Can we predict Heart Failure?

Zanotti Federico

Introduction

I have analyzed a dataset containing the medical records of 299 Heart Failure patients collected at the Faisalabad Institute of Cardiology and at the Allied Hospital in Faisalabad (Punjab, Pakistan), during April–December 2015 . The patients consists of 105 women and 194 men, and their ages range between 40 and 95 years old. The dataset contains 13 features:

- `Age` : age of patient
- `Anaemia`: binary variable that indicates if a patient has anaemia
- `Creatinine Phosphokinase`: states the level of the CPK enzyme in blood
- `Diabetes`: binary variable that indicates if a patient has diabetes
- `Ejection Fraction`: it states the percentage of how much blood the left ventricle pumps out with each contraction
- `High Blood Pressure`: binary variable that indicates if a patient has high_blood_pressure
- `Platelets`: component of blood whose function (along with the coagulation factors) is to react to bleeding from blood vessel injury by clumping, thereby initiating a blood clot.
- `Serum Creatinine`: waste product generated by creatine, when a muscle breaks down
- `Serum Sodium`: mineral that serves for the correct functioning of muscles and nerves
- `Sex`: binary variable that indicates if a patient is male or female
- `Smoking`: binary variable that indicates if a patient smokes
- `Time`: follow-up period
- `Death Event`: states if the patient died or survived before the end of the follow-up period

Unfortunately, there is no description on how smoking and high blood pressure are obtained, especially for the first one. In fact there is no parameter that indicates how long a patient has been smoking or how many cigarettes he has smoked per day

Cleaning and Filtering Data:

First of all we have to load the data and transform categorical variable in factor variable. To simplify this step I have defined a function ad hoc:

```
load<- function(path, levels='yes') {
  my_data <- read.csv(path, header=TRUE)
  my_data$anaemia<- as.factor(my_data$anaemia)
  my_data$diabetes<- as.factor(my_data$diabetes)
  my_data$high_blood_pressure<- as.factor(my_data$high_blood_pressure)
  my_data$smoking<-as.factor(my_data$smoking)
  if (levels=='yes') {
    my_data$sex<- as.factor(my_data$sex)
    levels(my_data$sex) = c("female","male")
    my_data$DEATH_EVENT<- as.factor(my_data$DEATH_EVENT)
    levels(my_data$DEATH_EVENT) = c("not dead","dead")
  }
```

```
  my_data
}
```

The load function read the file where the dataset is stored and then it transforms all categorical variables in factor variables. In particular for sex and death_event it adds also the levels.

```
data <- load(path)
attach(data)
head(data)
```

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

The next step is to check for NA values

```
dim_old<-dim(data)
data<-na.omit(data)
dim_new<-dim(data)
if (dim_old == dim_new) {
  print("No observations deleted, Dataset is clean")
} else {
  print("I have deleted some rows")
}
```

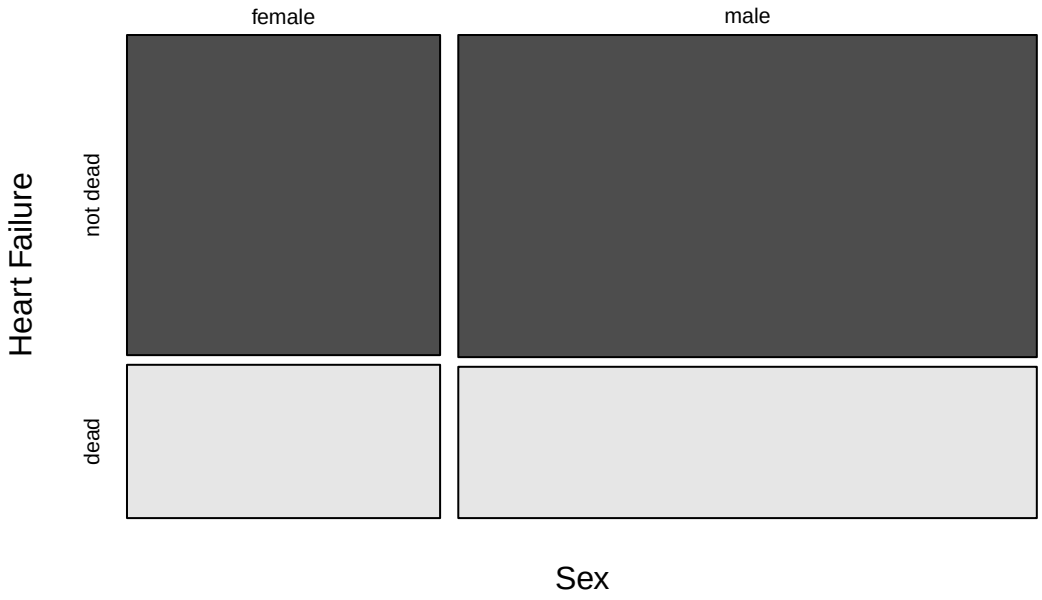
```
## [1] "No observations deleted, Dataset is clean"
```

Data Analysis

In this section I analyzed some features of the dataset in order to find some patterns and exploit them in the modeling phase.

From the plot below the dataset is balanced for male and female related to death event feature

```
mosaicplot(sex ~ DEATH_EVENT,
           main="Sex with Hearth Failure", shade=FALSE,color=TRUE,
           xlab="Sex", ylab="Heart Failure")
```



While we have a different situation for the death event feature alone, because there are more people with no Heart Failure:

Total number:

```
kable(table(DEATH_EVENT),  col.names = c("Heart Failure", "Count"))
```

Heart Failure	Count
not dead	203
dead	96

Percentage:

```
kable(round(prop.table((table(DEATH_EVENT)))*100,2),  col.names = c("Heart Failure", "Percentage"))
```

Heart Failure	Percentage
not dead	67.89
dead	32.11

In the next results it can be seen that the dataset is not very balanced:

```
kable(round(prop.table((table(sex)))*100,2), col.names = c("Sex", "Percentage"))
```

Sex	Percentage
female	35.12
male	64.88

```
kable(round(prop.table((table(high_blood_pressure)))*100,2), col.names = c("High Blood Pressure", "Perc
```

High Blood Pressure	Percentage
0	64.88
1	35.12

```
kable(round(prop.table((table(diabetes)))*100,2), col.names = c("Diabetes", "Percentage"))
```

Diabetes	Percentage
0	58.19
1	41.81

```
kable(round(prop.table((table(anaemia)))*100,2), col.names = c("Anaemia", "Percentage"))
```

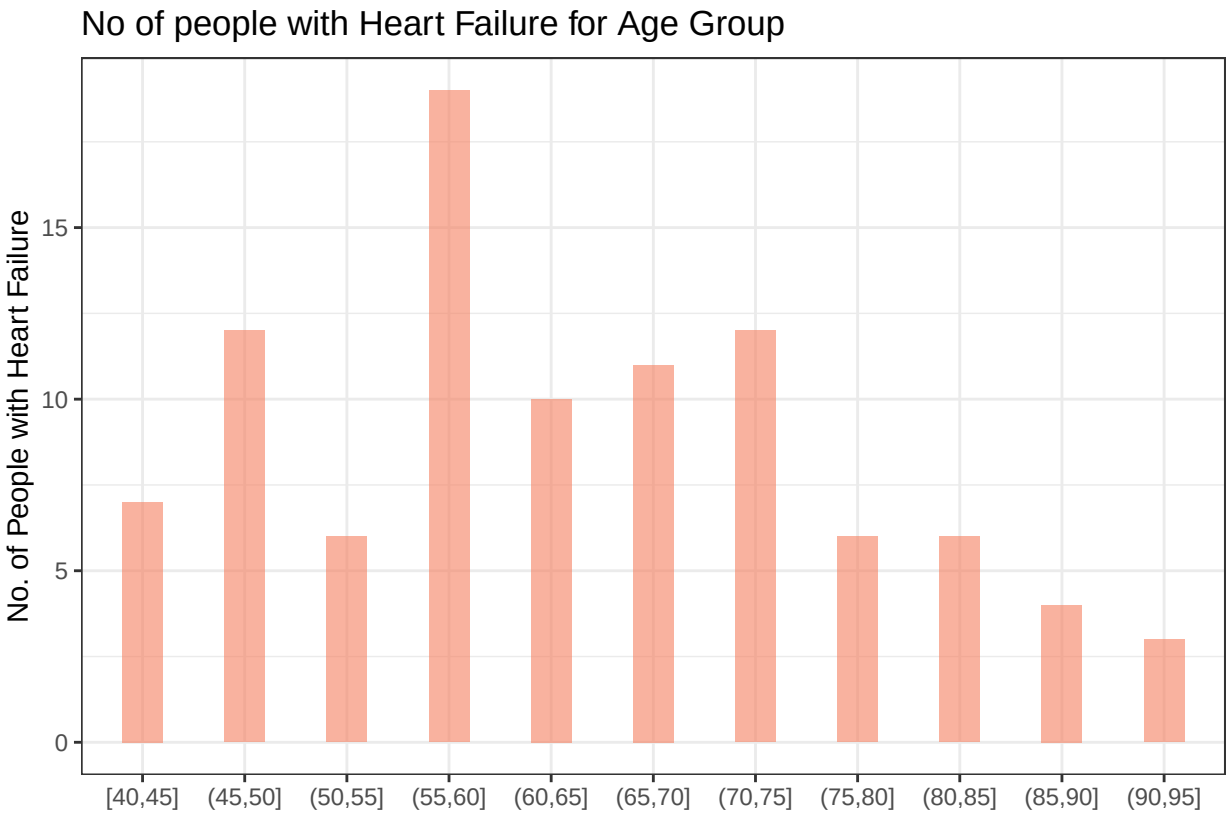
Anaemia	Percentage
0	56.86
1	43.14

```
kable(round(prop.table((table(smoking)))*100,2), col.names = c("Smoking", "Percentage"))
```

Smoking	Percentage
0	67.89
1	32.11

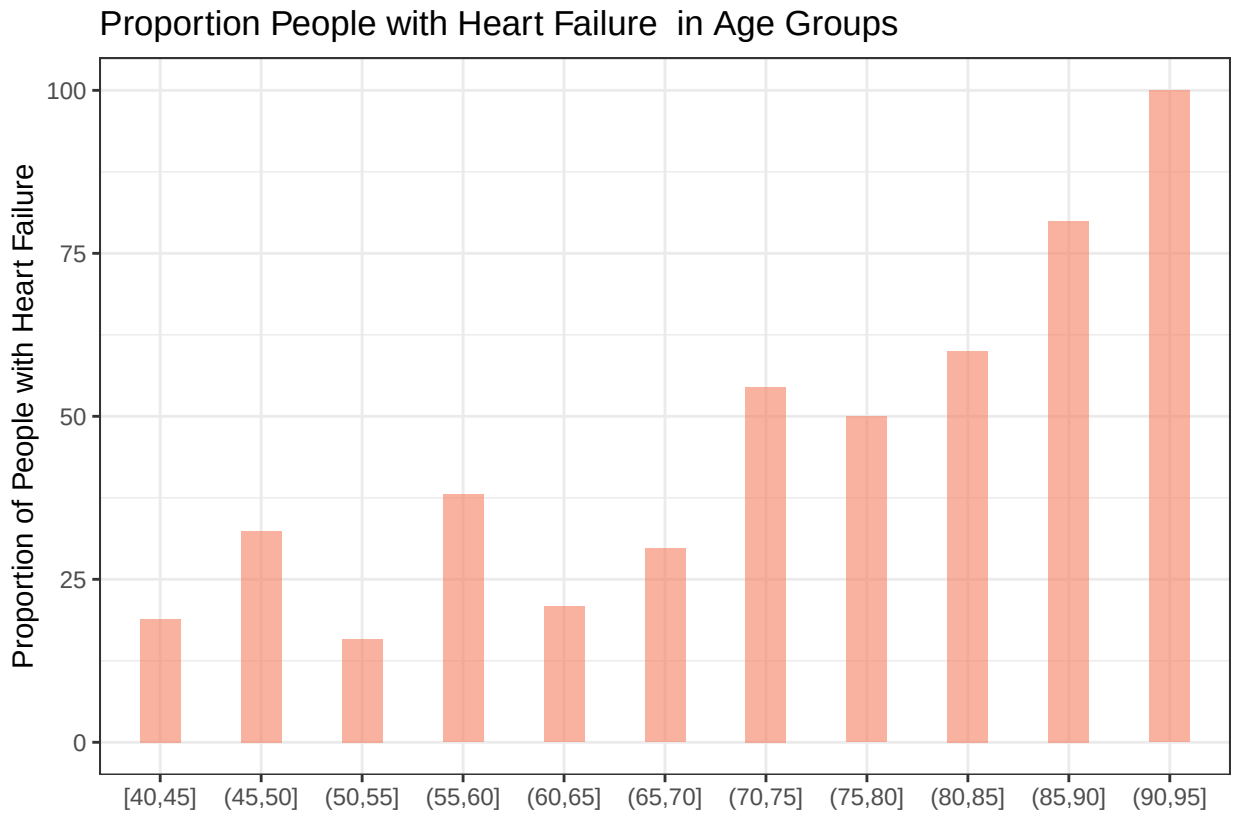
An important value to consider is how Heart Failure is distributed for the age of the patients. Ideally, there would be more old people with Heart Failure respect to younger people, but in our case it's different:

```
target_by_age %>%
  ggplot(aes(x=age_grp, y=dead)) +
    geom_bar(stat="identity", fill="#f68060", alpha=.6, width=.4) +
    xlab("") + ylab("No. of People with Heart Failure") + ggtitle("No of people with Heart Failure for
    theme_bw()
```



We have more people with Heart Failure in the range 55-60, but to have a complete view is better to consider the proportion for each age

```
prop_in_age %>%
  ggplot(aes(x=age_grp, y=dead_proportion)) +
  geom_bar(stat="identity", fill="#f68060", alpha=.6, width=.4) +
  xlab("") + ylab("Proportion of People with Heart Failure") + ggtitle("Proportion People with Heart Fa
  theme_bw()
```



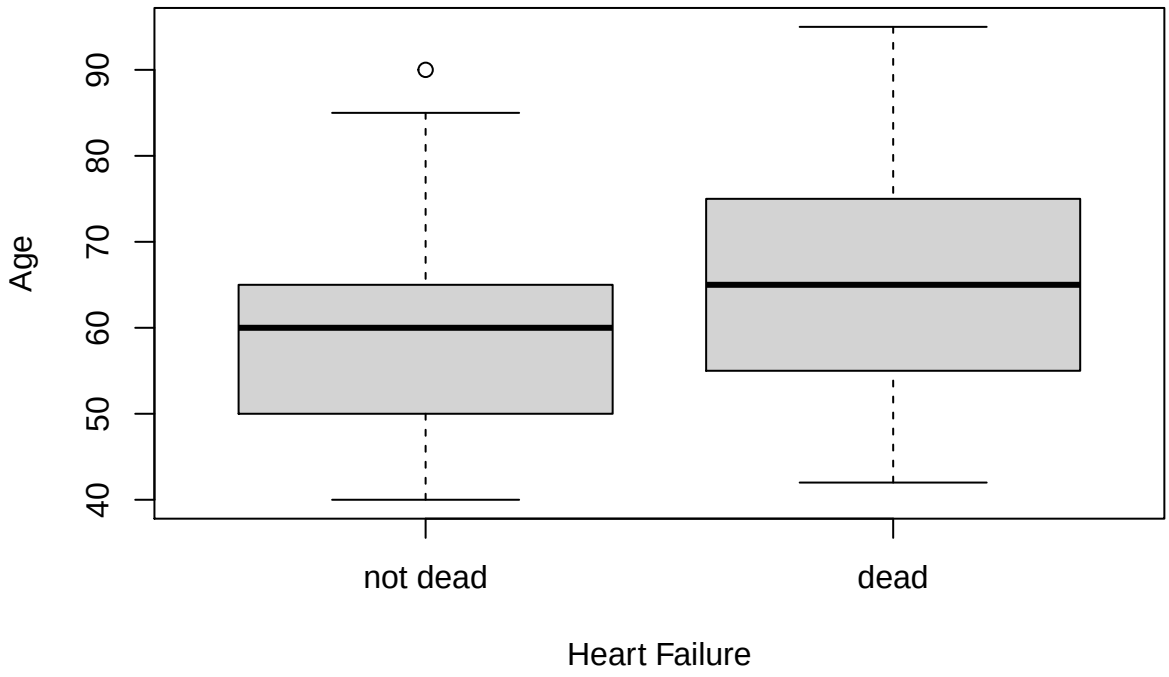
The proportion shows, in contrast to the previous barplot, that age is an important parameter to consider in the case of Heart Failure, because it increases when the age increases. This result could be misleading, because the age feature is unbalanced (as shown below), in fact we have only 3 people in age from 90 to 95 and they all have death event = 1

```
target_by_age
```

```
## # A tibble: 11 x 2
##    age_grp  dead
##    <fct>   <int>
##  1 [40,45]     7
##  2 (45,50]    12
##  3 (50,55]     6
##  4 (55,60]    19
##  5 (60,65]    10
##  6 (65,70]    11
##  7 (70,75]    12
##  8 (75,80]     6
##  9 (80,85]     6
## 10 (85,90]     4
## 11 (90,95]     3
```

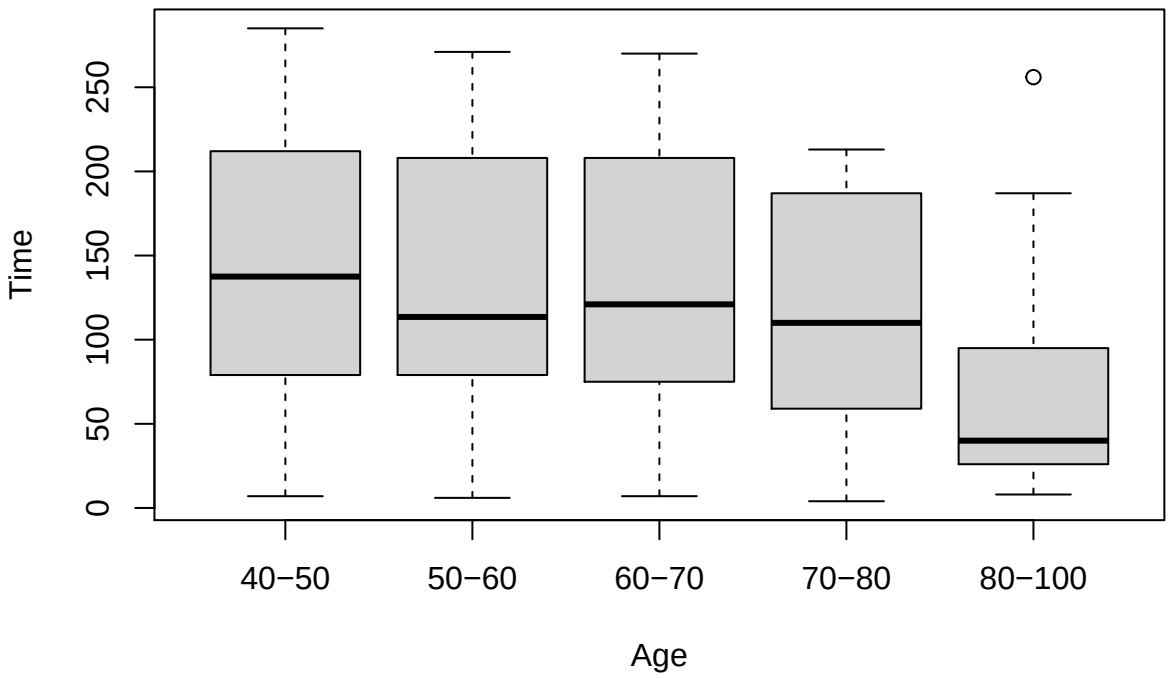
Despite the analysis just discussed on the unbalanced parameter age, in the following cell we can see that age has an impact on Heart Failure, so for the modeling we should take it into account

```
boxplot(age~DEATH_EVENT,  xlab = "Heart Failure", ylab= "Age")
```



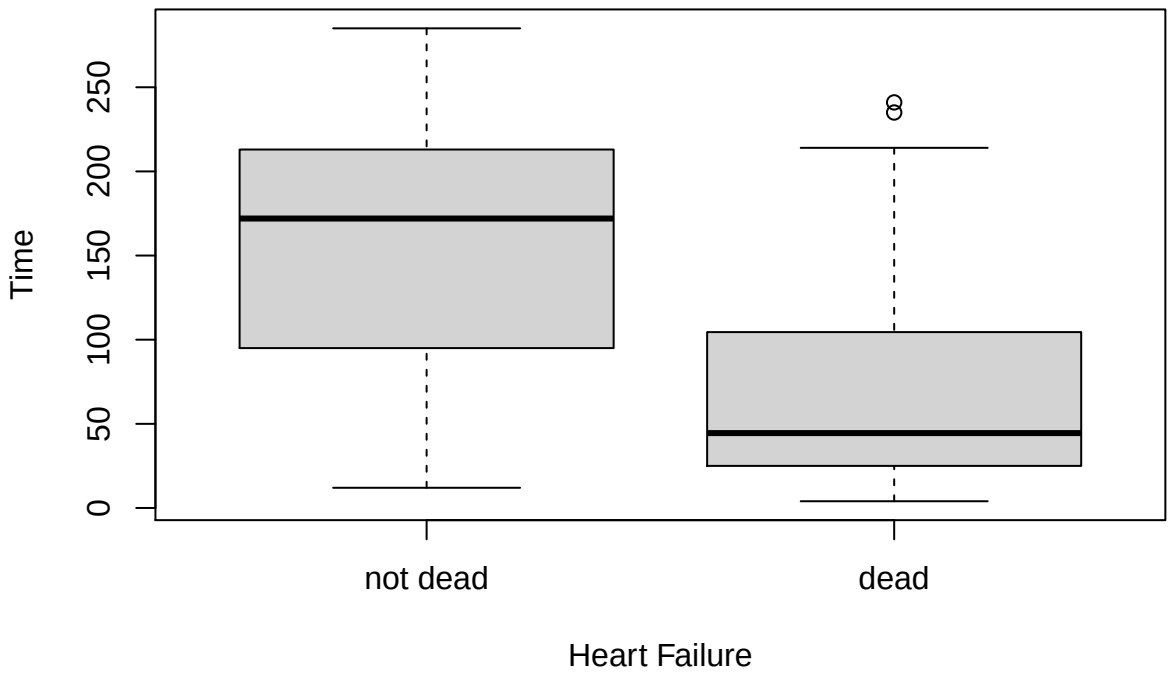
In the plot below we can see also that time is an important value, because the median changes for all the ages, in particular it drops dramatically for people with high age. This could mean that for the old people the survival time is very low, less than 2 months

```
data$age_group <- ''

data<-data %>%
  mutate(age_group = case_when(age<=50 ~ '40-50',
                               age<=60 ~ '50-60',
                               age<=70 ~ '60-70',
                               age<=80 ~ '70-80',
                               age<=100 ~ '80-100'
  ))

boxplot(time~data$age_group,  xlab = "Age", ylab= "Time")
```



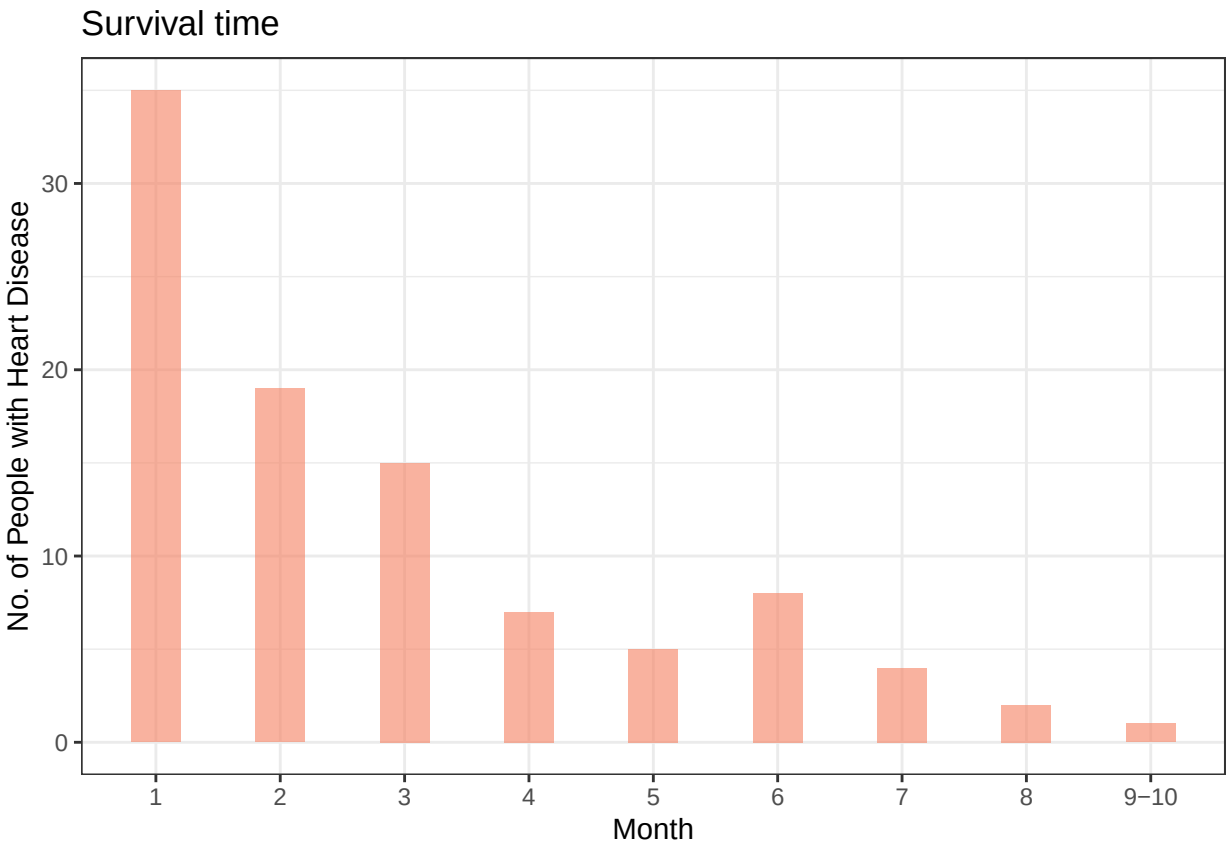
Moreover survival time is low for patients with Hearth Failure

```
boxplot(data$time~DEATH_EVENT, xlab = "Heart Failure", ylab= "Time")
```



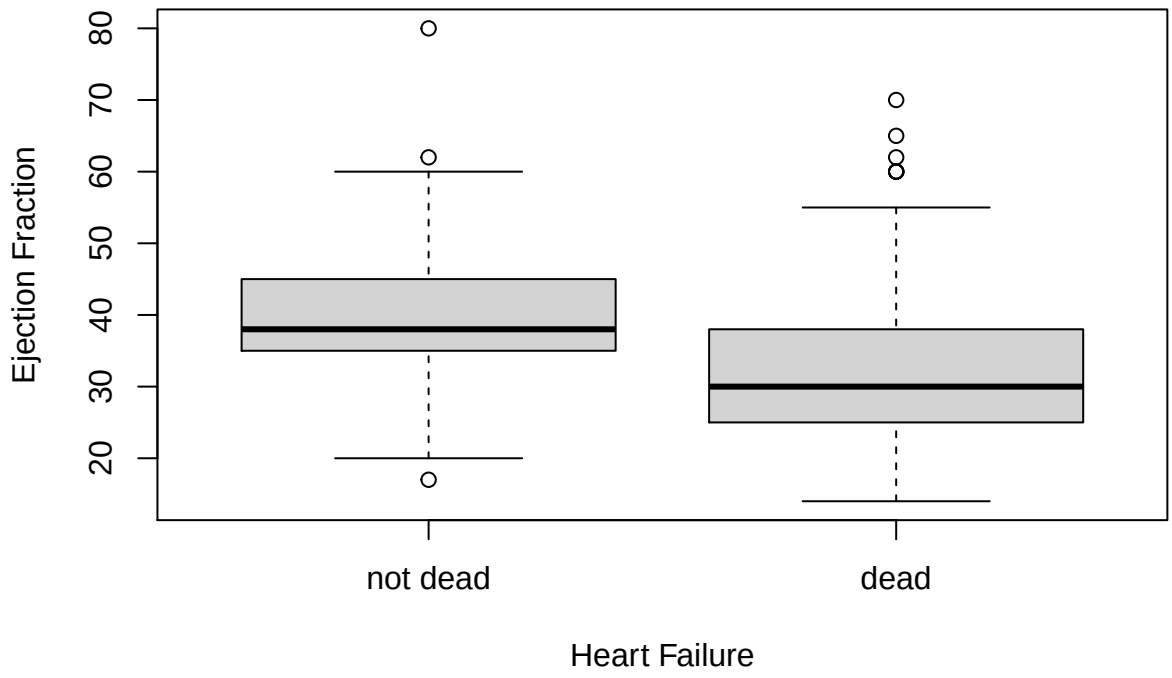
Another evidence that time is significant is the distribution of it respect to people with Heart Failure. This can be seen in the following bar plot, where there are a lot of patients that had an Heart Failure in the early months and the distribution is linear, indicating a strong effect of time parameter

```
data_time_group <- load(path, levels="no")
data_time_group$time_group <- ''
data_time_group<-data_time_group %>%
  mutate(time_group = case_when(time<=30 ~ '1',
                                time<=60 ~ '2',
                                time<=90 ~ '3',
                                time<=120 ~ '4',
                                time<=150 ~ '5',
                                time<=180 ~ '6',
                                time<=210 ~ '7',
                                time<=240 ~ '8',
                                time<=270 ~ '9-10',  ))
data_time_group<-na.omit(data_time_group)

target_time_grp_2 = data_time_group %>%
  group_by(time_group) %>%
  summarise(dead = sum(DEATH_EVENT))

target_time_grp_2 %>%
  ggplot(aes(x=time_group, y=dead)) +
  geom_bar(stat="identity", fill="#f68060", alpha=.6, width=.4) +
  xlab("Month") + ylab("No. of People with Heart Disease") + ggtitle("Survival time") +
  theme_bw()
```



Another feature related to death event is ejection fraction that is less on average for people with Heart Failure, so when the percentage of blood pumped out by the left ventricle is low then there could be an Hearth Failure. For this reason ejection fraction is a relevant feature to investigate

```
boxplot(ejection_fraction~DEATH_EVENT,  xlab = "Heart Failure", ylab= "Ejection Fraction")
```



From the results below we can see that the categorical variable referred to Heart Failure feature are a little balanced except for smoking, but we don't know if it depends by how data have been gathered or smoking has a negative effect on Heart Failure. FDA (U.S Food & Drug Administration) asserts that smoking can cause Heart Failure, but it's not the principal cause.

```
check_balance("smoking", smoking)
```

```
##  Proportion of patients died without  smoking :  68.75
##  Proportion of patients died with  smoking :  31.25
```

```
check_balance("anaemia", anaemia)
```

```
##  Proportion of patients died without  anaemia :  52.08
##  Proportion of patients died with  anaemia :  47.92
```

```
check_balance("diabetes", diabetes)
```

```
##  Proportion of patients died without  diabetes :  58.33
##  Proportion of patients died with  diabetes :  41.67
```

```
check_balance("high blood pressure", high_blood_pressure)
```

```
##  Proportion of patients died without  high blood pressure :  59.38
##  Proportion of patients died with  high blood pressure :  40.62
```

```
check_balance("anaemia", anaemia)
```

```
##  Proportion of patients died without  anaemia :  52.08
##  Proportion of patients died with  anaemia :  47.92
```

Model Data

The most important feature in this dataset is Death Event: binary variable that states if a patient is died in the follow up time or not. For the modeling phase I have used this feature as my response, in order to see if there are some medical parameters that can cause Heart Failure.

The first problem to face is the parameter time. The analysis shows that time is significant when related to death event, but what I would like to retrieve are the most important medical features and their values as a way to prevent an Heart Failure. Time tells us that in the early months there are more people dead, and this could lead to misinterpret medical features, therefore I have considered model with and without time variable.

A second problem is which metrics use to assert that the model is well-defined. The classical accuracy (Number of predicted right/total) is not a good choice, because the dataset is unbalanced (we have less people with Heart Failure) and if we created a trivial test with all the patients not dead, then our model would be very good, but if we did another test, with all patients dead, then our model would be very bad. To overcome this problem I have considered three metrics taken from the Confusion Matrix:

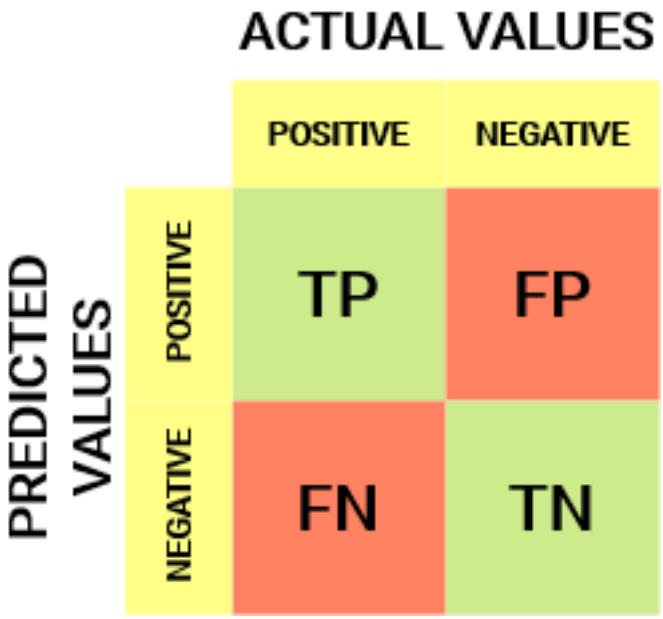


Figure 1: Confusion Matrix

- **precision**

$$\frac{TP}{TP + FP}$$

- **recall**

$$\frac{TP}{TP + FN}$$

- **f1-score**

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

Precision attempts to answer to this question: What proportion of positive identifications was actually correct?

Recall measures the proportion of positives that are correctly identified.

F1-score is the harmonic mean of precision and Recall and it represents the trade off among them.

In our case Precision represents the proportion of good prediction when we predict Hearth failure, while Recall is the proportion of those who have Hearth Failure who are correctly identified as having it.

Due to the unbalance of the dataset, accuracy is not a reliable measure, while recall represents the best way to measure the performance of our model, because it identifies Hearth Failure in patients that actually have it. As a consequence, also for a practical point, Recall is more important. In order to be more complete and also to present more models for the purpose of choosing the appropriate one on the basis of other measures (like precision) I always report all the metrics.

Moreover I have used ROC curve to analyze the trade off between Recall and Specificity

In this project I have created 3 models:

- **GLM**: Generalized Linear Model generalizes linear regression by allowing the linear model to be related to the response variable via a link function
- **LDA**: Linear discriminant analysis is a classification method that approximates the Bayes Classifier
- **QDA**: Quadratic discriminant analysis (QDA) is a variant of LDA that allows for non-linear separation of data. It is a more general version of the linear classifier.

Training and Test

For every model I have divided the data in train (70%) and test (30%). In the three following cells I report the function to split the data and the creation of Train and Test set without and with time variable

```
split_train_test <- function(data) {
  ## 70% of the sample size
  smp_size <- floor(0.7 * nrow(data))

  ## set the seed to make your partition reproducible
  set.seed(123)
  train_ind <- sample(seq_len(nrow(data)), size = smp_size)

  train <- data[train_ind, ]
  test <- data[-train_ind, ]
  return(list(train, test))
}
```

```
data<-load(path)
attach(data)
```

```
## The following objects are masked from data (pos = 3):
##
##     age, anaemia, creatinine_phosphokinase, DEATH_EVENT, diabetes,
##     ejection_fraction, high_blood_pressure, platelets,
##     serum_creatinine, serum_sodium, sex, smoking, time
```

```
data<-data[,-12] # Remove time column
l<-split_train_test(data)
train<-l[[1]]
test<-l[[2]]
cat("Number of rows of Train set WITHOUT time: ", nrow(train), "\n")
```

```
## Number of rows of Train set WITHOUT time:  209
```

```
cat("Number of rows of Test set WITHOUT time: ", nrow(test))
```

```
## Number of rows of Test set WITHOUT time:  90
```

```
data.time<-load(path)
l<-split_train_test(data.time)
train.time<-l[[1]]
test.time<-l[[2]]
cat("Number of rows of Train set WITH time: ", nrow(train.time), "\n")
```

```
## Number of rows of Train set WITH time:  209
```

```
cat("Number of rows of Test set WITH time: ", nrow(test.time))
```

```
## Number of rows of Test set WITH time:  90
```

some utility functions

In this paragraph I add some utility functions, that are useful for the modeling phase

Plot_ROC Function

This function plots the ROC curve of the model

```
plot_ROC <- function(variable, predictions, levels, add=FALSE, col='black', noauc=TRUE) {
  roc.out <- roc(variable, predictions, levels=levels)
  plot(roc.out,  print.auc=noauc, legacy.axes=TRUE, xlab="False positive rate", ylab="True positive rat
         add=add, col=col)

  coords(roc.out, "best")
}
```

Metrics Function

These functions calculate the metrics needed for the problem

```
my_precision<- function(matrix) {  # first varibale, second predictions in table
  # True positive
  tp <- matrix[2, 2]
  # false positive
  fp <- matrix[1, 2]
  return (tp / (tp + fp))
}

my_recall<- function(matrix) {
  tp <- matrix[2, 2]
  fn <- matrix[2, 1]
  return (tp / (tp + fn))
}


metrics <- function(matrix) {
  acc= sum(diag(matrix)) / sum(matrix)
  p = my_precision(matrix)
  r = my_recall(matrix)
  f1 =  2 * ((p * r) / (p + r))
  d<-data.frame("Accuracy"= acc, "Precision"=p, "Recall"=r, "F1 score"=f1)
  d
}
```

GLM

Without time

From the summary below we can see that age, ejection fraction and serum creatinine are important, because their p-value is very low, so very significant.

```
glm.fits<-glm(DEATH_EVENT~., family=binomial, data=data)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ ., family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3184  -0.7692  -0.4436   0.8293   2.4880
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                4.964e+00  4.601e+00   1.079 0.280625
## age                        5.569e-02  1.313e-02   4.241 2.23e-05 ***
## anaemia1                   4.179e-01  3.009e-01   1.389 0.164904
## creatinine_phosphokinase   2.905e-04  1.428e-04   2.034 0.041907 *
## diabetes1                  1.514e-01  2.974e-01   0.509 0.610644
## ejection_fraction         -7.032e-02  1.486e-02  -4.731 2.23e-06 ***
## high_blood_pressure1       4.189e-01  3.061e-01   1.369 0.171092
## platelets                 -7.094e-07  1.617e-06  -0.439 0.660857
## serum_creatinine           6.619e-01  1.734e-01   3.817 0.000135 ***
## serum_sodium              -5.667e-02  3.338e-02  -1.698 0.089558 .
## sexmale                   -3.990e-01  3.508e-01  -1.137 0.255394
## smoking1                   1.356e-01  3.486e-01   0.389 0.697300
## ---
## 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: 294.28  on 287  degrees of freedom
## AIC: 318.28
##
## Number of Fisher Scoring iterations: 5
```

In order to retrieve the best features that impact on Heart Failure, I have applied this procedure:

1. check if there is a variable with too high p-value
2. remove the variable with the biggest p-value
3. continue until there is no variable with high p-value

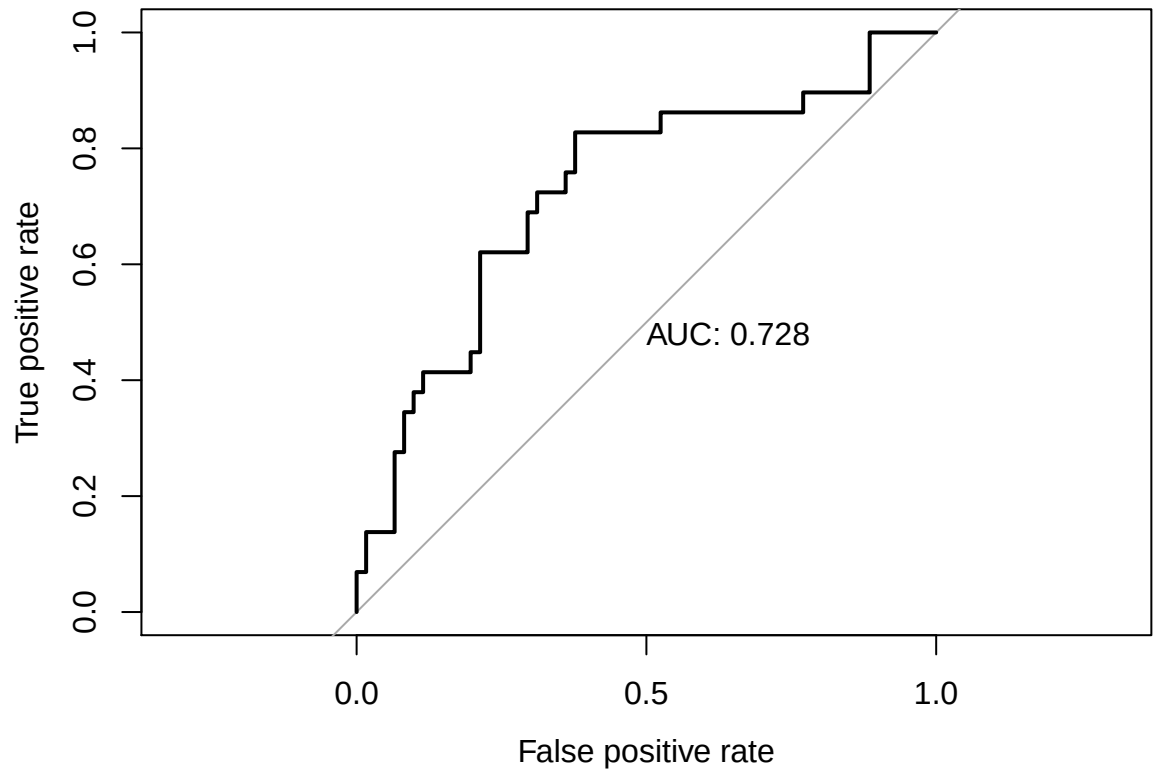
This procedure has led to consider Intercept, age, ejection fraction and serum creatinine

```
formula<-DEATH_EVENT~.-anaemia- creatinine_phosphokinase- smoking-sex-serum_sodium-
  platelets-high_blood_pressure- diabetes
glm.fits<-glm(formula, family=binomial, data=train)
summary(glm.fits)
```

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

```
## glm(formula = formula, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5682  -0.7531  -0.4793   0.7888   2.5253
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.16264    1.01049  -2.140   0.0323 *
## age                0.05672    0.01525   3.719   0.0002 ***
## ejection_fraction -0.09029    0.01909  -4.728 2.26e-06 ***
## serum_creatinine   0.79863    0.19100   4.181 2.90e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 262.21  on 208  degrees of freedom
## Residual deviance: 201.45  on 205  degrees of freedom
## AIC: 209.45
##
## Number of Fisher Scoring iterations: 5
```

```
predictions_glm<-predict(glm.fits, test, type="response")

p<-plot_ROC(test$DEATH_EVENT, predictions_glm, levels=c("not dead", "dead"))
```



```
cat("best threshold: ", p[[1]])
```

```
## best threshold:  0.2265663
```

```
table_mat_standard<-table(test$DEATH_EVENT, predictions_glm>0.5)
best_treshold = p[[1]]
table_mat_best <-table(test$DEATH_EVENT, predictions_glm>best_treshold)
```

From the plot above we can see that the ROC curve has not an high value, showing that we reached poor recall and specificity. ROC curve also suggests a threshold to improve the metrics.

The metrics of this model reached in the test set are in the following two cells

```
m1<-metrics(table_mat_standard)
m2<-metrics(table_mat_best)
```

From the results in the next cell it can be seen that the GLM model with threshold suggested by ROC reach a good recall, while precision slightly decreases and F1-score increases

Accuracy	Precision	Recall	F1.score
0.7	0.5454545	0.4137931	0.4705882

Accuracy	Precision	Recall	F1.score
0.6888889	0.5106383	0.8275862	0.6315789

With time

```
data.time<-load(path)

l<-split_train_test(data.time)
train.time<-l[[1]]
test.time<-l[[2]]
cat("Number of rows of Train set: ", nrow(train.time), "\n")
```

```
## Number of rows of Train set:  209
```

```
cat("Number of rows of Test set: ", nrow(test.time))
```

```
## Number of rows of Test set:  90
```

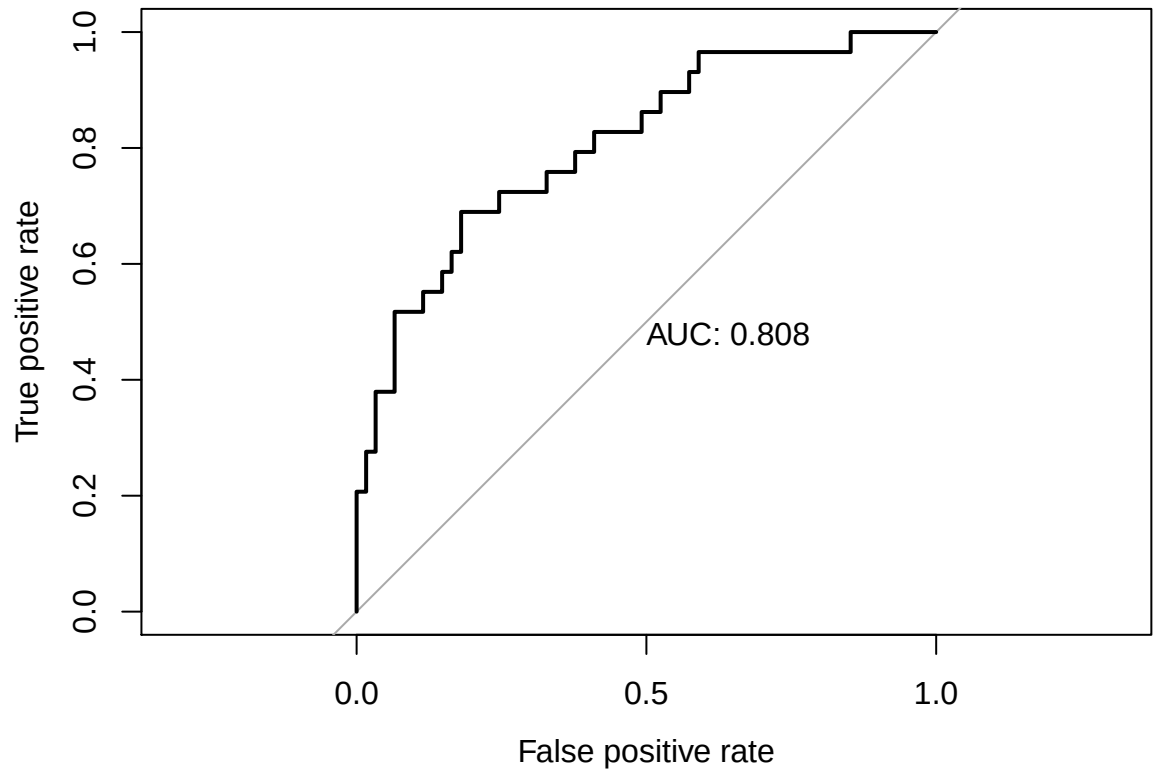
Here I did the same procedure of GLM without time, and I have obtained a different model, because now I have also time. An important thing to notice is that with time the ROC curve improves

```
formula.time<-DEATH_EVENT~.-anaemia- creatinine_phosphokinase- smoking-sex-serum_sodium-
  platelets-high_blood_pressure- diabetes-1
glm.fits.time<-glm(formula.time, family=binomial, data=train.time)
summary(glm.fits.time)
```

```
##
## Call:
## glm(formula = formula.time, family = binomial, data = train.time)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3321  -0.4658  -0.1597   0.3446   2.4563
```

```
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## age                0.063895   0.013414   4.763 1.90e-06 ***
## ejection_fraction -0.090609   0.020199  -4.486 7.27e-06 ***
## serum_creatinine   0.907882   0.211198   4.299 1.72e-05 ***
## time              -0.026066   0.004345  -5.999 1.99e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 289.74  on 209  degrees of freedom
## Residual deviance: 133.78  on 205  degrees of freedom
## AIC: 141.78
##
## Number of Fisher Scoring iterations: 6
```

```
predictions_glm.time<-predict(glm.fits.time, test.time, type="response")

p.time<-plot_ROC(test$DEATH_EVENT, predictions_glm.time, levels=c("not dead", "dead"))
```




```
table_mat_standard.time<-table(test.time$DEATH_EVENT, predictions_glm.time>0.5)

best_treshold = p.time[[1]]
table_mat_best.time <-table(test.time$DEATH_EVENT, predictions_glm.time>best_treshold)
```

```
m1.time<-metrics(table_mat_standard.time)
m2.time<-metrics(table_mat_best.time)
```

With time I don't reach a Recall measure like in GLM without time, but Precision is improved, and also the trade-off F1-score is improved. This model with time guarantees more balanced results (f1 score is higher)

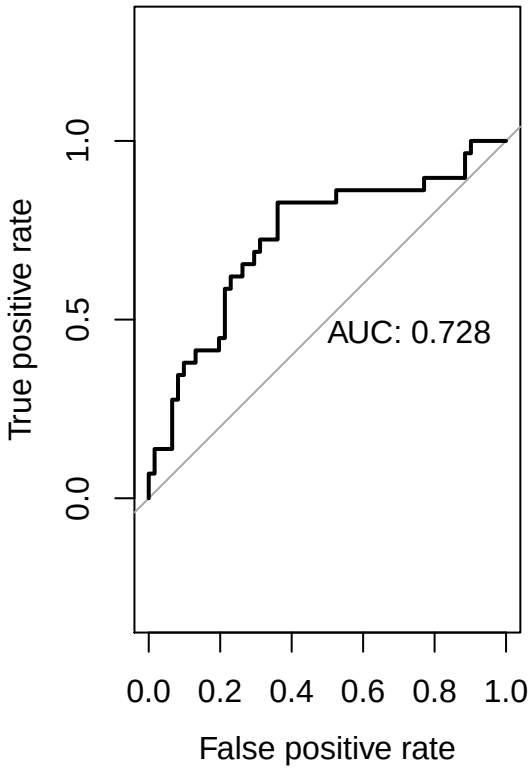
Accuracy	Precision	Recall	F1.score
0.7777778	0.6956522	0.5517241	0.6153846

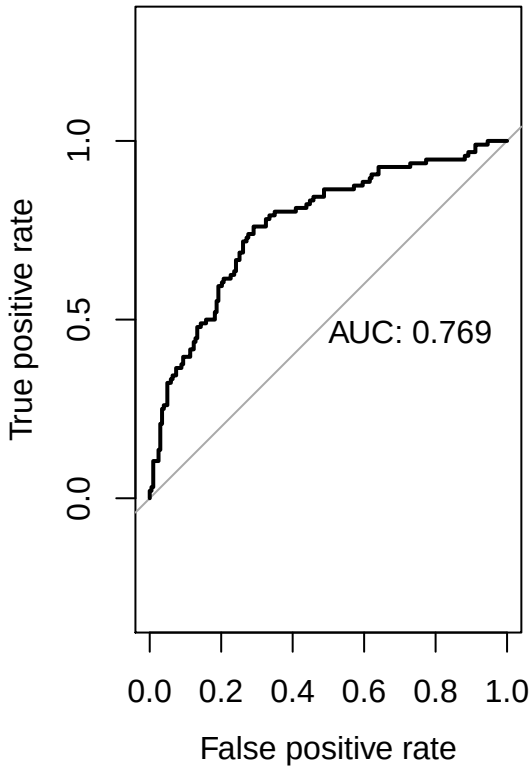
Accuracy	Precision	Recall	F1.score
0.7777778	0.6451613	0.6896552	0.6666667

LDA

In the cell below on the left there is ROC curve of LDA model and on the right there is LDA model used with Cross Validation. For the first ROC nothing changes with GLM, while in the second there is an improvement

```
formula<-DEATH_EVENT ~ . - anaemia - creatinine_phosphokinase - smoking -
  sex - serum_sodium - platelets - high_blood_pressure - diabetes

lda.fit <- lda(formula,data=train)
lda.fit.cv <-  lda(DEATH_EVENT~.,data=data, CV=TRUE)

lda.pred <- predict(lda.fit, test)

par(mfrow=c(1,2))


p.lda<-plot_ROC(test$DEATH_EVENT, lda.pred$posterior[,2], levels=c("not dead", "dead"))
p.lda.cv<-plot_ROC(data$DEATH_EVENT, lda.fit.cv$posterior[,2], levels=c("not dead", "dead"))
```






```
par(mfrow=c(1,1))
```

```
table_lda<-table(test$DEATH_EVENT, lda.pred$posterior[,2]>0.5)
table_lda_best_threshold<-table(test$DEATH_EVENT, lda.pred$posterior[,2]>p.lda[[1]] )
table_lda_CV<-table(data$DEATH_EVENT, lda.fit.cv$posterior[,2]>0.5)
table_lda_CV_best_threshold<-table(data$DEATH_EVENT, lda.fit.cv$posterior[,2]>p.lda.cv[[1]])

l1<-metrics(table_lda)
l1.b<-metrics(table_lda_best_threshold)
l1.cv<-metrics(table_lda_CV)
l1.cv.b<-metrics(table_lda_CV_best_threshold)
```

In the cell below we can see that:

- the maximum Precision is reached by LDA with Cross Validation
- the maximum Recall is reached by LDA with best threshold
- the maximum F1 Score is reached by LDA with Cross Validation and best threshold

```
## [1] "LDA"
```

Accuracy	Precision	Recall	F1.score
0.7111111	0.5714286	0.4137931	0.48

```
## [1] "LDA with best threshold"
```

Accuracy	Precision	Recall	F1.score
0.7	0.5217391	0.8275862	0.64

```
## [1] "LDA with Cross Validation"
```

Accuracy	Precision	Recall	F1.score
0.729097	0.6229508	0.3958333	0.4840764

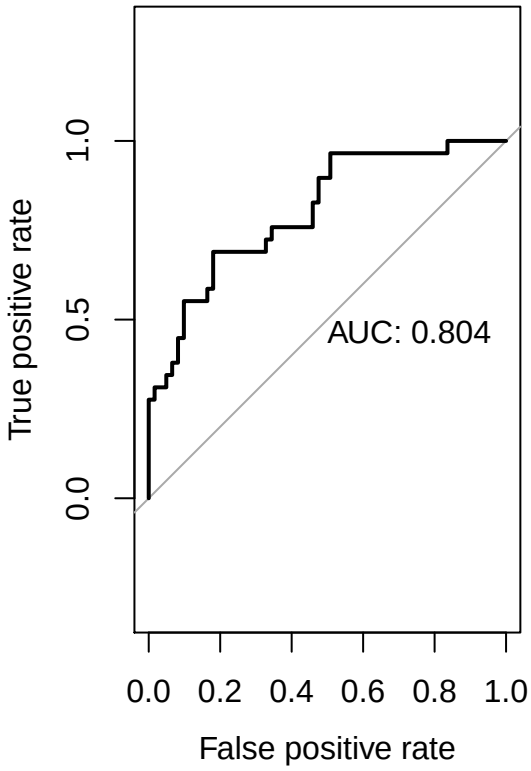
```
## [1] "LDA with Cross Validation and best threshold"
```

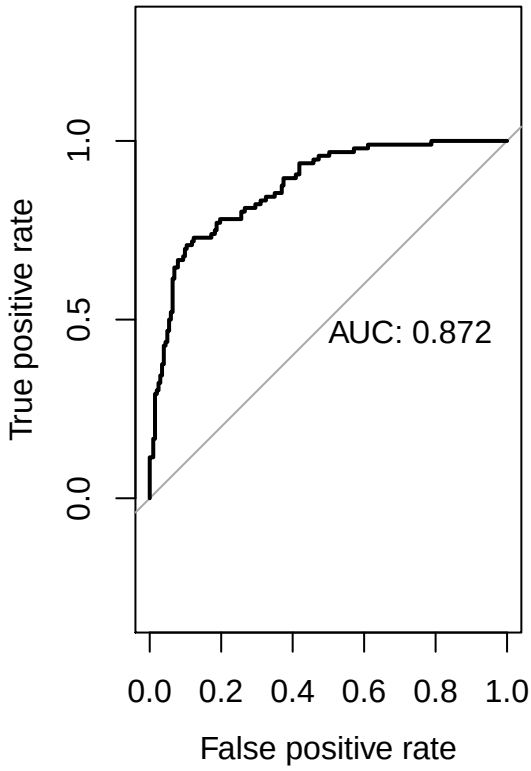
Accuracy	Precision	Recall	F1.score
0.7257525	0.5530303	0.7604167	0.6403509

LDA with Time

As in GLM we can see that ROC curves with the model that includes time are improved, especially for LDA with Cross Validation

```
formula<-DEATH_EVENT ~ . - anaemia - creatinine_phosphokinase - smoking -
  sex - serum_sodium - platelets - high_blood_pressure - diabetes

lda.fit.time <- lda(formula,data=train.time)
lda.fit.cv.time <-  lda(DEATH_EVENT~.,data=data.time, CV=TRUE)

lda.pred.time <- predict(lda.fit.time, test.time)



par(mfrow=c(1,2))


p.lda.time<-plot_ROC(test.time$DEATH_EVENT, lda.pred.time$posterior[,2], levels=c("not dead", "dead"))
p.lda.cv.time<-plot_ROC(data.time$DEATH_EVENT, lda.fit.cv.time$posterior[,2], levels=c("not dead", "dea
```






```
table_lda.time<-table(test.time$DEATH_EVENT, lda.pred.time$posterior[,2]>0.5)
table_lda_best_threshold.time<-table(test.time$DEATH_EVENT, lda.pred.time$posterior[,2]>p.lda.time[[1]]]

table_lda_CV.time<-table(data.time$DEATH_EVENT, lda.fit.cv.time$posterior[,2]>0.5)
table_lda_CV_best_threshold.time<-table(data.time$DEATH_EVENT, lda.fit.cv.time$posterior[,2]>p.lda.cv.t

l1.time<-metrics(table_lda.time)
l1.b.time<-metrics(table_lda_best_threshold.time)
l1.cv.time<-metrics(table_lda_CV.time)
l1.cv.b.time<-metrics(table_lda_CV_best_threshold.time)
```

In the cell below we can see that:

- the maximum Precision is reached by LDA with Cross Validation
- the maximum Recall is reached by LDA with Cross Validation and best threshold
- the maximum F1 Score is reached by LDA with Cross Validation and best threshold

```
## [1] "LDA"
```

Accuracy	Precision	Recall	F1.score
0.7777778	0.6956522	0.5517241	0.6153846

```
## [1] "LDA with best threshold"
```

Accuracy	Precision	Recall	F1.score
0.7777778	0.6451613	0.6896552	0.6666667

```
## [1] "LDA with Cross Validation"
```

Accuracy	Precision	Recall	F1.score
0.8294314	0.7647059	0.6770833	0.718232

```
## [1] "LDA with Cross Validation and best threshold"
```

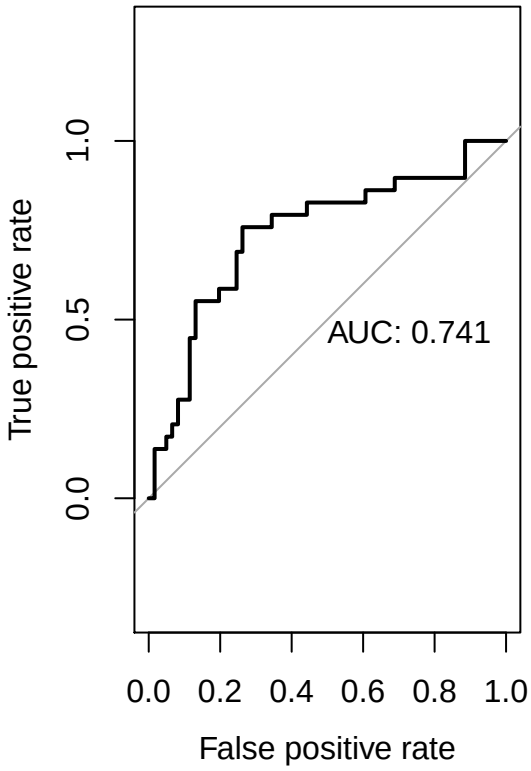
Accuracy	Precision	Recall	F1.score
0.8294314	0.7368421	0.7291667	0.7329843

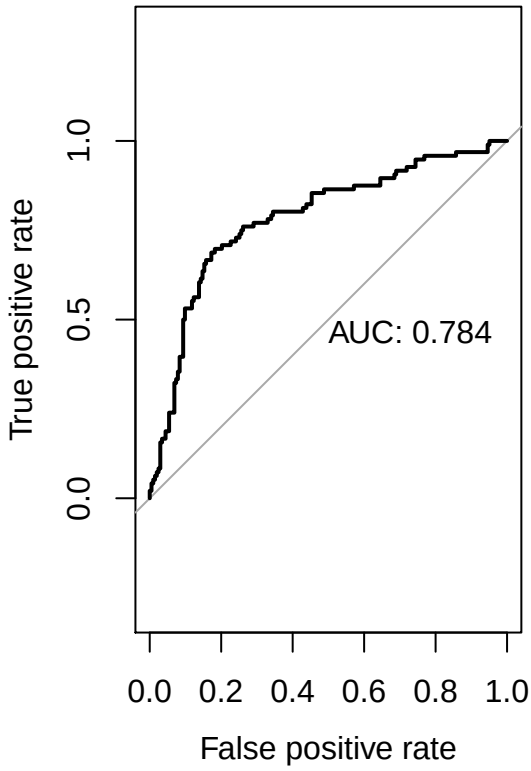
The difference from before now is that the different Precision measures of the models have higher values than LDA without time and as GLM, the maximum Recall is lower respect to LDA without time. A good result is that F1 Score increases, as in GLM with time, and this means that these are more balanced models

QDA

QDA approach has better values compared to all the previous models without time, as we can see in the plot of ROC curve below, where on the left there is QDA with Train and Test, while on the right there is QDA with Cross Validation

```
formula<-DEATH_EVENT ~ . - anaemia - creatinine_phosphokinase - smoking -
  sex - serum_sodium - platelets - high_blood_pressure - diabetes


qda.fit <- qda(formula,data=train)
qda.fit.cv <- qda(formula,data=data, CV=TRUE)

qda.pred <- predict(qda.fit,test)
table_qda<-table(test$DEATH_EVENT, qda.pred$class)
table_qda_CV<-table(data$DEATH_EVENT, qda.fit.cv$class)

par(mfrow=c(1,2))


p.qda<-plot_ROC(test$DEATH_EVENT, qda.pred$posterior[,2], levels=c("not dead", "dead"))
p.qda.cv<-plot_ROC(data$DEATH_EVENT, qda.fit.cv$posterior[,2], levels=c("not dead", "dead"))
```






```
table_qda<-table(test$DEATH_EVENT, qda.pred$posterior[,2]>0.5)
table_qda_best_threshold<-table(test$DEATH_EVENT, qda.pred$posterior[,2]> p.qda[[1]])
table_qda_CV<-table(data$DEATH_EVENT, qda.fit.cv$posterior[,2]>0.5)
table_qda_CV_best_threshold<-table(data$DEATH_EVENT, qda.fit.cv$posterior[,2]>p.qda.cv[[1]] )

q1<-metrics(table_qda)
q1.b<-metrics(table_qda_best_threshold)
q1.cv<-metrics(table_qda_CV)
q1.cv.b<-metrics(table_qda_CV_best_threshold)
```

In the cell below we can see that:

- the maximum Precision is reached by QDA with Cross Validation
- the maximum Recall is reached by QDA with best threshold
- the maximum F1 Score is reached by QDA with Cross Validation and best threshold

```
## [1] "QDA"
```

Accuracy	Precision	Recall	F1.score
0.6888889	0.5454545	0.2068966	0.3

```
## [1] "QDA with best threshold"
```

Accuracy	Precision	Recall	F1.score
0.7444444	0.5789474	0.7586207	0.6567164

```
## [1] "QDA with Cross Validation"
```

Accuracy	Precision	Recall	F1.score
0.729097	0.6744186	0.3020833	0.4172662

```
## [1] "QDA with Cross Validation and best threshold"
```

Accuracy	Precision	Recall	F1.score
0.7792642	0.6442308	0.6979167	0.67

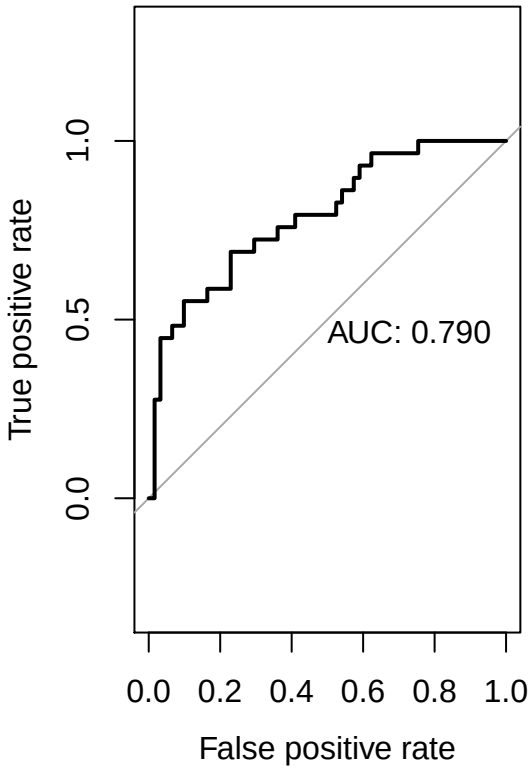
QDA with Time Variable

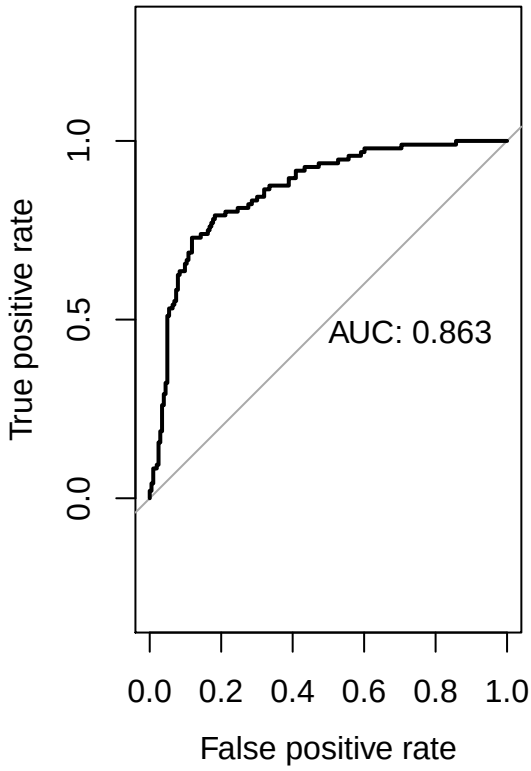
As in the previous models, the ROC curve improves when time is included.

```
formula<-DEATH_EVENT ~ . - anaemia - creatinine_phosphokinase - smoking -
  sex - serum_sodium - platelets - high_blood_pressure - diabetes

qda.fit.time <- qda(formula,data=train.time)
qda.fit.cv.time <- qda(formula,data=data.time, CV=TRUE)


qda.fit.time
```

```
## Call:
## qda(formula, data = train.time)
##
## Prior probabilities of groups:
##  not dead      dead
## 0.6794258 0.3205742
##
## Group means:
##               age ejection_fraction serum_creatinine      time
## not dead 58.49296          40.04225         1.189859 159.16901
## dead     64.74627          33.01493         1.958955  63.10448
```

```
qda.pred.time <- predict(qda.fit.time,test.time)
table_qda.time<-table(test.time$DEATH_EVENT, qda.pred.time$class)
table_qda_CV.time<-table(data.time$DEATH_EVENT, qda.fit.cv.time$class)


par(mfrow=c(1,2))

p.qda.time<-plot_ROC(test.time$DEATH_EVENT, qda.pred.time$posterior[,2], levels=c("not dead", "dead"))
p.qda.cv.time<-plot_ROC(data.time$DEATH_EVENT, qda.fit.cv.time$posterior[,2], levels=c("not dead", "dead
```






```
table_qda.time<-table(test.time$DEATH_EVENT, qda.pred.time$posterior[,2]>0.5)
table_qda_best_threshold.time<-table(test.time$DEATH_EVENT, qda.pred.time$posterior[,2]> p.qda.time[[1]]
table_qda_CV.time<-table(data.time$DEATH_EVENT, qda.fit.cv.time$posterior[,2]>0.5)
table_qda_CV_best_threshold.time<-table(data.time$DEATH_EVENT, qda.fit.cv.time$posterior[,2]>p.qda.cv.t

q1.time<-metrics(table_qda.time)
q1.b.time<-metrics(table_qda_best_threshold.time)
q1.cv.time<-metrics(table_qda_CV.time)
q1.cv.b.time<-metrics(table_qda_CV_best_threshold.time)
```

In the cell below we can see that:

- the maximum Precision is reached by QDA with Cross Validation
- the maximum Recall is reached by QDA with Cross Validation and best threshold
- the maximum F1 Score is reached by QDA with Cross Validation and best threshold

```
## [1] "QDA"
```

Accuracy	Precision	Recall	F1.score
0.7777778	0.7368421	0.4827586	0.5833333

```
## [1] "QDA with best threshold"
```

Accuracy	Precision	Recall	F1.score
0.7444444	0.5882353	0.6896552	0.6349206

```
## [1] "QDA with Cross Validation"
```

Accuracy	Precision	Recall	F1.score
0.8093645	0.7910448	0.5520833	0.6503067

```
## [1] "QDA with Cross Validation and best threshold"
```

Accuracy	Precision	Recall	F1.score
0.8327759	0.7446809	0.7291667	0.7368421

Model Summary

From all the tables discussed so far we have :

- best Precision: QDA with CV (with time)
- best Recall: GLM with best threshold (without time)
- best F-score: QDA and LDA with CV and best threshold (with time)

Conclusion and Results

In my work, an important and interesting evidence to point out is the fact that age, ejection fraction and serum creatinine are the most relevant features, and we only need those to detect if a people has an Heart Failure. In fact GLM with these features has an high recall (83%), meaning that the model well identifies when a patient has an Heart Failure. This result is encouraging for the hospitals, because even if some clinical features of a patient are missing or incomplete, the doctors are able to predict patient survival by analyzing the ejection fraction and serum creatinine values according also to the age. Moreover the effect of time discussed in the analysis part shows that doctors has to converge their efforts in the first months of the follow-up period.

In conclusion, the gathering of medical data related to patients has an high value because they can show patterns or correlation invisible to the Scientists, especially when the parameters are huge and of good quality.