Statistical Learning Project

Chiara Bigarella Student nr. 2004248
Silvia Poletti Student nr. 1239133
Johanna Weiss Student nr. 2013014

1 Dataset

We considered the Heart failure clinical records, stored at https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records. Data was collected at the Allied Hospital in Faisalabad (Punjab, Pakistan), form April to December 2015, and then elaborated at Krembil Research Institute in Toronto (Canada).
The dataset includes the medical records of 299 patients who had left ventricular systolic dysfunction and had previous heart failures. Data was collected during the follow-up period and each patient profile has 13 clinical features, that are listed in Table 1.

Feature	**Description**	**Type**
Age	Age of the patient in years	Integers (from 40 to 95)
Anaemia	If the patient has a decrease of red blood cells or hemoglobin	Booleans 0-1
High Blood Pressure	If the patient has hypertension	Booleans 0-1
Creatinine Phosphokinase	Level of the CPK enzyme in the blood in mcg/L	Reals (from 23 to 7861)
Diabetes	If the patient has diabetes	Booleans 0-1
Ejection Fraction	Percentage of blood pumped out with each heart contraction	Reals (from 14 to 80)
Platelets	Platelets in the blood in platelets/mL	Reals (from 25100 to 850000)
Sex	If the patient is a male or female	Booleans 0-1
Serum Creatinine	Level of serum creatinine in the blood in mg/dL	Reals (from 0.50 to 9.40)
Serum Sodium	Level of serum sodium in the blood in mEq/L	Reals (from 113 to 148)
Smoking	If the patient smokes	Booleans 0-1
Time	Follow-up period in days	Integers (from 4 to 285)
Death Event	If the patient deceased during the follow-up period	Booleans 0-1

Table 1: Features format.

The dataset has been imported as follows:

```
Heart <- read.csv("heart_failure_clinical_records_dataset.csv", sep=",")
Heart <- na.omit(Heart) # no missing values

dim(Heart)

## [1] 299  13

n <- dim(Heart)[1]      # sample size
d <- dim(Heart)[2] - 1  # number of features
```

```
colnames(Heart)
```

```
##  [1] "age"                       "anaemia"                   "creatinine_phosphokinase"
##  [4] "diabetes"                  "ejection_fraction"         "high_blood_pressure"
##  [7] "platelets"                 "serum_creatinine"          "serum_sodium"
## [10] "sex"                       "smoking"                   "time"
## [13] "DEATH_EVENT"
```

Death Event is the target in our binary classification study and resulted to be unbalanced:

```
sum(Heart$DEATH_EVENT==1)/length(Heart$DEATH_EVENT)  # proportion of survived patients
```

```
## [1] 0.3210702
```

```
sum(Heart$DEATH_EVENT==0)/length(Heart$DEATH_EVENT)  # proportion of dead patients
```

```
## [1] 0.6789298
```

Age and Time are discrete quantitative variables taking values in $\mathbb{N}^+$.
Creatinine Phosphokinase, Ejection Fraction, Platelets, Serum Creatinine, Serum Sodium and Time are continuous quantitative variables taking values in $\mathbb{R}^+$.
Anemia, Diabetes, High Blood Pressure, Sex and Smoking are nominal categorical binary variables.
Before applying regression on the data, the latter features need to be pre-processed as follows:

```
Heart$anaemia <- as.factor(Heart$anaemia)
Heart$diabetes <- as.factor(Heart$diabetes)
Heart$high_blood_pressure <- as.factor(Heart$high_blood_pressure)
Heart$sex <- as.factor(Heart$sex)
Heart$smoking <- as.factor(Heart$smoking)
```

Pre-processing also involves the split into train set (75% of the dataset) and test set (25% of the dataset) with a similar proportion of dead patients in the two sets. In fact, from the statistical point of view, classification can be carried out by using logistic regression to estimate the probability of the response variable (target) based on training data. Then the accuracy of the model is evaluated on the test set.

```
set.seed(1)
array <- sample(c(TRUE,FALSE),n,replace=TRUE,prob=c(0.75,0.25))

train <- Heart[array,]
dim(train)[1]
```

```
## [1] 232
```

```
test <- Heart[!array,]
dim(test)[1]
```

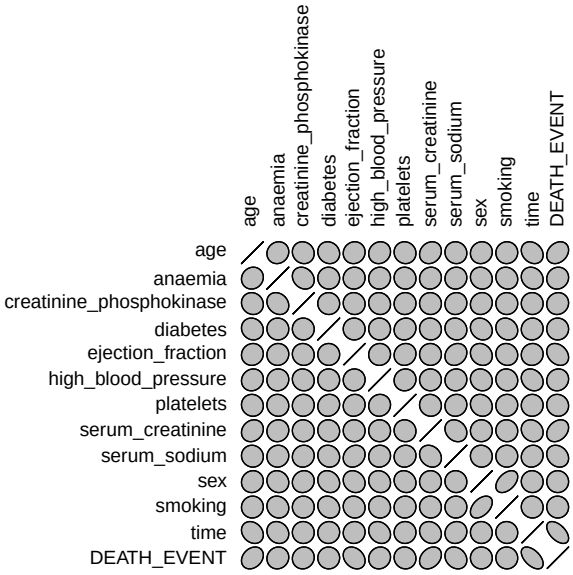
```
## [1] 67
```

```r
# Proportion of dead patients in the train and test set:
sum(train$DEATH_EVENT==1)/dim(train)[1]

## [1] 0.2974138

sum(test$DEATH_EVENT==1)/dim(test)[1]

## [1] 0.4029851
```

2 Data Exploration

The strenght of the linear association between two variables X and Y can be expressed in terms of the Covariance S_{XY} and the Pearson Correlation $r_{XY} = \frac{S_{XY}}{S_X S_Y}$.
In the following plot, the shape of the ellipses represents the Pearson coefficient for the corresponding variables: a flattened ellipse indicates that r_{XY} is near 1 or -1, while an almost circular ellipse indicates that r_{XY} is near 0.

```r
library(ellipse)
plotcorr(cor(Heart))
```



This plot highlights that Age and Serum Creatinine have a just hinted positive correlation with the response, while Serum Sodium and Ejection Fraction have a just hinted negative linear correlation with the response. This means that old age, high Serum Creatinine, low Serum Sodium and low Ejection Fraction are all factors that may increase the risk of death.
The quantitative value of these correlations are reported in the Table 2.

Feature	Correlation with Death Event
Age	0.254
Serum Creatinine	0.294
Serum Sodium	-0.195
Ejection Fraction	-0.269

Table 2: Relevant correlations with the response variable.

For what concerns the variable Time, the analysis is different. In fact the response variable is pretty correlated with Time (actual follow-up period), since it indicates whether the patient died or not before the end of the planned follow-up period. Therefore, a death often corresponds to a short value of time:

```
cor(DEATH_EVENT,time)

## [1] -0.5269638

mean(time[DEATH_EVENT==1])   # time mean for dead patients

## [1] 70.88542

mean(time[DEATH_EVENT==0])   # time mean for survived patients

## [1] 158.3399

# Proportion of dead patients in less than 90 days:
sum(time[DEATH_EVENT==1]<=90)/length(time[DEATH_EVENT==1])

## [1] 0.71875

# Proportion of dead patients in more than 90 days:
sum(time[DEATH_EVENT==1]>90)/length(time[DEATH_EVENT==1])

## [1] 0.28125
```

Taking into account this fact, we decided to not exclude the variable Time from the analysis, because the meaning of the response variable highly depends on the follow-up period. However, we preferred to focus more on the other clinical features and their influence on the response variable.

Collinearity stands for perfect linear correlation. Therefore couples of variables having correlation equal to 1 or -1 are said to be collinear. In our dataset the highest correlation (in absolute value) between features is $cor(sex, smoking) = 0.446$ and the others are lower than 0.230 (in absolute value). Therefore, our models will reasonably not suffer from collinearity of the features.

It's interesting to give some more insights about certain feature-response relationships and feature-feature relationships, that are consistent with well-known medical evidences.
First of all, our data shows that patients over the age of 70 have an higher risk to die than patients aged between 40 and 70:

```r
death_age <-matrix(c(length(DEATH_EVENT[age<=70&DEATH_EVENT==0]),
                     length(DEATH_EVENT[age<=70&DEATH_EVENT==1]),
                     length(DEATH_EVENT[age>70&DEATH_EVENT==0]),
                     length(DEATH_EVENT[age>70&DEATH_EVENT==1])), 2)
rownames(death_age) <- c("alive","dead")
colnames(death_age) <- c("40-70",">70")

death_age
```

```
##       40-70 >70
## alive   182  21
## dead     65  31
```

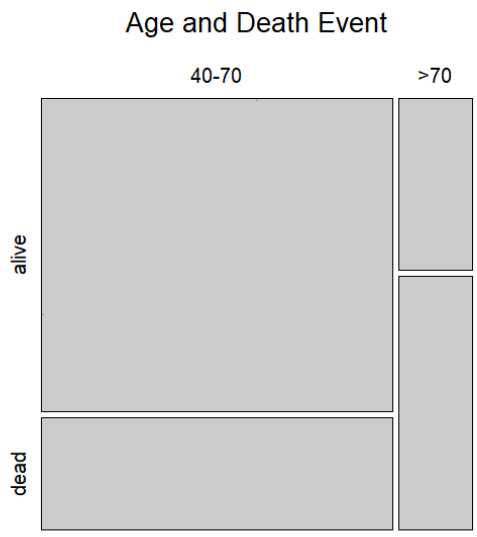
```r
margin_age <- margin.table(death_age, 2)

# Probability to die given the patient is over 70 years old:
death_age["dead",">70"]/margin_age[">70"]
```

```
##       >70
## 0.5961538
```

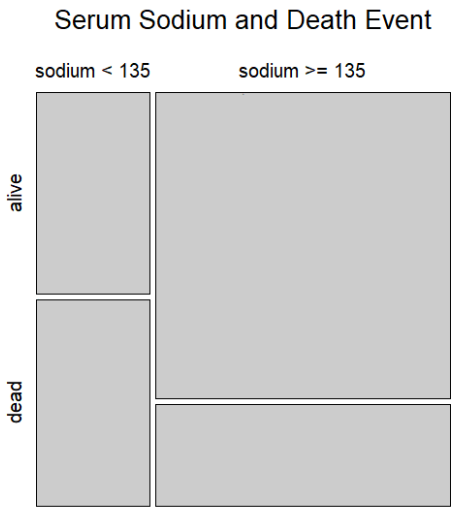
```r
# Probability to die given the patient is under 70 years old:
death_age["dead","40-70"]/margin_age["40-70"]
```

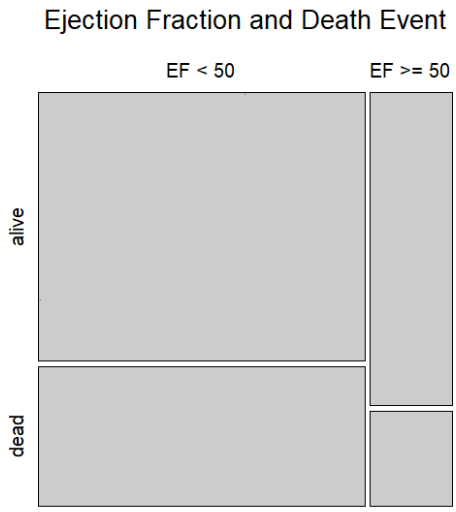
```
##     40-70
## 0.2631579
```

```r
# Qualitative plot for the relationship between Age and Death Event:
mosaicplot(death_age)
```



With similar computations, we can observe that patients having less than 135 mEq/L of Serum Sodium in the blood have an higher risk to die than patients with higher levels of this chemical compound; moreover, patients having less than 50% of Ejection Fraction have an higher risk to die than patients with an higher percentage of pumped blood:





Taking into account that the typical range for Serum Creatinine is $(0.5, 1.0)$ for females and $(0.7, 1.2)$ for males, it seems that patients having high levels of Serum Creatinine in the blood have an higher risk to die than patients with normal or low levels of this chemical compound:

```
sex_creatinine <-matrix(c(length(sex[serum_creatinine>=1&sex==0]),
                          length(sex[serum_creatinine>=1.2&sex==1]),
                          length(sex[serum_creatinine<1&sex==0]),
                          length(sex[serum_creatinine<1.2&sex==1])), 2)
colnames(sex_creatinine) <- c("high creatinine","normal creatinine")
rownames(sex_creatinine) <- c("female","male")

sex_creatinine
```

```
##        high creatinine normal creatinine
## female              72                33
## male                82               112
```

```
# Consider only patients that died:
death_creatinine_sex <-matrix(c(length(sex[serum_creatinine>=1&sex==0&DEATH_EVENT==1]),
                          length(sex[serum_creatinine>=1.2&sex==1&DEATH_EVENT==1]),
                          length(sex[serum_creatinine<1&sex==0&DEATH_EVENT==1]),
                          length(sex[serum_creatinine<1.2&sex==1&DEATH_EVENT==1])),2)
```

```r
colnames(death_creatinine_sex) <- c("high creatinine","normal creatinine")
rownames(death_creatinine_sex) <- c("female","male")

death_creatinine_sex
```

```
##        high creatinine normal creatinine
## female              32                 2
## male                39                23
```

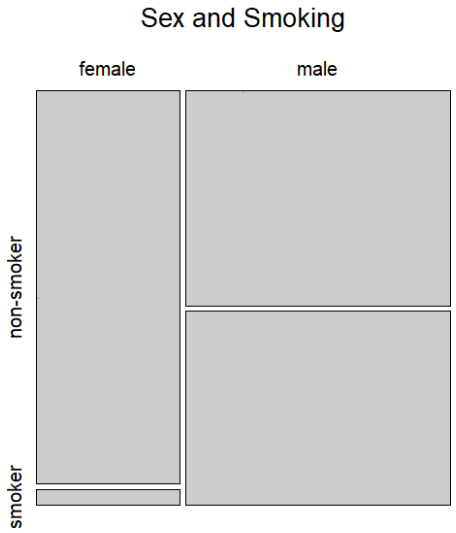
```r
# Proportion of patients with high creatinine that died:
death_creatinine_sex[,"high creatinine"]/sex_creatinine[,"high creatinine"]
```

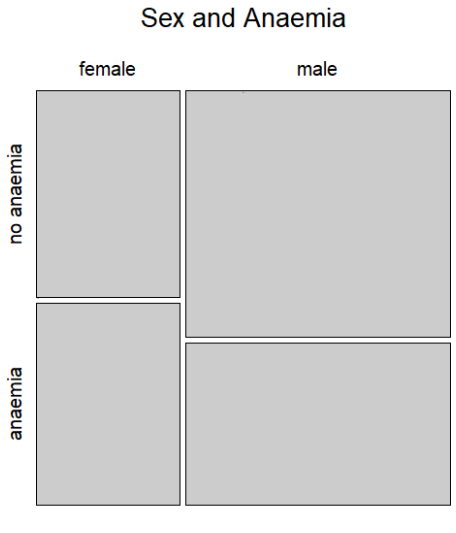
```
##    female      male
## 0.4444444 0.4756098
```

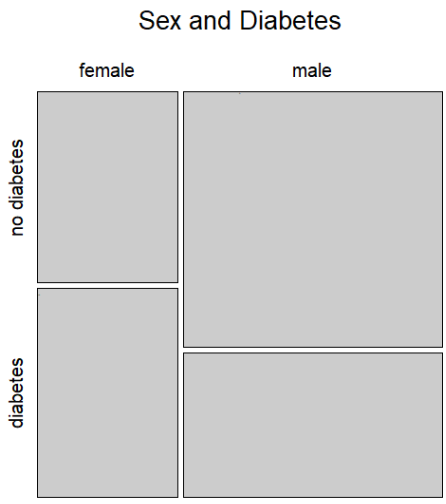
```r
# Proportion of patients with normal or low creatinine that died:
death_creatinine_sex[,"normal creatinine"]/sex_creatinine[,"normal creatinine"]
```

```
##     female       male
## 0.06060606 0.20535714
```

Again, using computations similar to those of the Age and Death Event relationship, we can observe that males have higher probability to smoke than females and females have higher probability to have anaemia or diabets:







Moreover, females often have more serious complications and a greater risk of death due to diabetes:

```
sex_diabetes <-matrix(c(length(sex[diabetes==0&sex==0]),
                        length(sex[diabetes==0&sex==1]),
                        length(sex[diabetes==1&sex==0]),
                        length(sex[diabetes==1&sex==1]) ),2, byrow = TRUE)
rownames(sex_diabetes) <- c("no diabetes","diabetes")
colnames(sex_diabetes) <- c("female","male")

# Consider only patients that died:
sex_diabetes_death <-matrix(c(length(sex[DEATH_EVENT==1&sex==0&diabetes==0]),
                      length(sex[DEATH_EVENT==1&sex==1&diabetes==0]),
                      length(sex[DEATH_EVENT==1&sex==0&diabetes==1]),
                      length(sex[DEATH_EVENT==1&sex==1&diabetes==1]) ),2, byrow = TRUE)
rownames(sex_diabetes_death) <- c("no diabetes","diabetes")
colnames(sex_diabetes_death) <- c("female","male")

# Probability to die given you have diabetes:
sex_diabetes_death["diabetes",]/sex_diabetes["diabetes",]

##    female      male
## 0.3636364 0.2857143
```

We can also observe that, after age 40, blood pressure is higher in females than in males:

```
sex_age <-matrix(c(length(sex[age<=50&sex==0]),length(sex[age<=50&sex==1]),
               length(sex[age>50&age<=58&sex==0]),length(sex[age>50&age<=58&sex==1]),
               length(sex[age>58&age<=64&sex==0]),length(sex[age>58&age<=64&sex==1]),
               length(sex[age>64&age<=70&sex==0]),length(sex[age>64&age<=70&sex==1]),
               length(sex[age>70&sex==0]),length(sex[age>70&sex==1])),2)
```

```
colnames(sex_age) <- c("40-50","50-58","58-64","64-70",">70")
rownames(sex_age) <- c("female","male")

# Consider only patients that have high blood pressure:
pressure_sex_age <-matrix(c(length(sex[age<=50&sex==0&high_blood_pressure==1]),
                      length(sex[age<=50&sex==1&high_blood_pressure==1]),
                      length(sex[age>50&age<=58&sex==0&high_blood_pressure==1]),
                      length(sex[age>50&age<=58&sex==1&high_blood_pressure==1]),
                      length(sex[age>58&age<=64&sex==0&high_blood_pressure==1]),
                      length(sex[age>58&age<=64&sex==1&high_blood_pressure==1]),
                      length(sex[age>64&age<=70&sex==0&high_blood_pressure==1]),
                      length(sex[age>64&age<=70&sex==1&high_blood_pressure==1]),
                      length(sex[age>70&sex==0&high_blood_pressure==1]),
                      length(sex[age>70&sex==1&high_blood_pressure==1])),2)
colnames(pressure_sex_age) <- c("40-50","50-58","58-64","64-70",">70")
rownames(pressure_sex_age) <- c("female","male")

# Proportion of males having high blood pressure:
pressure_sex_age["male",]/sex_age["male",]

##     40-50     50-58     58-64     64-70       >70
## 0.2558140 0.2432432 0.3076923 0.3714286 0.4000000

# Proportion of females having high blood pressure:
pressure_sex_age["female",]/sex_age["female",]

##     40-50     50-58     58-64     64-70       >70
## 0.4193548 0.3571429 0.3500000 0.4285714 0.5833333
```

This result also shows us that, regardless of sex, the proportion of patients having high blood pressure gets bigger as the age increases.

We can also deduce that smoking may lead to high pressure in females, while the proportion of smoking males having high blood pressure is pretty similar to non-smoking males having high blood pressure:

```
sex_smoking <-matrix(c(length(sex[smoking==0&sex==0]),
                       length(sex[smoking==0&sex==1]),
                       length(sex[smoking==1&sex==0]),
                       length(sex[smoking==1&sex==1]) ),2, byrow = TRUE)
rownames(sex_smoking) <- c("non-smoker","smoker")
colnames(sex_smoking) <- c("female","male")
```

```r
sex_smoke_pressure <-matrix(c(length(sex[smoking==0&sex==0&high_blood_pressure==1]),
              length(sex[smoking==0&sex==1&high_blood_pressure==1]),
              length(sex[smoking==1&sex==0&high_blood_pressure==1]),
              length(sex[smoking==1&sex==1&high_blood_pressure==1])),2, byrow = TRUE)
rownames(sex_smoke_pressure) <- c("non-smoker","smoker")
colnames(sex_smoke_pressure) <- c("female","male")

# Probability to have high pressure given that the patient smokes:
sex_smoke_pressure["smoker",]/sex_smoking["smoker",]
```

```
##    female      male
## 0.7500000 0.2934783
```

```r
# Probability to have high pressure given that the patient doesn't smoke:
sex_smoke_pressure["non-smoker",]/sex_smoking["non-smoker",]
```

```
##    female      male
## 0.4059406 0.3333333
```

With analogous calculations we deduce that diabetes or low Ejection Fraction may lead to high levels of Serum Creatinine in the blood:

```r
# Probability to have high Serum Creatinine given diabetes:
sex_diabetes_creatinine["diabetes",]/sex_diabetes["diabetes",]
```

```
##    female      male
## 0.7090909 0.4857143
```

```r
# Probability to have high Serum Creatinine given no diabetes:
sex_diabetes_creatinine["no diabetes",]/sex_diabetes["no diabetes",]
```

```
##    female      male
## 0.6600000 0.3870968
```

```r
# Probability to have high Serum Creatinine given low Ejection Fraction:
sex_ef_creatinine["low ef",]/sex_ef["low ef",]
```

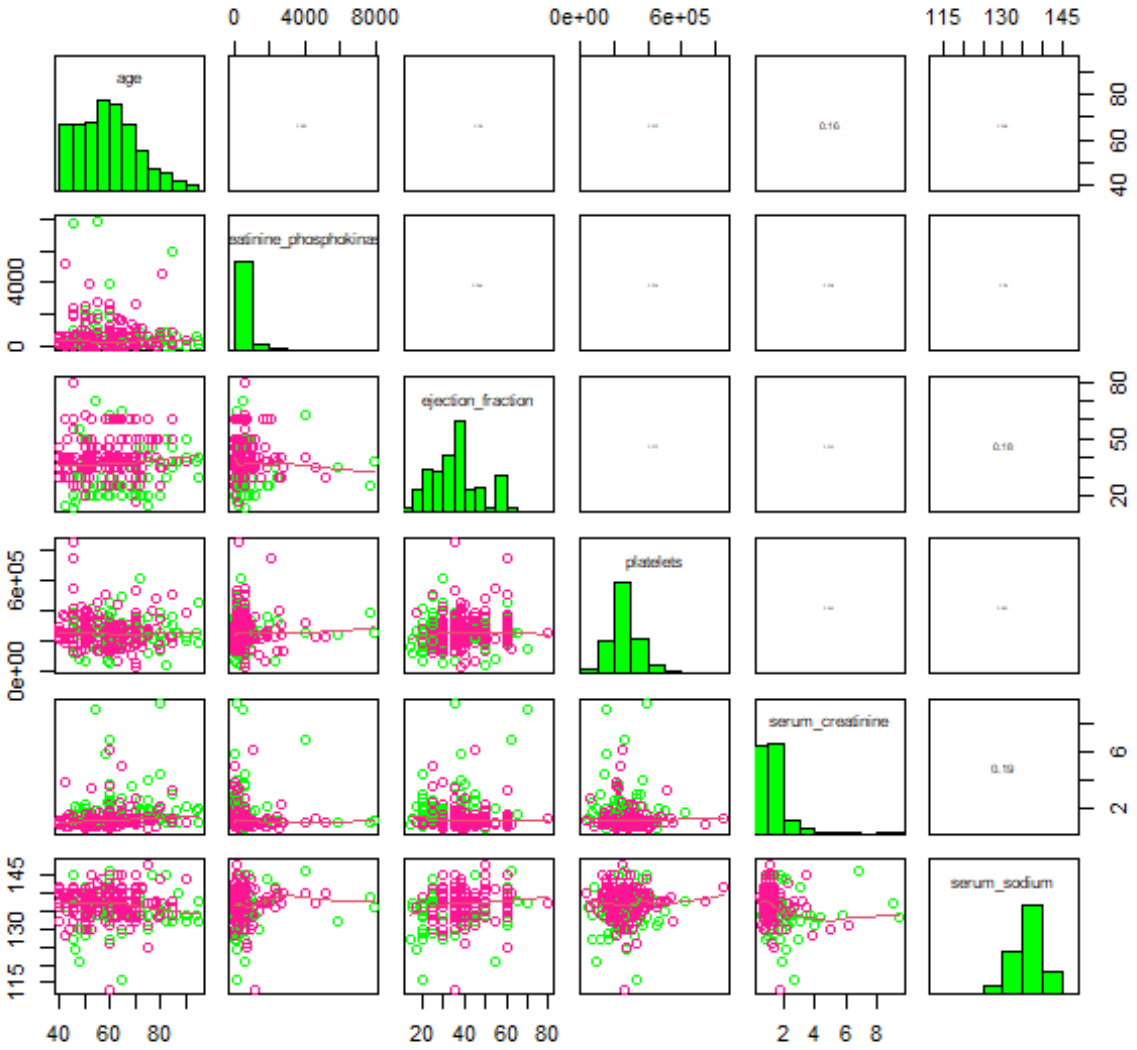
```
##    female      male
## 0.6883117 0.4506173
```

```r
# Probability to have high Serum Creatinine given not low Ejection Fraction:
sex_ef_creatinine["normal ef",]/sex_ef["normal ef",]
```

```
##    female      male
## 0.6785714 0.2812500
```

The following figure represents the scatter-plots (lower triangle) and the correlation coefficients (upper triangle) between features and the histograms of each feature on the main diagonal. Only quantitative features are included, except Time.

```
# Add colors to the scatter-plots to distinguish between the two classes:
colors <- c('deeppink', 'green')[unclass(as.factor(DEATH_EVENT))]
pairs(Heart[,-c(2,4,6,10,11,12,13)], col=colors, diag.panel=panel.hist,
              upper.panel=panel.cor, lower.panel=panel.smooth)
```

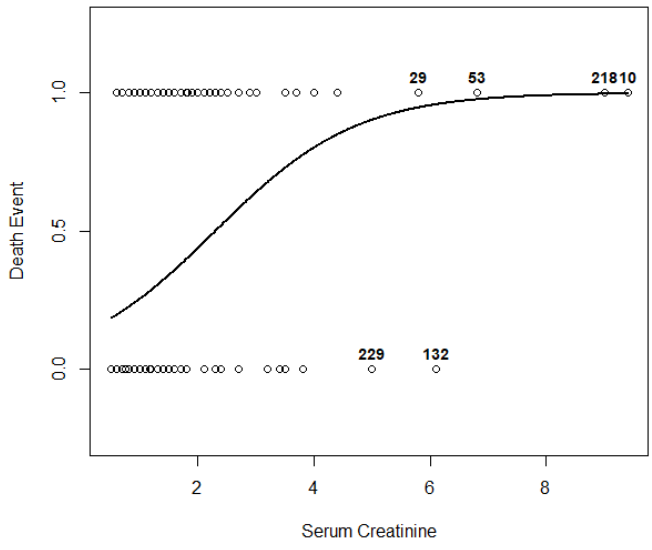


The scatter-plots do not reveal sistematic behaviours and the features look non-linearly separable. This confirms that our data is not affected by collinearity, but the classification task is quite difficult given the evident overlapping of the continuous features. Therefore, we can't expect to obtain a very high accuracy of the predictions.

2.1 Diagnostic

The analysis of this section concerns outliers and high leverage points for the logistic regression.
First, we analyzed the problematic points for each quantitative feature separately, just to give an intuition of which points fall out of the normal medical range and could have an influence on the regression.
Finally we considered all the features together (quantitative and qualitative), to extrapolate the true outliers and high leverage points.
Let's start by analyzing Serum Cratinine:

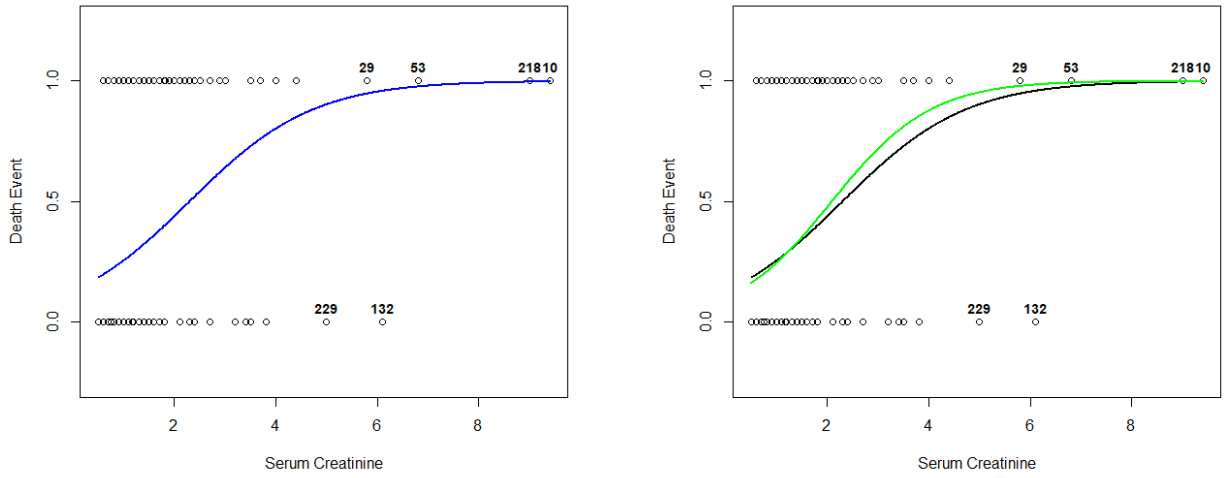
```
# Inverse transformation for computing pi_i in case of just one regressor:
inv.logit <- function(beta0, beta1, x) {
y <- exp(beta0+beta1*x)
return(y/(1+y))
}

logit.out <- glm(DEATH_EVENT~ serum_creatinine, data=Heart, family = binomial)
plot(platelets, DEATH_EVENT, xlim=c(min(serum_creatinine), max(serum_creatinine)),
        ylim=c(-.25, 1.25), xlab = "Serum Creatinine", ylab = "Death Event")

x <- seq(min(serum_creatinine), max(serum_creatinine), length=1000)
y <- inv.logit(coefficients(logit.out)[1], coefficients(logit.out)[2], x)
lines(x, y, col="black", lwd=2.5)  # regression curve in black

# Label anomalous with the corresponding indexes in the dataframe:
data = Heart[serum_creatinine>=5,]
text(data$DEATH_EVENT~ data$serum_creatinine, data=data, labels=rownames(data), pos=3)
```



For each anomalous point, we re-compute the logistic model and the regression curve on a restricted dataset that does not include the point. The following code shows the computations for the points having index 10 and 132:

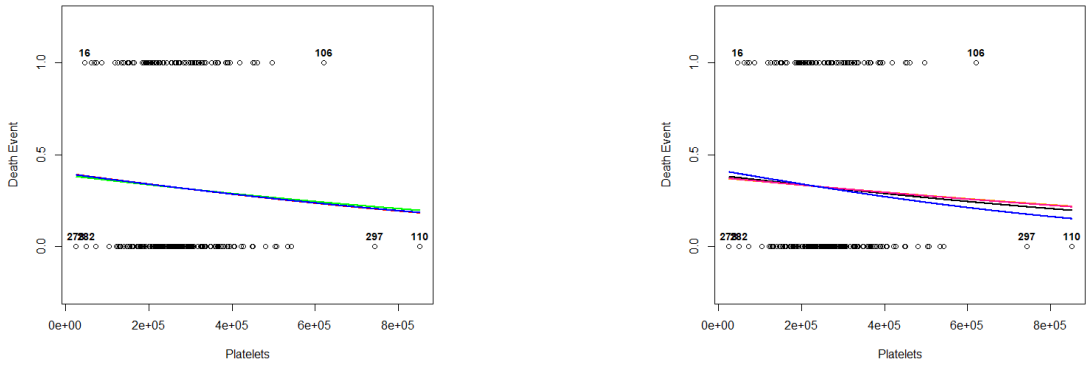
```
restricted_dataset <- Heart[-10,]
logit.r <- glm(DEATH_EVENT~ serum_creatinine, data=restricted_dataset, family = binomial)
y <- inv.logit(coefficients(logit.r)[1], coefficients(logit.r)[2], x)
lines(x, y, col="blue", lwd=2.5)
```

```
restricted_dataset <- Heart[-132,]
logit.r <- glm(DEATH_EVENT~ serum_creatinine, data=restricted_dataset, family = binomial)
y <- inv.logit(coefficients(logit.r)[1], coefficients(logit.r)[2], x)
lines(x, y, col="green", lwd=2.5)
```

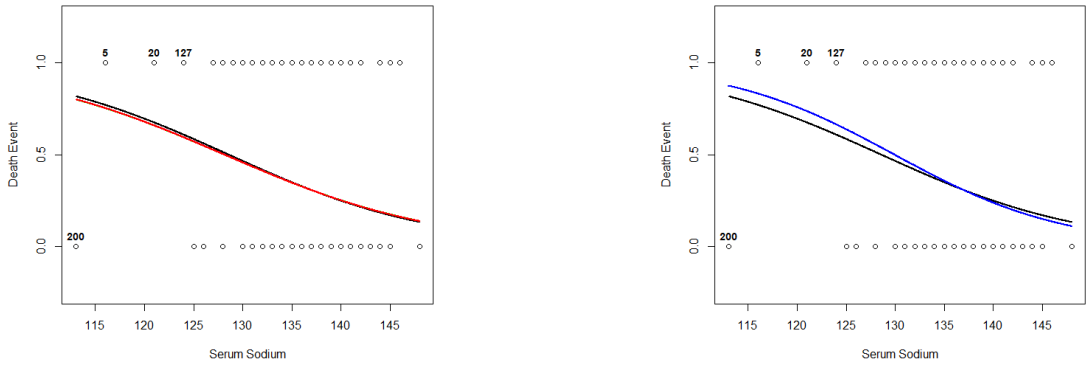


The sample 10 (and similarly the samples 29, 53 and 218) results to not influence the shape of the regression line, whereas the presence of sample 132 (and similarly the sample 229) highly influences the regression line. In fact, the typical range for Serum Creatinine is $(0.5, 1.0)$ for females and $(0.7, 1.2)$ for males, and here anomalous values are greater than 5.0. Therefore, one could expect that such high levels of waste in the blood will lead to death. Then it's reasonable that observing sample 132 will change the regression line, since it represents a survived patient who, however, presented a high level of Serum Creatinine.

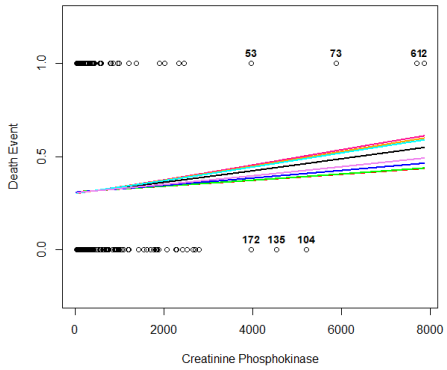
The same analysis was carried out for all the other quantitative features, except Time. In particular, Age and Ejection Fraction did not present strange values, and the results for Platelets, Serum Sodium and Creatinine Phosphokinase are reported in following figures.



Platelets: samples 16, 278 and 282 don't affect much the regression curve, unlike samples 106, 110 and 297.



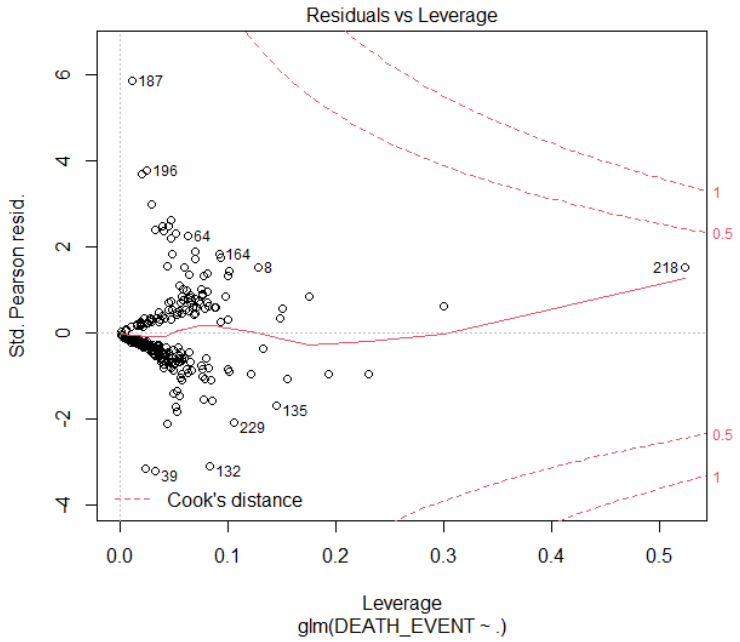
Serum Sodium: samples 5, 20 and 127 don't affect much the regression curve, unlike sample 200.



Creatinine Phosphokinase: samples 2, 53, 61, 73, 104, 135 and 172 influence the regression curve shape.

In conclusion, we fit a logistic regression model considering all the features together and produce the Residuals VS Leverage plot:

```
logit.out <- glm(DEATH_EVENT~., data=Heart, family = binomial)
plot(logit.out, id.n=10, which=5)    # shows the indexes of the 10 most extreme points:
```



The following tables report the problematic points, divided in outliers and high leverage points.

Index	Age	Anaemia	CFK	Diabetes	EF	HBP	Platelets	SC	SS	Sex	Smoke	Time	Death
187	50	0	582	0	50	0	153000	0.6	134	0	0	172	1
196	77	1	418	0	45	0	223000	1.8	145	1	0	180	1
64	45	0	582	0	35	0	385000	1	145	1	0	61	1
229	65	0	56	0	25	0	237000	5	130	0	0	207	0
132	60	1	1082	1	45	0	250000	6.1	131	1	0	107	0
39	60	0	2656	1	30	0	305000	2.3	137	1	0	30	0

Table 3: Outliers.

Index	Age	Anaemia	CFK	Diabetes	EF	HBP	Platelets	SC	SS	Sex	Smoke	Time	Death
185	58	1	145	0	25	0	219000	1.2	137	1	1	170	1

Table 4: High Leverage point.

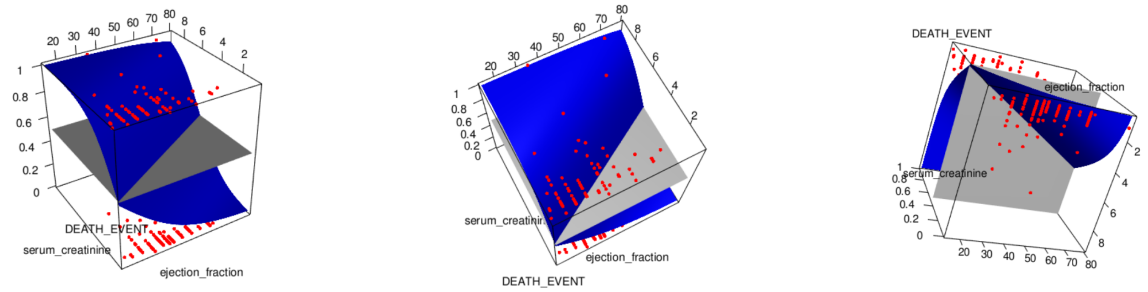
Patient 187 was just 50 years old and died after a long follow-up period without presenting any particularly critical clinical value.
Patient 196 presented a quite old age and both low Ejection Fraction and a high level of Serum Creatinine, but he only died after 180 days of follow-up.
Patient 64 died at early age.
Patients 229 and 132 did not die, even if they had low Ejection Fraction, slightly low Serum Sodium and extremely high Serum Creatinine.
Patient 39 had serius clinical values, such as low Ejection Fraction, pretty high Creatinine Phosphokinase, Diabetes and high Serum Creatinine, but his follow-up period terminated quickly.

Patient 185 presented a very low Ejection Fraction, while the other clinical values are acceptable or even fall in the normal range. He eventually died after a long follow-up period, at a relatively early age.

2.2 Interaction Effects

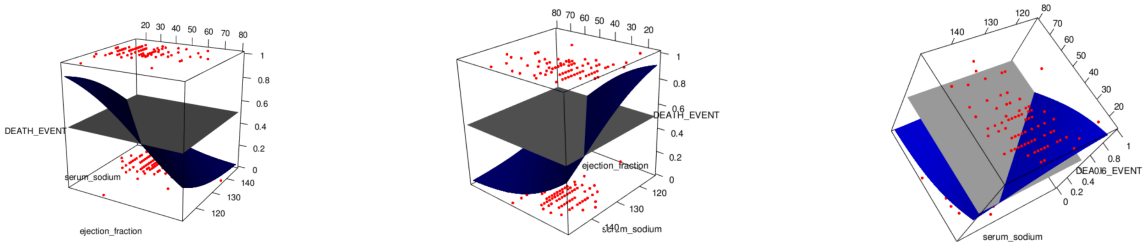
Before fitting our models, we have to check if there's any synergy or interaction between two features, i.e. if the response variable seems to benefit from the fact that two features are simultaneously present. This would indicate a systematic deviation from the randomness of the residuals.
To this aim, we analyzed the 3D plot having the response variable on the z axis and two features on the x and y axes. The following code produces the plot for Death Event, Ejection Fraction and Serum Creatinine:

```
# Inverse transformation for computing pi_i in case of two regressors:
fx <- function(u,v) u
fy <- function(u,v) v
fz <- function(u,v){
         exp(coefficients(logit.out)[1] + coefficients(logit.out)[2] * u +
         coefficients(logit.out)[3] * v) / (1 + exp(coefficients(logit.out)[1] +
         coefficients(logit.out)[2] * u + coefficients(logit.out)[3] * v))
}

logit.out <- glm(DEATH_EVENT~ ejection_fraction + serum_creatinine,
                 data=Heart, family = binomial)

library(misc3d)
library("rgl")
plot3d(ejection_fraction, serum_creatinine, DEATH_EVENT, col="red", size=5)
x0 <- seq(min(ejection_fraction), max(ejection_fraction), length=1000)
y0 <- seq(min(serum_creatinine), max(serum_creatinine), length=1000)
# Regression surface:
parametric3d(fx, fy, fz, u=x0, v=y0, color="blue", fill = FALSE, add=TRUE)
# Default threshold of 0.5:
planes3d(0,0,1,-0.5, col="gray", add=TRUE)
```

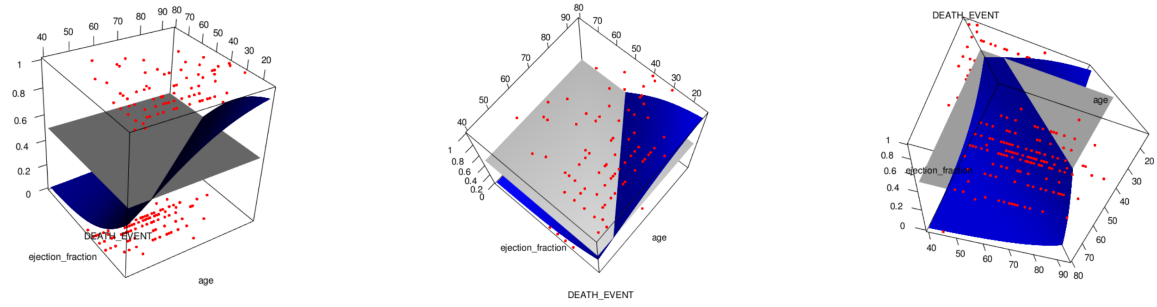


Low Ejection Fraction and high Serum Creatinine could reasonably lead to death. In fact the (blue) regression surface cut the (grey) horizontal plane - representing the default threshold 0.5 - forming an oblique line. However, in the second and third plot, we can observe some misclassified points, both for class 0 and class 1. Therefore we can't expect an interaction effect between Ejection Fraction and Serum Creatinine.

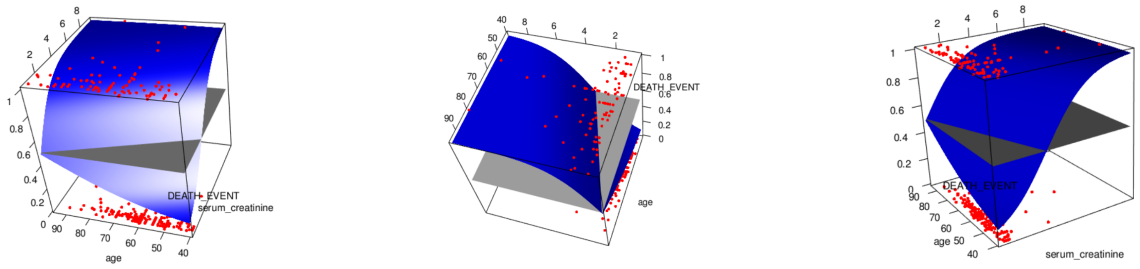
With analogous computations, we also checked other reasonable interactions leading to death and produced the following plots:



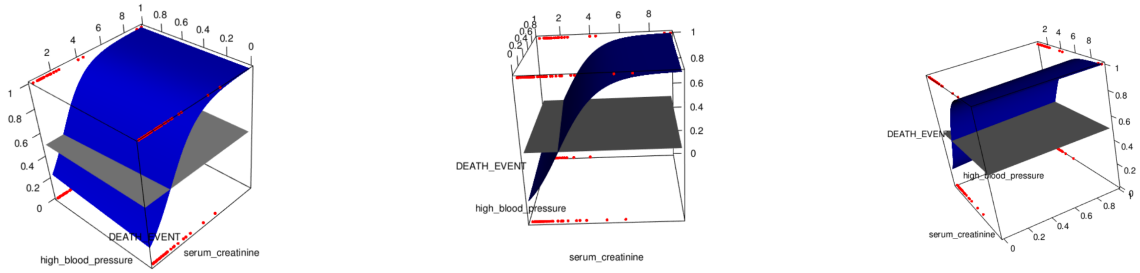
Low Ejection Fraction and low Serum Sodium: the regression surface cut the horizontal plane forming an oblique line. Moreover, all the six points in the portion of space described by Ejection Fraction lower than 30 and Serum Sodium lower than 130, belong to class 1. However, the other points seem more or less randomly dispersed among the two classes and therefore we can't expect that the interaction effect between Ejection Fraction and Serum Creatinine would lead to a better fit of the data.



Low Ejection Fraction and old Age: the regression surface cut the horizontal plane forming an oblique line. Even if the majority of points in the portion of space described by Ejection Fraction lower than 40 and Age greater than 70, belong to class 1, there's also a small amount of points in that region belonging to class 0, as shown in the third plot. In addition, there's a relevant amount of misclassified points in both classes and therefore we can exclude an interaction effect between Ejection Fraction and Age.



High Serum Creatinine and old Age: the regression surface cut the horizontal plane forming an oblique line. However, in this case it's clear that points don't present a systematic behaviour and therefore we can exclude an interaction effect between Serum Creatinine and Age.



High Serum Creatinine and High Blood Pressure: the regression surface cut the horizontal plane forming an almost perpendicular line to the Serum Creatinine axis. This indicates that the regression surface mostly depends on the value of Serum Creatinine rather than High Blood Pressure. In general, points are randomly dispersed and therefore there's no interaction effect between Serum Creatinine and High Blood Pressure.

Other interactions have been checked but none of them resulted to be relevant.

3 Methods

In our analysis, we must take into account that a trivial classifier predicting all 0s will have a better accuracy than a random model. This is due to the fact that the dataset is unbalanced:

```
trivial.pred <- rep(0, dim(train)[1])
# Confusion Matrix:
table(trivial.pred, train$DEATH_EVENT)

##
## trivial.pred   0   1
##            0 163  69

# Training Accuracy:
table(trivial.pred, train$DEATH_EVENT)[1]/dim(train)[1]

## [1] 0.7025862
```

3.1 Best Subset Selection

The Best Subset Selection is a method for strategically choosing predictors of a model until the best model with a given number of predictors is found. The algorithm starts with the null model, containing no predictors, which predicts the sample mean for each observation. Then predictors are incrementally added one by one to the null model. At the k-th increment, all possible models with k predictors are evaluated with a certain performance measure. This is done, until a selected maximum number of predictors is reached. For each number of predictors, the best model is selected: these models can be compared assessing the trade-off between the fit of the model and its simplicity.
The evaluation and comparison can be done using various performance measures, such as the Bayesian Information Criterion (BIC) or the Akaike Information Criterion (AIC). Both of them are using the maximum likelihood to evaluate the models. The BIC is placing a larger penalty on models with a larger number of variables and therefore results in a smaller selection of predictors.

BIC Analysis

```
library(bestglm)
# Prepare data: Xy must contain the dataframe X and response y
train.Xy <- within(train, {
     y <- train$DEATH_EVENT
     DEATH_EVENT <- NULL
})

# BSS using BIC:
regfit.full <- bestglm(Xy = train.Xy, family = binomial, method="exhaustive",
                       IC="BIC", nvmax=d)
best.model.BIC <- regfit.full$BestModel
summary(best.model.BIC)

##
## Call:
## glm(formula = y ~ ., family = family, data = Xi, weights =  weights)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0029  -0.5527  -0.1993   0.3613   2.3465
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.350276   1.250593   0.280 0.779410
## age                0.065414   0.018885   3.464 0.000532 ***
## ejection_fraction -0.095450   0.020539  -4.647 3.36e-06 ***
## serum_creatinine   0.487220   0.204414   2.383 0.017149 *
```

```
## time              -0.022360   0.003607  -6.198 5.71e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.42  on 231  degrees of freedom
## Residual deviance: 158.20  on 227  degrees of freedom
## AIC: 168.2
##
## Number of Fisher Scoring iterations: 6
```

```
# Accuracy on train set with default threshold:
pred.BIC <- predict(best.model.BIC, newdata=train, type="response")
pred.BIC[pred.BIC<0.5] <- 0
pred.BIC[pred.BIC>=0.5] <- 1
sum(train$DEATH_EVENT == pred.BIC)/dim(train)[1]
```

```
## [1] 0.8534483
```

```
# Accuracy on test set with default threshold:
pred.BIC <- predict(best.model.BIC, newdata=test, type="response")
pred.BIC[pred.BIC<0.5] <- 0
pred.BIC[pred.BIC>=0.5] <- 1
sum(test$DEATH_EVENT == pred.BIC)/dim(test)[1]
```

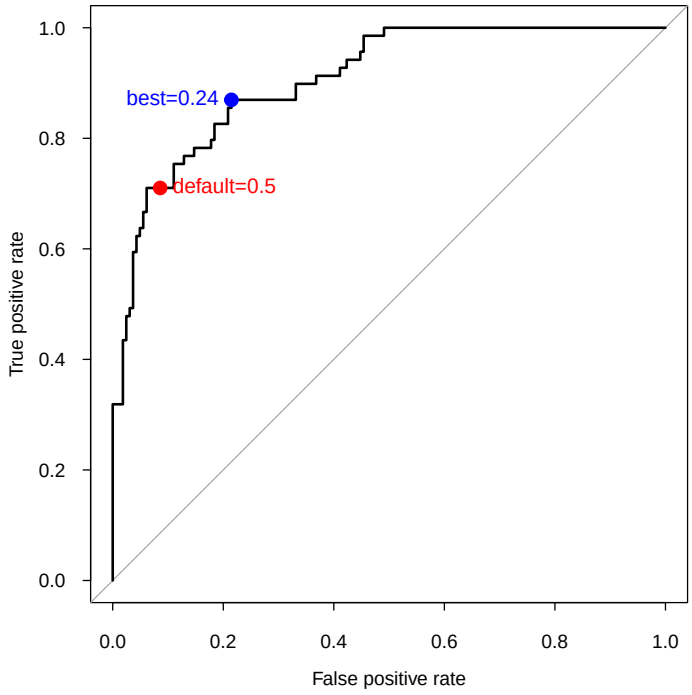
```
## [1] 0.7761194
```

Using BIC as a performance measure gives the following results: the model selected contains the variables age, ejection_fraction, serum_creatinine, and time; all of them being significant to at least 0.02 level. Using these predictors and a default threshold of 0.5 to calculate the performance on the training and test set gives an accuracy of 85.34% and 77.61% respectively.
In our model we prefer to minimize the False Negatives instead of the False Positives. In fact, saying that a person will survive the follow-up period while he/she won't, is worse compared to committing the opposite error. For this reason, the obtained results can be optimized by adjusting the default threshold through the ROC curve analysis, as follows:

```
# Confusion Matrix with default threshold:
table(pred.BIC, train$DEATH_EVENT)
```

```
##
## pred.BIC   0   1
##        0 149  20
##        1  14  49
```

```r
# ROC Curve:
library(pROC)
pred.BIC <- predict(best.model.BIC, newdata=train, type="response")
roc.out <- roc(train$DEATH_EVENT, pred.BIC, levels=c(0,1))
auc(roc.out)

## Area under the curve: 0.9067

# best threshold:
coords(roc.out, "best")

##   threshold specificity sensitivity
## 1  0.241733   0.7852761   0.8695652

# default threshold:
coords(roc.out, 0.5)

##   threshold specificity sensitivity
## 1       0.5   0.9141104   0.7101449

best.thr.BIC <- as.double(coords(roc.out, "best")[1])

# Confusion Matrix with best threshold:
pred.BIC <- predict(best.model.BIC, newdata=train, type="response")
pred.BIC[pred.BIC<best.thr.BIC] <- 0
pred.BIC[pred.BIC>=best.thr.BIC] <- 1
sum(train$DEATH_EVENT == pred.BIC)/dim(train)[1]
table(pred.BIC, train$DEATH_EVENT)

##
## pred.BIC   0   1
##        0 128   9
##        1  35  60

# Graphical representation of the ROC curve:
plot(roc.out, legacy.axes=TRUE, xlab="False positive rate", ylab="True positive rate")
points(coords(roc.out, "best")[2], coords(roc.out, "best")[3], col="blue", pch=19, lwd=4)
text(coords(roc.out, "best")[2], coords(roc.out, "best")[3], labels="best=0.24",
     pos=2,offset=0.5, col="blue")

# compare with default threshold:
points(coords(roc.out, 0.5)[2], coords(roc.out, 0.5)[3], col="red", pch=19, lwd=4)
text(coords(roc.out, 0.5)[2], coords(roc.out, 0.5)[3], labels="default=0.5",
     pos=4,offset=0.5, col="red")
```




```
# Accuracy on test set with best threshold:
pred.BIC <- predict(best.model.BIC, newdata=test, type="response")
pred.BIC[pred.BIC<best.thr.BIC] <- 0
pred.BIC[pred.BIC>=best.thr.BIC] <- 1
sum(test$DEATH_EVENT == pred.BIC)/dim(test)[1]

## [1] 0.7313433
```

The test accuracy with the best threshold of 0.24 drops to 73.13% but the model gains the best possible sensitivity and specificity.

AIC Analysis

The same calculations have been carried out for the AIC measure. Using the AIC led to the selection of the following seven variables: age, serum_creatinine, time, ejection_fraction, sex, serum_sodium, serum_creatinine. Compared to the BIC selected variables, four of them are the same and three additionals are chosen. This shows the characteristic of BIC to select fewer variables. The AIC choice of variables and their level of significance can be found below, it is to notice that not all variables selected are significant.

```
regfit.full <- bestglm(Xy = train.Xy, family = binomial, method="exhaustive",
                       IC="AIC", nvmax=d)

best.model.AIC <- regfit.full$BestModel
summary(best.model.AIC)
```

```
##
## Call:
## glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2968  -0.5177  -0.1806   0.2922   2.5802
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              13.1423082  7.4565289   1.763 0.077981 .
## age                       0.0697594  0.0202161   3.451 0.000559 ***
## creatinine_phosphokinase  0.0005194  0.0003716   1.398 0.162177
## ejection_fraction        -0.1011710  0.0218588  -4.628 3.69e-06 ***
## serum_creatinine          0.3517129  0.2215746   1.587 0.112437
## serum_sodium             -0.0900178  0.0532378  -1.691 0.090863 .
## sex1                     -0.8921974  0.4414368  -2.021 0.043267 *
## time                     -0.0231042  0.0037491  -6.163 7.16e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 282.42  on 231  degrees of freedom
## Residual deviance: 149.78  on 224  degrees of freedom
## AIC: 165.78
##
## Number of Fisher Scoring iterations: 6
```

The Tables 5 and 6 summarize the results obtained for the Best Subset Selection using BIC and AIC.

	Train Accuracy	**Test Accuracy**	**Specificity (train set)**	**Sensitivity (train set)**	**AUC**
BIC	85.34%	77.61%	91.41%	71.01%	0.9067
AIC	87.50%	74.62%	93.87%	72.46%	0.914

Table 5: Results with the default threshold.

	Best Threshold	**Test Accuracy**	**Specificity (train set)**	**Sensitivity (train set)**
BIC	0.24	73.13%	78.53%	86.95%
AIC	0.44	76.12%	91.41%	76.81%

Table 6: Results with the best threshold.

In conclusion, BIC selects less but more significant features than AIC and obtains a higher test accuracy with the default threshold, whereas the specificities and sensitivities are almost equal. On the other hand, AIC leads to a better test accuracy with its best threshold, whereas BIC gets a better sensitivity and the test accuracy is still acceptable. Taking all these facts into account, we can say that BIC offers a better selection of features than AIC, without losing much in terms of accuracy.

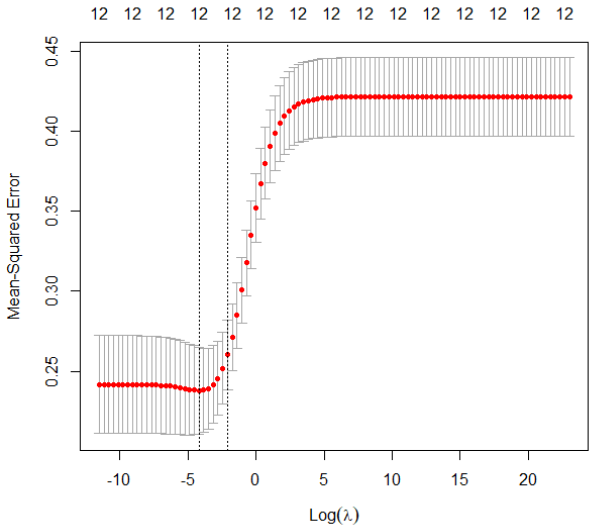
3.2 Shrinkage Methods

Ridge and Lasso regression are other methods of selecting the best-performing subset of predictors by using least squares. In contrast to the best subset selection method described in the previous section, the Ridge and Lasso regression do not exclude predictors completely, instead, they consider all variables, but set a penalty on the coefficients of the model. In practice, that means that parameters that do not contribute strongly enough to the performance of the model are forced to go to zero. A tuning parameter determines the severity with which the model is regularized. Using a lambda of zero equivalates to using the least-squares, whereas very high values for lambda force all coefficients to be zero, which would result in the null model. The main difference between Ridge and Lasso regression is the way they regularize the parameter coefficients. Ridge regression uses quadratic shrinking of the parameters, whereas Lasso regression uses the absolute values of the coefficients.

Ridge Regression

The main hyperparameter in Ridge Regression is lambda, which decides the magnitude of the regularization. A grid search with various values of lambda is done, and their performance in terms of the MSE is evaluated using cross-validation. A plot of the outcome can be found below.
After the best value for lambda is obtained, the model is fitted again on the whole training set.

```
library(glmnet)
X <- model.matrix(DEATH_EVENT~., data=train)
X <- X[,-1]
y <- train$DEATH_EVENT
grid <- 10^seq(10, -5, length=100)

# LEAVE-ONE-OUT Cross-Validation for finding optimal lambda:
cv.out <- cv.glmnet(X, y, family = "binomial", alpha=0, nfolds=n, lambda=grid,
                    type.measure = "mse")
# Optimal lambda:
bestlam.ridge <- cv.out$lambda.min
# Use the whole train set to fit the model with optimal lambda:
ridge.mod <- glmnet(X, y, family = "binomial", alpha=0, lambda=bestlam.ridge)

# Plot of the Cross-Validated MSE (with error bar) for all the lambda values:
plot(cv.out)
```

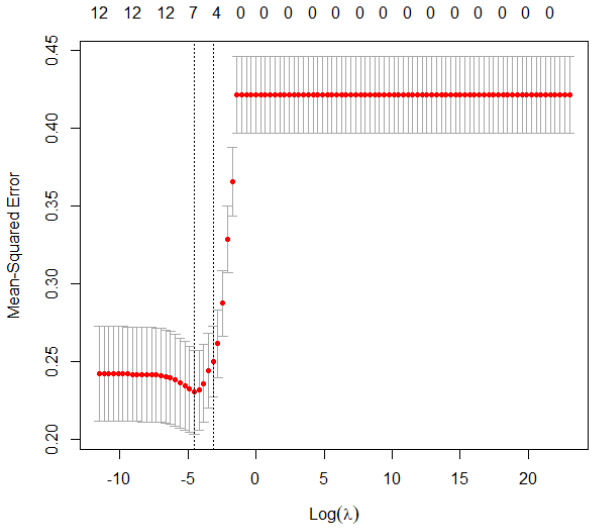



```
coefficients(ridge.mod)

## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                9.574842e+00
## age                        5.334455e-02
## anaemia1                  -6.751698e-02
## creatinine_phosphokinase   3.422384e-04
## diabetes1                 -3.538898e-02
## ejection_fraction         -7.580627e-02
## high_blood_pressure1       1.433041e-01
## platelets                  6.999627e-07
## serum_creatinine           3.518456e-01
## serum_sodium              -6.922480e-02
## sex1                      -6.489109e-01
## smoking1                   8.723818e-02
## time                      -1.756132e-02
```

Lasso Regression

Similarly, as for the Ridge Regression, a grid search is performed to find the best hyperparameter for this model. Below, the plot of the various values for lambda and the resulting MSE using cross-validation can be found. Compared to the same graph for Ridge Regression, this one shows that Lasso model obtains more or less the same MSE with a lower number of features (7 instead of 12) having the coefficient different from zero. In particular, these features are the same selected by AIC in the Best Subset Selection and the coefficients are also very similar to the ones of AIC model.

```r
# LEAVE-ONE-OUT Cross-Validation for finding optimal lambda:
cv.out <- cv.glmnet(X, y, family = "binomial", alpha=1, nfolds=n, lambda=grid,
                    type.measure = "mse")
# Optimal lambda:
bestlam.lasso <- cv.out$lambda.min
# Use the whole train set to fit the model with optimal lambda:
lasso.mod <- glmnet(X, y, family = "binomial", alpha=1, lambda=bestlam.lasso)

# Plot of the Cross-Validated MSE (with error bar) for all the lambda values:
plot(cv.out)
```




```r
coefficients(lasso.mod)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)                8.385508496
## age                        0.053388281
## anaemia1                   .
## creatinine_phosphokinase   0.000250073
## diabetes1                  .
## ejection_fraction         -0.079516185
## high_blood_pressure1       .
## platelets                  .
## serum_creatinine           0.311404551
## serum_sodium              -0.056269168
## sex1                      -0.540494679
## smoking1                   .
## time                      -0.019419829
```

At this point, for both Ridge and Lasso models, we repeated the same computations for test and train accuracies, confusion matrices and ROC curve as the previous section.
The results are reported in the Table 7 and Table 8.

	Train Accuracy	**Test Accuracy**	**Specificity (train set)**	**Sensitivity (train set)**	**AUC**
Ridge	87.07%	73.13%	94.48%	69.56%	0.913
Lasso	87.07%	76.12%	93.87%	71.02%	0.913

Table 7: Results with the default threshold.

	Best Threshold	**Test Accuracy**	**Specificity (train set)**	**Sensitivity (train set)**
Ridge	0.43	77.61%	92.02%	76.81%
Lasso	0.44	77.61%	93.25%	75.36%

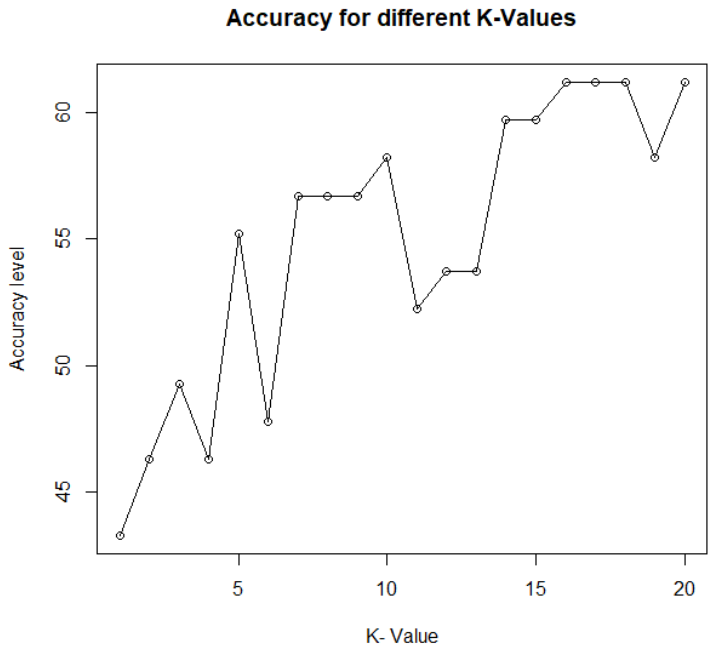
Table 8: Results with the best threshold.

In terms of both, accuracy and confusion table, these results are comparable with the results of the AIC model in the previous section.
Using the optimized threshold for Ridge and Lasso models gives a slightly higher test accuracy.
However, it needs to be noted that even if the test accuracy is at the same level, the composition of true positives (sensitivity) is slightly different, which would make the Lasso model preferable.

3.3 K-Nearest Neighbors

The K-nearest neighbor algorithm (KNN) is a non-parametric classification algorithm. the algorithm is based on the assumption that data points with similar attributes also share the same class label. For each unclassified coordinate, the algorithm considered the k most similar data points and classifies the point according to the majority class of the neighbors. Especially for binary classification, uneven numbers for K are preferred to avoid tied votes, however, in the case of a tie, the label is determined at random.
The only parameter to be selected in this algorithm is K, the number of neighbors that are considered to classify a coordinate. In order to choose the right value for K, one can split the data used into a training and test set, try different values for the parameter, and evaluate its accuracy of classification on the test set. Finding the best value for K is difficult, as KNN is strongly dependent on the starting condition of the algorithm. Therefore, with every time running the algorithm, the outcomes vary. The resulting accuracy for various K values can be seen in the figure below. Using all variables, it can be said that larger values give better accuracy. One problem is that the underlying data distribution is not uniform across the two classes, as class 0 is being overrepresented with a share of approximately 60%. Therefore, with larger values for K, the algorithm will classify the data generally as 0, and achieving an acceptable accuracy of above 60%. Logically, a model like that is not useful. It is important to look closer at the classifications and create an upper limit for K to avoid this behavior.
Visually, this phenomenon can be observed with a converging accuracy rate at a level of 60% starting from a K value from 20 onwards. Limiting the K values makes sure the algorithm classifies at least some instances as 1, which, however, gives a lower accuracy than before.

```r
library(class)
library(caret)
library(ROCR)

# Useful Functions:
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

best.k.acc <- function(train, test, y_train, y_test, k.max) {
     k.optm <- 0
     best.acc <- 0
     best.tab <- 0
     for (i in 1:k.max){
          pred <- knn(train=train, test=test, cl=y_train, k=i)
          tab <- table(pred,y_test)
          acc <- accuracy(tab)
          if(acc>best.acc){
               best.acc <- acc
               best.tab <- tab
          }
          k.optm[i]  <-  acc
     }
     cat("\nMax Accuracy: ", max(k.optm))
     cat("\nBest K: ",which(k.optm==max(k.optm)), "\n")
     print(best.tab)
     return(k.optm)
}


# KNN (Full Model):
k.optm <- best.k.acc(train,test,train$DEATH_EVENT,test$DEATH_EVENT,20)

##
## Max Accuracy:  61.19403
## Best K:  16 17 18 20
##     y_test
## pred  0  1
##    0 40 26
##    1  0  1

plot(k.optm, type="o", xlab="K- Value", ylab="Accuracy level",
     main="Accuracy for different K-Values")
```

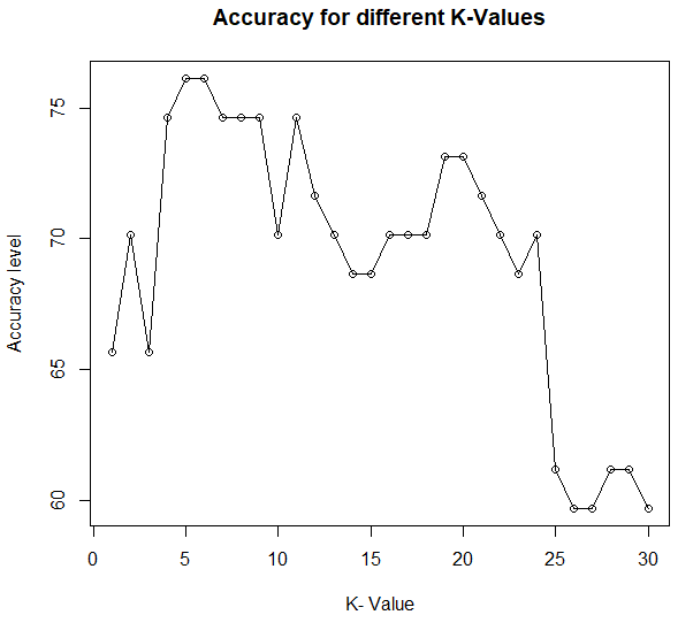


In order to improve the model, we will make use of the best subset variable selection, described in the previous chapter. The KNN algorithm is tested for both, the AIC selected model and the BIC selected model.

The AIC selected model of seven predictors shows an increase of accuracy of about 15% compared to the use of all the variables. The plot below shows how various values for K are impacting the results. It needs to be noticed that the overall data set and therefore also the test set was relatively small. This means even though the accuracy seems to be quite a bit better, this is caused by only a few more correctly classified data points. One more correct classification increases the accuracy by about 1.%. Generally, it seems that lower values for k, between 5 and 10 give the best results with a maximum accuracy on the test set of 76.11%. A possible explanation for this increased accuracy is that with the removal of some variables, also noise in the data is removed, which makes the remaining features more expressive. Also, the overall prediction has improved, the model correctly classifies positive and negative classes. It gives a sensitivity of 82.3% and a specificity of 74%.

```
# KNN (AIC Best Model):
AIC_var_train = train[c("age","creatinine_phosphokinase","time", "ejection_fraction","sex",
                        "serum_sodium","serum_creatinine", "DEATH_EVENT")]
AIC_var_test = test[c("age","creatinine_phosphokinase","time", "ejection_fraction", "sex",
                        "serum_sodium","serum_creatinine","DEATH_EVENT")]
AIC_var = Heart[c("age","creatinine_phosphokinase","time", "ejection_fraction", "sex",
                        "serum_sodium","serum_creatinine","DEATH_EVENT")]

k.optm <- best.k.acc(AIC_var_train,AIC_var_test,AIC_var_train$DEATH_EVENT,
                        AIC_var_test$DEATH_EVENT, 30)
```

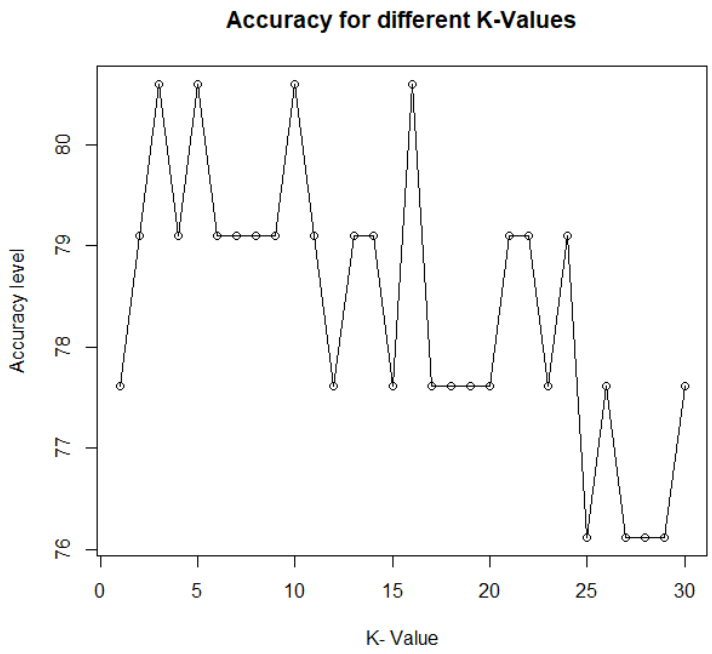
```
##
## Max Accuracy:  76.1194
## Best K:  5 6 8
##     y\_test
## pred  0  1
##    0 37 13
##    1  3 14

plot(k.optm, type="o", xlab="K- Value", ylab="Accuracy level",
        main="Accuracy for different K-Values")
```



To see if the model benefits from a further reduction of predictors, the model obtained from BIC best subset selection is used. Below, the confusion matrix and the behavior of the accuracy for various values for k can be found. Again, there is always some variation due to the starting condition of the algorithm, but it can be clearly seen that the BIC model improves the results even further as it reaches a test accuracy of above 80%. The best accuracy was obtained with values between 3 and 10 which gives a score of just above 80.5%. Compared to the full model and even the AIC model, using the four predictors improves the results in every aspect. The sensitivity increases to 85% and the specificity to 78.7%.

```
# KNN (BIC Best Model):
BIC_var_train = train[c("age","serum_creatinine","time", "ejection_fraction", "DEATH_EVENT")]
BIC_var_test = test[c("age","serum_creatinine","time", "ejection_fraction", "DEATH_EVENT")]
BIC_var = Heart[c("age","serum_creatinine","time", "ejection_fraction", "DEATH_EVENT")]

k.optm <- best.k.acc(BIC_var_train, BIC_var_test,BIC_var_train$DEATH_EVENT,
                     BIC_var_test$DEATH_EVENT, 30)
```

```
##
## Max Accuracy:  80.59701
## Best K:  3 5 10
##     y\_test
## pred  0  1
##    0 37 10
##    1  3 17

plot(k.optm, type="o", xlab="K- Value",ylab="Accuracy level",
     main="Accuracy for different K-Values")
```



To conclude this section about KNN, the best model is looked at closer by plotting the ROC curve for the K values 5, 10, 15, and 20. The following code reports the lines for K = 5, and then similar commands have been ran for the other K values.

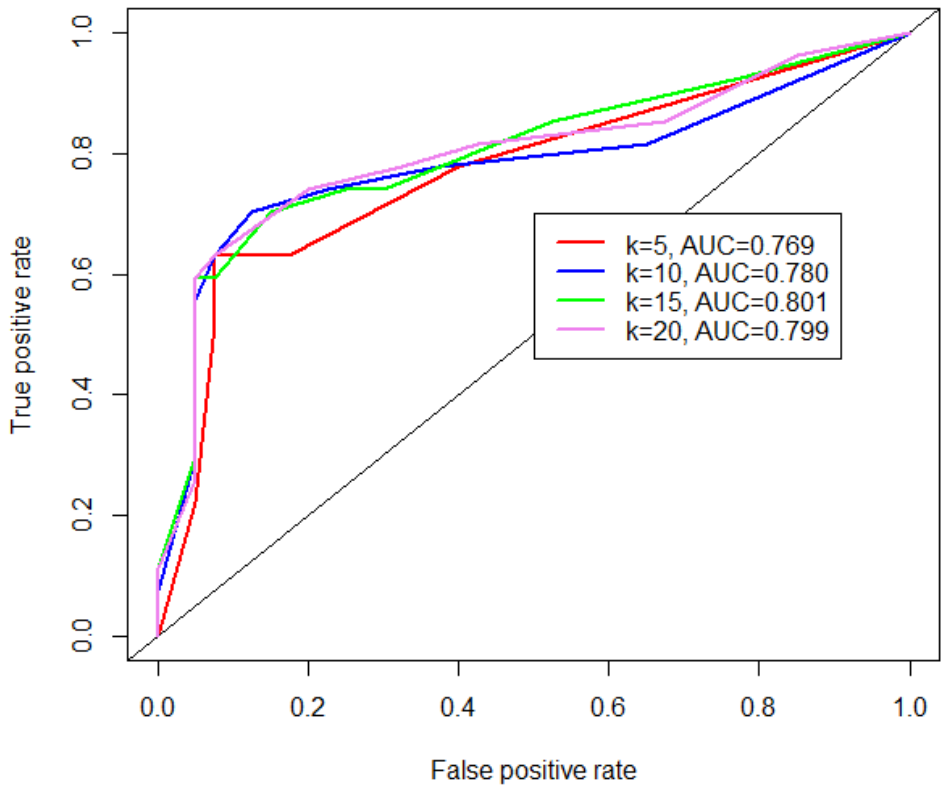
```
kNN.mod.5 <- knn(train=BIC_var_train, test=BIC_var_test, cl=BIC_var_train$DEATH_EVENT,
                 k=5, prob=TRUE)
prob <- attr(kNN.mod.5, 'prob')
prob <- ifelse(kNN.mod.5 == "0", 1-prob, prob)
pred.knn.5 <- prediction(prob, BIC_var_test$DEATH_EVENT)
perf.knn.5 <- performance(pred.knn.5, measure='tpr', x.measure='fpr')

auc <- performance(pred.knn, measure='auc')
auc@y.values[[1]]

## [1] 0.7694444
```

```
# Test accuracy:
kNN.mod.5 <- knn(train=BIC_var_train, test=BIC_var_test, cl=BIC_var_train$DEATH_EVENT, k=5)
tab = table(kNN.mod.5,BIC_var_test$DEATH_EVENT)
accuracy(tab)

## [1] 80.59701
```

The resulting test accuracies are: 80.60% (K = 5, K = 10), 77.61% (K = 15) and 79.11% (K = 20).

The final plot of all the ROC Curves and the AUC is the following:

```
plot(perf.knn.5, lwd= 2, lty=1, col="red")
plot(perf.knn.10, lwd= 2, lty=1, col="blue", add=TRUE)
plot(perf.knn.15, lwd= 2, lty=1, col="green",  add=TRUE)
plot(perf.knn.20, lwd= 2, lty=1, col="violet", add=TRUE)
abline(a=0,b=1)
legend(x=0.5,y=0.7, legend=c("k=5, AUC=0.769","k=10, AUC=0.780","k=15, AUC=0.801",
            "k=20, AUC=0.799"), col=c("red","blue","green","violet"), lty=1, lwd=2)
```



3.4 Linear Discriminant Analysis

Linear Discriminant Analysis is a classification method that approximates the Bayes classifier.
It models the distribution of the predictors **X** separately in each of the response classes and gives as a result the posterior probability that an observation **X** = **x** belongs to a certain class, given the predictors' values for that observation.
LDA is based on two assumptions:

1. The data belonging to the same class G_j are normally distributed:

$$X \mid G_j \sim N(\mu_j, \Sigma_j) \tag{1}$$

2. The covariance matrix Σ is the same for all the classes:

$$\Sigma_j = \Sigma \tag{2}$$

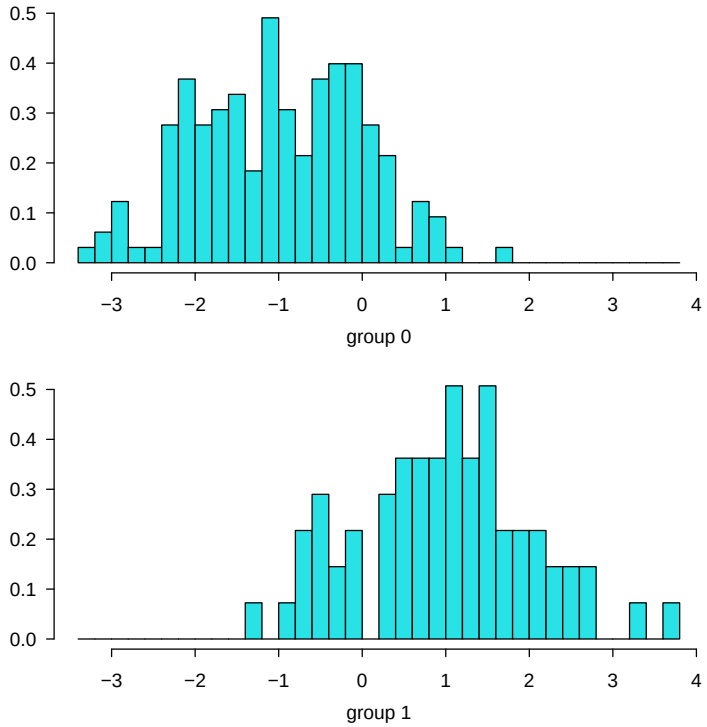
Therefore, LDA makes stronger assumptions on the predictors compared to the Logistic Regression.
However, it works better than the Logistic Regression if the classes are well separated, or the sample size is small and the assumptions are satisfied, or the response variable has more than 2 classes.
In our case we have two classes and the predictors are not well separated (as shown in the pairs-plot of Section 2), but we have a small sample size and the LDA assumptions (1) and (2) seem to be satisfied.

Analysis

We first ran the analysis with the full model and then with the BIC best model, in order to compare the results. For each model, we computed the training accuracy and we used the ROC curve to select the best threshold. We then computed the test accuracy using both the default threshold (0.5) and the best threshold selected by the ROC curve.

```
# Useful function:
library(MASS)
accuracy <- function(y, predicted, threshold){
   a1 = sum(rep(1, length(y[predicted$posterior[,2]>= threshold])) ==
            y[predicted$posterior[,2] >= threshold])
   a0 = sum(rep(0, length(y[predicted$posterior[,2]< threshold])) ==
            y[predicted$posterior[,2] < threshold])
   return((a1 + a0) / length(y))
}
# Full Model:
lda.fit <- lda(DEATH_EVENT~.,data=train)
lda.fit

## Call:
## lda(DEATH_EVENT ~ ., data = train)
```

```
##
## Prior probabilities of groups:
##         0         1
## 0.7025862 0.2974138
##
## Group means:
##        age  anaemia1 creatinine_phosphokinase diabetes1 ejection_fraction
## 0 58.50920 0.4171779                 493.6810 0.4171779          40.69939
## 1 66.96619 0.4492754                 654.4638 0.4057971          33.30435
##   high_blood_pressure1 platelets serum_creatinine serum_sodium      sex1  smoking1
## 0            0.3128834  263221.5         1.205951     137.3067 0.6564417 0.2822086
## 1            0.4347826  264264.1         1.833043     135.3188 0.6086957 0.3043478
##        time
## 0 159.73620
## 1  69.17391
##
## Coefficients of linear discriminants:
##                                   LD1
## age                       3.287122e-02
## anaemia1                 -1.031906e-01
## creatinine_phosphokinase  2.511448e-04
## diabetes1                -1.107500e-02
## ejection_fraction        -4.990861e-02
## high_blood_pressure1      1.008912e-01
## platelets                 6.685597e-07
## serum_creatinine          2.263989e-01
## serum_sodium             -3.634713e-02
## sex1                     -4.441681e-01
## smoking1                  9.650835e-02
## time                     -1.143882e-02

plot(lda.fit)
```



From the graph, we can see that the LDA assumptions (1) and (2) seem to be sufficiently satisfied.

```
# Training Accuracy:
lda.train.pred <- predict(lda.fit, train)
mean(lda.train.pred$class==train$DEATH_EVENT)

## [1] 0.8793103

# ROC Curve:
lda.roc.out <- roc(train$DEATH_EVENT, lda.train.pred$posterior[,2], levels=c(0,1))
auc(lda.roc.out)

## Area under the curve: 0.9133

coords(lda.roc.out, "best")

##   threshold specificity sensitivity
## 1 0.4338429   0.9202454   0.7826087

# Test Accuracy:
lda.pred <- predict(lda.fit, test)
mean(lda.pred$class==test$DEATH_EVENT)

## [1] 0.7462687

best_threshold = 0.43
accuracy(test$DEATH_EVENT, lda.pred, best_threshold)

## [1] 0.761194
```

```r
# BIC Best Model:
lda.reduced <- lda(train$DEATH_EVENT~age+time+ejection_fraction+serum_creatinine,
                   data=train)
lda.reduced
```

```
## Call:
## lda(train$DEATH_EVENT ~ age + time + ejection_fraction + serum_creatinine,
##     data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.7025862 0.2974138
##
## Group means:
##        age      time ejection_fraction serum_creatinine
## 0 58.50920 159.73620          40.69939         1.205951
## 1 66.96619  69.17391          33.30435         1.833043
##
## Coefficients of linear discriminants:
##                            LD1
## age                 0.03156625
## time               -0.01176881
## ejection_fraction -0.04945959
## serum_creatinine    0.26476426
```

```r
# Training Accuracy:
lda.train.pred <- predict(lda.reduced, train)
mean(lda.train.pred$class==train$DEATH_EVENT)
```

```
## [1] 0.8491379
```

```r
# ROC Curve:
lda.roc.out <- roc(train$DEATH_EVENT, lda.train.pred$posterior[,2], levels=c(0,1))
auc(lda.roc.out)
```

```
## Area under the curve: 0.9066
```

```r
coords(lda.roc.out, "best")
```

```
##   threshold specificity sensitivity
## 1 0.2570787    0.791411   0.8695652
```

```r
# Test Accuracy:
lda.pred <- predict(lda.reduced, test)
mean(lda.pred$class==test$DEATH_EVENT)

## [1] 0.7761194

# Test Accuracy with best threshold:
best_threshold = 0.26
accuracy(test$DEATH_EVENT, lda.pred, best_threshold)

## [1] 0.7462687
```

Results

	Training Accuracy	Best Threshold	Test Accuracy (default thr.)	Test Accuracy (best thr.)
Full Model	88%	0.43	75%	76%
BIC Best Model	85%	0.26	78%	75%

Table 9: LDA results

As we can see from table 9, the full model has a better training accuracy than the BIC best model. However, the test accuracy of the BIC best model with the default threshold of 0.5 is better compared to the full model's one and the test accuracy with the models' best threshold is almost the same for the two models, Therefore, the BIC best model is simpler than the full model (it contains only 4 predictors) and has a similar performance, so we can conclude that it is preferable to the full model.

3.5 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis is another classification method that approximates the Bayes classifier. Contrary to LDA, it doesn't make the assumption that the covariates have the same covariance matrix Σ in every class. This is a very strong assumption and can be quite restrictive, therefore QDA is more flexible compared to LDA. However, there need to be many observations in each class in order for QDA to perform well. Since our dataset is not balanced and it has many more observations associated to class 0 compared to class 1, we expect that QDA's performance on the data set will be a bit worse compared to LDA's performance.

Analysis

The analysis was carried out with computations analogous to those of LDA, both on the full model and the BIC best model.

Results

	Training Accuracy	Best Threshold	Test Accuracy (default thr.)	Test Accuracy (best thr.)
Full Model	83%	0.19	64%	67%
BIC Best Model	83%	0.28	75%	76%

Table 10: QDA results

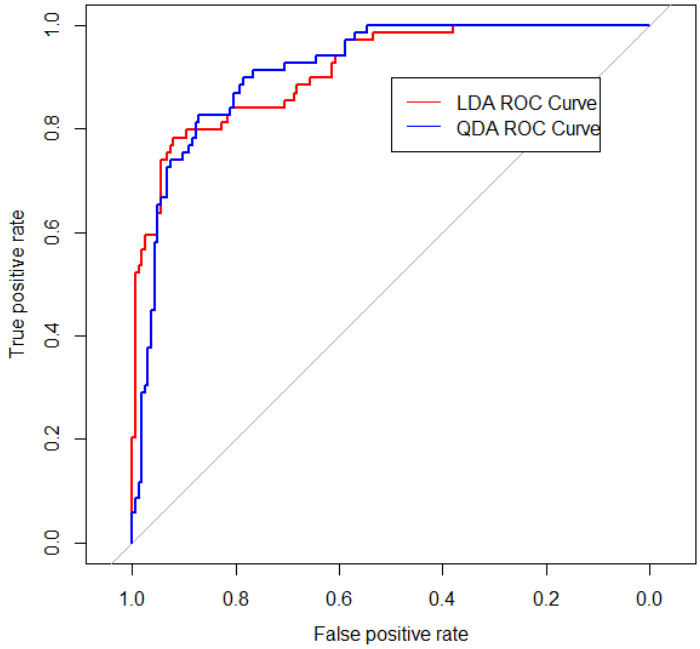
As we can see from table 10, while the training accuracy is the same for both the full and reduced models, we can clearly see an improvement in the test accuracy of the BIC best model compared to the test accuracy of the full model. In particular, the test accuracy corresponding to a threshold of 0.28 is the best test accuracy obtained using QDA.

As expected, the training accuracy of QDA is smaller than the training accuracy of LDA for both models. This is probably due to the fact that our dataset is unbalanced and there are just a few observations associated to a response variable equals to 1, i.e. dead patients. Nevertheless, the QDA test accuracy of the BIC best model is comparable with the LDA test accuracy of both full model and BIC best model.

LDA and QDA comparison

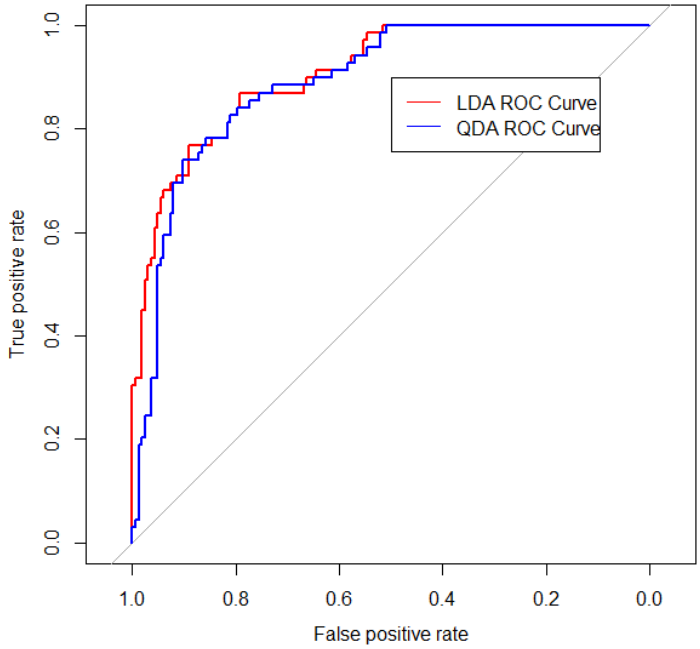
In order to better compare the LDA and QDA methods, we plotted together the ROC curves computed on the training data using the two methods, first on the full model and secondly on the BIC best model.

```
# Full model
lda.fit <- lda(DEATH_EVENT~.,data=train)
lda.train.pred <- predict(lda.fit, train)
lda.roc.out <- roc(train$DEATH_EVENT, lda.train.pred$posterior[,2], levels=c(0,1))

qda.fit <- qda(DEATH_EVENT~.,data=train)
qda.train.pred <- predict(qda.fit, train)
qda.roc.out <- roc(train$DEATH_EVENT, qda.train.pred$posterior[,2], levels=c(0,1))

plot(lda.roc.out, col = "red", xlab="False positive rate", ylab="True positive rate")
plot(qda.roc.out, add = TRUE, col = "blue")
legend(x=0.5,y=.9,legend=c("LDA ROC Curve", "QDA ROC Curve"), col=c("red","blue"), lty=1)
```



From this graph we can see that the two ROC curves are very similar. Furthermore, both the two areas under the curves can be approximated to 0.91.

```
# BIC best model
lda.reduced <- lda(train$DEATH_EVENT~age+time+ejection_fraction+serum_creatinine,
                   data=train)
lda.train.pred <- predict(lda.reduced, train)
lda.roc.out <- roc(train$DEATH_EVENT, lda.train.pred$posterior[,2], levels=c(0,1))

qda.reduced <- qda(train$DEATH_EVENT~age+time+ejection_fraction+serum_creatinine,
                   data=train)
qda.train.pred <- predict(qda.reduced, train)
qda.roc.out <- roc(train$DEATH_EVENT, qda.train.pred$posterior[,2], levels=c(0,1))

plot(lda.roc.out, col = "red", xlab="False positive rate", ylab="True positive rate")
plot(qda.roc.out, add = TRUE, col = "blue")
legend(x=0.5,y=.9,legend=c("LDA ROC Curve", "QDA ROC Curve"), col=c("red","blue"), lty=1)
```



From this graph we can see that the two ROC curves almost overlap. Nevertheless, the area under the curve for LDA (approximately 0.91) is slightly larger compared to the area under the curve for QDA (approximately 0.89). Therefore, we can conclude that the LDA method has a slightly better performance compared to the QDA method.

4 Conclusions

In table 11 we reported the training and test accuracies for the models computed using the default and best thresholds. The K-Nearest Neighbors model was analyzed in a different way and therefore we report just the test accuracy with the default threshold.

	Training Accuracy	**Test Accuracy (default thr.)**	**Test Accuracy (best thr.)**
BIC best model	85%	78%	73%
AIC best model	88%	75%	76%
Ridge Regression	87%	73%	78%
Lasso Regression	87%	76%	78%
KNN - Full model	-	61%	-
KNN - BIC best model	-	81%	-
KNN - AIC best model	-	76%	-
LDA - Full model	88%	75%	76%
LDA - BIC best model	85%	78%	75%
QDA - Full model	83%	64%	67%
QDA - BIC best model	83%	75%	76%

Table 11: Models' accuracy.

We have chosen as best model, the model for which we obtained the best test accuracy, i.e. the KNN BIC best model with K = 5, which has a test accuracy of 81%. For this model, we computed the confusion matrix on the test set as follows:

```
# KNN (BIC Best Model):
BIC_var_train= train[c("age","serum_creatinine","time", "ejection_fraction",
                       "DEATH_EVENT")]
BIC_var_test= test[c("age","serum_creatinine","time", "ejection_fraction", "DEATH_EVENT")]

pred <- knn(train=BIC_var_train, test=BIC_var_test, cl=train$DEATH_EVENT, k=5)
confusion_matrix <- table(pred, test$DEATH_EVENT)
confusion_matrix

##
## pred  0  1
##    0 37 10
##    1  3 17

TN <- confusion_matrix[1]
TP <- confusion_matrix[4]
FP <- confusion_matrix[2]
FN <- confusion_matrix[3]
P <- sum(Heart$DEATH_EVENT==1)
N <- sum(Heart$DEATH_EVENT==0)
P_ <- TP + FP
N_ <- TN + FN

false.negative.rate <- FN/P
false.negative.rate

## [1] 0.1041667

false.positive.rate <- FP/N
false.positive.rate

## [1] 0.01477833

true.positive.rate <- TP/P
true.positive.rate

## [1] 0.1770833

positive.predicted.value <- TP/P_
positive.predicted.value

## [1] 0.85
```

```
negative.predicitve.value <- TN/N_
negative.predicitve.value

## [1] 0.787234
```

Table 12 shows the metrics derived from the confusion matrices computed with the default threshold.

FN Rate	FP Rate	TP Rate	Positive Predicted Value	Negative Predicted Value
0.10	0.01	0.18	0.85	0.79

Table 12: Confusion Matrix

In conclusion KNN is the model having the best fit on unseen data and its error rates are more than acceptable for our problem, considering that medical dataset are often difficult to predict.
In particular, within the scope of medicine, our dataset is a "lucky" one because it is unbalanced, but the sizes of two classes are not extremely different. However the total sample size is quite small and this contributes to worsen the accuracy.
In addition, the dataset doesn't describe an heterogeneous population, but just patients that had left ventricular systolic dysfunction and had previous heart failures. Therefore our conclusions don't apply to any patient in general, but just to ones that fall in this specific category.
Nevertheless, the main goal of this report is giving insights about the relationships between clinical features and death, wereas the predictive power of the models (values of the test accuracies) must be put in relation with the fact that we're including in the analysis the variable Time.
Moreover, BIC, AIC and Ridge coefficients are coherent with the descriptions reported in the Data Exploration section. In fact, the signs of the coefficients reflect the positive or negative influences of the clinical features on the response variable. For example, low Ejection Fraction or high levels of Serum Creatinine in the blood reasonably increase the probability of death. Therefore, the coefficients of these two feature are, respectively, negative and positive.
On the other hand, Ridge, LDA and QDA models are much more difficult to interpret and are not particularly suitable for insights.