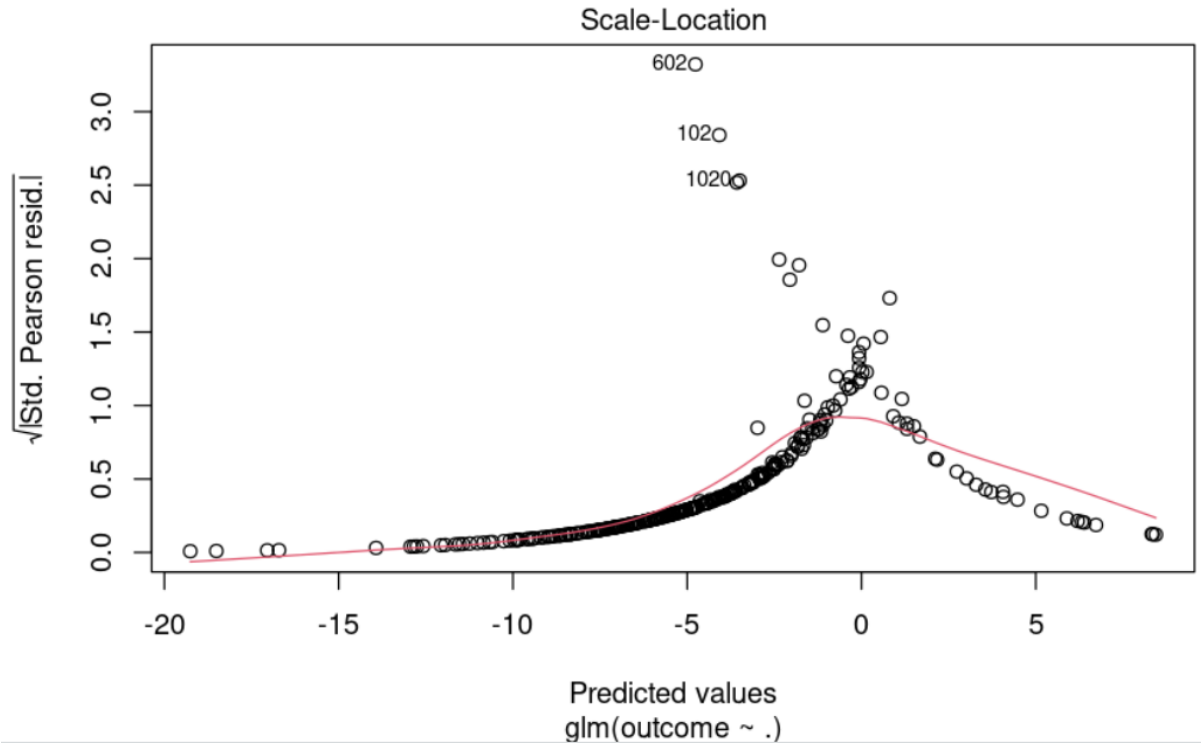
```
## Model 3: Logistic Regression
```{r}
# Training the model
logistic_model <- glm(outcome~.,
                      data = training_data,
                      family = "binomial")
logistic_model
```
```

```
Call:  glm(formula = outcome ~ ., family = "binomial", data = training_data)

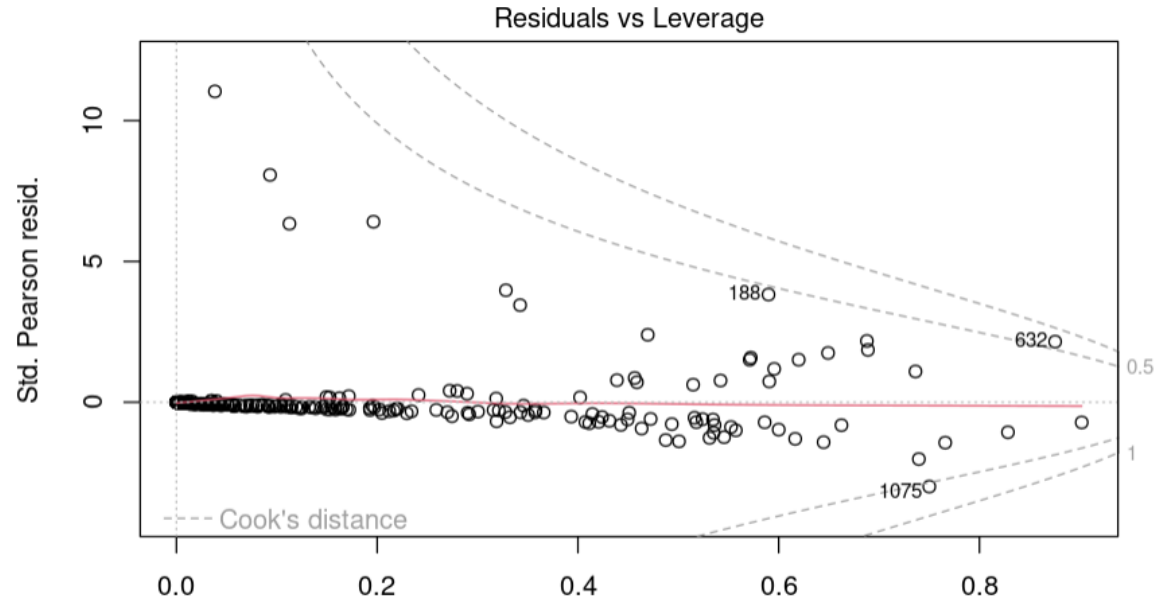
Coefficients:
             (Intercept)                      age                   gendera
                -19.58363                  0.04726                  -0.32320
                     BMI             hypertensive       atrialfibrillation
                 -1.57571                 -0.83468                   0.81813
          CHD.with.no.MI                 diabetes         deficiencyanemias
                 -1.09270                 -0.08967                  -1.45402
              depression             Hyperlipemia               Rel.failure
                 -0.54612                 -0.18155                  -2.25264
                    COPD               heart.rate   Systolic.blood.pressure
                 -2.65935                  0.38987                   4.48410
Diastolic.blood.pressure         Respiratory.rate               temperature
                 -4.01108                  3.28167                  -1.16126
                   SP.O2             Urine.output                hematocrit
                  1.25231                  0.75006                 -39.28073

Degrees of Freedom: 320 Total (i.e. Null);  272 Residual
Null Deviance:      249.1
Residual Deviance: 94.24        AIC: 192.2
```

```
```{r}
plot(logistic_model)
```
```


Residuals vs Fitted
Residuals
602
102
65
12
10
8
6
4
2
0
-2
-20
-15
-10
-5
0
5


Normal Q-Q
Std. Pearson resid.
602
102
1020
10
5
0
-3
-2
-1
0
1
2
3


Scale-Location
602
102
1020
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
-20
-15
-10
-5
0
5
Predicted values
glm(outcome ~ .)


Residuals vs Leverage
Std. Pearson resid.
0
5
10
188
632
1075
0.5
1
Cook's distance
0.0
0.2
0.4
0.6
0.8

```
# Summary of the model
summary(logistic_model)
```

```
Call:
glm(formula = outcome ~ ., family = "binomial", data = training_data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.53440  -0.22311  -0.08160  -0.01853   3.08940

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -19.58363   27.38484  -0.715   0.4745
age                        0.04726    2.45625   0.019   0.9846
gendera                   -0.32320    0.84053  -0.385   0.7006
BMI                       -1.57571    3.12928  -0.504   0.6146
hypertensive              -0.83468    0.80279  -1.040   0.2985
atrialfibrillation         0.81813    0.83522   0.980   0.3273
CHD.with.no.MI            -1.09270    1.37826  -0.793   0.4279
diabetes                  -0.08967    0.85905  -0.104   0.9169
deficiencyanemias         -1.45402    1.02532  -1.418   0.1562
depression                -0.54612    1.18629  -0.460   0.6453
Hyperlipemia              -0.18155    0.77301  -0.235   0.8143
Rel.failure               -2.25264    1.02037  -2.208   0.0273 *
COPD                      -2.65935    1.58694  -1.676   0.0938 .
heart.rate                 0.38987    3.02638   0.129   0.8975
Systolic.blood.pressure    4.48410    2.54828   1.760   0.0785 .
Diastolic.blood.pressure  -4.01108    4.16337  -0.963   0.3353
Respiratory.rate           3.28167    2.35782   1.392   0.1640
temperature               -1.16126    3.31748  -0.350   0.7263
SP.O2                      1.25231    4.07540   0.307   0.7586
Urine.output               0.75006    2.68189   0.280   0.7797

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 249.085  on 320  degrees of freedom
Residual deviance:  94.241  on 272  degrees of freedom
AIC: 192.24

Number of Fisher Scoring iterations: 8
```

```{r}
# Predicting on test data
predict_reg <- predict(logistic_model,
                       testing_data, type = "response")
predict_reg
```

```
           1            3           18           22           40           46           47
1.642957e-04 2.330439e-03 1.705553e-06 3.999103e-04 1.829028e-02 3.811438e-05 5.457722e-05
          49           66           67           77           89          107          108
3.501901e-05 1.081138e-05 3.403493e-04 2.875941e-03 1.197566e-01 2.769984e-03 3.176698e-01
         109          130          131          140          143          144          145
5.705318e-04 9.177734e-02 6.065375e-02 5.969365e-03 2.130816e-01 1.774232e-02 3.363017e-02
         156          160          168          183          213          216          219
1.344078e-03 2.446125e-03 4.994924e-04 9.129937e-04 6.878798e-05 6.583662e-02 1.679579e-03
         235          237          259          263          264          269          278
1.355658e-03 9.666634e-05 7.377978e-01 3.385293e-01 9.655207e-03 2.072567e-01 1.096900e-02
         281          290          299          332          335          338          363
1.480014e-01 3.410291e-01 2.564907e-03 1.847041e-03 9.135429e-06 4.102991e-02 9.510315e-01
         366          376          379          381          384          397          398
```

```{r}
# Changing probabilities
predict_reg <- ifelse(predict_reg >0.5, 1, 0)
predict_reg
```

```
   1    3   18   22   40   46   47   49   66   67   77   89  107  108  109  130  131  140  143  144
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
 145  156  160  168  183  213  216  219  235  237  259  263  264  269  278  281  290  299  332  335
   0    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0
 338  363  366  376  379  381  384  397  398  405  418  448  450  451  462  463  479  482  486  488
   0    1    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
 497  498  512  528  553  554  555  571  572  579  587  593  595  606  607  617  627  647  649  650
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0
 663  664  829  832  835  836  847  850  857  878  895  899  900  920  923  934  937  939  940  952
   0    1    0    0    0    0    0    0    0    0    0    0    1    0    0    0    1    0    0    0
 959  965  982 1011 1012 1013 1026 1028 1046 1051 1053 1055 1064 1065 1072 1084
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```{r}
# Confusion matrix
confusionMatrix(cm)
```

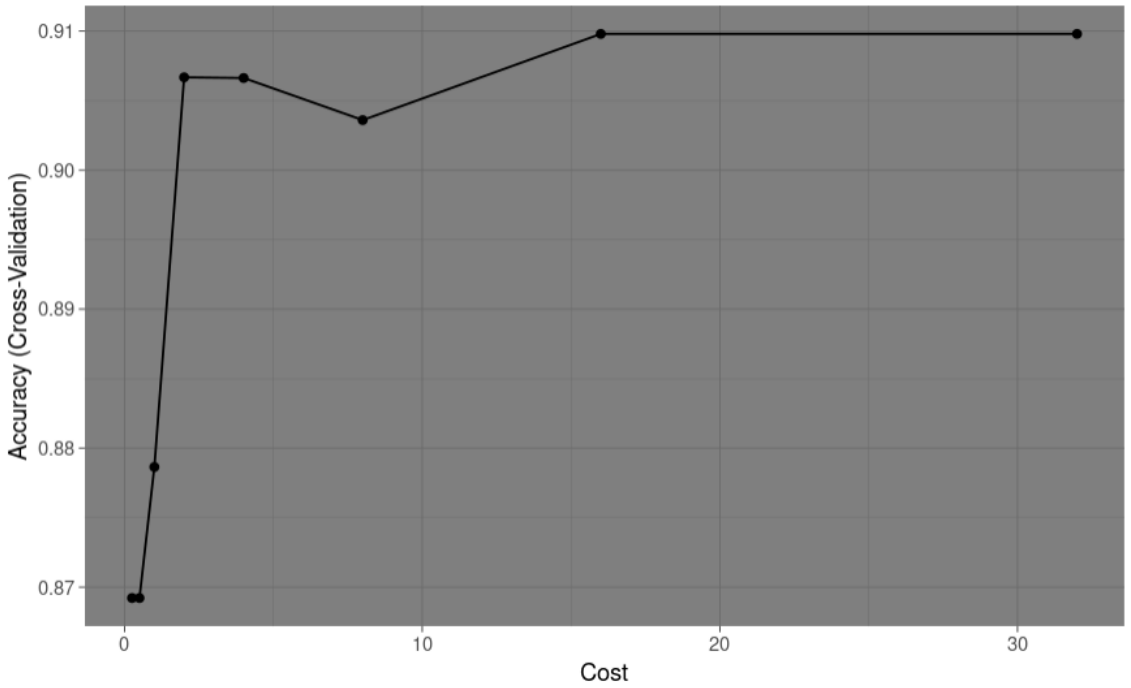
```
Confusion Matrix and Statistics

   predict_reg
     0  1
  0 88  5
  1 20  3

               Accuracy : 0.7845
                 95% CI : (0.6985, 0.8554)
    No Information Rate : 0.931
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.1016

 Mcnemar's Test P-Value : 0.00511

            Sensitivity : 0.8148
            Specificity : 0.3750
         Pos Pred Value : 0.9462
         Neg Pred Value : 0.1304
             Prevalence : 0.9310
         Detection Rate : 0.7586
   Detection Prevalence : 0.8017
      Balanced Accuracy : 0.5949

       'Positive' Class : 0
```

Model 4: Support Vector Machine

```{r}
set.seed(23) # for reproducibility
svm <- train(as.factor(outcome) ~ .,
data = training_data,
method = "svmRadial",
trControl = trainControl(method = "cv", number = 5),
tuneLength = 8
)

svm
```

```
Support Vector Machines with Radial Basis Function Kernel

321 samples
 48 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 256, 258, 257, 257, 256
Resampling results across tuning parameters:

  C      Accuracy   Kappa
   0.25  0.8692186  0.0000000
   0.50  0.8692186  0.0000000
   1.00  0.8786432  0.1685691
   2.00  0.9066751  0.4519232
   4.00  0.9066255  0.4915281
   8.00  0.9035981  0.5043186
  16.00  0.9098001  0.5262462
```

```{r}
ggplot(svm) + theme_dark()
```




```{r}
ctrl <- trainControl(
method = "cv",
number = 15,
classProbs = TRUE,
summaryFunction = twoClassSummary # also needed for AUC/ROC
)
ctrl
```

```
$method
[1] "cv"

$number
[1] 15
```

```{r}
training_data$outcome<-as.factor(training_data$outcome)
levels(training_data$outcome) <- c("alive", "dead")
```

```{r}
# Tune an SVM
set.seed(23) # for reproducibility
outcome_svm_auc <- train(
outcome ~ .,
data = training_data,
method = "svmRadial",
metric = "ROC", # area under ROC curve (AUC)
trControl = ctrl,
tuneLength = 15)

outcome_svm_auc
```

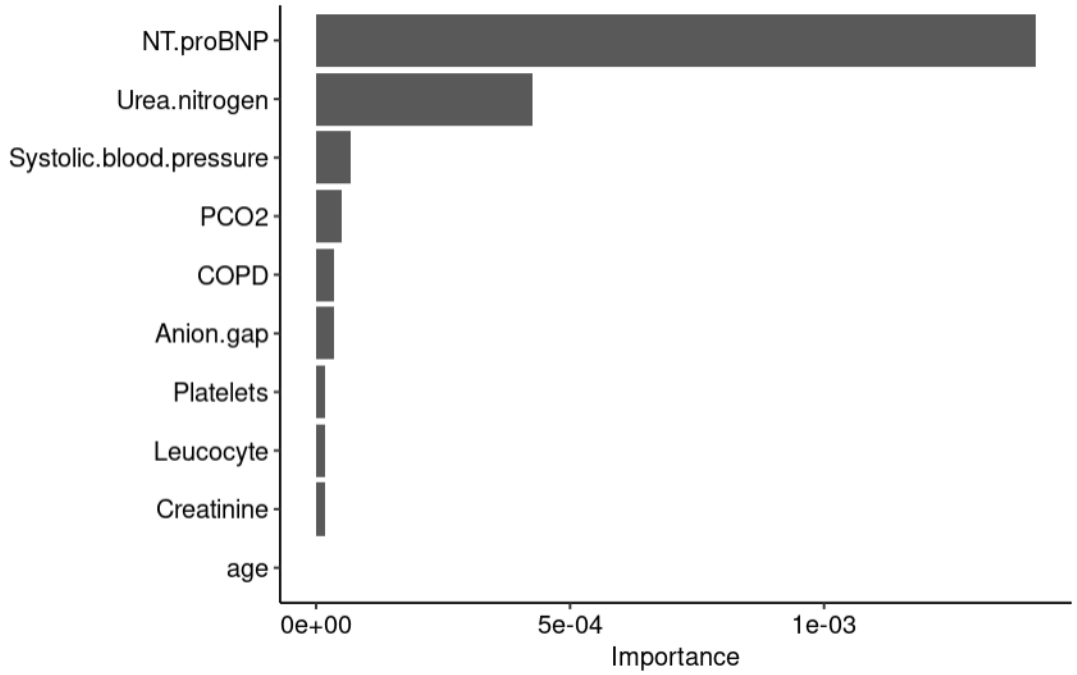
```
Support Vector Machines with Radial Basis Function Kernel

321 samples
 48 predictor
  2 classes: 'alive', 'dead'

No pre-processing
Resampling: Cross-Validated (15 fold)
Summary of sample sizes: 299, 300, 300, 299, 299, 300, ...
Resampling results across tuning parameters:

  C     ROC        Sens       Spec
  0.25  0.8632878  0.9789474  0.5000000
```

```{r}
confusionMatrix(outcome_svm_auc)
```

```
Cross-Validated (15 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction alive dead
     alive  85.7  7.2
     dead    1.2  5.9

 Accuracy (average) : 0.9159
```

```{r}
# Arranging variables in terms of importance
prob_alive <- function(object, newdata) {
predict(object, newdata = newdata, type = "prob")[, "dead"]
}
```

```{r}
# Importing necessary libraries
library(vip)
set.seed(2827) # for reproducibility
# Visualizing variables in terms of their importance
vip(outcome_svm_auc, method = "permute", nsim = 5, train = training_data,
target = "outcome", metric = "auc", reference_class = "dead",
pred_wrapper = prob_alive)
```



Out of all factors, NT-proBNP, Urea-nitrogen levels, blood pressure, high partial pressure of carbon dioxide and increasing age are the leading causes of deaths in ICU-admitted heart failure patients. It's likely that you have heart failure if your BNP or NT-proBNP levels were higher than normal. The higher the amount, the more serious your disease is likely to be.

```{r}
pred <- predict(outcome_svm_auc, testing_data)
pred
```

```
  [1] alive alive alive alive alive alive alive alive alive alive alive alive alive alive alive alive
 [17] alive alive alive alive alive alive alive alive alive alive alive alive alive alive alive alive
 [33] alive alive alive alive alive alive alive alive alive dead  alive alive dead  alive alive alive
 [49] alive alive alive alive alive alive alive alive alive alive alive dead  alive alive alive alive
 [65] alive alive alive alive alive alive alive alive alive dead  alive alive alive dead  dead  alive
 [81] alive dead  alive alive alive alive alive alive alive alive alive alive dead  alive alive alive
 [97] alive alive alive alive alive alive alive alive alive dead  alive alive alive alive alive alive
[113] alive alive alive alive
Levels: alive dead
```

```{r}
testing_data$outcome<-as.factor(testing_data$outcome)
levels(testing_data$outcome) <- c("alive", "dead")
confusionMatrix(table(pred,as.factor(testing_data$outcome)))
```

```
Confusion Matrix and Statistics


pred    alive dead
  alive    89   18
  dead      4    5

               Accuracy : 0.8103
                 95% CI : (0.7271, 0.8772)
    No Information Rate : 0.8017
    P-Value [Acc > NIR] : 0.462936

                  Kappa : 0.2262

 Mcnemar's Test P-Value : 0.005578

            Sensitivity : 0.9570
            Specificity : 0.2174
         Pos Pred Value : 0.8318
         Neg Pred Value : 0.5556
             Prevalence : 0.8017
         Detection Rate : 0.7672
   Detection Prevalence : 0.9224
      Balanced Accuracy : 0.5872

       'Positive' Class : alive
```

The Support Vector Machine gives an accuracy of 81.03%.

Algorithm Analysis

Finally, models are trained and the cross-validation results are obtained. The K-Fold Cross Validation predicts each model's accuracy.

```{r}
## Comparative analysis of all the prediction models
```{r}
models <- c("NB", "LGR", "SVM", "RF")
accuracy <- c(81.03, 78.45, 81.03, 82.76)
mdls <- data.frame(models,accuracy)
```
```

```{r}
p<-ggplot(data=mdls, aes(x=models, y=accuracy)) +
  geom_bar(stat="identity", fill="black")+
  geom_text(aes(label=accuracy), vjust=1.6, color="white", size=3.5)+
  theme_minimal()
p
```

As shown above, the names and accuracy of the four different algorithms are displayed. The highest accuracy is given by the Random Forests(RF), followed by the Support Vector Machine(SVM), and Naive Bayes Algorithm(NB). The least accuracy is given by Logistic Regression(LR). Figure hows a graphical comparison chart for the different machine learning classifiers using a bar plot.

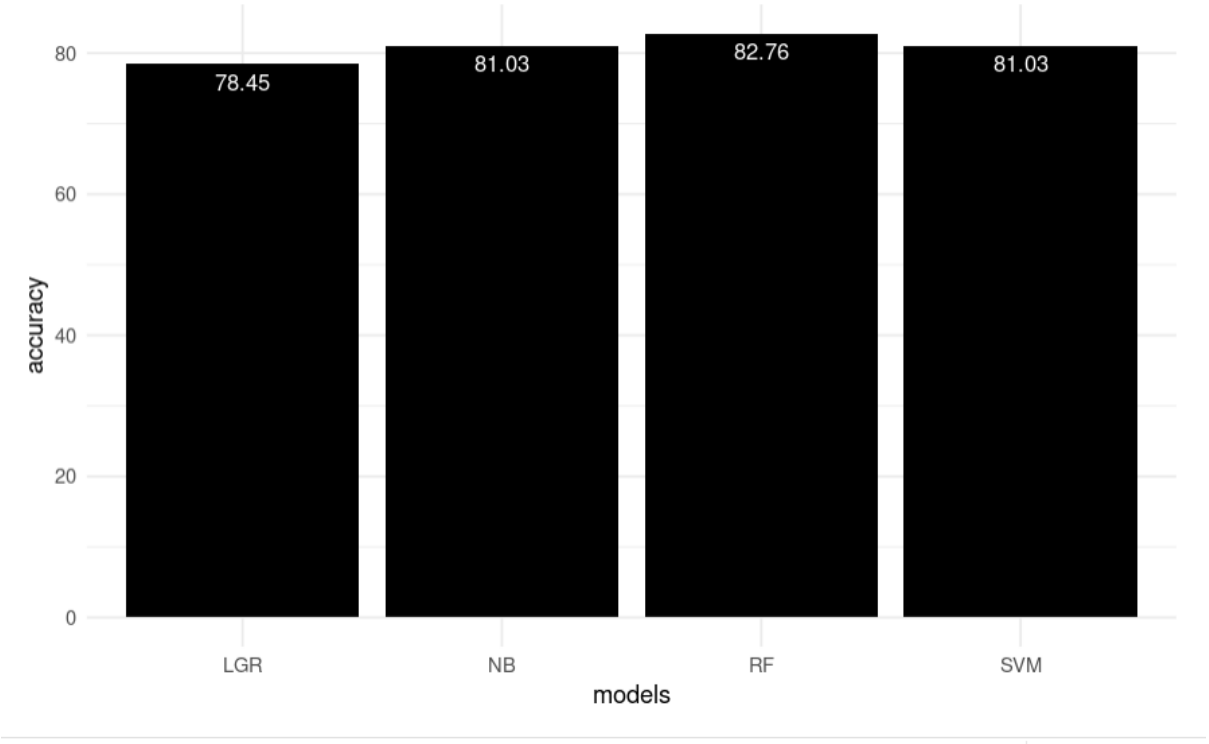


Figure 10. Comparison of all the machine learning algorithms

10. Conclusion

Different classification techniques that can be used for predicting the mortality status were analyzed in this study. The performance of the five machine learning algorithms- Naive Bayes, Logistic Regression, Support Vector Machine, Linear Regression and Random Forests was investigated. The Random Forest algorithm gave the maximum accuracy rate. The algorithm's effectiveness in recognising and classifying near-to-death patients was demonstrated by the best results, which were obtained with very little computational effort. Another advantage of this method is the ability to detect mortality at an early stage. By training it with a wide range of train datasets, it can be extended to detect many more griveously-ill patients. As a result, the model incorporates information technology into the medical domain and is conducive to the long-term growth of medical industry. Out of all factors, renal failure, high level of leukocytes, anemia deficiencies, high partial pressure of carbon dioxide and increasing age are the leading causes of deaths in ICU-admitted patients. To further improve the recognition rate in the classification process, Artificial Neural Network, Convolutional Neural Networks, Fuzzy Logic and hybrid algorithms can also be used.

11. References

[1] Kong, G., Lin, K. & Hu, Y. Using machine learning methods to predict in-hospital mortality of sepsis patients in the ICU. (2020).

[2] Awad A, Bader-El-Den M, McNicholas J, Briggs J. Early hospital mortality prediction of intensive care unit patients using an ensemble learning approach. (2017)

[3] Li M, Chen H, Yan S, Xu X, Xu H. Application of Deep Learning Technology in Predicting the Risk of Inpatient Death in Intensive Care Unit. (2021)

[4] Stewart, K., Choudry, M. I., & Buckingham, R. (2016). Learning from hospital mortality. Clinical medicine (London, England)

[5] Rosenthal N, Cao Z, Gundrum J, Sianis J, Safo S. Risk Factors Associated With In-Hospital Mortality in a US National Sample of Patients With COVID-19.