Statistical learning project

Sercan Albut, Farid Rustamli, Emanuele Zangrando

12/07/2021

Contents

Import libraries and useful functions

```
knitr::opts_chunk$set(echo = TRUE)
library(MASS);
library(pROC);
library(dplyr);
library(xtable);
library(glmnet);
library(corrplot);
library(class);
library(randomForest);
library(caret);

dot<-function(x,y){
  d = 0
  for(i in 1:length(x)){
    d=d+x[i]*y[i];
  }
  return(d)
}

sigmoid<-function(x,weights){
  y = c(1,x);
  result=exp(dot(y,weights))/(1+exp(dot(y,weights)));
  return(result)
}

predglm<-function(mod,x){
```

```
  w = coefficients(mod);
  if(length(x)==length(w)-1){
    return(sigmoid(x,w))
  }else{
    X = split(x,1:floor((2*length(x))/length(w))) ;
    fits = c()
    for(i in X){
      fits=c(fits,sigmoid(i,w));
    }
    return(fits)
  }

}


hard_classif<-function(x,threshold){     #given the probability preditions and a
  #threshold, returns an hard binary classification
  hard_classification = c()
  for (i in 1:length(x)){
    if (x[i]>threshold){
      hard_classification = c(hard_classification,1)

    } else{
      hard_classification = c(hard_classification,0)
    }


  }
  return(hard_classification)
}
```

Heart failure dataset

For this project we decided to use the "Heart failure" dataset which is available on Kaggle. We choose this dataset because all the group members are interested in medical data and also because of its size: it is low dimensional and there are also a handful number of observations, in this way we were able to focus on the statistical analysis. This data is a collection of the observation of 13 features on 299 patients that suffered from a heart failure. Half of the features are binary classification such as sex,smoking,diabetes etc. One of the variables is called "time" and it represent the time for which the patient has been at the hospital under observation. This feature would have been a valuable information if it would have been a time series of a patient's observations until the death or the return home. This variable can be possibly used to do a survival analysis, but we did not do that. This is why we got rid of it, because it had no value in our analysis. The only cleaning we did was just to convert the platelets measure (it was platelets/L) into Kiloplatelets/L. We have also checked for missing observations and they are not present.

The variables are the following:

```
data=read.csv(file = "/Users/emanu/Downloads/heart_failure_clinical_records_dataset.csv");
data$platelets = data$platelets / 1000;  #conversion in kiloplatelets
attach(data);
data = subset(data,select=-time);
data = data.frame(data);
data_num = data;
DEATH_EVENT[DEATH_EVENT==1]='yes';
DEATH_EVENT[DEATH_EVENT==0]='no';
diabetes[diabetes==1]='yes';
diabetes[diabetes==0]='no';
sex[sex==1]='man';
sex[sex==0]='woman';
high_blood_pressure[high_blood_pressure==1]='yes';
high_blood_pressure[high_blood_pressure==0]='no';
smoking[smoking==1]='yes';
smoking[smoking==0]='no';
anaemia[anaemia==1]='yes';
anaemia[anaemia==0]='no';
x=subset(data,select=-DEATH_EVENT);
y=data$DEATH_EVENT;
glimpse(data)
```

```
## Rows: 299
## Columns: 12
## $ age                       <dbl> 75, 55, 65, 50, 65, 90, 75, 60, 65, 80, 75...
## $ anaemia                   <int> 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, ...
## $ creatinine_phosphokinase <int> 582, 7861, 146, 111, 160, 47, 246, 315, 15...
## $ diabetes                  <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...
## $ ejection_fraction         <int> 20, 38, 20, 20, 20, 40, 15, 60, 65, 35, 38...
## $ high_blood_pressure       <int> 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, ...
## $ platelets                 <dbl> 265.000, 263.358, 162.000, 210.000, 327.00...
## $ serum_creatinine          <dbl> 1.90, 1.10, 1.30, 1.90, 2.70, 2.10, 1.20, ...
## $ serum_sodium              <int> 130, 136, 129, 137, 116, 132, 137, 131, 13...
## $ sex                       <int> 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, ...
## $ smoking                   <int> 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, ...
## $ DEATH_EVENT               <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
```

Some of the numerical variables are binary and they can be interpreted as categorical (sex=1 stands for men, and anaemia, high_blood_pressure, diabetes, smoking and death_event = 1 means, the patient has it, does it and is deceased).In particular there are 6 binary variables and 6 numerical ones. Just as a reference, these

are the normal levels of the least common variables in the dataset.

variable	normal range	brief description
creatinine phosphokinase	10 to 120 mcg/L	enzime present in muscles in part responsible for muscles contraction.
platelets	150-400 kplatelets/mL	platelets concentration in the blood
ejection fraction	45%-75%	percentage of blood leaving the heart at each contraction
serum creatinine	0.5-1.2 mg/dL	concentration of creatinine in the blood
serum sodium	135-145 mEq/L	concentration of sodium in the blood

This dataset is also pretty unbalanced, more or less $\frac{1}{3}$ of the observations lead to a death event.

Q: What does it mean when your CPK(Creatine phosphokinase) level is high? A: When the total CPK level is very high, it most often means there has been injury or stress to muscle tissue, the heart, or the brain. Muscle tissue injury is most likely. When a muscle is damaged, CPK leaks into the bloodstream,

Q: What is the danger level of platelet count? A: The platelet count is a test that determines the number of platelets in your sample of blood. When there is an injury to a blood vessel or tissue and bleeding begins, platelets help stop bleeding. Dangerous internal bleeding can occur when your platelet count falls below 10,000 platelets per microliter.

Q: What is Ejection Fraction and what does it measures? A: Ejection fraction (EF) refers to how well your left ventricle (or right ventricle) pumps blood with each heart beat. Most times, EF refers to the amount of blood being pumped out of the left ventricle each time it contracts. The left ventricle is the heart's main pumping chamber. EF is expressed as a percentage.

Q: What does serum creatinine measures? A: A creatinine test is a measure of how well your kidneys are performing their job of filtering waste from your blood. Creatinine is a chemical compound left over from energy-producing processes in your muscles. Healthy kidneys filter creatinine out of the blood. High serum creatinine levels create the symptoms of muscle cramps, chest pain, nausea and high blood pressure.

Q: What does serum sodium measures? A: Sodium is an electrolyte present in all body fluids and is vital to normal body function, including nerve and muscle function. This test measures the level of sodium in the blood and/or urine.

Data is also unbalanced, more or less $\frac{1}{3}$ of the observations lead to a death event.

By looking at this data one can immediately notice that the $y = Death\_event$ variable can naturally be interpreted as a target variable. This is the first question one can try to answer with this data: is it actually possible to predict the death risk given the features of the patient? if yes, is it possible to extract a subset of the features that is sufficient to be able to predict the death occurrence? Answering these questions could be interesting from a medical point of view: if one is able to extract specified features from the patients and if these are useful for predicting the risk of death, then one can act in advance on patients with a high predicted death risk. We also noticed that half of the variables are pretty much of immediate information collection (the ones that are binary: age,sex,high blood pressure,anemia,diabetes and smoking) while the others require examinations. Hopefully variable selection will tell us that important variables are the ones easily collectible, this could also improve the speed in assessing the risk.

With this tasks in mind we organized the work as suggested. We started with exploratory data analysis just to understand the main features of this dataset. After this step we tried different methods for variable selection and we compared them. Finally, we tried different models for predicting the death probability of patients conditioned on the features. For all this models we made a brief analysis of the results and we compared them. The project ends with a brief conclusion section.

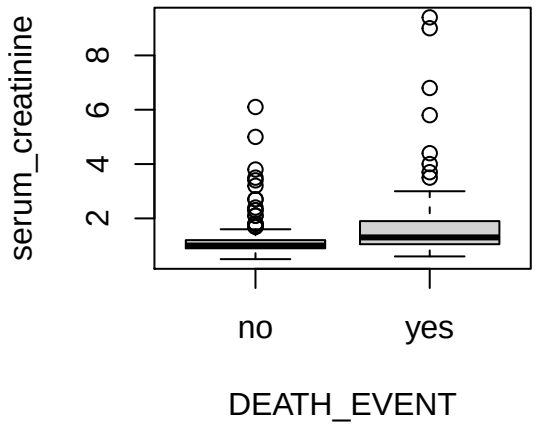
Exploratory data analysis

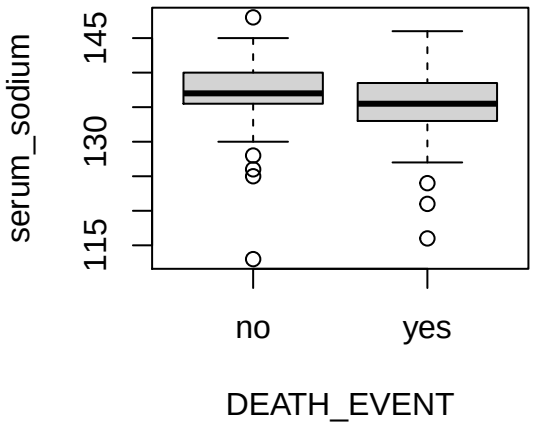
The first thing we did was to look at the distributions of the continuous variables conditioned on the death event using boxplots. By doing so we would like to detect differences in the distributions of specific variables between the people that passed away and the people that survived, and if we find a difference between them, this indicates that this specific variable might be useful for classification. We also included a table with the summary statistics of the continuous variables.

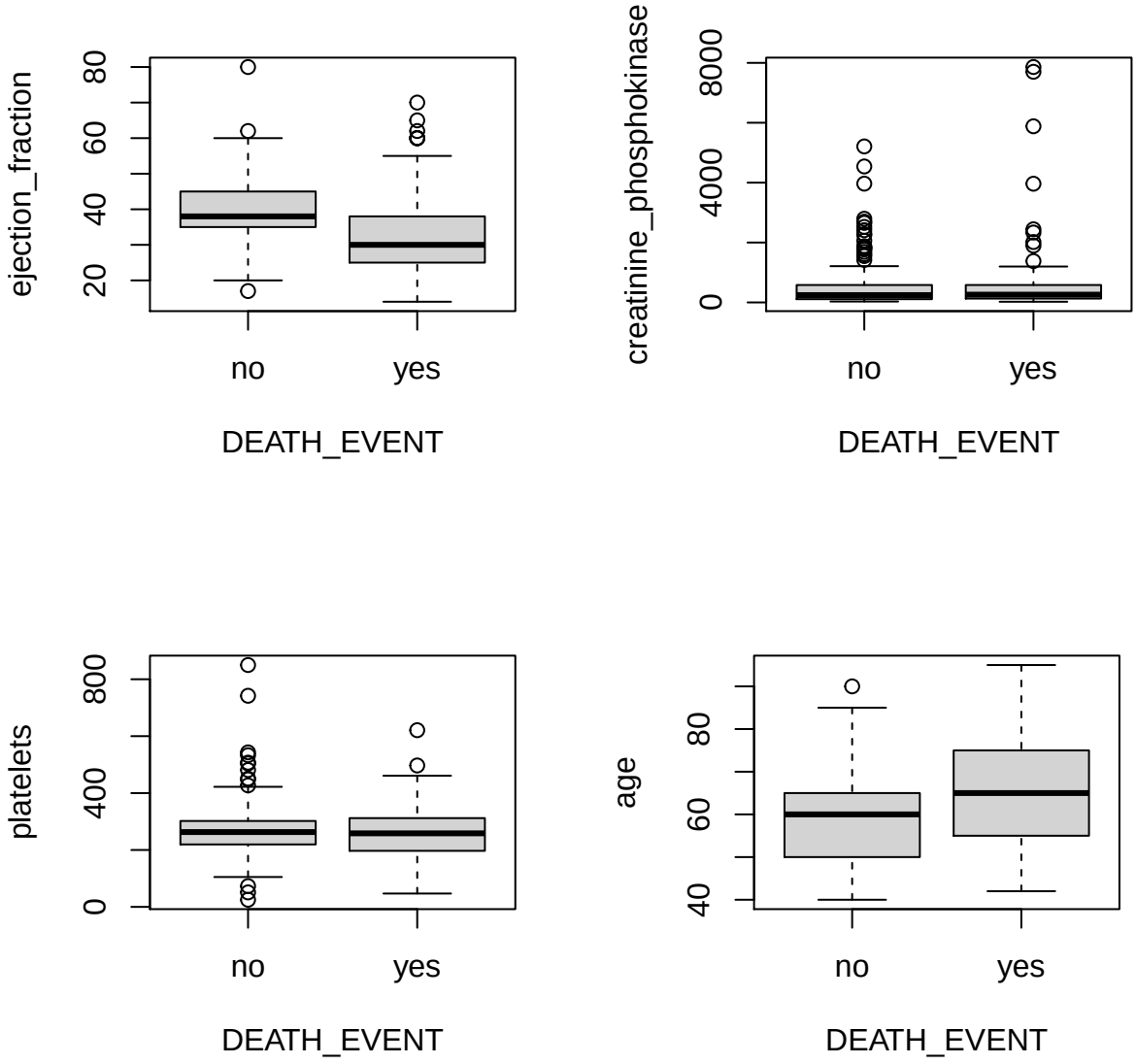
```
cont.vars = subset(data,select=-c(sex,anaemia,high_blood_pressure,
                                  diabetes,smoking,DEATH_EVENT));
summary(cont.vars)
```

```
##       age        creatinine_phosphokinase ejection_fraction   platelets
##  Min.   :40.00   Min.   :  23.0           Min.   :14.00     Min.   : 25.1
##  1st Qu.:51.00   1st Qu.: 116.5           1st Qu.:30.00     1st Qu.:212.5
##  Median :60.00   Median : 250.0           Median :38.00     Median :262.0
##  Mean   :60.83   Mean   : 581.8           Mean   :38.08     Mean   :263.4
##  3rd Qu.:70.00   3rd Qu.: 582.0           3rd Qu.:45.00     3rd Qu.:303.5
##  Max.   :95.00   Max.   :7861.0           Max.   :80.00     Max.   :850.0
##  serum_creatinine  serum_sodium
##  Min.   :0.500    Min.   :113.0
##  1st Qu.:0.900    1st Qu.:134.0
##  Median :1.100    Median :137.0
##  Mean   :1.394    Mean   :136.6
##  3rd Qu.:1.400    3rd Qu.:140.0
##  Max.   :9.400    Max.   :148.0
```

Already from this summary statistic of the data we noticed that there are some strange values, in particular some values of creatinine-phosphokinase seems too high from a medical point of view (patient 2 and patient 61). We did not remove them (even thought some of them are probably measuration errors), but it's better to be aware of them because they can weight highly in the wrong direction of our analysis since we have only 299 observation.





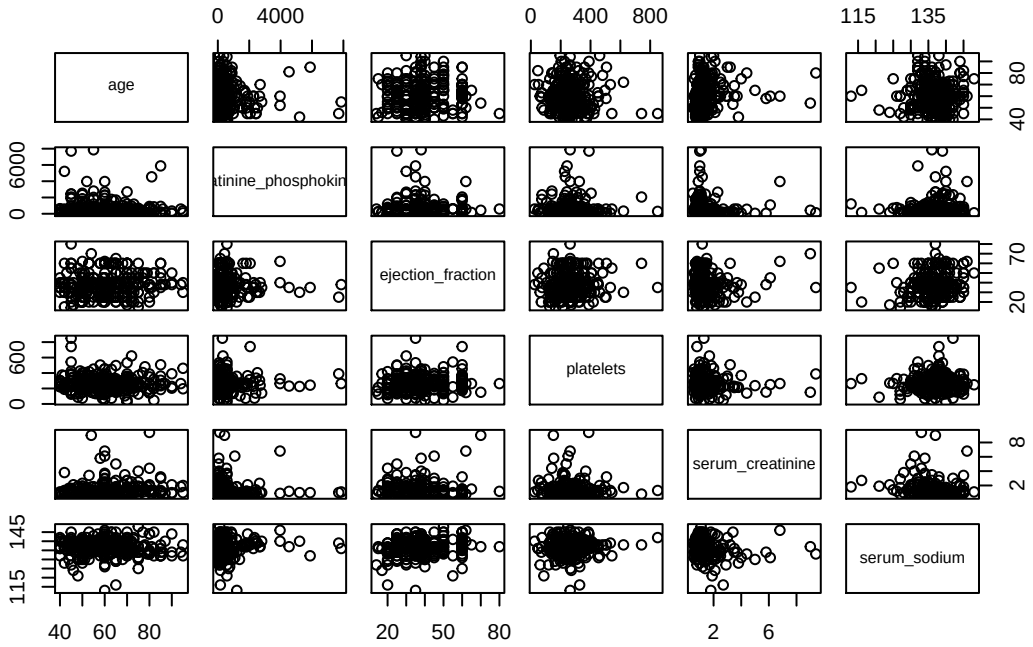


We also noticed the presence of other possible outliers, in particular also some values for serum creatinine are strange (patients 10, 53,132 218).

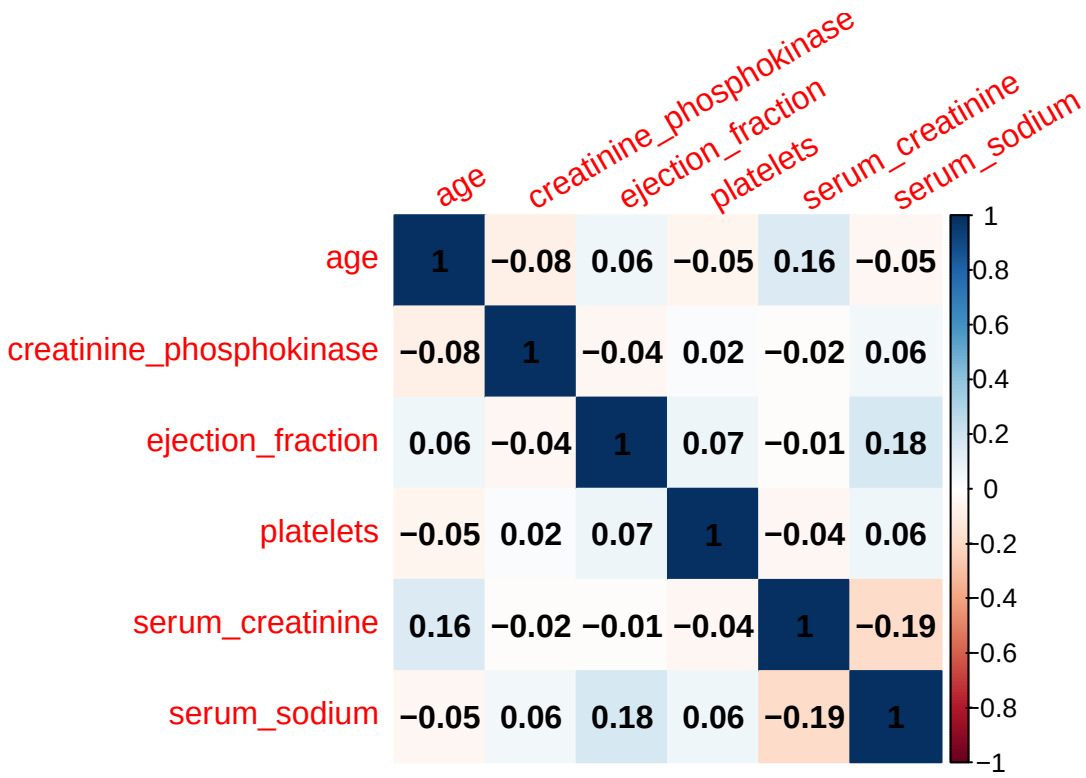
From this plots we can see that the distribution of some variables seem to be influenced with the death event. In particular we can notice some differences in ejection fraction, serum creatinine, serum sodium and age. Since these differences seem minimal, we don't expect to obtain a good classifier using single variables. In the next section we will try to understand which variables jointly are able to predict death.

We also expect some variables to be correlated, it's better to check that before fitting some model. For continuous variables we checked the correlation plots and the correlation matrix to get a visual idea.

```
plot(subset(data,select = -c(smoking,DEATH_EVENT,sex,high_blood_pressure,
                             anaemia,diabetes)))
```




```
cor.mat=cor(subset(data,select = -c(smoking,DEATH_EVENT,sex,high_blood_pressure,
                                    anaemia,diabetes)));
corrplot(cor.mat, method="color",addCoef.col = "black",tl.srt=30)
```



From this plots we noticed that there are not so strong correlations between regressors. The highest

ones are between serum_sodium and serum creatinine (-0.19), ejection_fraction and serum_sodium (0.18), serum_creatinine and age (0.16).

For binary variables instead we produced a table containing all the conditioned proportions.

Conditioned ratios	Full_sample	Dead patients	Survived patients
sex (women)	0.3511706	0.3541667	0.3497537
sex (men)	0.6488294	0.6458333	0.6502463
anaemia (no)	0.5685619	0.5208333	0.591133
anaemia (yes)	0.4314381	0.4791667	0.408867
high_blood_press (no)	0.6488294	0.59375	0.6748768
high_blood_press (yes)	0.3511706	0.40625	0.3251232
smoking (no)	0.6789298	0.6875	0.6748768
smoking (yes)	0.3210702	0.3125	0.3251232
diabetes (no)	0.5819398	0.5833333	0.5812808
diabetes(yes)	0.4180602	0.4166667	0.4187192
Death_event (no)	0.6789298	0	1
Death_event (yes)	0.3210702	1	0

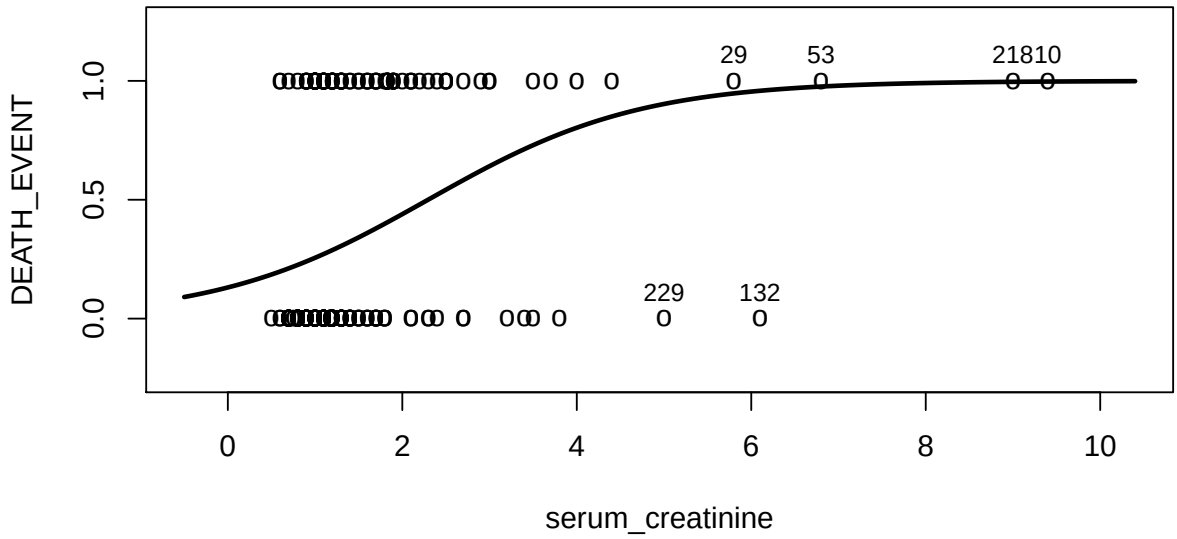
From this last table we can see that the variables for which conditioning on the death event seems to matter the most are high_blood_pressure and anaemia, so we expect them to be the most informative binary variables.

We looked also at the distributions of continuous variables conditioned on the other binary variables, but they were not visually informative.

After that we tried to fit logistic regression models for single predictors. Probably these models will not be useful (often a medical parameter is predicting a risk when it's either too low or too high), but they can be useful for interpreting each variable's role. In the produced plots we also evidenced the points that in the boxplots seemed to be outliers.

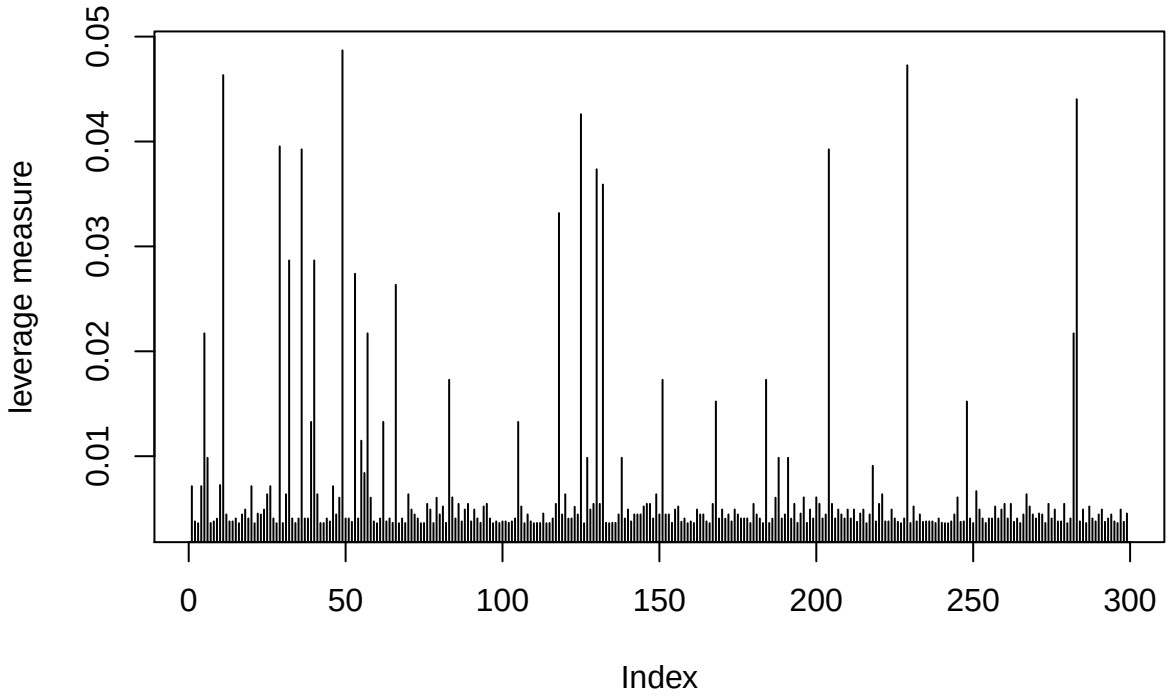
The first variable we looked at was serum_creatinine. We fitted a logistic regression model and then looked at the significance of the parameters. Just for the purpose of comparing we showed also the Z-test automatically computed in the glm function, but for the reliability we choose to use the deviance test.

```
g = glm(DEATH_EVENT~serum_creatinine,data=data,family=binomial);
xgrid = seq(min(serum_creatinine)-1,max(serum_creatinine)+1,length=1000);
ypred<-predglm(g,xgrid);

r = data[serum_creatinine>4.8,];
plot(xgrid,ypred,type='l',xlab='serum_creatinine',
     ylab='DEATH_EVENT',xlim=c(min(serum_creatinine)-1,
                               max(serum_creatinine)+1),ylim=c(-0.25,1.25)   ,
     col ='black',lwd = 2.5)

points(data$serum_creatinine,data$DEATH_EVENT,pch='o')
text(r$DEATH_EVENT~r$serum_creatinine
     ,data=r,labels=rownames(r),pos=3,cex=0.8)
```



From this first plot we can already see that this single variable is not able to discriminate sufficiently the death event. The points we evidence are the ones that seemed to high in the boxplots, and in fact they are isolated from the big group of observations (patients 132 and 229 are also outliers for this model). Visually they don't seem to be leverage points, but just to be sure we checked the leverage of those points.

```
plot(hatvalues(g), type = 'h',ylab='leverage measure')
```



The leverage plot up here shows that there are some points with a leverage higher than the average but they don't seem to be a problem. Then we checked for the significance of the parameters, confronting the Z and the deviance test.

```
summary(g)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ serum_creatinine, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5213  -0.7966  -0.7417   1.2644   1.7990
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.8917     0.2939  -6.438 1.21e-10 ***
## serum_creatinine   0.8242     0.1972   4.180 2.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 375.35  on 298  degrees of freedom
## Residual deviance: 347.25  on 297  degrees of freedom
## AIC: 351.25
```

```
##
## Number of Fisher Scoring iterations: 5
```

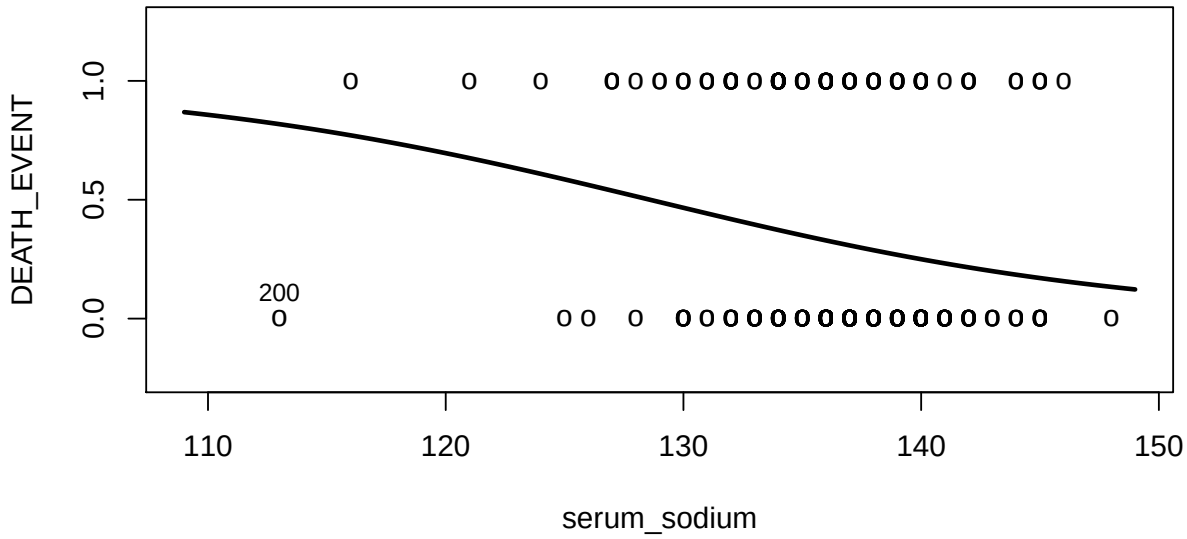
```
anova(g,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: DEATH_EVENT
##
## Terms added sequentially (first to last)
##
##
##                   Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                298     375.35
## serum_creatinine  1   28.097       297     347.25 1.154e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

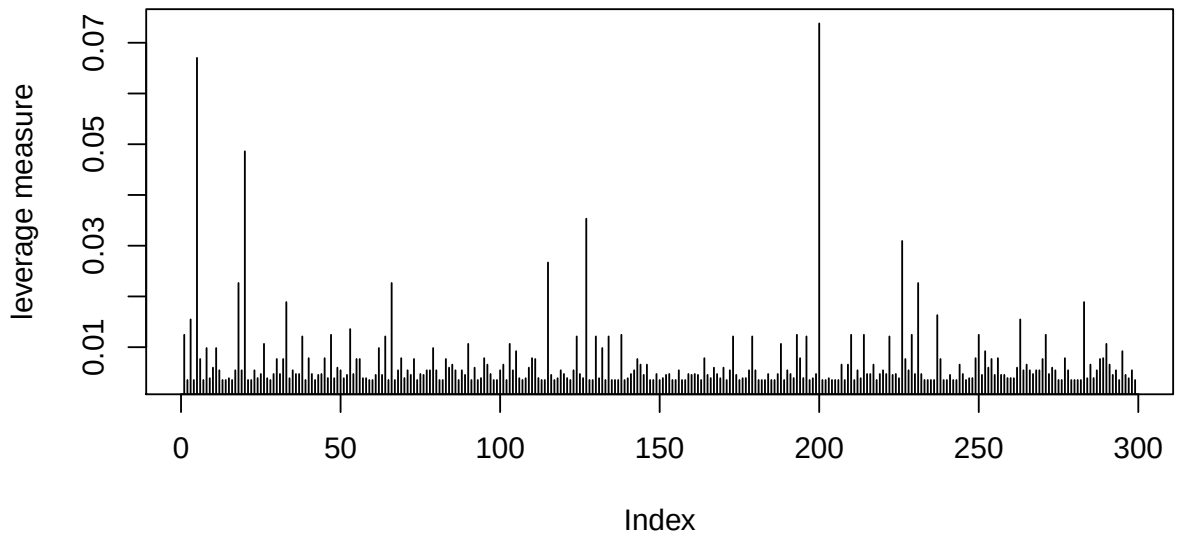
The Z-test tells us that the coefficients fitted in the glm are significantly different from zero and also the deviance test tells us that the difference in deviance with the null model is highly significant. This is an insight that serum_creatinine may be an import regressor.

After serum_creatinine, we tried with serum_sodium. We produced the exact same plots as the one above.

```
g = glm(DEATH_EVENT~serum_sodium,data=data,family=binomial);
xgrid = seq(min(serum_sodium)-4,max(serum_sodium)+1,length=1000);
ypred<-predglm(g,xgrid);

r=data[serum_sodium<115,];
plot(xgrid,ypred,type='l',xlab='serum_sodium',
     ylab='DEATH_EVENT',xlim=c(min(serum_sodium)-4,
                               max(serum_sodium)+1),ylim=c(-0.25,1.25)   ,
     col ='black',lwd = 2.5)

points(data$serum_sodium,data$DEATH_EVENT,pch='o')
text(r$DEATH_EVENT~r$serum_sodium
     ,data=r,labels=rownames(r),pos=3,cex=0.8)
```




```
plot(hatvalues(g), type = 'h',ylab='leverage measure')
```



As before, patient 200 seems to be an outlier for this model and again there seems to be no significant leverage points (the highest of the leverage measure is attained by patient 200).

```
summary(g)
```

```
##
## Call:
```

```
## glm(formula = DEATH_EVENT ~ serum_sodium, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8459  -0.8928  -0.7582   1.3197   1.9231
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  12.39442    4.07264   3.043  0.00234 **
## serum_sodium -0.09639    0.02989  -3.224  0.00126 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 375.35  on 298  degrees of freedom
## Residual deviance: 364.02  on 297  degrees of freedom
## AIC: 368.02
##
## Number of Fisher Scoring iterations: 4
```

```
anova(g,test = 'Chisq')
```

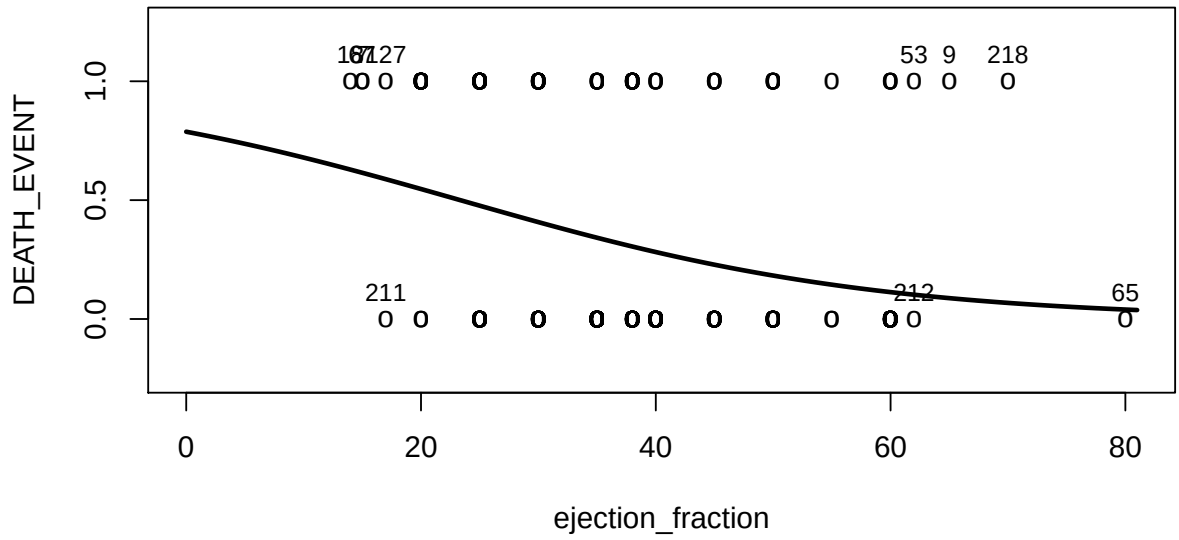
```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: DEATH_EVENT
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                           298     375.35
## serum_sodium  1   11.327       297     364.02 0.000764 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As before, both the Z-test and the deviance tests show that this model is significantly better then the null one. Nevertheless, from this tests it seems that serum_sodium is less important than serum_creatinine for predictions.

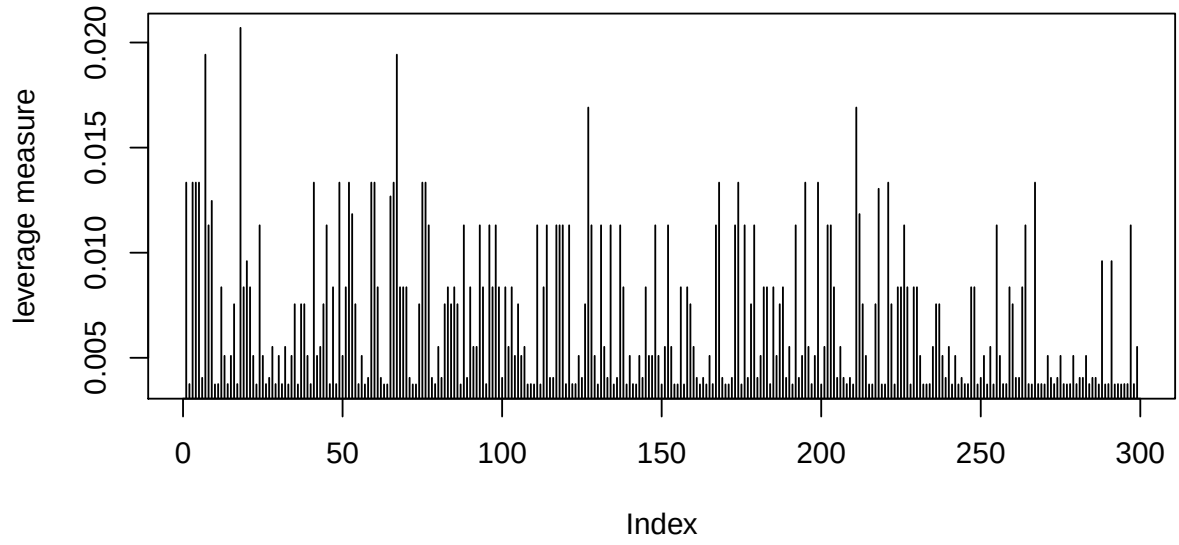
The last interesting single variable regression we fitted was with ejection_fraction as a predictor.

```
g = glm(DEATH_EVENT~ejection_fraction,data=data,family=binomial);
xgrid = seq(0,max(ejection_fraction)+1,length=1000);
ypred<-predglm(g,xgrid);

r = data[ejection_fraction>60 | ejection_fraction<20,];
plot(xgrid,ypred,type='l',xlab='ejection_fraction',
     ylab='DEATH_EVENT',xlim=c(0,max(ejection_fraction)+1),
     ylim=c(-0.25,1.25)  ,
     col ='black',lwd = 2.5)

points(data$ejection_fraction,data$DEATH_EVENT,pch='o')
```

```
text(r$DEATH_EVENT~r$ejection_fraction
     ,data=r,labels=rownames(r),pos=3,cex=0.8)
```




```
plot(hatvalues(g), type = 'h',ylab='leverage measure')
```




```
summary(g)
```

```
##
```

```
## Call:
## glm(formula = DEATH_EVENT ~ ejection_fraction, family = binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3320  -0.9146  -0.7201   1.2173   2.3205
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.31169    0.46278   2.834  0.00459 **
## ejection_fraction -0.05620    0.01258  -4.468 7.88e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 375.35  on 298  degrees of freedom
## Residual deviance: 351.97  on 297  degrees of freedom
## AIC: 355.97
##
## Number of Fisher Scoring iterations: 4
```

```
anova(g,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: DEATH_EVENT
##
## Terms added sequentially (first to last)
##
##
##                   Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                298     375.35
## ejection_fraction  1   23.381       297     351.97 1.329e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Also here, there seems to be no significant leverage points. In this model there are some outliers (patients 211,53,9 and 218). Both the Z and the deviance test lead us to reject the null hypothesis up to a significance level of order 10^{-6}.

The three plots above are the three most interesting single variables for logistic regressions that we found. We evidenced points that seems to be outliers in the boxplots and made deviance tests to check if these models are significantly better than the null one. They all turned out to be highly significant (the highest p values are around 7×10^{-4}).

Variable Selection

For selecting variables, we compared different methods. The first one we tried just to get a rough idea was to fit logistic regression with all the regressors and performing a deviance significance test. This won't select immediately a subset of the variables, but it gives an insight of what important variables can be.

```r
g = glm(DEATH_EVENT~.,data=data,family = binomial);

anova(g,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: DEATH_EVENT
##
## Terms added sequentially (first to last)
##
##
##                           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                        298     375.35
## age                        1  19.3560       297     355.99 1.085e-05 ***
## anaemia                    1   0.6712       296     355.32   0.41262
## creatinine_phosphokinase   1   2.7190       295     352.60   0.09916 .
## diabetes                   1   0.2857       294     352.32   0.59302
## ejection_fraction          1  28.9248       293     323.39 7.525e-08 ***
## high_blood_pressure        1   1.3568       292     322.04   0.24410
## platelets                  1   0.3141       291     321.72   0.57518
## serum_creatinine           1  23.3897       290     298.33 1.323e-06 ***
## serum_sodium               1   2.7424       289     295.59   0.09772 .
## sex                        1   1.1571       288     294.43   0.28207
## smoking                    1   0.1513       287     294.28   0.69728
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This test fits the models introducing a variable at a time (starting from the one on the top) and calculates the difference with the deviance of the model before, then it tests for the null hypothesis that the difference of the two deviance is zero. We observed that the highly significant decrements in deviance happened then we added to the model the following variables : ejection_fraction, age and serum_creatinine. Also the introduction of serum_sodium and diabetes produced a difference in deviance, but less significant (more or less the p values are 0.1).

After this first analysis we tried stepwise selection with both Akaike information criteria (AIC) and Bayesian information criteria (BIC). This measures are really similar to the deviance, the only difference is that they are also penalizing a bigger number of regressors.

$$AIC = Dev(\hat{\theta}) + 2d \qquad BIC = Dev(\hat{\theta}) + log(n)d$$

```r
#AIC
glm.mod = glm(DEATH_EVENT~.,family = "binomial",data=data);
g = stepAIC(glm.mod,direction='both',trace=0)
summary(g)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + anaemia + creatinine_phosphokinase +
##     ejection_fraction + high_blood_pressure + serum_creatinine +
##     serum_sodium, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.2358  -0.7597  -0.4619   0.8415   2.4993
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)               4.7298351  4.5190849   1.047  0.29527
## age                       0.0530244  0.0127859   4.147 3.37e-05 ***
## anaemia                   0.4309325  0.2980595   1.446  0.14824
## creatinine_phosphokinase  0.0002674  0.0001405   1.902  0.05713 .
## ejection_fraction        -0.0679014  0.0145626  -4.663 3.12e-06 ***
## high_blood_pressure       0.4606311  0.2992103   1.539  0.12368
## serum_creatinine          0.6619016  0.1711437   3.868  0.00011 ***
## serum_sodium             -0.0568437  0.0330681  -1.719  0.08562 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 375.35  on 298  degrees of freedom
## Residual deviance: 296.05  on 291  degrees of freedom
## AIC: 312.05
##
## Number of Fisher Scoring iterations: 5
```

```
#BIC
g = stepAIC(glm.mod,direction='both',trace=0,k=log(nrow(data)))
summary(g)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine,
##     family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3072  -0.7704  -0.5177   0.9127   2.4175
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.35306    0.83954  -2.803  0.00507 **
## age                0.05173    0.01231   4.202 2.65e-05 ***
## ejection_fraction -0.07000    0.01423  -4.918 8.72e-07 ***
## serum_creatinine   0.66592    0.15916   4.184 2.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 375.35  on 298  degrees of freedom
## Residual deviance: 305.28  on 295  degrees of freedom
## AIC: 313.28
##
## Number of Fisher Scoring iterations: 4
```

As expected, the BIC is penalizing more than the AIC (in fact AIC is selecting a model with 7 out of 11 regressors, the BIC instead suggested to use just three of them, exactly the three that were evidenced in the

deviance test). The comparison between the two reduced models obtained by AIC and BIC is summarized in the following table.

variables	p-val (AIC selection, Z test)	p-val (BIC selection, Z test)
anaemia	0.14824	not present
high_blood_pressure	0.12368	not present
serum_sodium	0.08562	not present
creatinine_phosphokinase	0.05713	not present
age	3.37e-05	2.65e-05
serum_creatininine	0.00011	2.86e-05
ejection_fraction	3.12e-06	8.72e-07

After this we tried to test the significance of this model compared with the full one. We did this with the BIC reduced model by using the Chi-square test for deviances.

```
full.mod<-glm(DEATH_EVENT~.,data=data,family=binomial);
bic.mod<-glm(DEATH_EVENT~ejection_fraction+age+serum_creatinine,data=data,family=binomial);
anova(bic.mod,full.mod,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: DEATH_EVENT ~ ejection_fraction + age + serum_creatinine
## Model 2: DEATH_EVENT ~ age + anaemia + creatinine_phosphokinase + diabetes +
##     ejection_fraction + high_blood_pressure + platelets + serum_creatinine +
##     serum_sodium + sex + smoking
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       295     305.28
## 2       287     294.28  8   11.002   0.2016
```

This test tells us that we cannot reject the null hypothesis (up to an α level of 20.16%) that the coefficients of the full model associated with the deleted variables are null. This in practice mean that all the variables deleted by the BIC selection are not able to jointly give a significant increase in the deviance, this is a good sign that these three variables alone may be a good choice. Also the interpretation of the coefficients is interesting: if we fix two of the three variables, the sign of the other tells us if lower or higher values are augmenting the death probability.

Since in this case the models selected by AIC and BIC are nested, we confronted them also using the deviance test. We did this to try to understand if AIC was significantly better than BIC in explaining the DEATH_EVENT.

```
full.aic.mod<-glm(DEATH_EVENT~ejection_fraction+age+
                    serum_creatinine+creatinine_phosphokinase+
                    serum_sodium+high_blood_pressure+anaemia
                    ,data=data,family=binomial);


anova(bic.mod,full.aic.mod,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: DEATH_EVENT ~ ejection_fraction + age + serum_creatinine
## Model 2: DEATH_EVENT ~ ejection_fraction + age + serum_creatinine + creatinine_phosphokinase +
##     serum_sodium + high_blood_pressure + anaemia
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       295     305.28
```

```
## 2       291     296.05  4   9.2304  0.05559 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This test tells us that for significance levels lower than approximately 0.05 we should not reject the null hypothesis. In practice the model selected by AIC is not significantly better than the one selected by BIC (at least according to this test). For this reason in the following section we used more the BIC variables than the AIC ones.

We tried also to see what results Lasso and Ridge regression give. We expect Lasso regression to produce sparse coefficients (and thus perform variable selection), in this way we can confront it with the methods seen before. Ridge regression is not producing sparse coefficients, so it won't select variables.

```
l=0.1;
lasso.glm<-glmnet(x,y,family = 'binomial',alpha=1,lambda=l);
coefficients(lasso.glm)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)              -0.912003986
## age                       0.005716948
## anaemia                   .
## creatinine_phosphokinase  .
## diabetes                  .
## ejection_fraction        -0.010354480
## high_blood_pressure       .
## platelets                 .
## serum_creatinine          0.145338827
## serum_sodium              .
## sex                       .
## smoking                   .
```

```
threshold <-0.32;
lasso.class<-hard_classif(predict(lasso.glm,newx=as.matrix(x),
                                  type='response'),threshold);
confusionMatrix(as.factor(lasso.class),as.factor(y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 146  25
##          1  57  71
##
##                Accuracy : 0.7258
##                  95% CI : (0.6714, 0.7755)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0458071
##
##                   Kappa : 0.4217
##
##  Mcnemar's Test P-Value : 0.0006185
##
##             Sensitivity : 0.7396
##             Specificity : 0.7192
##          Pos Pred Value : 0.5547
##          Neg Pred Value : 0.8538
```

```
##              Prevalence : 0.3211
##          Detection Rate : 0.2375
##    Detection Prevalence : 0.4281
##       Balanced Accuracy : 0.7294
##
##        'Positive' Class : 1
##
```

```
ridge.glm<-glmnet(x,y,family = 'binomial',alpha=0,lambda=l);
coefficients(ridge.glm)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)                4.1201213003
## age                        0.0311716068
## anaemia                    0.2077156252
## creatinine_phosphokinase   0.0001447147
## diabetes                   0.0506673342
## ejection_fraction         -0.0366395947
## high_blood_pressure        0.2475966193
## platelets                 -0.0004311448
## serum_creatinine           0.3735529510
## serum_sodium              -0.0443559803
## sex                       -0.1341306949
## smoking                    0.0024782770
```

We tried Lasso for different values of the penalizing coefficient. We left here the one with $\lambda = 0.1$ and it selected age, ejection_fraction and serum_creatinine. By lowering the threshold to $\lambda = 0.05$ the next variable that appears is serum_sodium (and by lowering it more it appears creatinine_phosphokinase and high_blood_pressure). We also checked the performance measures of this model using an optimized threshold that minimizes a convex combination of FNR and FPR (details in next chapter). We obtained a 0.74 of TPR, 0.72 of TNR and an overall accuracy around 0.72, but we are aware that the performance is tested on the training data. Perhaps one could try to obtain less biased error measures by using techniques like LOOCV or K-fold cross validation (but the problem here is the unbalance and the size of the dataset).

We did the same thing with Ridge regression.

```
ridge.glm<-glmnet(x,y,family = 'binomial',alpha=0,lambda=l);
coefficients(ridge.glm)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)                4.1201213003
## age                        0.0311716068
## anaemia                    0.2077156252
## creatinine_phosphokinase   0.0001447147
## diabetes                   0.0506673342
## ejection_fraction         -0.0366395947
## high_blood_pressure        0.2475966193
## platelets                 -0.0004311448
## serum_creatinine           0.3735529510
## serum_sodium              -0.0443559803
## sex                       -0.1341306949
## smoking                    0.0024782770
```

```
threshold <-0.32;
ridge.class<-hard_classif(predict(ridge.glm,newx=as.matrix(x),
```

```
                                    type='response'),threshold);
confusionMatrix(as.factor(ridge.class),as.factor(y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 146  19
##          1  57  77
##
##                Accuracy : 0.7458
##                  95% CI : (0.6925, 0.7942)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.007011
##
##                   Kappa : 0.4721
##
##  Mcnemar's Test P-Value : 2.194e-05
##
##             Sensitivity : 0.8021
##             Specificity : 0.7192
##          Pos Pred Value : 0.5746
##          Neg Pred Value : 0.8848
##              Prevalence : 0.3211
##          Detection Rate : 0.2575
##    Detection Prevalence : 0.4482
##       Balanced Accuracy : 0.7606
##
##        'Positive' Class : 1
##
```

As expected, Lasso enforces sparsity on the coefficients. The variables selected by Lasso are the same that BIC returned, but the coefficients are different (the loss function is different in the two cases). As mentioned before, Ridge regression instead does not enforce sparsity in the coefficients, so the variable selection is 'smooth'. Beside this, Ridge regression seems to have good performance measures at least on training data (with a lowered threshold for predicting death), with a 0.8 of TPR and a 0.72 of TNR (overall accuracy of 0.74). Beside the better performance measures of Ridge, the big advantage of Lasso is that it is actually telling us which of the variables seems to be necessary for prediction.

As an overall conclusion of this section: Lasso and BIC in this case gave us the same set of selected variables, while AIC gave us a model with a double number of regressors (but the deviance test told us that the difference in deviance is not highly significant).

In the next section we fitted different models using the insights we have extracted up to now on this dataset. We also confronted some measures of accuracy for all these models.

Logistic Regression

To predict $p(death\_event = 1 \,|\, patient's\, parameters)$ the first thing we tried was logistic regression. With this model we tried to find a good compromise between miss-classifying in the false positive and in the false negative sense.

In the medical scenario a false negative is much more dangerous (we would like to use this model to predict in advance people that are in a critical situation), so we tried to minimize for a fixed ϵ the function:

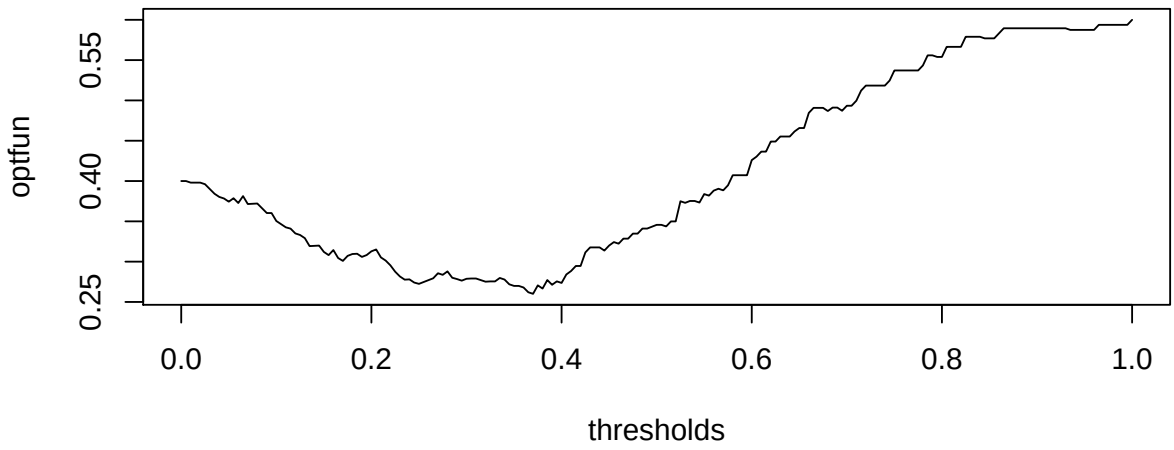
$$\delta \to (1-\epsilon)p(\hat{y} < \delta | y = 1) + \epsilon p(\hat{y} > \delta | y = 0)$$

This kind of minimization corresponds in some sense in putting a prior distribution on the death probability. Since we do not know the real distributions we have no other choice than to minimize the empirical version of the last function, that is:

$$\delta \to (1-\epsilon)\frac{\sum_{x\in Data} I_{\{y(x)=1,\hat{y}(x)<\delta\}}}{\sum_{x\in Data} I_{\{y(x)=1\}}} + \epsilon\frac{\sum_{x\in Data} I_{\{y(x)=0,\hat{y}(x)>\delta\}}}{\sum_{x\in Data} I_{\{y(x)=0\}}} = (1-\epsilon)FNR + \epsilon FPR$$

. Unluckily, we need to make a choice on how do we want to penalize a false negative miss-classification compared to false positive one. With $\epsilon = 0$ we would classify everyone in the dying risk category (and that's the safest choice to avoid false negative miss-classifications) but that's not so useful. We tried initially with $\epsilon = 0.4$ on the Logistic regression model with BIC selected variables.

```
#BIC
g = bic.mod;
thresholds = seq(0,1,0.005);
probs = fitted(g);
optfun = seq(0,1,0.005);
eps = 0.4;
for(i in seq(1,length(thresholds))){
  optfun[i]=(1-eps)*sum(probs<thresholds[i] & DEATH_EVENT=='yes')/sum(DEATH_EVENT=='yes');
  optfun[i]=optfun[i]+
    eps*sum(probs>thresholds[i] & DEATH_EVENT=='no')/sum(DEATH_EVENT=='no');
}
optimal_thresh=thresholds[which.min(optfun)];

plot(thresholds,optfun,type='l')
```




```
fitted_vals <-hard_classif(predict.glm(g,type='response'),0.5);
confusionMatrix(as.factor(fitted_vals),as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 183  49
##          1  20  47
```

```
##
##                Accuracy : 0.7692
##                  95% CI : (0.7173, 0.8158)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0003752
##
##                   Kappa : 0.4249
##
##  Mcnemar's Test P-Value : 0.0007495
##
##             Sensitivity : 0.4896
##             Specificity : 0.9015
##          Pos Pred Value : 0.7015
##          Neg Pred Value : 0.7888
##              Prevalence : 0.3211
##          Detection Rate : 0.1572
##    Detection Prevalence : 0.2241
##       Balanced Accuracy : 0.6955
##
##        'Positive' Class : 1
##
```

```
fitted_vals.optimal_thresh<-hard_classif(predict.glm(g,type='response'),
                                         optimal_thresh);

confusionMatrix(as.factor(fitted_vals.optimal_thresh),
                as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 163  29
##          1  40  67
##
##                Accuracy : 0.7692
##                  95% CI : (0.7173, 0.8158)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0003752
##
##                   Kappa : 0.4862
##
##  Mcnemar's Test P-Value : 0.2286443
##
##             Sensitivity : 0.6979
##             Specificity : 0.8030
##          Pos Pred Value : 0.6262
##          Neg Pred Value : 0.8490
##              Prevalence : 0.3211
##          Detection Rate : 0.2241
##    Detection Prevalence : 0.3579
##       Balanced Accuracy : 0.7504
##
##        'Positive' Class : 1
##
```

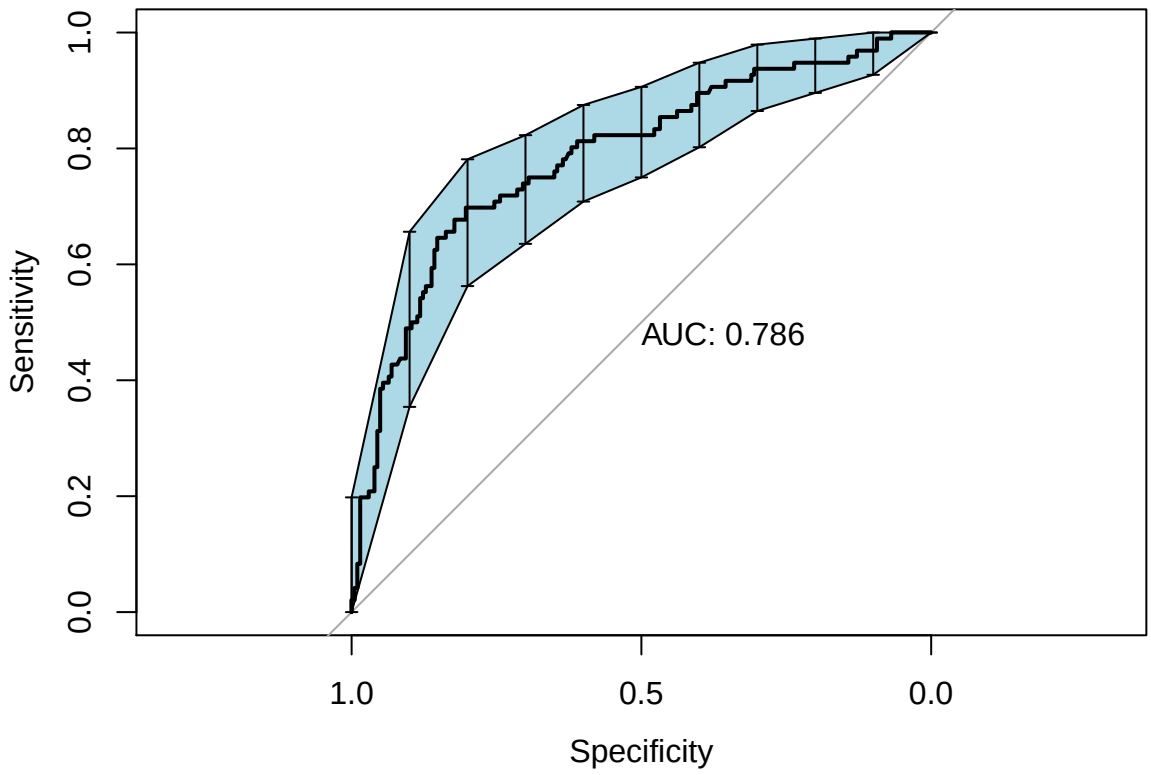
Just up here we put the two confusion matrices for the hard classificator obtained from BIC (we tried with the AIC variables too) logistic regression by using two diffent thresholds. In the first one we use 0.5 as threshold, and in this case the false negative rate is around 0.51. In the second one we used the optimized threshold, and this lead to a false negative rate of around 0.3. We can also notice that even if we gained in sensitivity with the second version, we lost specificity. The optimal threshold we found for this model is $\delta = 0.37$. Clearly even with the optimized threshold the error is too high to make reliable predictions. We tried to decrease ϵ but it seems that the optimal threshold is increasing too fast as a function of ϵ, so this was not useful.

```
pROC_obj <- roc(data$DEATH_EVENT,fitted(g),
            smoothed = TRUE,
            ci=FALSE, ci.alpha=0.9, stratified=FALSE,
            plot=TRUE, auc.polygon=FALSE, max.auc.polygon=FALSE, grid=FALSE,
            print.auc=TRUE, show.thres=FALSE);
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

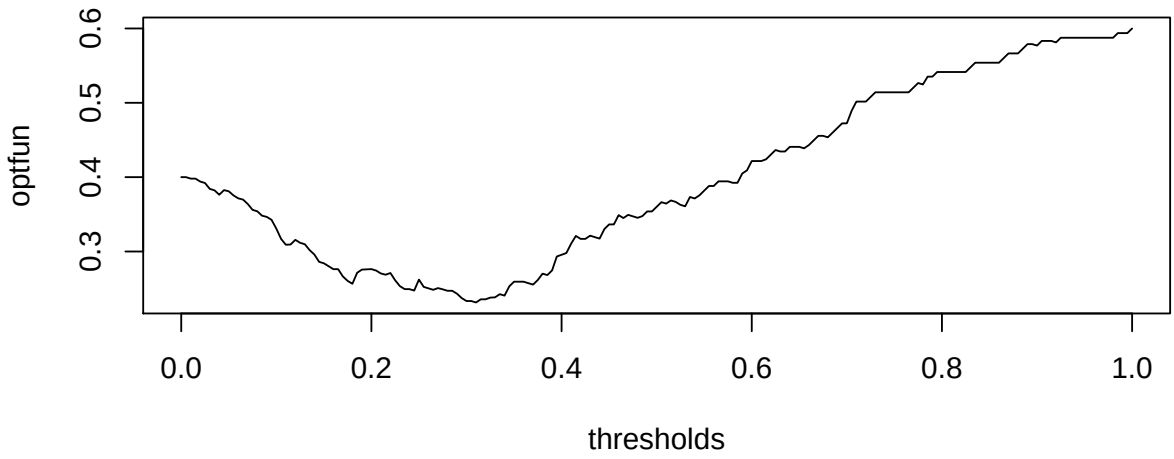
```
sens.ci <- ci.se(pROC_obj);
plot(sens.ci, type="shape", col="lightblue");
plot(sens.ci, type="bars");
```



From the ROC curve we can see sensitivity and specificity for all possible thresholds. From the shape of this curve we can see that if we want a high sensitivity, we would get a pretty low specificity.AUC provides an aggregate measure of performance across all possible classification thresholds.The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.In this case AUC is high enough to distinguish between the positive and negative classes.

We did the same for the AIC model, just for completeness we are sharing the results here:

```
#AIC
g = full.aic.mod;
thresholds = seq(0,1,0.005);
probs = fitted(g);
optfun = seq(0,1,0.005);
eps = 0.4;
for(i in seq(1,length(thresholds))){
  optfun[i]=(1-eps)*sum(probs<thresholds[i] & DEATH_EVENT=='yes')/sum(DEATH_EVENT=='yes');
  optfun[i]=optfun[i]+
    eps*sum(probs>thresholds[i] & DEATH_EVENT=='no')/sum(DEATH_EVENT=='no');
}
optimal_thresh=thresholds[which.min(optfun)];

plot(thresholds,optfun,type='l')
```




```
fitted_vals <-hard_classif(predict.glm(g,type='response'),0.5);
confusionMatrix(as.factor(fitted_vals),as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 182  51
##          1  21  45
##
##                Accuracy : 0.7592
##                  95% CI : (0.7066, 0.8066)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0014572
##
##                   Kappa : 0.3981
##
##  Mcnemar's Test P-Value : 0.0006316
```

```
##
##             Sensitivity : 0.4688
##             Specificity : 0.8966
##          Pos Pred Value : 0.6818
##          Neg Pred Value : 0.7811
##              Prevalence : 0.3211
##          Detection Rate : 0.1505
##    Detection Prevalence : 0.2207
##       Balanced Accuracy : 0.6827
##
##        'Positive' Class : 1
##
```

```
fitted_vals.optimal_thresh<-hard_classif(predict.glm(g,type='response'),
                                         optimal_thresh);

confusionMatrix(as.factor(fitted_vals.optimal_thresh),
                as.factor(data_num$DEATH_EVENT),positive='1')
```

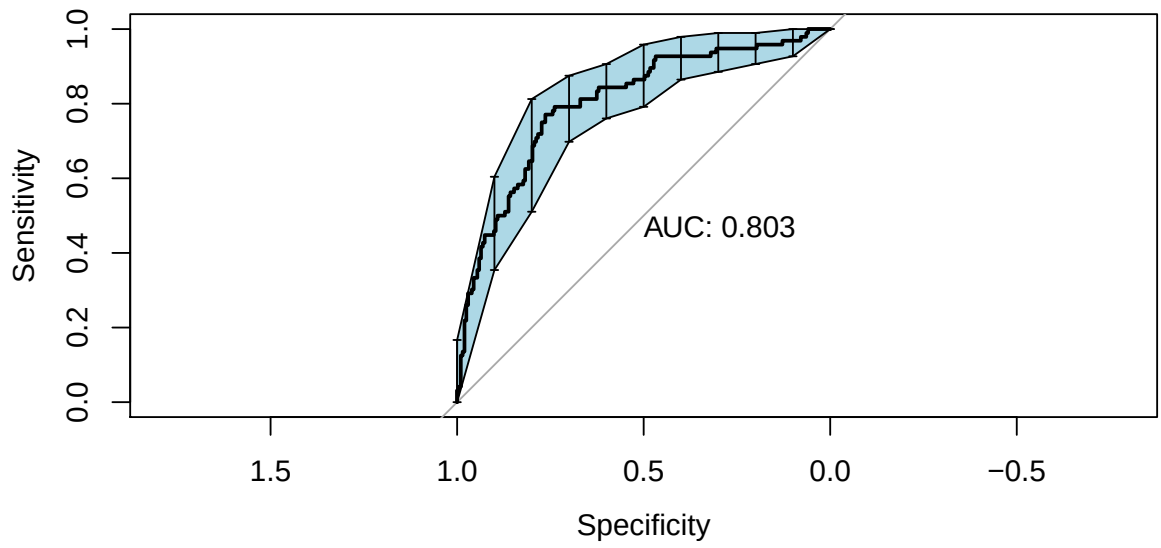
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 149  20
##          1  54  76
##
##                Accuracy : 0.7525
##                  95% CI : (0.6996, 0.8004)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.003304
##
##                   Kappa : 0.4808
##
##  Mcnemar's Test P-Value : 0.000125
##
##             Sensitivity : 0.7917
##             Specificity : 0.7340
##          Pos Pred Value : 0.5846
##          Neg Pred Value : 0.8817
##              Prevalence : 0.3211
##          Detection Rate : 0.2542
##    Detection Prevalence : 0.4348
##       Balanced Accuracy : 0.7628
##
##        'Positive' Class : 1
##
```

```
pROC_obj <- roc(data$DEATH_EVENT,fitted(g),
            smoothed = TRUE,
            ci=FALSE, ci.alpha=0.9, stratified=FALSE,
            plot=TRUE, auc.polygon=FALSE, max.auc.polygon=FALSE, grid=FALSE,
            print.auc=TRUE, show.thres=FALSE);
```

```
## Setting levels: control = 0, case = 1
```

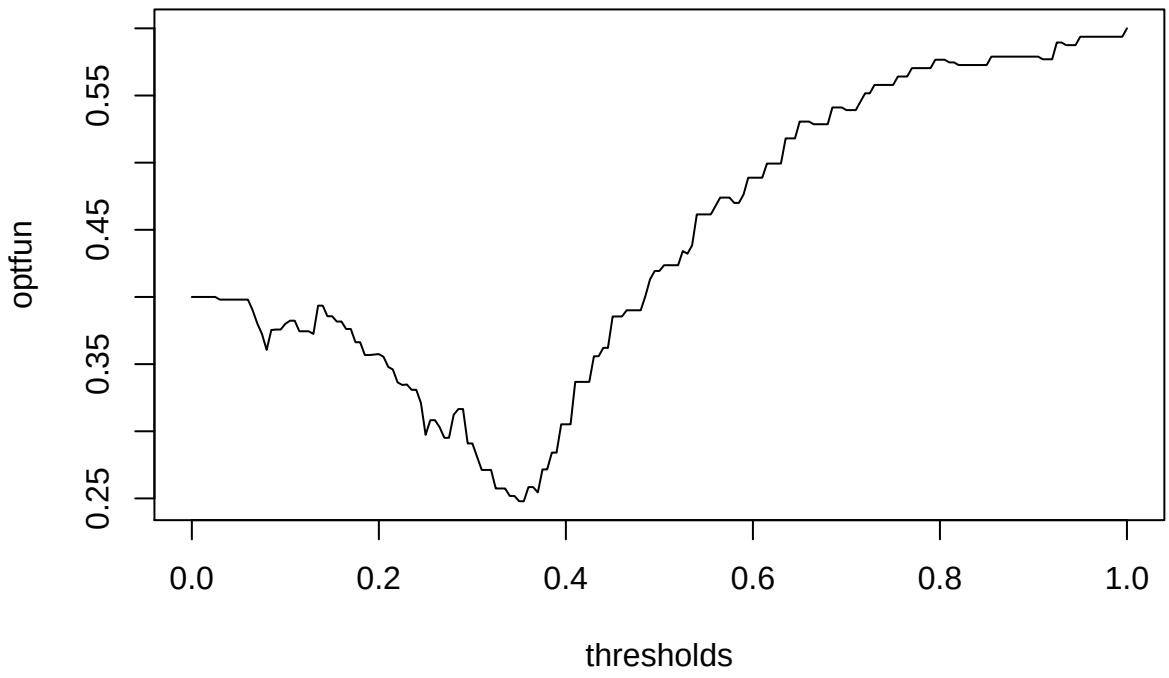
```
## Setting direction: controls < cases
```

```
sens.ci <- ci.se(pROC_obj);
plot(sens.ci, type="shape", col="lightblue",ylab='TPR');
plot(sens.ci, type="bars");
```



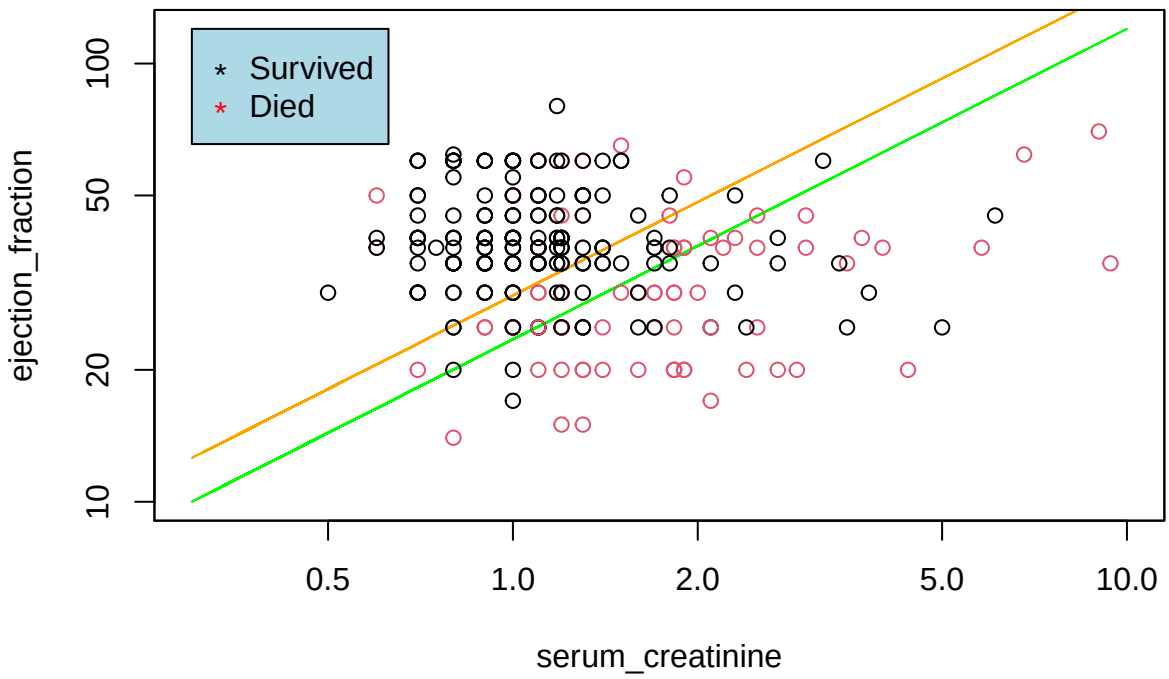
Starting from the BIC we tried also a two variables model, and it turned out that the best one was the couple serum_creatinine-ejection_fraction. By dropping the age we loose a little bit in sensitivity but with a bigger gain in specificity (and in overall accuracy) as you can see in the AUC.

```
g<-glm(DEATH_EVENT~serum_creatinine+ejection_fraction
       ,data=data,family=binomial);

thresholds = seq(0,1,0.005);
probs = fitted(g);
optfun = seq(0,1,0.005);
eps = 0.4
for(i in seq(1,length(thresholds))){
  optfun[i]=(1-eps)*sum(probs<thresholds[i] & DEATH_EVENT=='yes')/sum(DEATH_EVENT=='yes');
  optfun[i]=optfun[i]+
    eps*sum(probs>thresholds[i] & DEATH_EVENT=='no')/sum(DEATH_EVENT=='no');
}
plot(thresholds,optfun,type='l')
```

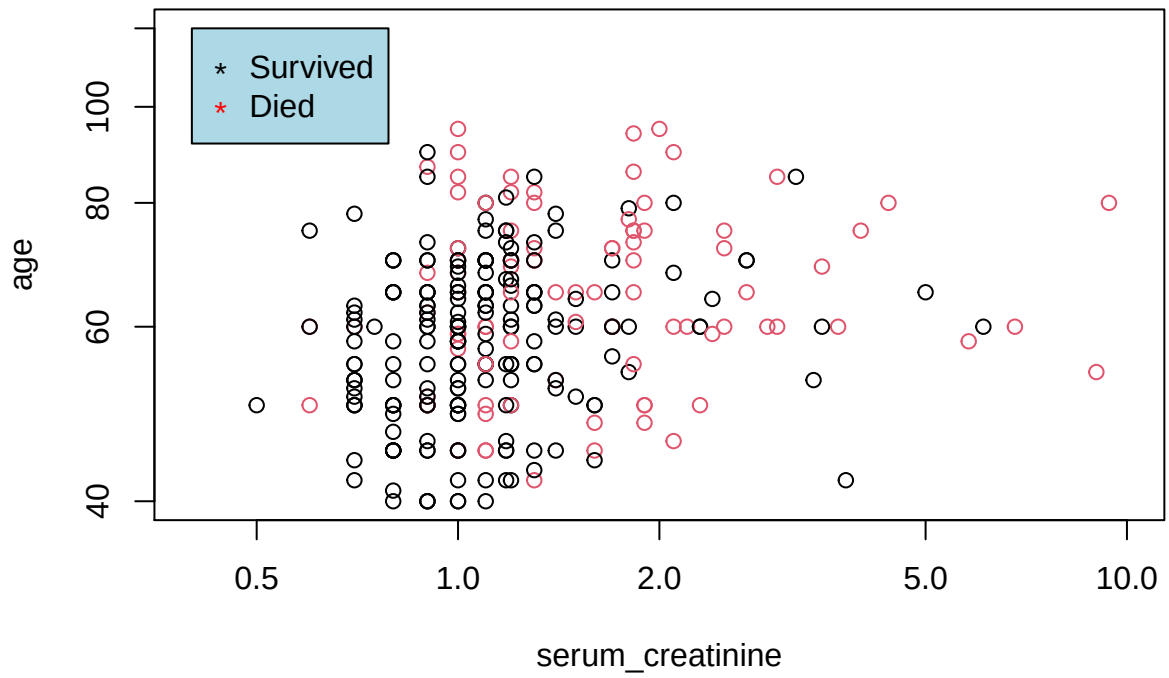



```
optimal_thresh=thresholds[which.min(optfun)];



x<-serum_creatinine;
threshold<-0.5;
decision.boundary<-(1/g$coefficients[3])*
  log(threshold/(1-threshold))-
  (g$coefficients[2]/g$coefficients[3])*x-
  (g$coefficients[1]/g$coefficients[3]);

decision.boundary.opt<-(1/g$coefficients[3])*
  log(optimal_thresh/(1-optimal_thresh))-
  (g$coefficients[2]/g$coefficients[3])*x-
  (g$coefficients[1]/g$coefficients[3]);


l<-DEATH_EVENT=='yes'
plot(x,decision.boundary,col='green',type='l',
    xlab='',ylab='',xaxt='n',yaxt='n')
lines(x,decision.boundary.opt,col='orange',type='l',
     xlab='',ylab='',xaxt='n',yaxt='n')
par(new=TRUE)
plot(serum_creatinine,ejection_fraction,log='xy',type=,
     col=l+1,ylim=c(10,120),xlim=c(0.3,10))
legend(0.3,120,legend = c('Survived','Died'),col=c('black','red'), bg='lightblue',
```

```
       cex=1,pch='*')
```




```
plot(serum_creatinine,age,log='xy',col=1+1,ylim=c(40,120),
     xlim=c(0.4,10))
legend(0.4,120,legend = c('Survived','Died'),col=c('black','red'), bg='lightblue',
       cex=1,pch='*')
```




```
plot(age,ejection_fraction,log='xy',col=1+1,xlim=c(30,100),
     ylim=c(12,100))
legend(30,100,legend = c('Survived','Died'),col=c('black','red'), bg='lightblue',
       cex=1,pch='*')
```


```
fitted_vals <-hard_classif(predict.glm(g,type='response'),0.5);
confusionMatrix(as.factor(fitted_vals),as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 190  63
##          1  13  33
##
##                Accuracy : 0.7458
##                  95% CI : (0.6925, 0.7942)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.007011
##
##                   Kappa : 0.3242
##
##  Mcnemar's Test P-Value : 1.902e-08
##
##             Sensitivity : 0.3438
##             Specificity : 0.9360
##          Pos Pred Value : 0.7174
##          Neg Pred Value : 0.7510
##              Prevalence : 0.3211
##          Detection Rate : 0.1104
##    Detection Prevalence : 0.1538
```

```
##       Balanced Accuracy : 0.6399
##
##        'Positive' Class : 1
##
```

```
fitted_vals.optimal_thresh<-hard_classif(predict.glm(g,type='response'),
                                         optimal_thresh);

confusionMatrix(as.factor(fitted_vals.optimal_thresh),
                as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 166  28
##          1  37  68
##
##                Accuracy : 0.7826
##                  95% CI : (0.7315, 0.828)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 4.787e-05
##
##                   Kappa : 0.5134
##
##  Mcnemar's Test P-Value : 0.3211
##
##             Sensitivity : 0.7083
##             Specificity : 0.8177
##          Pos Pred Value : 0.6476
##          Neg Pred Value : 0.8557
##              Prevalence : 0.3211
##          Detection Rate : 0.2274
##    Detection Prevalence : 0.3512
##       Balanced Accuracy : 0.7630
##
##        'Positive' Class : 1
##
```

Up here we put the scatter plots for the joint couples of the three BIC variables. In the first scatter plot we put also the decision boundaries of the model with ejection_fraction and serum_creatinine (orange for the optimized threshold and green for the 0.5 threshold). From the train performance measures, it seems it would be better to use logistic regression on the two variable instead of the three suggested by BIC.

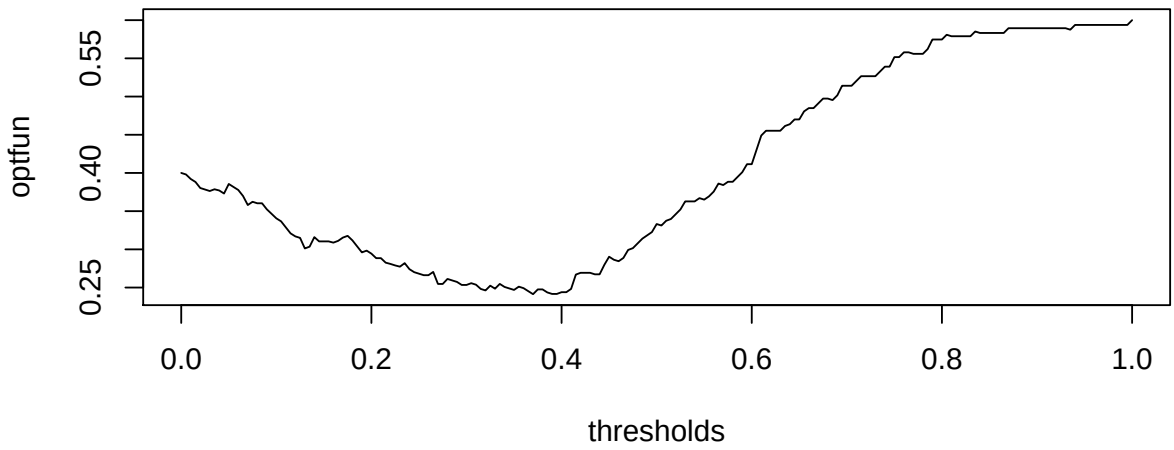
Another thing we tried to do was to fit a polynomial logistic regression model (degree two and three) with all interaction terms to see if we could find a better fit with the data. We tried also to understand if there are some interesting interaction terms, to do this we fitted the quadratic logistic regression model and we performed variable selection.

```
#g = glm(DEATH_EVENT~polym(data$age,data$ejection_fraction,
  #                         data$serum_creatinine,raw=TRUE,degree=3),
  #    family = "binomial",data=data);

g=glm(DEATH_EVENT~age+ejection_fraction+serum_creatinine+
        ejection_fraction:serum_creatinine+
        ejection_fraction:ejection_fraction
```

```
      +serum_creatinine:serum_creatinine+
        age:age+
        age:serum_creatinine+
        age:ejection_fraction,
      family = "binomial",data=data);

g = step(g,trace=0,direction = 'backward');
summary(g)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ age + ejection_fraction + serum_creatinine +
##     age:ejection_fraction, family = "binomial", data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3096  -0.7865  -0.4879   0.9177   2.6318
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)             3.069247   2.925196   1.049  0.29407
## age                    -0.035252   0.046406  -0.760  0.44747
## ejection_fraction      -0.223732   0.081873  -2.733  0.00628 **
## serum_creatinine        0.698253   0.155643   4.486 7.25e-06 ***
## age:ejection_fraction   0.002412   0.001252   1.926  0.05405 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 375.35  on 298  degrees of freedom
## Residual deviance: 301.47  on 294  degrees of freedom
## AIC: 311.47
##
## Number of Fisher Scoring iterations: 5
```

```
thresholds = seq(0,1,0.005);
probs = fitted(g);
optfun = seq(0,1,0.005);
eps = 0.4
for(i in seq(1,length(thresholds))){
  optfun[i]=(1-eps)*sum(probs<thresholds[i] & DEATH_EVENT=='yes')/sum(DEATH_EVENT=='yes');
  optfun[i]=optfun[i]+
    eps*sum(probs>thresholds[i] & DEATH_EVENT=='no')/sum(DEATH_EVENT=='no');
}
plot(thresholds,optfun,type='l')
```




```
fitted_vals<-hard_classif(predict.glm(g,type='response'),0.5);
confusionMatrix(as.factor(fitted_vals),as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 183  47
##          1  20  49
##
##                Accuracy : 0.7759
##                  95% CI : (0.7244, 0.8219)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.000139
##
##                   Kappa : 0.4449
##
##  Mcnemar's Test P-Value : 0.001491
##
##             Sensitivity : 0.5104
##             Specificity : 0.9015
##          Pos Pred Value : 0.7101
##          Neg Pred Value : 0.7957
##              Prevalence : 0.3211
##          Detection Rate : 0.1639
##    Detection Prevalence : 0.2308
##       Balanced Accuracy : 0.7059
##
##        'Positive' Class : 1
##
```

```
fitted_vals.optimal_thresh<-hard_classif(predict.glm(g,type='response'),
                                         optimal_thresh);
```

```
confusionMatrix(as.factor(fitted_vals.optimal_thresh),
                as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 157  25
##          1  46  71
##
##                Accuracy : 0.7625
##                  95% CI : (0.7102, 0.8096)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0009433
##
##                   Kappa : 0.485
##
##  Mcnemar's Test P-Value : 0.0176174
##
##             Sensitivity : 0.7396
##             Specificity : 0.7734
##          Pos Pred Value : 0.6068
##          Neg Pred Value : 0.8626
##              Prevalence : 0.3211
##          Detection Rate : 0.2375
##    Detection Prevalence : 0.3913
##       Balanced Accuracy : 0.7565
##
##        'Positive' Class : 1
##
```

```
anova(bic.mod,g,test='Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: DEATH_EVENT ~ ejection_fraction + age + serum_creatinine
## Model 2: DEATH_EVENT ~ age + ejection_fraction + serum_creatinine + age:ejection_fraction
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       295     305.28
## 2       294     301.47  1   3.8162  0.05076 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The deviance test isn't rejecting the null hypothesis (up to a 5% level) so there's not a strong evidence that this model is better than the one without the interaction term. The interaction terms in this model does not seem to influence the performance that much, even with the optimized threshold (the false negative rate is more or less 0.27). The only interesting insight is that variable selection suggests to use the BIC variables and just add an interaction term ejection_fraction:age.

LDA and QDA

Since the selected variables with the BIC test are all continuos (except for the age), we tried also with linear and quadratic discriminant analysis. Since the data is unbalanced (more than half of the observation are survived people), we tried also to change the prior choosing $p(DEATH = 1) > p(DEATH = 0)$ (in this way we can try to compensate the unbalance and get an higher sensitivity, it can be chosen by minimizing the

same function as before).

```
lda.mod = lda(DEATH_EVENT~ejection_fraction+serum_creatinine+age,
              data=data)
print(lda.mod)
```

```
## Call:
## lda(DEATH_EVENT ~ ejection_fraction + serum_creatinine + age,
##     data = data)
##
## Prior probabilities of groups:
##         0         1
## 0.6789298 0.3210702
##
## Group means:
##   ejection_fraction serum_creatinine      age
## 0          40.26601         1.184877 58.76191
## 1          33.46875         1.835833 65.21528
##
## Coefficients of linear discriminants:
##                            LD1
## ejection_fraction -0.05806427
## serum_creatinine   0.60473979
## age                0.04753473
```

```
lda.pred=predict(lda.mod,subset(data,select=-DEATH_EVENT));
confusionMatrix(as.factor(lda.pred$class),as.factor(data$DEATH_EVENT),positive='1');
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 183  52
##          1  20  44
##
##                Accuracy : 0.7592
##                  95% CI : (0.7066, 0.8066)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0014572
##
##                   Kappa : 0.3945
##
##  Mcnemar's Test P-Value : 0.0002588
##
##             Sensitivity : 0.4583
##             Specificity : 0.9015
##          Pos Pred Value : 0.6875
##          Neg Pred Value : 0.7787
##              Prevalence : 0.3211
##          Detection Rate : 0.1472
##    Detection Prevalence : 0.2140
##       Balanced Accuracy : 0.6799
##
##        'Positive' Class : 1
##
```

```
lda.mod = lda(DEATH_EVENT~ejection_fraction+serum_creatinine+age,
           data=data,prior=c(0.4,0.6));
print(lda.mod)
```

```
## Call:
## lda(DEATH_EVENT ~ ejection_fraction + serum_creatinine + age,
##     data = data, prior = c(0.4, 0.6))
##
## Prior probabilities of groups:
##   0   1
## 0.4 0.6
##
## Group means:
##   ejection_fraction serum_creatinine      age
## 0          40.26601         1.184877 58.76191
## 1          33.46875         1.835833 65.21528
##
## Coefficients of linear discriminants:
##                            LD1
## ejection_fraction -0.05806427
## serum_creatinine   0.60473979
## age                0.04753473
```

```
lda.pred = predict(lda.mod,subset(data,select=-DEATH_EVENT));
confusionMatrix(as.factor(lda.pred$class),as.factor(data$DEATH_EVENT),positive='1');
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 122  19
##          1  81  77
##
##                Accuracy : 0.6656
##                  95% CI : (0.609, 0.7188)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.7131
##
##                   Kappa : 0.3444
##
##  Mcnemar's Test P-Value : 1.061e-09
##
##             Sensitivity : 0.8021
##             Specificity : 0.6010
##          Pos Pred Value : 0.4873
##          Neg Pred Value : 0.8652
##              Prevalence : 0.3211
##          Detection Rate : 0.2575
##    Detection Prevalence : 0.5284
##       Balanced Accuracy : 0.7015
##
##        'Positive' Class : 1
##
```

As mentioned before, by changing the priors (we choose (0.4,0.6) by trying different values) we have been

able to get better training sensitivity results without loosing so much in the overall error (but we loose in specificity).

Then we tried with the QDA.

```
qda.mod = qda(DEATH_EVENT~ejection_fraction+serum_creatinine+age
              ,data=data)
print(qda.mod)
```

```
## Call:
## qda(DEATH_EVENT ~ ejection_fraction + serum_creatinine + age,
##     data = data)
##
## Prior probabilities of groups:
##         0         1
## 0.6789298 0.3210702
##
## Group means:
##   ejection_fraction serum_creatinine      age
## 0          40.26601         1.184877 58.76191
## 1          33.46875         1.835833 65.21528
```

```
qda.pred = predict(qda.mod,subset(data,select=-DEATH_EVENT))
confusionMatrix(as.factor(qda.pred$class),as.factor(data$DEATH_EVENT),positive ='1');
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 189  67
##          1  14  29
##
##                Accuracy : 0.7291
##                  95% CI : (0.6749, 0.7787)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.0348
##
##                   Kappa : 0.2728
##
##  Mcnemar's Test P-Value : 7.569e-09
##
##             Sensitivity : 0.30208
##             Specificity : 0.93103
##          Pos Pred Value : 0.67442
##          Neg Pred Value : 0.73828
##              Prevalence : 0.32107
##          Detection Rate : 0.09699
##    Detection Prevalence : 0.14381
##       Balanced Accuracy : 0.61656
##
##        'Positive' Class : 1
##
```

```
qda.mod = qda(DEATH_EVENT~ejection_fraction+serum_creatinine+age,
            data=data,prior=c(0.25,0.75));
print(qda.mod)
```

```
## Call:
## qda(DEATH_EVENT ~ ejection_fraction + serum_creatinine + age,
##     data = data, prior = c(0.25, 0.75))
##
## Prior probabilities of groups:
##    0    1
## 0.25 0.75
##
## Group means:
##   ejection_fraction serum_creatinine      age
## 0          40.26601         1.184877 58.76191
## 1          33.46875         1.835833 65.21528
```

```
qda.pred = predict(qda.mod,subset(data,select=-DEATH_EVENT));
confusionMatrix(as.factor(qda.pred$class),as.factor(data$DEATH_EVENT),positive='1');
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 125  19
##          1  78  77
##
##                Accuracy : 0.6756
##                  95% CI : (0.6193, 0.7283)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.5765
##
##                   Kappa : 0.3596
##
##  Mcnemar's Test P-Value : 3.885e-09
##
##             Sensitivity : 0.8021
##             Specificity : 0.6158
##          Pos Pred Value : 0.4968
##          Neg Pred Value : 0.8681
##              Prevalence : 0.3211
##          Detection Rate : 0.2575
##    Detection Prevalence : 0.5184
##       Balanced Accuracy : 0.7089
##
##        'Positive' Class : 1
##
```

We had to set an harder prior with the QDA to obtain reasonable results for sensitivity. In this tests we can see that QDA is better than LDA as expected, but the difference compared with the added complexity is not rewarding. Actually, the train performance of LDA is really surprising giving the simplicity of the model. We tried to fit LDA also deleting the age variable (with the same death prior), we got 2% more in sensitivity but with a big drop in specificity and overall accuracy. We leave here the confusion matrix:

```
lda.mod = lda(DEATH_EVENT~ejection_fraction+serum_creatinine,
            data=data,prior=c(0.4,0.6));
print(lda.mod)
```

```
## Call:
## lda(DEATH_EVENT ~ ejection_fraction + serum_creatinine, data = data,
```

```
##     prior = c(0.4, 0.6))
##
## Prior probabilities of groups:
##   0   1
## 0.4 0.6
##
## Group means:
##   ejection_fraction serum_creatinine
## 0          40.26601         1.184877
## 1          33.46875         1.835833
##
## Coefficients of linear discriminants:
##                            LD1
## ejection_fraction -0.06152077
## serum_creatinine   0.77268584
```

```
lda.pred = predict(lda.mod,subset(data,select=-DEATH_EVENT));
confusionMatrix(as.factor(lda.pred$class),as.factor(data$DEATH_EVENT),positive='1');
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  94  17
##          1 109  79
##
##                Accuracy : 0.5786
##                  95% CI : (0.5204, 0.6352)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.2283
##
##  Mcnemar's Test P-Value : 5.192e-16
##
##             Sensitivity : 0.8229
##             Specificity : 0.4631
##          Pos Pred Value : 0.4202
##          Neg Pred Value : 0.8468
##              Prevalence : 0.3211
##          Detection Rate : 0.2642
##    Detection Prevalence : 0.6288
##       Balanced Accuracy : 0.6430
##
##        'Positive' Class : 1
##
```

If one does not care that much about specificity, this is the best model by now.

K-nearest neighbor classifier

We decided to try also a knn classifier. To assest a performance measure we used LOOCV.

```
knn_classifier <- knn.cv(train = data_num,cl = data_num$DEATH_EVENT,k=3);

confusionMatrix(as.factor(knn_classifier),as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173  62
##          1  30  34
##
##                Accuracy : 0.6923
##                  95% CI : (0.6366, 0.7442)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 0.334542
##
##                   Kappa : 0.2263
##
##  Mcnemar's Test P-Value : 0.001229
##
##             Sensitivity : 0.3542
##             Specificity : 0.8522
##          Pos Pred Value : 0.5312
##          Neg Pred Value : 0.7362
##              Prevalence : 0.3211
##          Detection Rate : 0.1137
##    Detection Prevalence : 0.2140
##       Balanced Accuracy : 0.6032
##
##        'Positive' Class : 1
##
```

The performance on the test set does not seems to be good. The situation in this case gets worse if we use more neighbors because of the unbalance of the data (it would predict a lot of false negatives). We thought about some possible solutions to this problem: instead of using a majority rule to decide the classification, one can try to compensate this unbalance by deciding to classify one person as "dead" if the percentage of its "dead neighbors" is bigger than the percentage of dead people observed in the whole dataset. In this way we can also use a little bit more neighbors without destroying the performance.

```
knn_classifier <- knn.cv(train =data_num,
                         cl=data_num$DEATH_EVENT,k=4,prob = TRUE);

threshold = sum(DEATH_EVENT==1)/nrow(data_num);

prob.class <-attributes(knn_classifier)$prob;

classif <- knn_classifier[1:nrow(data_num)];

neigh_death_percentage = c()

for(i in 1:length(prob.class)){

  if(classif[i]==1){

    neigh_death_percentage = c(neigh_death_percentage,prob.class[i])

  }else{

     neigh_death_percentage = c(neigh_death_percentage,1-prob.class[i])
```

```
  }

}

biased.class <-hard_classif(neigh_death_percentage,threshold);

confusionMatrix(as.factor(biased.class),as.factor(data_num$DEATH_EVENT),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  73  24
##          1 130  72
##
##                Accuracy : 0.4849
##                  95% CI : (0.427, 0.5432)
##     No Information Rate : 0.6789
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0849
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7500
##             Specificity : 0.3596
##          Pos Pred Value : 0.3564
##          Neg Pred Value : 0.7526
##              Prevalence : 0.3211
##          Detection Rate : 0.2408
##    Detection Prevalence : 0.6756
##       Balanced Accuracy : 0.5548
##
##        'Positive' Class : 1
##
```

Ensemble of logistic classifiers

The last thing we tried (just for curiosity cause it has been mentioned in class) to improve the performace was to try ensemble methods, in particular we tried with a random forest of logistic regressors. In this model every classifier is trained using bagging. We tried to solve the unbalance problem between death and alive people by putting a stronger prior on the probability of death (0.6 in our case).

```
set.seed(42)

rf = randomForest(formula=as.factor(DEATH_EVENT)~ejection_fraction+
                serum_creatinine+age+creatinine_phosphokinase,
                ntree=200,data=data_num,
                replace=FALSE, classwt=c(0.4,0.6))




print(rf$confusion)
```

```
##     0  1 class.error
## 0 172 31   0.1527094
## 1  45 51   0.4687500
```

To see an unbiased version of the error, we can look at the out of bag confusion matrix provided by the random forest. This confusion matrix tells us that with this prior, the false negative rate is around 38% and the false positive rate is around 21.04% (with 200 trees). The conclusion is that we have not been able to solve the unbalance problem in the random sampling, this model is probably not adequate for this dataset without other shrewdness.

Discussion of the results and summary

In conclusion, we are now giving our answers to the questions posed at the beginning. To us, this dataset is not sufficient to make good risk prediction for two main reasons: the first one is that probably the collected features are not sufficient to explain the risk of dying from heart failure and the second is a lack of observations. Even if we would have more features, it is not easy to asset if the models are actually useful with this small number of observations. It is also accurately same even if one tries to get around this problem by using LOOCV or k-fold cross validations (because of the unbalance). We have been able to extract from the features the ones that seems more useful for predictions, and the model selected by BIC seems to be the good compromise. Moreover, the two variable models work surprisingly well given the complexity. The best model we found in terms of sensitivity is LDA, even thought specificity and overall accuracy are pretty low. Just to be more schematic, we put here a table with the performance of all the model for prediction we tried.

model / performance	train accuracy	train sensitivity(TPR)	train specificity(TNR)
BIC Logistic regression	0.7692	0.6979	0.8030
BIC (serum creatinine+ejection fraction)	0.7826	0.7083	0.8177
AIC Logistic regression	0.7525	0.7917	0.7340
LASSO 0.1	0.7258	0.7396	0.7192
Ridge Logistic regression	0.7458	0.8021	0.7192
Polynomial Logistic regression (quadratic)	0.7726	0.7292	0.7931
Polynomial Logistic regression (cubic)	0.8127	0.7604	0.8374
LDA	0.6656	0.8021	0.6010
LDA (serum creatinine+ejection fraction)	0.5786	0.8229	0.4631
QDA	0.6756	0.8021	0.6158
knn classifier	0.4849	0.7500	0.3596
Random forest	0.7458	0.6219	0.7926