ADA 442 - Statistical Learning

Predicting Heart Failure With Different Models

Arda Gemalmaz 16561060858 Ebru Kılıç 14449202680

04 December, 2023

Contents

1 Introduction

✓ Cardiovascular diseases are one of the biggest dangers to human life. As observed, approximately 17.9 million people are dying from cardiovascular diseases. Some people which have cardiovascular risk or disease need to be detected before it is too late. This project will be a great help for that. The goal of this project is to predict heart disease on sample data and make a classification to identify which category belongs to based on a training set while we are doing observations on the dataset. The dataset that we are using contains 13 clinical features that we will use to predict the death rate.

2 Methodology

✓ The methodology of this work is to train Logistic Regression and Decision Tree models. The logistic regression model is trained in two ways. Firstly trained with all of the predictors and secondly with some chosen predictors. Also, the decision tree model is trained by using the unpruned, pruned, Rpart library, and 5-fold cross-validation decision tree models.

3 Dataset

✓ The dataset is from Kaggle. This can be findable in the references part. The columns are "age, anaemia, creatinine_phosphokinase, diabetes, ejection_fraction, high_blood_pressure, platelets, serum_creatinine, serum_sodium, and sex".

```r
# Data Loading
HeartFailure = read.csv("HeartFailureClinicalRecords.csv")
head(HeartFailure)
```

```
##   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
```

4 Libraries

```
set.seed(85868)
library(corrplot)
library(caret)
library(tree)
library(tidyverse)
library(rpart)
library(mlbench)
library(glm2)
library(ISLR2)
library(boot)
```

5 Data Analysis

There are 13 clinical features and 299 observations. Predictors should be analyzed by getting more precise judgments about the data. In our data, there are numerous binaural and categorical data. Numerical data values are age, creatinine_phosphokinase, ejection_fraction, platelets, and serum_creatinine.

```
# Data descriptives.
dim(HeartFailure)
```

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

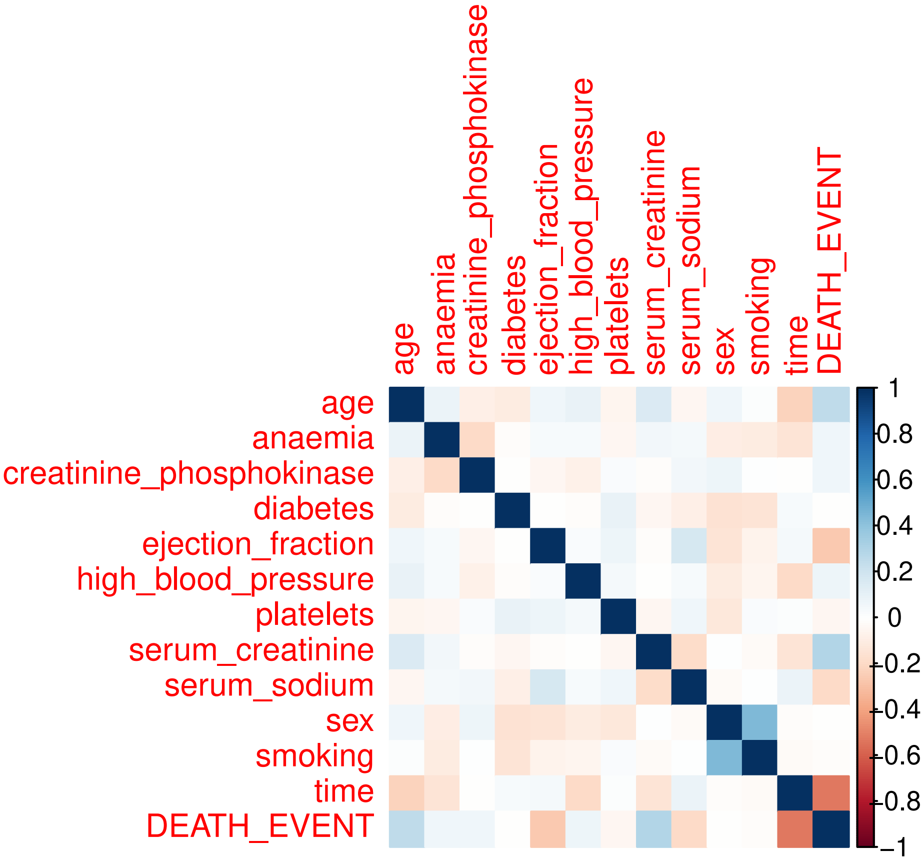
```
summary(HeartFailure)
```

```
##       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
```

5.1 Corralation Matrix

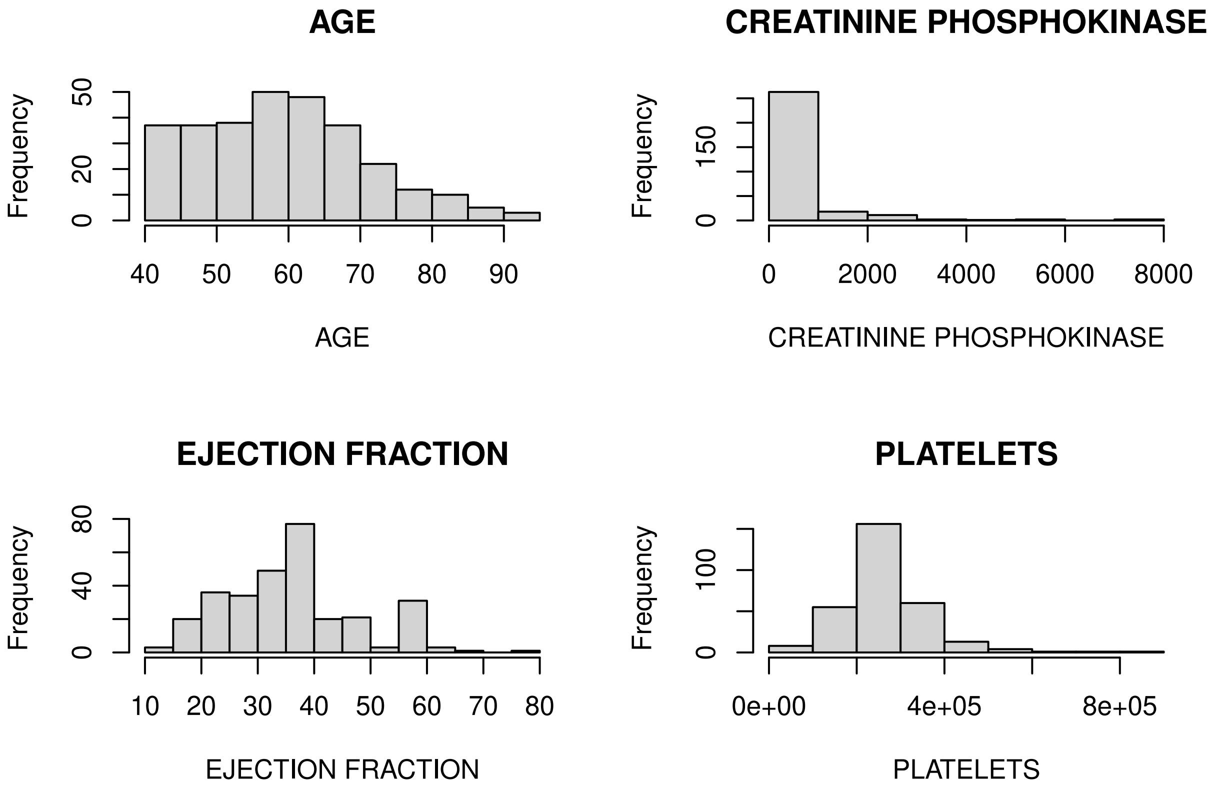
✓ To see whether or not redundant predictors exist in the data correlation matrix should be checked. The correlation matrix was created with numeric data.

```
# Creating correlation matrix.
ColumnNumber = unlist(lapply(HeartFailure, is.numeric))
numericdata = HeartFailure[ , ColumnNumber]
correlationmatrix = cor(numericdata)
corrplot(correlationmatrix, method = "color")
```

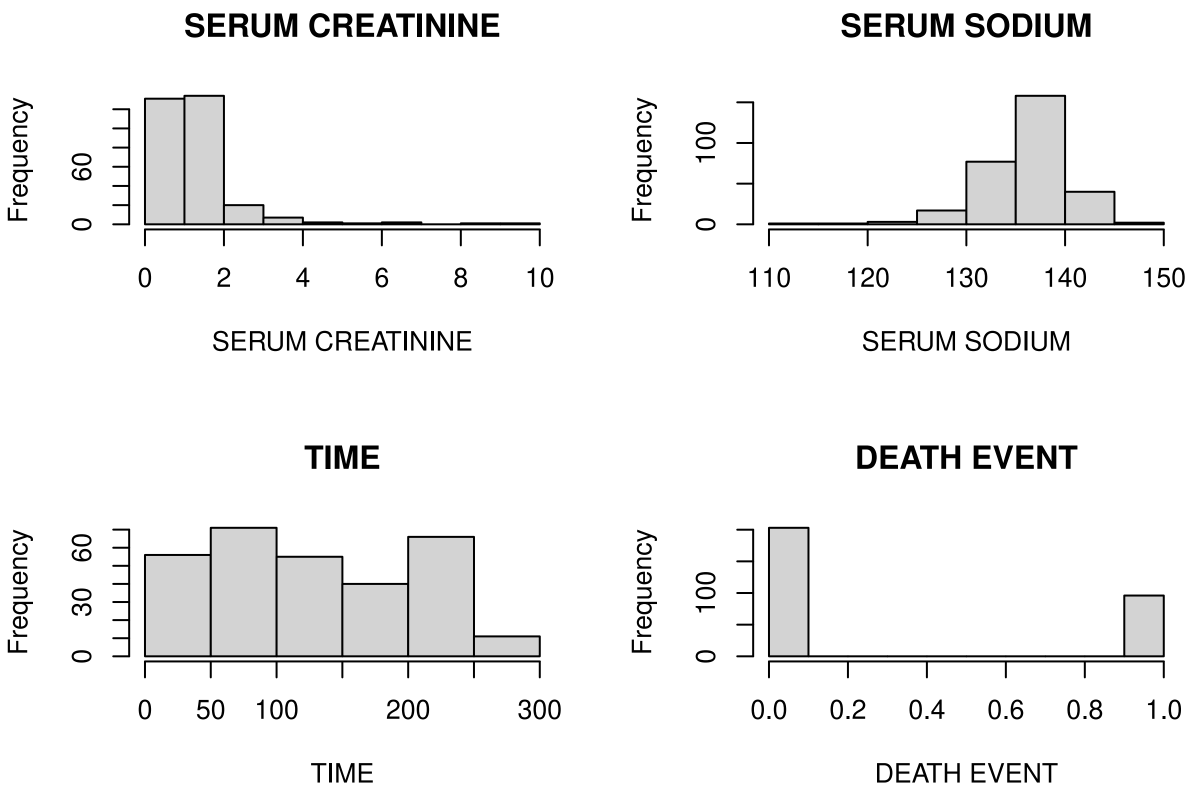


As seen in the correlation matrix there are no redundant predictors. ? how

```
# Histogram.
par(mfrow = c(2, 2))
hist(HeartFailure$age, main = "AGE", xlab = "AGE")
hist(HeartFailure$creatinine_phosphokinase, main = "CREATININE PHOSPHOKINASE", xlab = "CREATININE PHOSP
hist(HeartFailure$ejection_fraction, main = "EJECTION FRACTION", xlab = "EJECTION FRACTION")
hist(HeartFailure$platelets, main = "PLATELETS", xlab = "PLATELETS")
```

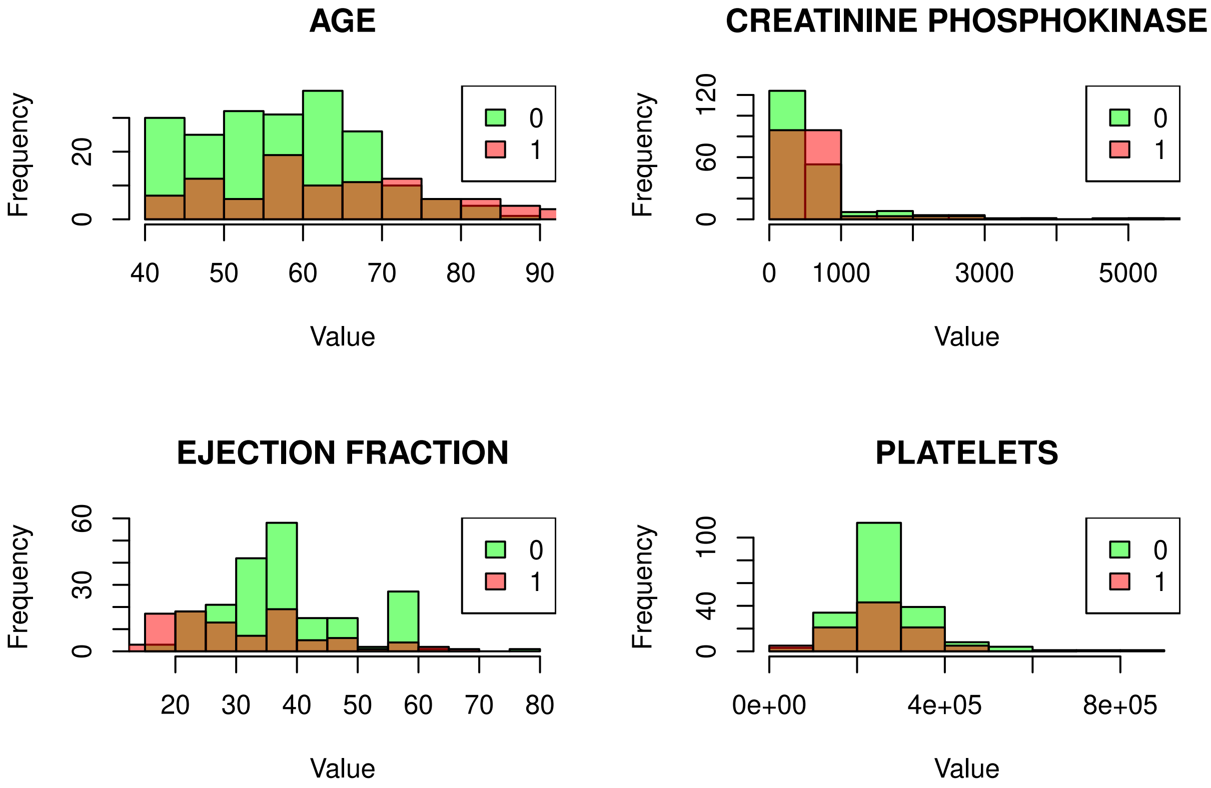


```
hist(HeartFailure$serum_creatinine, main = "SERUM CREATININE", xlab = "SERUM CREATININE")
hist(HeartFailure$serum_sodium, main = "SERUM SODIUM", xlab = "SERUM SODIUM")
hist(HeartFailure$time, main = "TIME", xlab = "TIME")
hist(HeartFailure$DEATH_EVENT, main = "DEATH EVENT", xlab = "DEATH EVENT")
```

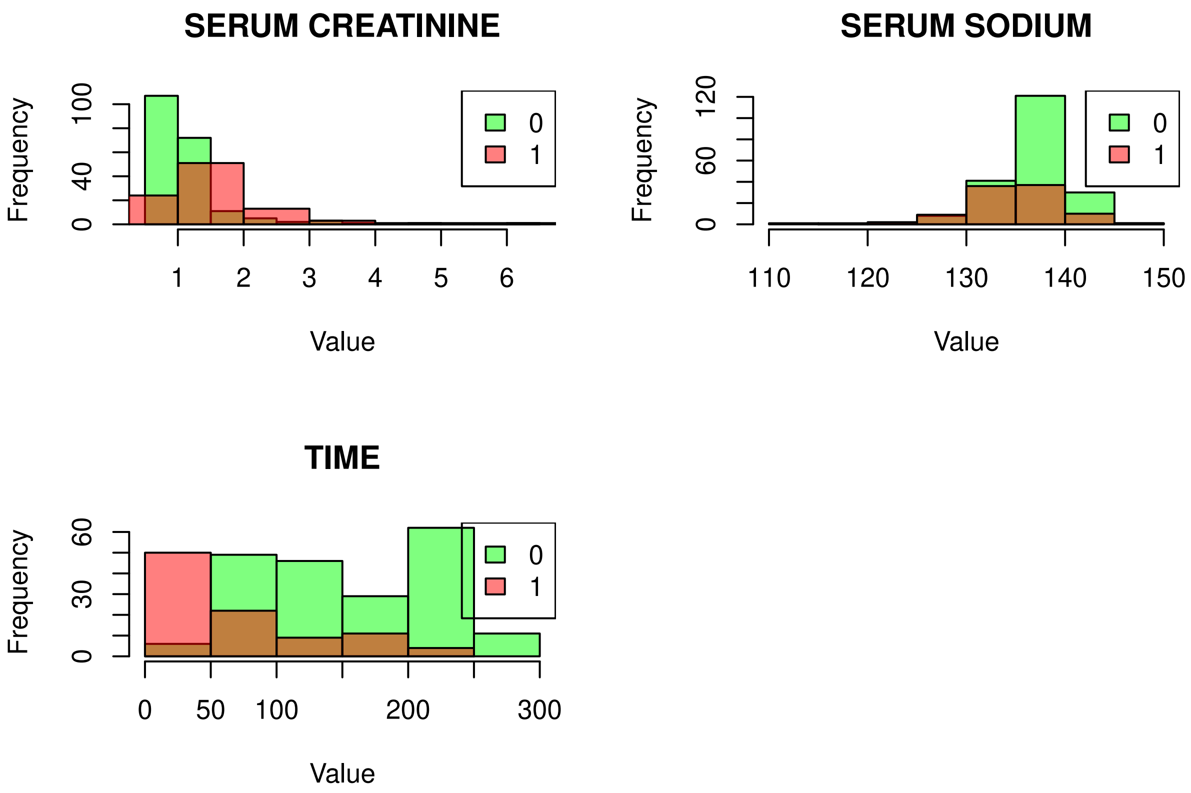


5.2 Numerical Data Analysis

```
par(mfrow = c(2, 2))
Histogram(HeartFailure$age, "AGE")
Histogram(HeartFailure$creatinine_phosphokinase, "CREATININE PHOSPHOKINASE")
Histogram(HeartFailure$ejection_fraction, "EJECTION FRACTION")
Histogram(HeartFailure$platelets, "PLATELETS")
```



```
Histogram(HeartFailure$serum_creatinine, "SERUM CREATININE")
Histogram(HeartFailure$serum_sodium, "SERUM SODIUM")
Histogram(HeartFailure$time, "TIME")
```



✓

5.2.1 AGE

The age data has accumulated around 60. Also, patients after 70 are more likely to die from heart failure.

5.2.2 CREATININE PHOSPHOKINASE

The creatinine phosphokinase (CPK) data represent the level of the CPK enzyme in the blood (mcg/L). The healthy range for CPK is between 10 and 120 (mcg/L). Generally, deaths from heart disease happened the value lower than 1000.

✓

5.2.3 EJECTION FRACTION

The ejection fraction data is the percentage of blood leaving the heart at each contradiction. A healthy person's ejection fraction is between %50 and %75. In our data, most deaths from heart failure happened when the ejection fraction is below %25.

5.2.4 PLATELETS

The platelets data represents platelets in the blood (kiloplatelets/mL). Normal platelet in adults should be between 150k and 450k. If the platelet value is less than 150k stopping the bleeding will be a problem. In the data patients who has platelet lower than 100k has a higher risk of heart failure.

5.2.5 SERUM CREATININE

The serum creatinine data represent the level of the serum creatinine in the blood (mg/dL). For healthy patients, the range for serum creatinine should be 0.59 to 1.35 (mg/dL). If the values are not between 0.59 and 1.35 (mg/dL) patients have a high risk of dying from heart failure.

5.2.6 SERUM SODIUM

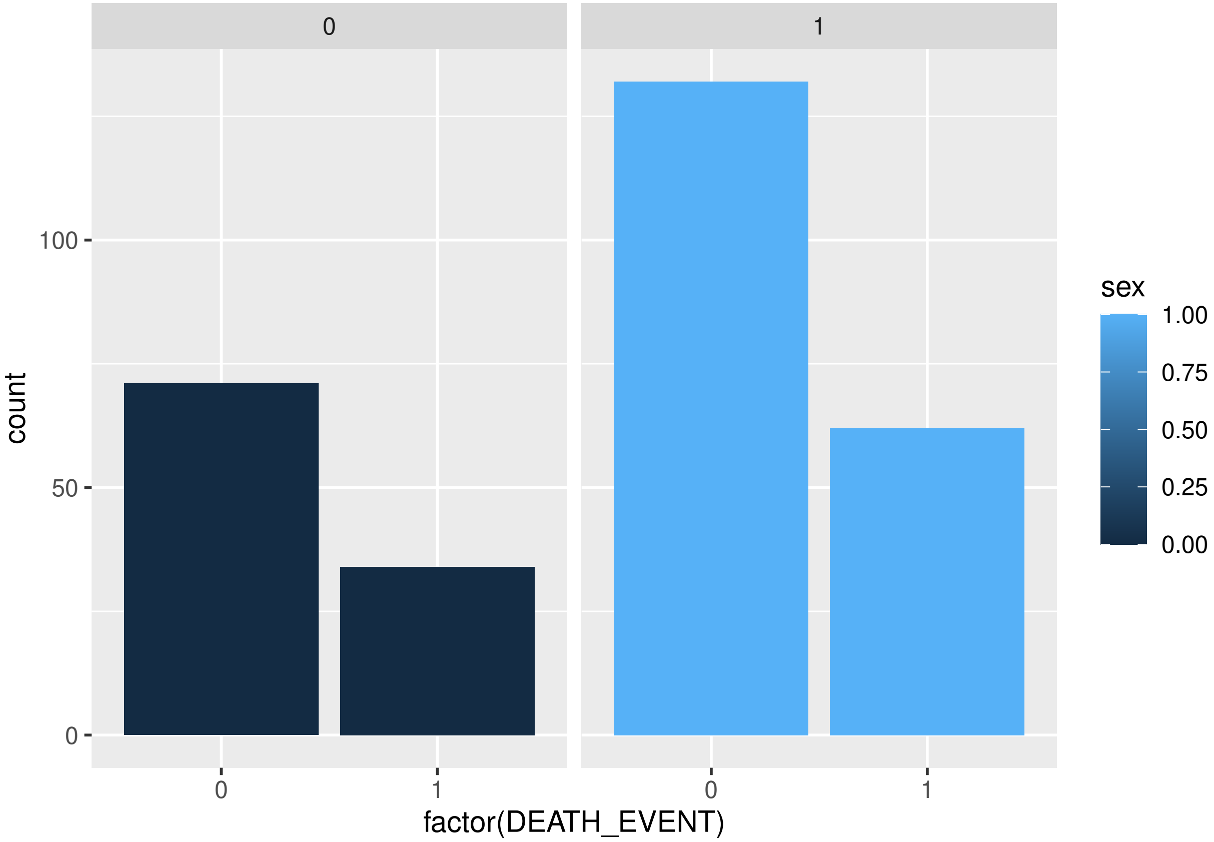
The serum sodium data represent the level of serum sodium in the blood (mEq/L). For healthy patients, the range for serum sodium should be 135 to 145 (mEq/L). In the data as seen roughly the range other than 130 to 145 (mEq/L) has a higher risk of dying from heart failure.

5.2.7 TIME

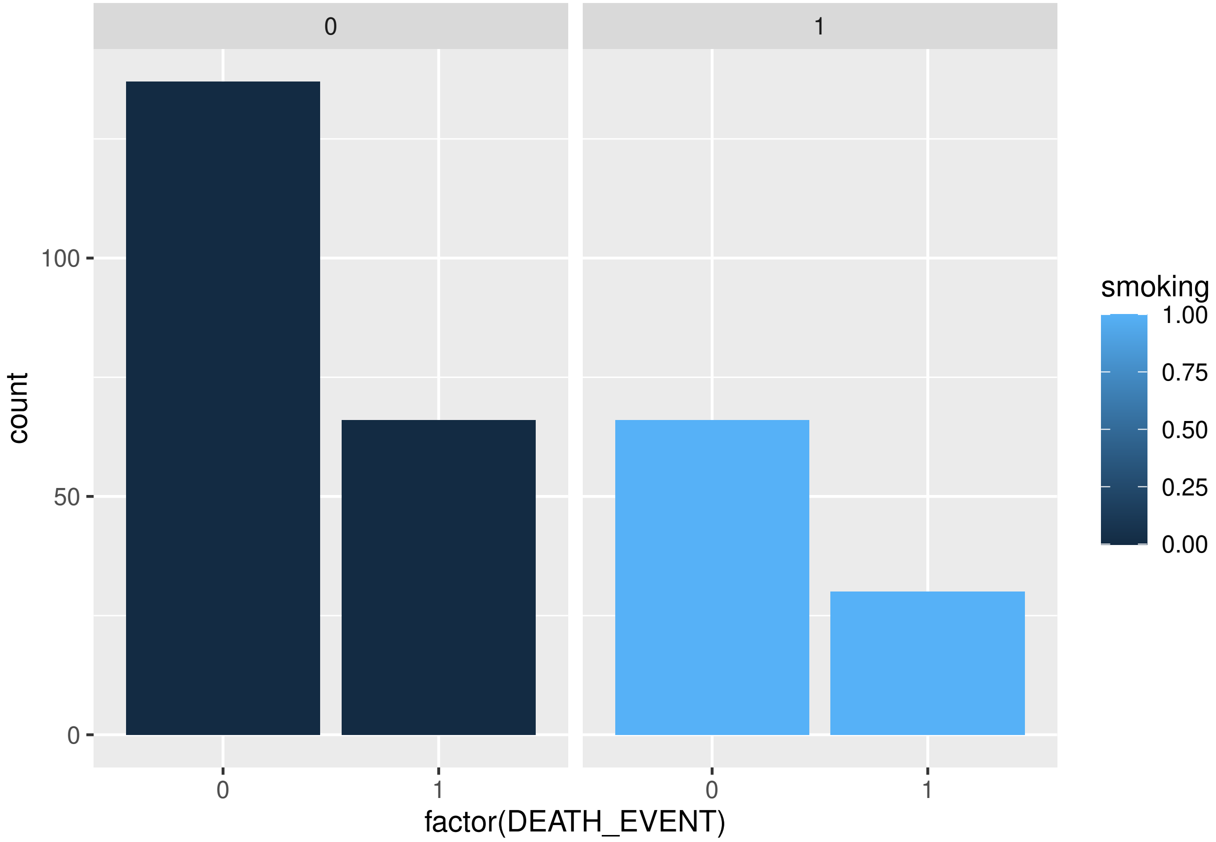
Time is the follow-up period as in days. If the follow-up period is between 0 and 50 it is likely the patient will die from heart failure.

5.3 Categorical Data Analysis

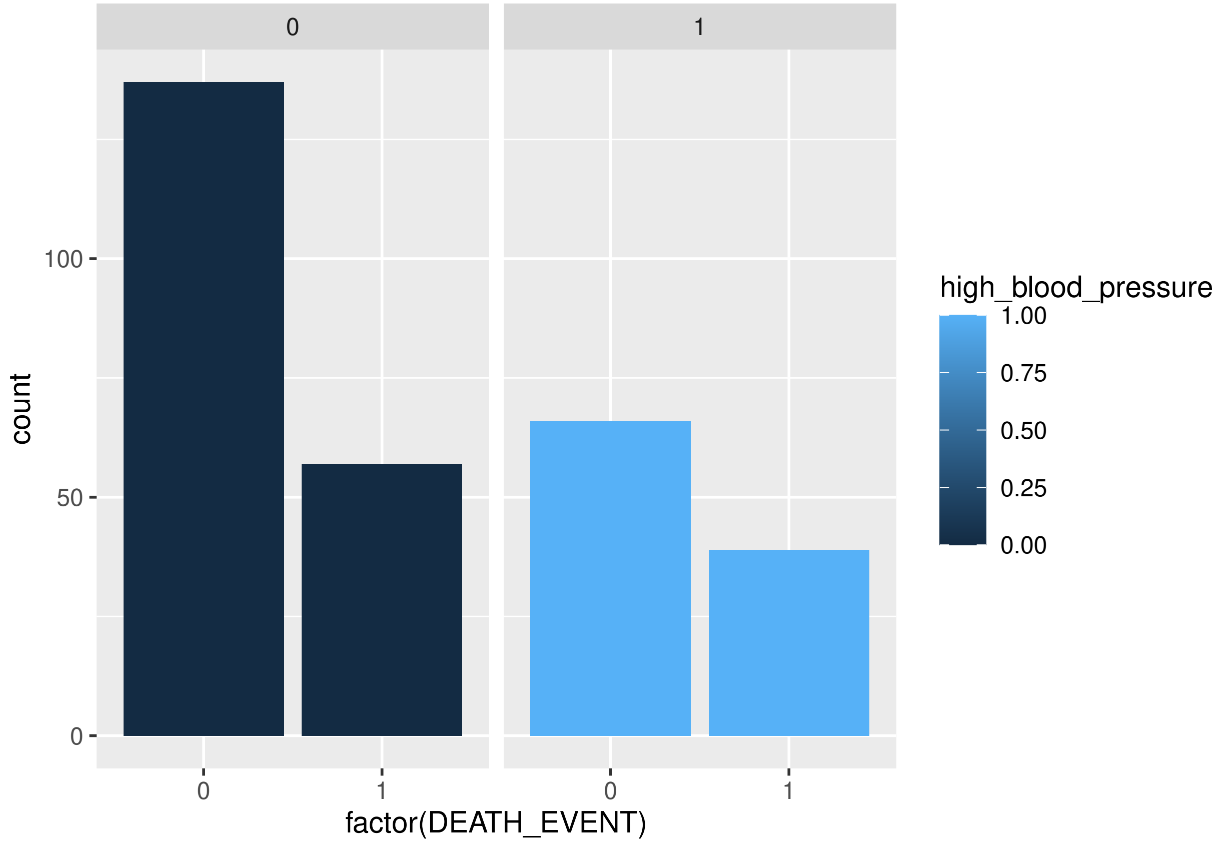
```
ggplot(HeartFailure, aes(x = factor(DEATH_EVENT), fill = sex)) +
  geom_bar(position = position_dodge(preserve = "single",))+
  facet_wrap(~sex)
```



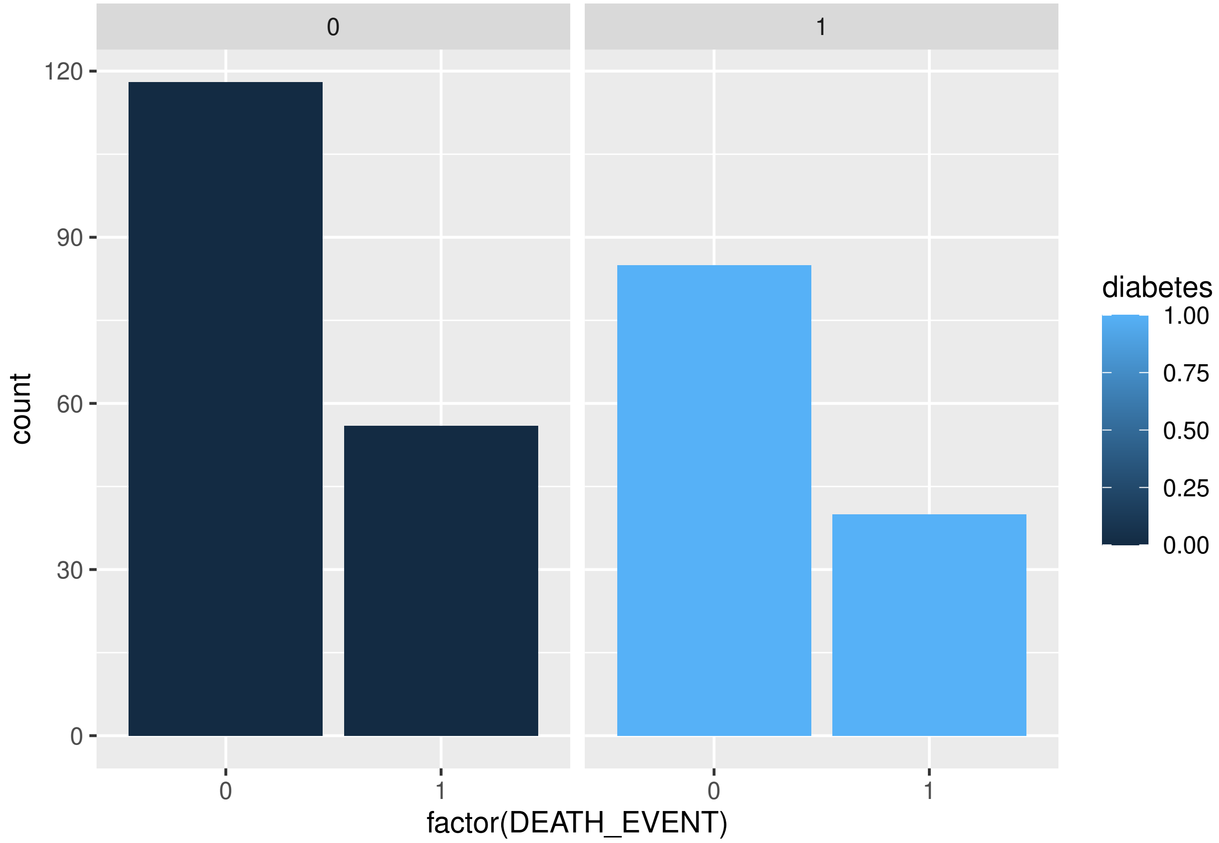
```
ggplot(HeartFailure, aes(x = factor(DEATH_EVENT), fill = smoking)) +
  geom_bar(position = position_dodge(preserve = "single",))+
  facet_wrap(~smoking)
```



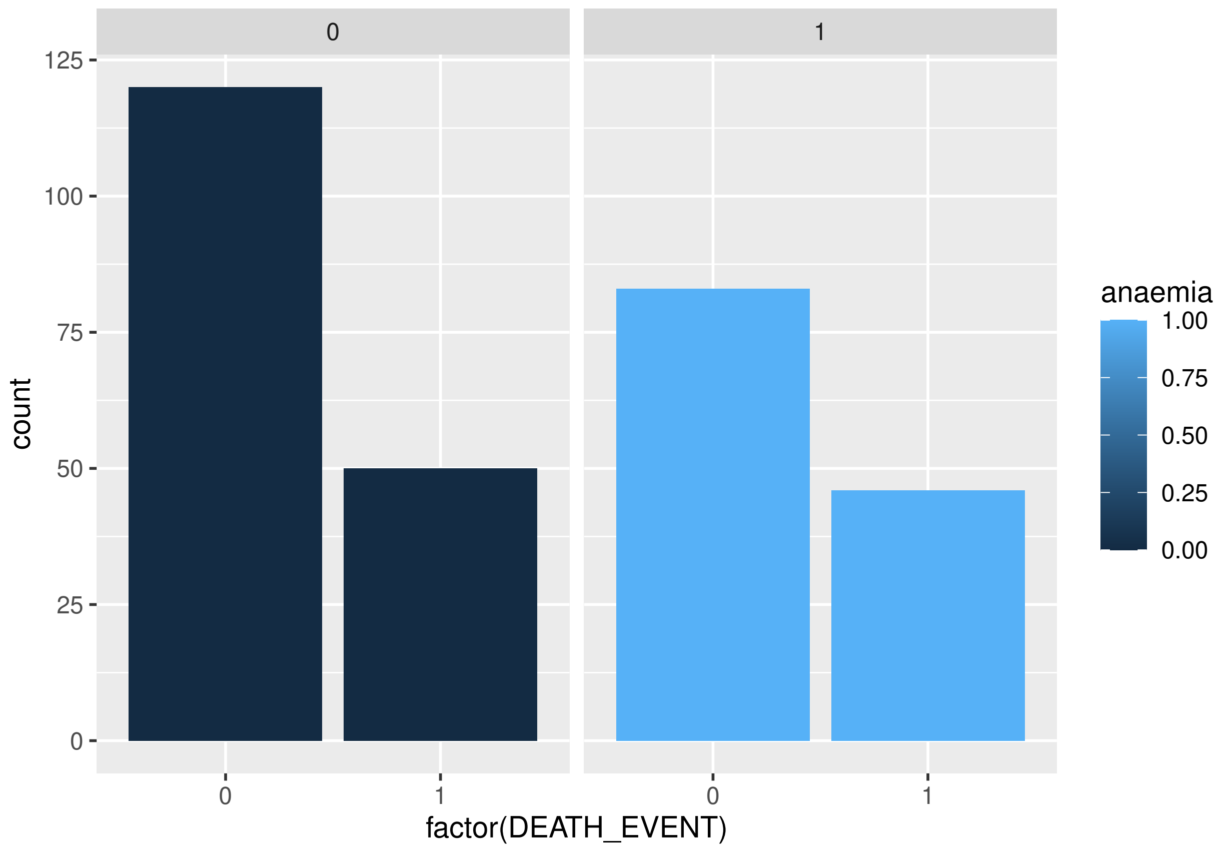
```
ggplot(HeartFailure, aes(x = factor(DEATH_EVENT), fill = high_blood_pressure)) +
  geom_bar(position = position_dodge(preserve = "single",))+
  facet_wrap(~high_blood_pressure)
```



```
ggplot(HeartFailure, aes(x = factor(DEATH_EVENT), fill = diabetes)) +
  geom_bar(position = position_dodge(preserve = "single",))+
  facet_wrap(~diabetes)
```



```
ggplot(HeartFailure, aes(x = factor(DEATH_EVENT), fill = anaemia)) +
  geom_bar(position = position_dodge(preserve = "single",))+
  facet_wrap(~anaemia)
```



The death event value divides categorical features in half. A score of 0 indicates no death event, while a score of 1 indicates that there is death event. Most of the estimations in the presented analysis come from the charts above.

5.3.1 SEX

✓

From the graph, we can say that heart disease is more common in male individuals. There are approximately 300 patients. Heart disease affects more than half of these people. Female patients are also more likely to be heart disease-free since their “0” scores are higher than their “1” ones. This is, clearly distinguishes between heart disease, and will be a suitable fit for the model training phase.

5.3.2 SMOKING

From the graph, approximately 100 people are smoking. In the smoking part, if we look at the gap between people who have heart disease and who do not, it is less than people who is not smoking. The ratio between those who have heart disease and who do not is bigger in people who are smoking. From that, we can say that people who are smoking are prone to get heart disease. This column will be a good fit for the model training phase since this column make a clear the distinction between heart disease.

5.3.3 HIGH BLOOD PRESSURE

As same as the smoking graph, if we look at the graph the number of people who have high blood pressure and heart disease is lower than the people who do not have high blood pressure. This is because around 105 people have a high blood pressure of 194. If you look at the ratio between them, people who have high blood pressure has bigger ratio, the gap is smaller. This is, clearly distinguishes between heart disease, and will be a suitable fit for the model training phase.

5.3.4 DIABETES

The number of people who has diabetes is 125, and who do not are 174. Of the people who have diabetes, approximately 40 people have heart disease, and 85 people has not. From the part that people have diabetes, approximately 40 people have heart disease, and 85 people have not. And from the part that people do not have diabetes, approximately 55 people has heart disease, and 120 people have not. The ratio of those who have heart disease to those who have no heart disease is bigger in people who have diabetes with a small difference. Therefore, people with diabetes are more likely to have heart disease. As a last word, this column is suitable for the model training phase.

5.3.5 ANAEMIA

For anaemia, it is similar to the diabetes graph. The gap between people who have anemia is smaller than between people who do not. Also, their ratio is bigger for people having heart disease or not. Therefore, this column is suitable for the model training phase.

6 Data Preprocessing

✓ To determine if there is a missing value in the data. Rows with NA values can simply be removed. But since this is about health and still the particular rows would still include valuable information, filling them using imputation would be more suitable.

```
# Missing value determination.
sum(is.na(HeartFailure))
```

```
## [1] 0
```

As seen above there is not any missing value. But there might be some technical mistake about missing data. There might be patients that has unknown history and have 0 values in columns as interpreted as NA. Since we had a doubt about it, check it to see are there any unusual 0 values.

```
sum(HeartFailure$creatinine_phosphokinase == 0)
```

```
## [1] 0
```

```
sum(HeartFailure$ejection_fraction == 0)
```

```
## [1] 0
```

```
sum(HeartFailure$platelets == 0)
```

```
## [1] 0
```

```
sum(HeartFailure$serum_creatinine == 0)
```

```
## [1] 0
```

```
sum(HeartFailure$serum_sodium == 0)
```

```
## [1] 0
```

```
sum(HeartFailure$time == 0)
```

```
## [1] 0
```

All numerical values have not have any 0 values. Therefore, there seems nothing unusual in values. There cannot be missing values.

In the data, all 12 clinical features with help determine death events will be used. Because there is no unrelated value such as patient number in the data.

7 Data Partioning

We split the data into 80% training and 20% testing.

```
# Data Partitioning as %80.
Index = sample(nrow(HeartFailure), 0.8*nrow(HeartFailure))
# Training data.
Train = HeartFailure[Index, ]
# Testing data.
Test = HeartFailure[-Index, ]
```

8 Model Fit Comparison

8.1 Logisitic Regression

Couple logistic regression models fitting. This is the summary:

```
# Logisitic Regression
regmodel <- glm(data = Train, formula = DEATH_EVENT ~ ., family = "binomial")
summary(regmodel)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ ., family = "binomial", data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1847  -0.5803  -0.2405   0.4416   2.7428
##
## Coefficients:
```

```
##                           Estimate Std. Error z value Pr(>|z|)    
## (Intercept)              7.266e+00  6.058e+00   1.199  0.23038    
## age                      4.785e-02  1.749e-02   2.737  0.00621 ** 
## anaemia                  3.519e-01  3.964e-01   0.888  0.37471    
## creatinine_phosphokinase 3.392e-04  2.132e-04   1.591  0.11152    
## diabetes                 2.572e-01  3.852e-01   0.668  0.50438    
## ejection_fraction       -8.090e-02  1.865e-02  -4.337 1.44e-05 ***
## high_blood_pressure     -5.979e-02  3.996e-01  -0.150  0.88105    
## platelets               -1.365e-06  1.987e-06  -0.687  0.49205    
## serum_creatinine         6.489e-01  1.888e-01   3.436  0.00059 ***
## serum_sodium            -4.657e-02  4.260e-02  -1.093  0.27430    
## sex                     -6.651e-01  4.547e-01  -1.463  0.14354    
## smoking                  3.213e-01  4.629e-01   0.694  0.48757    
## time                    -2.028e-02  3.252e-03  -6.236 4.50e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 303.32  on 238  degrees of freedom
## Residual deviance: 176.97  on 226  degrees of freedom
## AIC: 202.97
## 
## Number of Fisher Scoring iterations: 6
```

```
modelPre = ifelse(predict(regmodel, newdata = Test, type = "response") > 0.5, 1, 0)
```

Difference between predicted value (Pr) and an observed value is known as a residual. The median of the deviance residuals is -0.2405. The first quantile range is -0.5803. The third quantile range is 0.4416.
The maximum value is 2.7428. The minimum value is -2.1847.

? Lower p-values suggest that the variable in the model is meaningful. The Pr(<|z|) provides the p-value linked with the z-value. The predictors that more likely to chance for changing the null hypothesis' conclusion are, age, ejection_fraction, serum_creatinine, time. We are taking all to train since they have stars. Hypothesis might be, the patient has heart disease. ✓

✓

```
modelchosen <- glm(data = Train, formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + ti
modelchosenPred = ifelse(predict(modelchosen, newdata = Test, type = "response") > 0.5, 1, 0)
```

By cross validation, error rate has calculated below. We did not use separate validation since it can host high variability. When divided, it can reduces training samples' number.

The leave-one-out cross validation, LOOCV is a special case of k-fold CV in which k = n. We choose k = 5 and k = 10 because empirical evidence suggests that these values produce test error rate estimates that are do not cause overly high in bias and not cause overly high variance. The data is divided by LOOCV into m-1 samples for training and 1 sample for validation. The data is divided into 5 equal chunks for 5-fold cross validation, and one of the chunks is validated for each fold in 10-fold cross validation as well.

✓ The second component provides the bias-corrected cross-validation error, whereas the delta array provides the raw cross-validation prediction error. By using the provided formula, the errors for cross validation are determined. Formula:

MSE $= \frac{1}{m}\Sigma_{i=1}^{m}(y-\hat{y})^2$

m : the number of data points

y : the actual values

$\hat{y}$: the predicted values

```r
library(boot)
# without any additional model-fitting LOOCV estimates

cv = cv.glm(Train, regmodel)
cv$delta
```

```
## [1] 0.1394682 0.1394236
```

```r
cv5 = cv.glm(Train, regmodel, K = 5) # Corresponds 5-fold CV
cv5$delta
```

```
## [1] 0.1331577 0.1316076
```

```r
cv10 = cv.glm(Train, regmodel, K = 10) # Corresponds 10-fold CV
cv10$delta
```

```
## [1] 0.1424933 0.1412517
```

✓

```r
sprintf("Raw error of LOOCV: %.7f", cv$delta[2])
```

```
## [1] "Raw error of LOOCV: 0.1394236"
```

```r
sprintf("Raw error of LOOCV 5-Fold: %.7f", cv5$delta[1])
```

```
## [1] "Raw error of LOOCV 5-Fold: 0.1331577"
```

```r
sprintf("Raw error of LOOCV 10-Fold: %.7f", cv10$delta[1])
```

```
## [1] "Raw error of LOOCV 10-Fold: 0.1424933"
```

```r
sprintf("bias corrected error of LOOCV: %.7f", cv$delta[2])
```

```
## [1] "bias corrected error of LOOCV: 0.1394236"
```

```r
sprintf("bias corrected error of 5-Fold: %.7f", cv5$delta[2])
```

```
## [1] "bias corrected error of 5-Fold: 0.1316076"
```

```r
sprintf("bias corrected error of 10-Fold: %.7f", cv10$delta[2])
```

```
## [1] "bias corrected error of 10-Fold: 0.1412517"
```

The smallest raw error is 0.1331577 with 5-fold. The highest one is 0.1424933 with 10-fold. Once the bias values have been applied to the raw error, the smallest one is, again 5-fold with 0.1316076. And again highest one is 10-fold with 0.1412517.

LOOCV first appears to be the ideal cross validation method because of its low bias corrected error and low raw error. However, it runs the danger of overfitting the data, which can lead to misleading error rates.

Moreover, the error difference between the 5-fold CV and LOOCV can be disregarded because there is a small difference.

Furthermore, compared to 5-fold CV, 10-fold cross validation has the highest error rate and requires more processing power. 5-fold cv, wants the most computing power, which would be a significant disadvantage if the data were substantially larger.

With the aforementioned justifications, LOOCV is the preffered one.

8.2 Logisitic Regression Result

True positive (TP) which is predicted as positive value and it is true. True negative (TN) which is predicted as negative value and it is true. False negative (FN) which is type 2 error is predicted as negative value and it is false. False positive (FP) which is type 1 error is predicted as positive value and it is false.

Accuracy: (TP + TN) / TP + TN + FP + FN

Predictive analytics’ concept of precision describes how well the model’s predictions match the actual data. The data points closely match the predictions the more accurate the model is. It is in predicted positive classes.

Recall is looking for false negatives that were thrown into prediction mix. It is in all positive classes.

If a patient has heart disease, it is generally understood in real life that this is a bad scenario. Having no heart disease is a good thing for the patient. Suppose that the 0 0 index is TP while looking at the confusion matrix.

```r
# Confusion matrix and statistics.
TestDEATH_EVENTLogisitic = factor(Test$DEATH_EVENT, levels = c(0, 1))
ModelPreFactor = factor(modelPre, levels = c(0, 1))
confusionMatrix(data = ModelPreFactor, reference = TestDEATH_EVENTLogisitic)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 38  5
##          1  5 12
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.02687
##
##                   Kappa : 0.5896
##
##  Mcnemar’s Test P-Value : 1.00000
##
##             Sensitivity : 0.8837
##             Specificity : 0.7059
##          Pos Pred Value : 0.8837
##          Neg Pred Value : 0.7059
##              Prevalence : 0.7167
##          Detection Rate : 0.6333
##    Detection Prevalence : 0.7167
##       Balanced Accuracy : 0.7948
##
##        'Positive' Class : 0
##
```

We can observe that, in the confusion matrix of logistic regression matrix, 83.33% is the percentage of logistic regression's accuracy. The sum of the correct predictions which is True Positives and True Negatives is 50 (38 + 12 = 50). Total cases are 38 + 12 + 5 + 5 = 60. 50 / 60 is approximately 0.83333.

Recall from the formula: TP / (TP + FN)

38 / (38 + 5) is approximately 0.88372.

Precision from the formula: TP / Predictive Results or TP / (TP + FP)

38 / (38 + 5) is approximately 0.88372.

```
modelchosenFac = factor(modelchosenPred, levels=c(0, 1))
confusionMatrix(data= modelchosenFac, reference= TestDEATH_EVENTLogisitic)
```

✓

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 39  6
##          1  4 11
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.02687
##
##                   Kappa : 0.5745
##
##  Mcnemar's Test P-Value : 0.75183
##
##             Sensitivity : 0.9070
##             Specificity : 0.6471
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 0.7333
##              Prevalence : 0.7167
##          Detection Rate : 0.6500
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.7770
##
##        'Positive' Class : 0
##
```

Recall from the formula: TP / (TP + FN)

39 / (39 + 4) is approximately 0.9070.

Precision from the formula: TP / Predictive Results or TP / (TP + FP)

39 / (39 + 6) is approximately 0.8667.

The confusion matrix shown above is from a model that was trained on certain columns and while its recall value is higher than that of a standard logistic regression model, its precision value is lower. The F score should be examined in order to decide which one is appropriate for the heart disease prediction instance.

```
sprintf("Specific Logistic model F score: %.5f", F_meas(data=modelchosenFac, reference=TestDEATH_EVENTL
```

```
## [1] "Specific Logistic model F score: 0.88636"
```

```
sprintf("Logistic model F score: %.5f", F_meas(data=modelchosenFac, reference=TestDEATH_EVENTLogisitic)
```

✓

```
## [1] "Logistic model F score: 0.88636"
```

From the comparison, specific logistic regressions and normal logistic regressions F score is equal. Therefore, do not have to involve to comparison part the Specific logistic regression model.

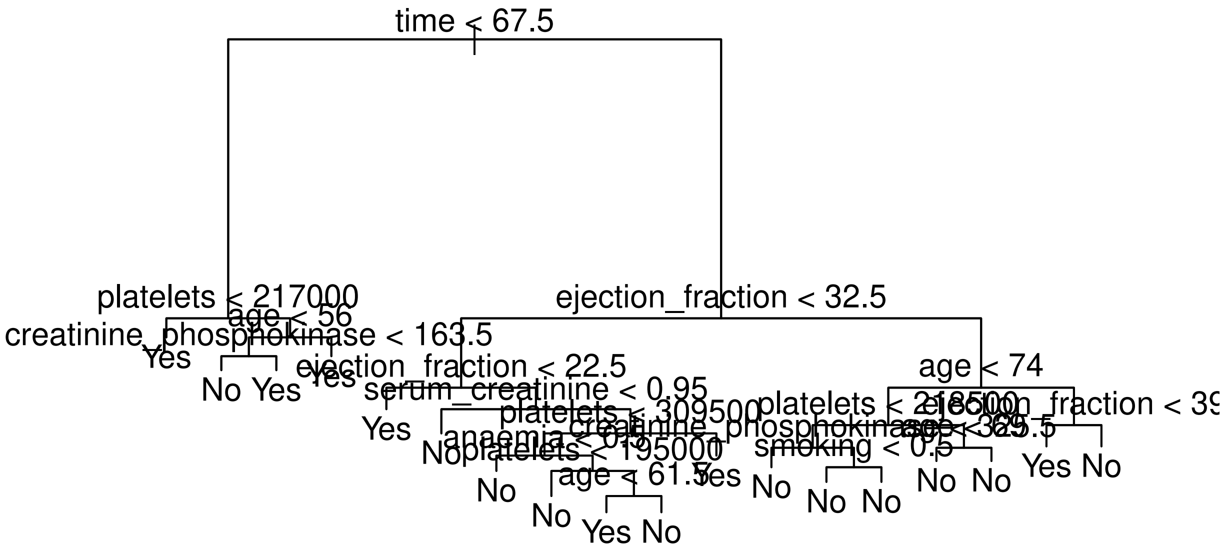
8.3 Decision Tree

Before applying the decision tree model in order to make a classification tree model, the response should be converted from '1' and '0' to 'Yes' and 'No'. The conversion is made by using the factor method.

20.1

✓

```
# Factor method.
HeartFailure$DEATH_EVENT <- factor(ifelse(HeartFailure$DEATH_EVENT < 0.5, "No", "Yes"))
TestDEATH_EVENT <- HeartFailure$DEATH_EVENT[-Index]
# Decision Tree.
DecisionTreeModel1 <- tree(DEATH_EVENT~., HeartFailure, subset = Index)
# Plot.
plot(DecisionTreeModel1)
text(DecisionTreeModel1, pretty = 0)
```



```
summary(DecisionTreeModel1)
```

```
##
## Classification tree:
## tree(formula = DEATH_EVENT ~ ., data = HeartFailure, subset = Index)
## Variables actually used in tree construction:
## [1] "time"                     "platelets"
## [3] "age"                      "creatinine_phosphokinase"
## [5] "ejection_fraction"        "serum_creatinine"
## [7] "anaemia"                  "smoking"
## Number of terminal nodes:  18
## Residual mean deviance:  0.3707 = 81.92 / 221
## Misclassification error rate: 0.07113 = 17 / 239
```

The summary method is telling variables used in tree construction and the number of nodes. Variables are randomly selected.

8.4 Decision Tree Result

The result of the above decision tree model.

```
# Confusion matrix and statistics.
TreePred <- predict(DecisionTreeModel1, Test, type = "class")
confusionMatrix(data = TreePred, reference = TestDEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  33   3
##        Yes 10  14
##
##                Accuracy : 0.7833
##                  95% CI : (0.658, 0.8793)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.15784
##
##                   Kappa : 0.5255
##
##  Mcnemar's Test P-Value : 0.09609
##
##             Sensitivity : 0.7674
##             Specificity : 0.8235
##          Pos Pred Value : 0.9167
##          Neg Pred Value : 0.5833
##              Prevalence : 0.7167
##          Detection Rate : 0.5500
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.7955
##
##        'Positive' Class : No
##
```

True observations is 33 + 14 = 47. False observations is 3 + 10 = 13. The models accuracy level is 78.33%. Recall of the model is calculated as 33/(33+10) = 0.7674. Precision of the model is calculated as 33/(33+3) = 0.9167.

8.5 Pruning Decision Tree

Pruning will reduce the missclassification error rate of the model. For further decisions like pruning level, plotting will made according to k (alpha), deviance, and size.

```
# Pruning level.
cv.HeartFailure <- cv.tree(DecisionTreeModel1, FUN = prune.misclass)
names(cv.HeartFailure)
```

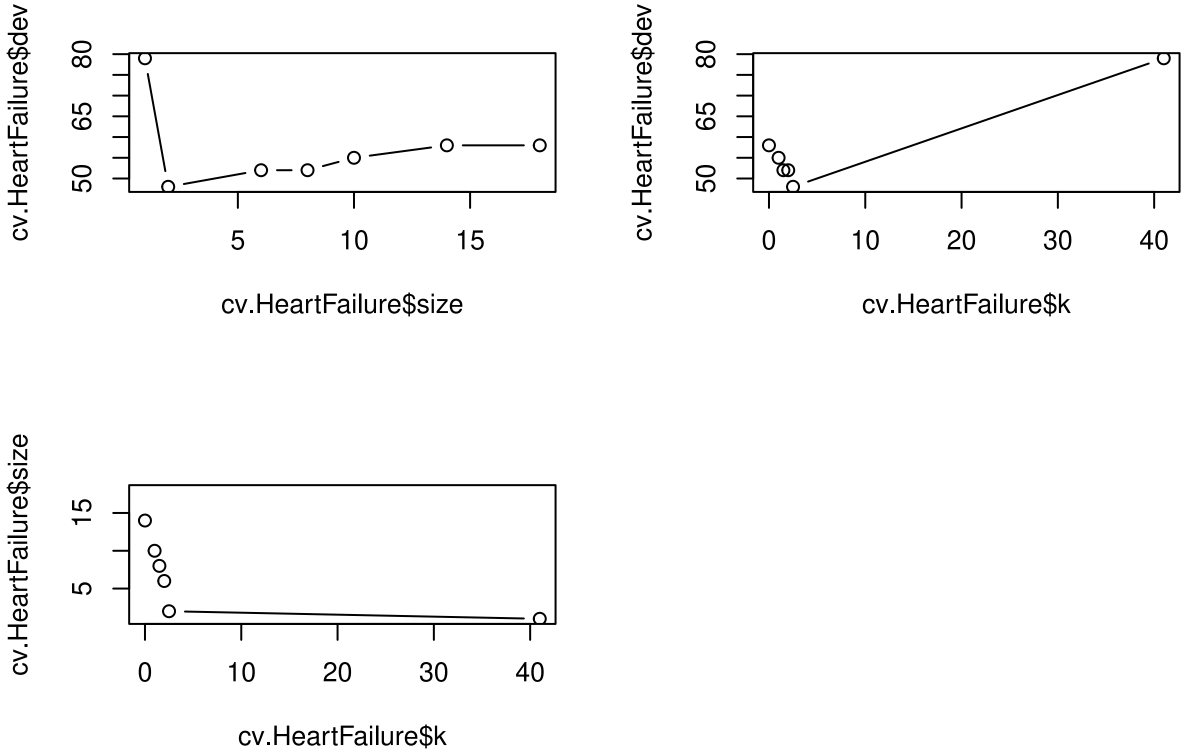
```
## [1] "size"   "dev"    "k"      "method"
```

```
cv.HeartFailure
```

```
## $size
## [1] 18 14 10  8  6  2  1
##
## $dev
## [1] 58 58 55 52 52 48 79
##
## $k
## [1] -Inf  0.0  1.0  1.5  2.0  2.5 41.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
# Plot.
par(mfrow = c(2,2))
plot(cv.HeartFailure$size, cv.HeartFailure$dev, type = "b")
plot(cv.HeartFailure$k, cv.HeartFailure$dev, type = "b")
plot(cv.HeartFailure$k, cv.HeartFailure$size, type = "b")
```

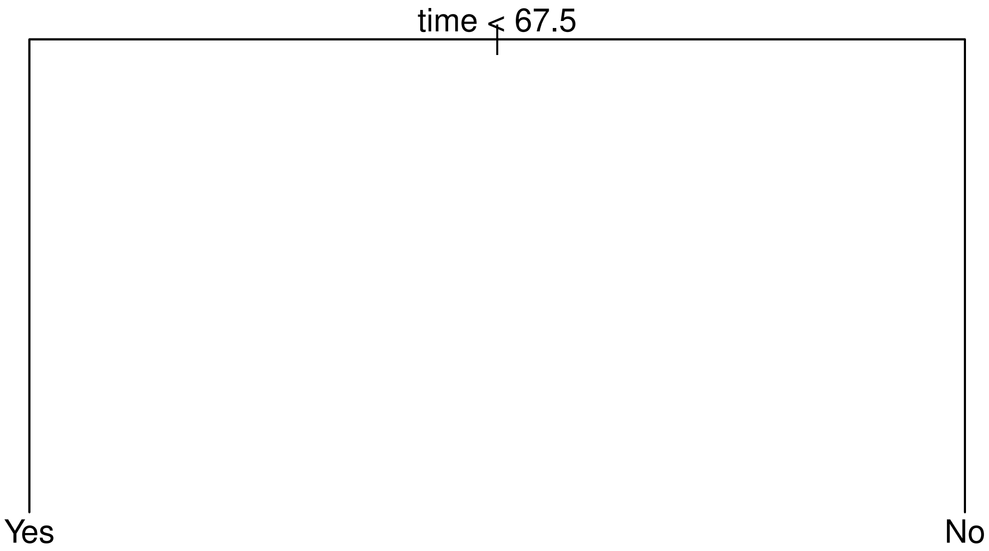
✓



We see that the smallest deviance is starting from 2.

```
# Pruning.
Prune.DecisionTreeModel1 <- prune.tree(DecisionTreeModel1, best = 2)
# Pruned tree model.
plot(Prune.DecisionTreeModel1)
text(Prune.DecisionTreeModel1, pretty = 0)
```

✓



```
summary(Prune.DecisionTreeModel1)
```

```
##
## Classification tree:
## snip.tree(tree = DecisionTreeModel1, nodes = 2:3)
## Variables actually used in tree construction:
## [1] "time"
## Number of terminal nodes:  2
## Residual mean deviance:  0.881 = 208.8 / 237
## Misclassification error rate: 0.159 = 38 / 239
```

Terminal nodes reduced to 2.

8.6 Pruning Decision Tree Result

After pruning the tree model prediction and confusion matrix of the model are made. Pruning can sometimes increase the accuracy of the model.

```
# Confusion matrix and statistics.
TreePredictPrune <- predict(Prune.DecisionTreeModel1, Test, type = "class")
confusionMatrix(data = TreePredictPrune, reference = TestDEATH_EVENT)
```

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

✓

```
##           Reference
## Prediction No Yes
##        No  40   5
##        Yes  3  12
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
##                   Kappa : 0.6596
##
##  Mcnemar's Test P-Value : 0.723674
##
##             Sensitivity : 0.9302
##             Specificity : 0.7059
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.8000
##              Prevalence : 0.7167
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : No
##
```

✓

True observations is 40 + 12 = 52. False observations is 5 + 3 = 8. The models accuracy level is 86.67%. Recall of the model is calculated as 40/(40+3) = 0.9302. Precision of the model is calculated as 40/(40+5) = 0.8889. As seen accuracy of the pruned model is higher.

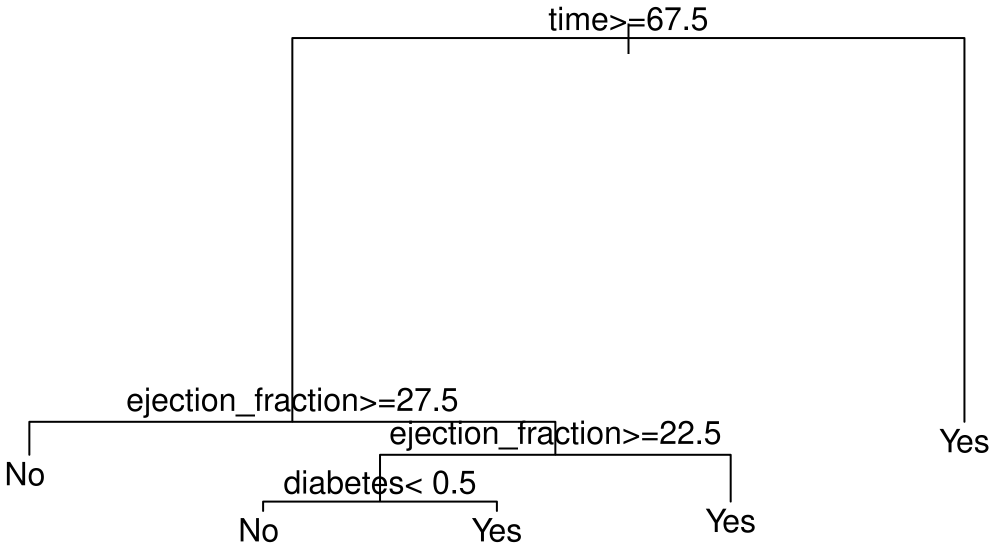
8.7 Rpart Decision Tree

In this section decision tree will made using Rpart library. Like the first part factor operation will be made for classification tree.

```
Test$DEATH_EVENT <- factor(ifelse(Test$DEATH_EVENT < 0.5, "No", "Yes"))
Train$DEATH_EVENT <- factor(ifelse(Train$DEATH_EVENT < 0.5, "No", "Yes"))
# Rpart
DecisionTreeModel2 <- rpart(DEATH_EVENT~., data = Train, method = "class")
DecisionTreeModel2
```

```
## n= 239
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 239 79 No (0.6694561 0.3305439)
##    2) time>=67.5 184 31 No (0.8315217 0.1684783)
##      4) ejection_fraction>=27.5 153 16 No (0.8954248 0.1045752) *
##      5) ejection_fraction< 27.5 31 15 No (0.5161290 0.4838710)
##       10) ejection_fraction>=22.5 24  9 No (0.6250000 0.3750000)
##         20) diabetes< 0.5 13  3 No (0.7692308 0.2307692) *
##         21) diabetes>=0.5 11  5 Yes (0.4545455 0.5454545) *
##       11) ejection_fraction< 22.5 7  1 Yes (0.1428571 0.8571429) *
##    3) time< 67.5 55  7 Yes (0.1272727 0.8727273) *
```

```
# Plot the Rpart
par(xpd = NA) # Avoid clipping the text in some device
plot(DecisionTreeModel2)
text(DecisionTreeModel2, digits = 3)
```



✓

```
# Importance
DecisionTreeModel2$variable.importance
```

```
##                     time        ejection_fraction                      age
##               42.7214104               12.2274598                2.9727705
## creatinine_phosphokinase                 diabetes         serum_creatinine
##                1.7418865                1.1800699                1.0855014
##                platelets             serum_sodium                  smoking
##                0.9344397                0.7636642                0.2145582
```

Terminal nodes reduced to five.

8.8 Rpart Decision Tree Result

```
# Prediction.
Predicted <- DecisionTreeModel2 %>%
  predict(Test, type = "class")
head(Predicted)
```

```
##   1   3  10  12  17  21
## Yes Yes Yes Yes Yes Yes
## Levels: No Yes
```

```
# Confusion matrix.
confusionMatrix(data = Predicted, reference = TestDEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  37   4
##        Yes  6  13
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.02687
##
##                   Kappa : 0.6037
##
##  Mcnemar's Test P-Value : 0.75183
##
##             Sensitivity : 0.8605
##             Specificity : 0.7647
##          Pos Pred Value : 0.9024
##          Neg Pred Value : 0.6842
##              Prevalence : 0.7167
##          Detection Rate : 0.6167
##    Detection Prevalence : 0.6833
##       Balanced Accuracy : 0.8126
##
##        'Positive' Class : No
##
```

True observations is 37 + 13 = 50. False observations is 4 + 6 = 10. The models accuracy level is 83.33%. Recall of the model is calculated as 37/(37+6) = 0.8605. Precision of the model is calculated as 37/(37+4) = 0.9024. As seen the accuracy of the Rpart library model is lower than pruned model.

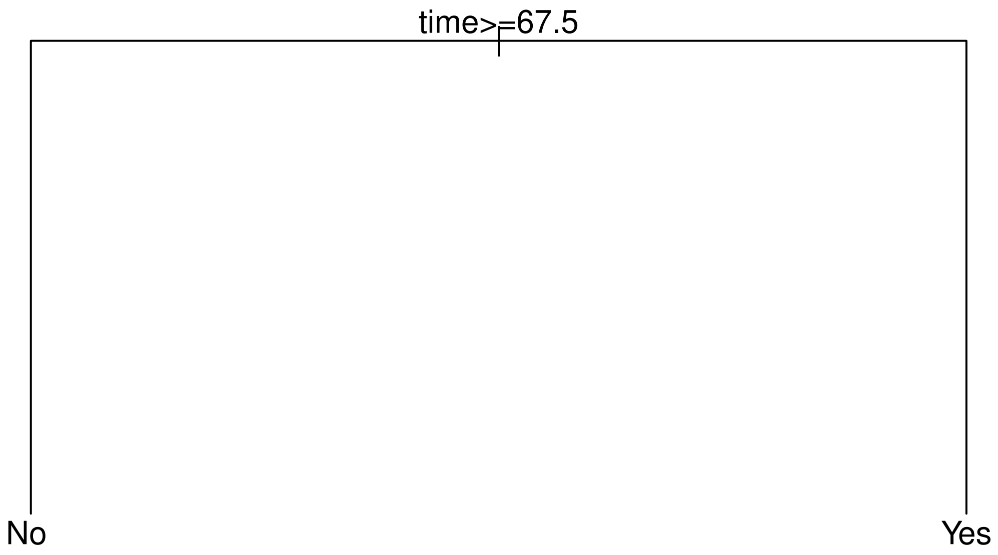
8.9 Rpart Decision Tree With Cross Validation

In this section the aim is to do 5-fold cross validation in Rpart library.

```
# 5 fold cross validation
DecisionTreeModel3 <- train(DEATH_EVENT ~., data = Train, method = "rpart", trControl = trainControl("c
```

8.10 Rpart Decision Tree With Cross Validation Result

```
# Plot.
par(xpd = NA)
plot(DecisionTreeModel3$finalModel)
text(DecisionTreeModel3$finalModel, digits = 3)
```



```
# Prediction.
Predicted3 <- DecisionTreeModel3 %>% predict(Test)
# Confusion matrix.
confusionMatrix(data = Predicted3, reference = TestDEATH_EVENT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  40   5
##        Yes  3  12
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7167
##     P-Value [Acc > NIR] : 0.004937
##
```

```
##                   Kappa : 0.6596
##
##  Mcnemar's Test P-Value : 0.723674
##
##             Sensitivity : 0.9302
##             Specificity : 0.7059
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.8000
##              Prevalence : 0.7167
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7500
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : No
##
```

True observations is 40 + 12 = 52. False observations is 5 + 3 = 8. The models accuracy level is 86.67%. Recall of the model is calculated as 40/(40+3) = 0.9302. Precision of the model is calculated as 40/(40+5) = 0.8889. As seen, the model has gave the same results as pruned decision tree.

✓

9 Model Performance Comparison

For checking a model's performance considering only accuracy will not give a good outcome. Thus for performance comparison recall, precision and F1 score values must be considered.

- Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$
- Recall (Sensitivity) = $\frac{TP}{TP+FN}$
- Precision = $\frac{TP}{TP+FP}$
- F1 score = $2*(\frac{Precision*Recall}{Precision+Recall})$

Values for Unpruned Decision Tree Model:

```
## [1] "Recall: 0.7674"
```

```
## [1] "Precision: 0.9167"
```

```
## [1] "F1 score: 0.8354"
```

Values for Pruned Decision Tree Model:

```
## [1] "Recall: 0.9302"
```

```
## [1] "Precision: 0.8889"
```

```
## [1] "F1 score: 0.9091"
```

Values for Rpart Decision Tree Model:

```
## [1] "Recall: 0.8605"
## [1] "Precision: 0.9024"
```

```
## [1] "F1 score: 0.8810"
```

Values for Rpart Decision Tree With Cross Validation Model:

```
## [1] "Recall: 0.9302"
```

```
## [1] "Precision: 0.8889"
```

```
## [1] "F1 score: 0.9091"
```

Values for Logisitic Regression Model:

```
## [1] "Recall: 0.8837"
```

```
## [1] "Precision: 0.8837"
```

```
## [1] "F1 score: 0.8837"
```

✓ Where the case is intolerable to a false negative, the recall value will be significant. The precision value considerably is increasing when the false positive is getting significant. The primary objective of the F score is to lessen the impact of variation in recall and precision values. For instance, the F score will be low if the Recall value is high but the Precision value is low. The other way around, the F score is going to be low again. Only when both Recall and Precision values are high would the F score be high. For this reason, the F score is more useful in deciding the better model when compared with the accuracy.

About the precision values, Unpruned Decision Tree Model has the best precision value with 0.9167. Next Rpart Decision Tree Model coming next with 0.9024. Rpart Decision Tree With Cross Validation Model and Pruned Decision Tree Model 0.8889. The last one is Logistic Regression Model with 0.8837

Comparing the above models, Rpart Decision Tree With Cross Validation Model and Pruned Decision Tree Model has the best recall value with 0.9302. It is followed by Logistic Regression Model with 0.8837. After that, Rpart Decision Tree Model coming next with 0.8605. The last one is Unpruned Decision Tree Model with 0.7674.

Comparing the F score values, again Rpart Decision Tree With Cross Validation Model and Pruned Decision Tree Model has the best value with 0.9091. And after that Logistic Regression Model with 0.8837. After that, Unpruned Decision Tree Model coming next with 0.8357. The last one is Rpart Decision Tree Model with 0.8810.

Furthermore, best accuracy value is 0.8667 with Rpart Decision Tree With Cross Validation Model and Pruned Decision Tree Model. The next ones are Logisitic Regression Result Accuracy and Rpart Decision Tree Result Accuracy with 0.8333. The last one is Decision Tree Accuracy with 0.7833.

Various parameters can be used to choose the best model. The accuracy can be the primary consideration when choosing a model if the relative percentages of correct classifications were significant. But, accuracy does not provide data on false positives and false negatives. The models with high False negative counts will not be suitable for heart disease prediction since predicting sick patients as healthy is a risky outcome that is unacceptable in this situation.

Recall values can be used to estimate high FN counts. Therefore, the two models with the lowest recall values are eliminate. Unpruned Decision Tree Model and Rpart Decision Tree Model has eliminated. The highest one is Rpart Decision Tree With Cross Validation Model.

For comparing the models with F scores, the less ones are Unpruned Decision Tree Model and Rpart Decision Tree Model. The highest one is Rpart Decision Tree With Cross Validation Model.

It is clear from the above observations that models with high recall and high F scores are better suited for cases of heart disease. Therefore, the best and most suitable model for heart disease is Rpart Decision Tree With Cross Validation Model and Pruned Decision Tree Model since the results are exactly same with the highest values.

✓

10 Conclusions

In this report the aim was by using Logistic Regression and Decision Tree models, classify the heart failure as accurate as possible. We predict heart failure as we do observations on the dataset, we classify the data to determine which group each prediction about heart disease belongs to using a training set. Finally, we decided to choose Rpart Decision Tree With Cross Validation Model as the best model since it has the best values.

✓

11 References

1) Aman Chauhan (2023, January 2), [Online]. Available: https://www.kaggle.com/datasets/whenamancodes/heart-failure-clinical-records

2) yashchuahan (2023, January 3), [Online]. Available: https://www.geeksforgeeks.org/correlation-matrix-in-r-programming/

3) Subha Ganapathi (2023, January 3), [Online]. Available: https://medium.com/nerd-for-tech/implementing-decision-trees-in-r-regression-problem-using-rpart-c74cbd9e0b7b

Index of comments

20.1 Is it really necessary ?