Heart Failure Clynical Dataset Analysis

Francesco Montagna

Contents

1 Background

In this work we analyze a dataset of 299 patients with heart failure, applying some regression and classification models to predict patients survival.
Heart failure (HF) occurs when the heart is unable to pump enough blood to meet the body needs. To give an intuitive dimension of the problem, we report from Wikipedia (2020) that in year 2015 over 40 million cases had been recorded. Globally, 2% of adults are affected by heart failures; this rate increases to 6-10% for people of age over 65, and above 75 years old it is greater than 10%. Moreover, HF is the main cause of hospitalization in people aged more than 65. These statistics motivate and value the presented research. The available medical records of patiences allow to quantify symptoms, body features, and clinical laboratory test results, which can be used to perform analysis aimed at highlighting patterns and correlations otherwise undetectable by medical doctors.

The following is a summary of the document's structure and of its milestones.
First, we present our dataset and its attributes description. Then we focus on EDA (Exploratory Dataset Analysis), divided in **visualization** and **processing** parts. This step is preparatory for the models creation, which goal is to predict a boolean response on patience survival given the sample predictors realizations: we define several models to approach the problem, showing the inadequacy of **linear regression** in predicting a Bernoulli random variable and then, moving to **logistic regression**. The subsequent section of the document exploits different methods to choose one among the defined models.
Since we are dealing with a classification problem, we also provide a soft margin Support Vector Machine implementation. In this way we explore solutions outside regression domain, and we use this as starting point for an analysis on the interpretability of the classifiers that have been illustrated.

1.1 Development Environment

All methods are implemented with the open source R programming language, while the code is publically available at this github repository.
We use ggplot2 library for the visualization of results and data, and e1071 machine learning library for off the shelf implementation of SVM. Several other libraries are exploited for minor use.
This document is an R Markdown notebook.

2 Dataset Overview

We train and validate our models using the Heart failure clinical records Dataset, which current version has been created by D. Chicco (2020) and was downloaded from UCI repository.
The dataset contains the medical records of 299 patients with heart failure, and has been collected at the Faisalabad Institute of Cardiology and at the Allied Hospital in Faisalabad (Punjab, Pakistan), during April–December 2015. All 299 patients had left ventricular systolic dysfunction and had previous heart failures.
No missing values are reported in the dataset.

2.1 Attributes

Below table describes dataset attributes, along with their measurement unit and the range of covered values.

```
# Heart failure dataset from UCI repository
hf.df = read.csv("heart_failure_clinical_records_dataset.csv", header = T)


# Compute range for attributes
range <- c()
```

```
for (col in colnames(hf.df)) {
     min <- format(min(hf.df[col]))
     max <- format(max(hf.df[col]))
     colrange <- paste(min, max, sep = "-")
     range <- append(range, colrange)
}

meanings <- c("Age of the patient",
              "Anomalous decrease of red blood cells or hemoglobin",
              "Level of the CPK enzyme in the blood. High values are associated with muscular
              distrophy",
              "If the patient has diabetes",
              "Percentage of blood leaving the heart at each contraction",
              "If a patient has hypertension",
              "Platelets in the blood",
              "Level of creatinine in the blood. High values are associated with renal
              disfunction",
              "Level of sodium in the blood",
              "Gender: 0 for woman, 1 for man",
              "If the patient smokes",
              "Follow-up period",
              "If the patient died during the follow-up period")

attributes <- colnames(hf.df)

unit.measure <- c("Years",
                  "Boolean",
                  "mcg/L",
                  "Boolean",
                  "Percentage",
                  "Boolean",
                  "kiloplatelets/mL",
                  "mg/dL",
                  "mEq/L",
                  "Binary",
                  "Boolean",
                  "Days",
                  "Boolean")

features.table = data.frame("attribute" = attributes,
                            "description" = meanings,
                            "unit measure" = unit.measure,
                            "range" = range)

formattable(features.table, col.names = c("Attribute", "Description", "Unit Measure", "Range"), align =
```

Attribute

Description

Unit Measure

Range

age

Age of the patient

Years

40-95

anaemia

Anomalous decrease of red blood cells or hemoglobin

Boolean

0-1

creatinine_phosphokinase

Level of the CPK enzyme in the blood. High values are associated with muscular distrophy

mcg/L

23-7861

diabetes

If the patient has diabetes

Boolean

0-1

ejection_fraction

Percentage of blood leaving the heart at each contraction

Percentage

14-80

high_blood_pressure

If a patient has hypertension

Boolean

0-1

platelets

Platelets in the blood

kiloplatelets/mL

25100-850000

serum_creatinine

Level of creatinine in the blood. High values are associated with renal disfunction

mg/dL

0.5-9.4

serum_sodium

Level of sodium in the blood

mEq/L

113-148

sex

Gender: 0 for woman, 1 for man

Binary

0-1

smoking

If the patient smokes

Boolean

0-1

time

Follow-up period

Days

4-285

DEATH_EVENT

If the patient died during the follow-up period

Boolean

0-1

mcg/L: micrograms per liter
mL: microliter
mEq/L: milliequivalents per litre

Inspecting the table, we see how the dataset is made of 5 categorical variables and 7 quantitatives. Our model will need to predict *DEATH_EVENT* value. Having a binary response suggests to use a logistic regression model.

Now we want to proceed analyzing the sample data, and eventually perform some processing activity using the insights gained from the exploratory activity.

2.2 EDA

2.2.1 Visualization

We start the Exploratory Data Analysis (EDA) plotting **sample frequencies** of our features. This can be easily done using histograms with the number of occurrences on the y axis and binned variables on the x axis.

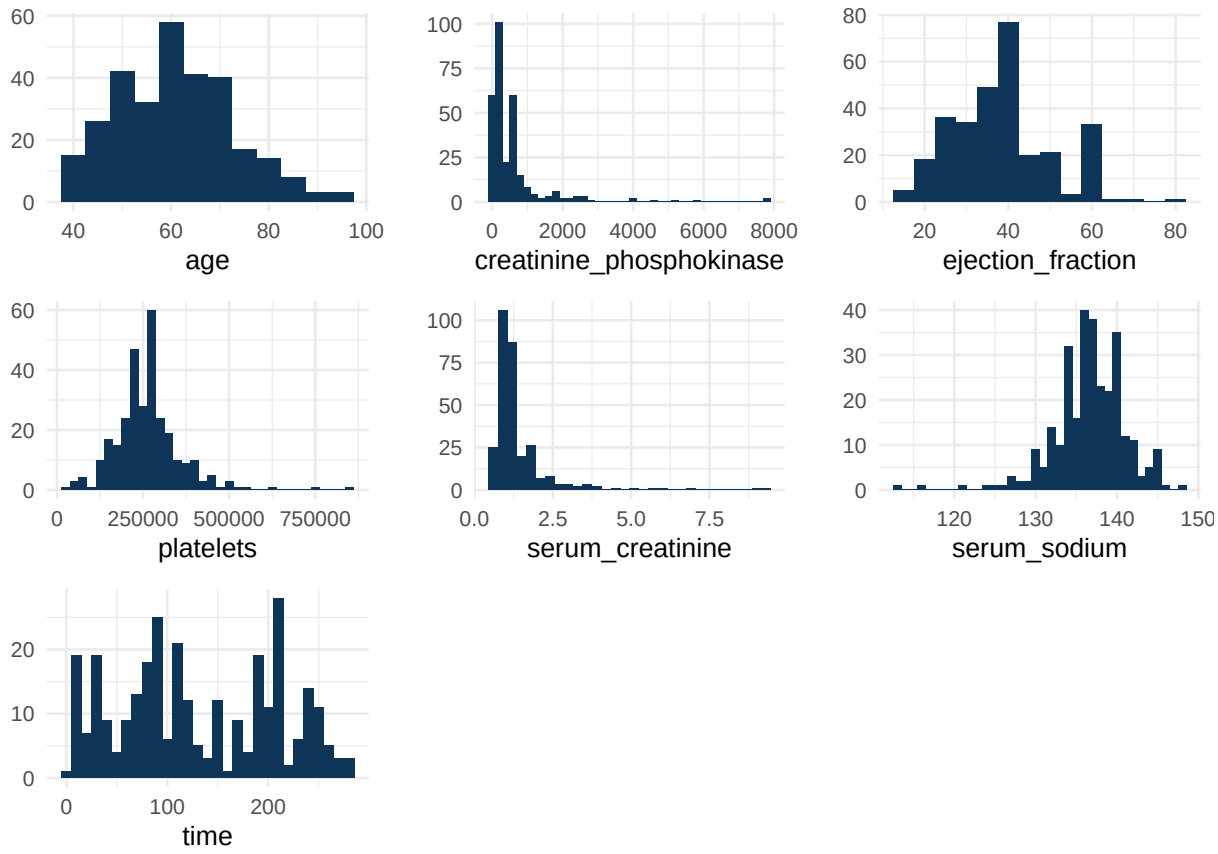
```
attach(hf.df)

plist <- list()
i <- 1
bwidths <- c(5, 200, 5, 25000, 0.3, 1, 10)

for (col in colnames(hf.df)) {
  # select only quantitative attributes
  mask <- features.table$attribute == col
  if (features.table[mask,]$unit.measure != "Boolean" &
      features.table[mask,]$unit.measure != "Binary" &
      features.table[mask,]$attribute != "DEATH_EVENT") {
    newplot <- ggplot(data= hf.df)+
```

```
    geom_histogram(aes_string(col), binwidth = bwidths[i], fill = "#0E3557") +
    xlab(col)+ylab("")+
    theme_minimal(base_size = 10)
    plist[[i]] <- newplot
    i <- i+1
  }
}

do.call("grid.arrange", c(plist, ncol=3))
```



Next section is devoted to predictors' graphs interpretation and resulting transformations.

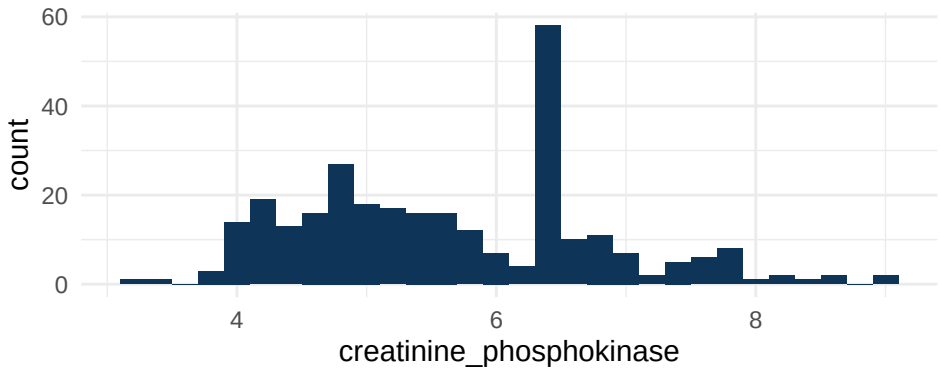
2.2.2 Processing

We see that *creatinine_phosphokinase* sample frequency is skewed and characterized by a wide range of values: since we know that it is easier for a line to fit more homogeneous values, we **scale logarithmically** our data applying $ln(x)$ function, which returns a shape more evenly spreaded in a narrowed range.
Histogram plots also shows a large difference in the **order of magnitudes** covered by our features. Focusing on *platelets* attribute we see that its maximum is larger than *800 000*. We could decide to apply some rescaling techniques, but this would not be influent in the model: since the β coefficients reflect predictors' unit change on the response, they adapt to the order of magnitude of the variable they refer to. For this reason this transformation is discarded.

After the processing step, *creatinine_phosphokinase* count histogram looks like this

```r
hf.df$creatinine_phosphokinase = log(hf.df$creatinine_phosphokinase, base = exp(1))

ggplot(data= hf.df)+
    geom_histogram(aes(creatinine_phosphokinase), binwidth = 0.2,
                   fill = "#0E3557") +
    xlab("creatinine_phosphokinase")+
    theme_minimal(base_size = 11)
```

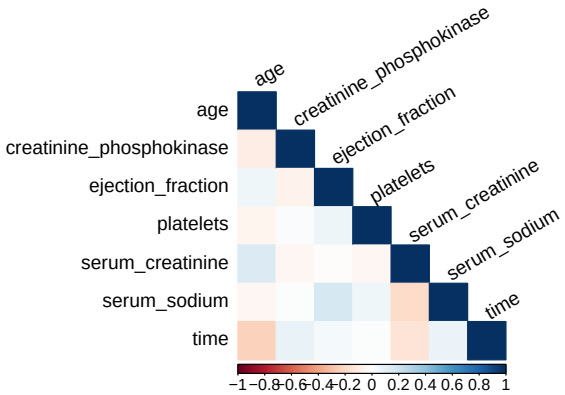


2.2.3 Correlation

When using regression, we would like to interpret the estimated regression coefficient β_i as the average effect on the response, or on the logodds, of a unit change in X_i predictor. For this reason it is important to verify correlation among features: if it is weak, the above reading holds since a variation on X_i fully explains the consequent variation on Y_i response.

```r
# Filter categorical and response variables
mask <- (features.table$unit.measure != "Binary" & features.table$unit.measure != "Boolean"
& features.table$attribute != "DEATH_EVENT")

corrplot(cor(hf.df[mask]), method = "color",
         type = "lower",
         tl.col = "black",
         tl.srt  = 30)
```



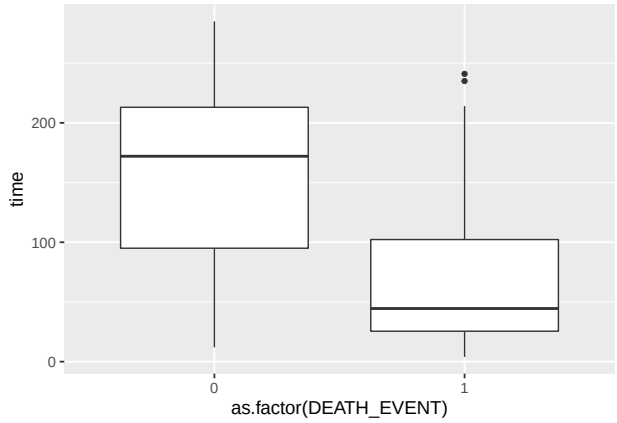
The heatmap graphically represents the sample correlation coefficients among pairs. We can see that features are weakly correlated allowing us to mantain this straightforward interpretation of our regression models.

2.2.4 Time attribute

We now want to focus on time attribute to decide whether it should be included as predictor. Our hypothesis is that using time lead to a distorted, unprecise algorithm: it is in fact intuitive to see that for large time value d, the outcome is more likely to be that the patience survived during this *follow-up* period, for the simple reason that we can reasonably think that until day $d-1$ the patience was alive. In this sense, it is important to notice that time does not reflect the health state of a patience, and we think that in the model it results being a hint to the classifier towards the right prediction. If this is true, our model becomes weaker in identifying patterns between clinical health features and the response.

We use a boxplot to further explore this intuition

```
ggplot(hf.df,aes(x = as.factor(DEATH_EVENT), y = time))+
  geom_boxplot()+
  theme_gray(base_size = 14)
```



A first look is sufficient to identify a link between death event and time: when time is small, the outcome is more likely to be *death = 1* (as we would suppose) and vice versa for larger number of days. In fact, we see a large shift in the time median for the 2 groups, where approximately for 0 response its value is 170 days versus 50 days in case of outcome equals 1.

The explainations of the hypothesis and this brief analysis of the plot leads us to **discard time attribute** from the predictors, and the goal of our model becomes to predict death or survival of the patience exclusively on the base of its clinical features. Please refer to the appendix for further exploration of the topic.

3 Regression Models

We start introducing the **General Linear Model** theory, in order to show why it is not suitable for the current prediction task. Then we move to **Generalized Linear Model** , defining some distinct logistic regression models on our dataset and trying to choose the best one using different methods.

3.1 General Linear Model

In the General Linear Model we focus on a set of **quantitative responses** $y_1, ..., y_n$, considered as realizations of a **normal** random variable $Y_1, ..., Y_n$. The goal is to explain the response behaviour in terms of

predictors, collected in a matrix X of dimension $n \times p$, which **linearly** affects Y components. In particular, n is the number of samples while p is the number of predictors.

$$Y = \beta X + \varepsilon$$

β is the vector of X's **coefficients**, a set of unknown parameters and the main object of the statistical inference. As already mentioned, the goal is to find $\hat{\beta}$ estimates such that a unit change in X_i is reflected as $\hat{\beta}_i$ variation on the response.
ε is the **error** vector, which account for sources of uncertainty (*e.g.* measurement error, natural variability ...). It is a set of unobservable random variables which entries are assumed to be *IID* (Identically Independently distributed) from a normal distribution $\mathcal{N}_n(0, \sigma^2 I_{n\times n})$. This implies our response to be

$$Y \sim \mathcal{N}_n(X\beta, \sigma^2 I_{n\times n})$$

such that we have *homoschedastic* Y_i components, *i.e.* sharing same variance. The distribution form offers another important insight: given Y_i, its expected value is

$$\mathbf{E}(Y_i) = \sum_{k=0}^{p-1} \beta_k x_{ik}$$

which means that, given a sample realization of the predictors, the fitted line describes the mean response.

3.1.1 Least Square Estimates

β coefficients, being an unknown parameter of the response distribution, can be found as *Maximum Likelihood Estitmators*. Given the likelihood L of the observable Y, we find β that maximizes it. This is a common way to proceed, and it turns out that maximizing the likelihood with respect to β is equivalent to minimizing

$$\sum_{i=0}^{n-1} (y_i - \mu_i)^2$$

where μ_i is the expected value of y_i itself.

Generelizing to vectors, the expression become

$$||y - \mu||^2$$

with $||\cdot||$ representing the Euclidean norm.

We've already noticed that μ vector is equals to βX. Therefore, solving the minimization problem

$$min_\beta ||y - \mu||^2$$

we find a general expression in matricial form for β estimator

$$\hat{\beta} = (X'X)^{-1}X'Y$$

which is true if $(X'X)$ is invertible. From the above equation, we can see that $\hat{\beta}$ is a linear transformation of Y, thus it is itself a normally distributed random vector. In particular

$$\hat{\beta} \sim \mathcal{N}_n(\beta, \sigma^2(X'X)^{-1})$$

It is interesting to notice that $\hat{\beta}$ is an **unbiased estimator**, as we can easily prove

$$\mathbf{E}(\hat{\beta}) = \mathbf{E}((X'X)^{-1}X'Y) = (X'X)^{-1}X'X\beta = \beta$$

3.1.2 Toy Model

It should already be clear that linear regression is not suitable for the current task: in fact we want to predict a Bernoulli random variable, while in the general model holds the assumption of normal response. Using a toy model, made simply with *serum_creatinine* continuous predictor, we can graphically show why it is inadequate.
If we apply linear regression to model a Bernoulli response, then the fitted line is the estimated probability of a sample of belonging to $Y = 1$, since the following relation holds

$$\mu_i = P(Y_i = 1) = \sum_{j=0}^{p-1} \beta_j x_{ij}$$

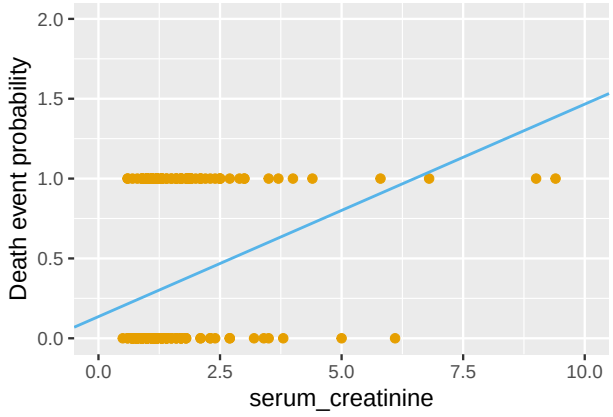
The below plot shows the fitted regression line and the sample responses

```
# define the linear model
toymodel <- lm(DEATH_EVENT ~ serum_creatinine)

# extract coefficients \beta_0 and \beta_1
coeffs <- coefficients(toymodel)

toyplot <- ggplot(hf.df, aes(x=serum_creatinine, y=DEATH_EVENT)) +
  geom_point(color = "#E69F00",size = 3)+
  xlim(0, 10) +
  ylim(0, 2) +
  ylab("Death event probability")

toyplot +
  geom_abline(slope = coeffs[2], intercept = coeffs[1],
                        color = "#56B4E9",
                        size = 1) +
  theme_gray(base_size = 18)
```



We can see that after a certain value, we have a probability larger than 1, which is clearly an issue. This analysis motivates the need of a different model, that we now proceed to introduce.

3.2 Generalized Linear Model (GLM)

Discrete binary response like in our case are better modeled with logistic regression, belonging to the GLM family. As for the general linear model, GLM is characterized by a response vector $Y_1, ..., Y_n$ of independent

random variables, with means $\mu_i = \mathbf{E}(Y_i)$.
We also define η, the **linear combination** of p categorical and quantitative predictors,

$$\eta = \beta X$$

Finally we have a **link function** $g(\cdot)$ connecting the means μ_i and the linear combinations

$$g(\mu_i) = \eta_i$$

Different response distributions and link functions define different GLM families. For example, the general linear model has normal homoschedastic response vector, and link function equals to the identity matrix I.

In case of bernoullian response, for **logistic regression** the link function is the logit

$$g(\mu_i) = logit(p_i) = ln(\frac{p_i}{1-p_i})$$

This function is also known as *logodds*, since it is the logarithm of the odds: we can think to the odds as a reparametrization of probability. Given

$$p_i = P(Y_i = 1|X_i)$$

the corresponding odds are

$$p_i \to \frac{p_i}{1-p_i}$$

Finally, if we take the inverse of the *logits*, we are modeling the probability p of the response of belonging to class 1.

$$p_i = \frac{e^{logit}}{1+e^{logit}}$$

As it has been mentioned, the logits are used as link function. For this reason it holds that

$$logit(p_i) = \sum_{j=0}^{p-1} \beta_j X_{ij}$$

which tells us that logistic regression model is linear in the logits, and β_j coefficient is nothing but the effect of a unit change of X_{ij} over the logit. This interpretation reconnects with the one defined in linear regression where we considered the effect of predictors directly over the response.

In order to estimate β coefficients, maximum likelihood estimation is exploited.

3.2.1 Logistic Regression Models

Now we want to create in R some logistic regression models to predict our response, the death event of a patient. We start defining the complete model, using all available predictors, with R `glm()` function. Before doing that, we specify which predictors are categorical, or else, which are our *factors*. Below, we display `summary()` results

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

big <- glm(DEATH_EVENT ~ . -time, data = hf.df,family = binomial)
summary(big)
```

```
##
## Call:
## glm(formula = DEATH_EVENT ~ . - time, family = binomial, data = hf.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2494  -0.7949  -0.4557   0.8163   2.4434
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                3.510e+00  4.574e+00   0.767 0.442936
## age                        5.444e-02  1.297e-02   4.197 2.71e-05 ***
## anaemia1                   4.063e-01  3.035e-01   1.339 0.180672
## creatinine_phosphokinase   2.010e-01  1.332e-01   1.509 0.131239
## diabetes1                  1.309e-01  2.967e-01   0.441 0.659016
## ejection_fraction         -6.942e-02  1.480e-02  -4.690 2.73e-06 ***
## high_blood_pressure1       4.227e-01  3.060e-01   1.382 0.167090
## platelets                 -5.818e-07  1.600e-06  -0.364 0.716159
## serum_creatinine           6.768e-01  1.741e-01   3.889 0.000101 ***
## serum_sodium              -5.324e-02  3.295e-02  -1.616 0.106158
## sex1                      -3.554e-01  3.483e-01  -1.020 0.307513
## smoking1                   1.291e-01  3.473e-01   0.372 0.710023
## ---
## 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: 296.17  on 287  degrees of freedom
## AIC: 320.17
##
## Number of Fisher Scoring iterations: 5
```

For the majority of the predictors, ***p_value*** is large respect to the *significance level* of 0.05: when this is the case, we interpret that we do not have enough evidence to reject the null hypothesis $H_0 : \beta_i = 0$ for a variable i, when all other variables are in the model. From this inspection, we are safely lead to keep only *age*, *serum_creatinine* and *ejection_fraction* attributes in the model.
In order to have a deeper insight, we define 95% **confidence intervals** of the β parameters.
The use of confidence interval is justified by the fact that usually, when making point estimates for parameters we don't expect them to be equals to the real value, but "close" to it. It is sometimes valuable to be able to specify an interval for which we have a certain degree of confidence that our parameter lies within. In this sense they are a useful tool which allows to **quantify the uncertainty** not expressed by point estimates.

```
suppressMessages(round(confint(big),6))
```

```
##                               2.5 %    97.5 %
## (Intercept)              -5.436886 12.606162
## age                       0.029619  0.080657
## anaemia1                 -0.186625  1.006716
## creatinine_phosphokinase -0.059407  0.464858
## diabetes1                -0.452206  0.714398
## ejection_fraction        -0.099704 -0.041493
## high_blood_pressure1     -0.178126  1.024936
```

```
## platelets                -0.000004  0.000003
## serum_creatinine          0.366298  1.059280
## serum_sodium             -0.119200  0.010831
## sex1                     -1.045035  0.324936
## smoking1                 -0.553380  0.812951
```

The boundaries of the intervals give us an interesting view: we see how all predictors have *0* value included in their confidence interval, except those that we decided to retain. This is important confirm of our previous analysis on point estimates based on their p-value.

Now we proceed defining 3 models significantly smaller than the complete.

- small model: `DEATH_EVENT ~ ejection_fraction + serum_creatinine`
  It has been proposed by Chicco and Jurman (2020), with the goal of creating good quality prediction with a very simple model.
- medium model: `DEATH_EVENT ~ ejection_fraction + serum_creatinine + age`
  Here we add *age* predictor, exploiting the relevant insights and analysis of summary results of the complete model.
- big model: `DEATH_EVENT ~ ejection_fraction + serum_creatinine + age + anaemia + high_blood_pressure`
  Finally, we also include several binary predictors, which selection is done in accordance with Ahmad (2017) research work.

```
small <- glm(DEATH_EVENT ~ ejection_fraction + serum_creatinine,
             data = hf.df, family = "binomial")

medium <- glm(DEATH_EVENT ~ age + ejection_fraction + serum_creatinine,
             data = hf.df, family = "binomial")

big <- glm(DEATH_EVENT ~ age + anaemia + ejection_fraction +
             serum_creatinine + high_blood_pressure,
             data = hf.df, family = "binomial")
```

We now want to focus on big model, to interpret the meaning of the estimated coefficient for a boolean predictor: extracting $\hat{\beta_{AN}}$ for *anaemia* we get

```
coeffs <- coefficients(big)
cat(paste0("anaemia coefficient: ",  round(coeffs[3], 3)))
```

```
## anaemia coefficient: 0.266
```

Let us define c as the contribute to the logit function from all attributes except *anaemia (AN)*: then we can rewrite the logodds as

$$logit(p_i) = c + 0.266AN$$

Being AN boolean, the logit results to be

$$logit(p_i) = \begin{cases} c, & \text{if } AN = 0 \\ c + 0.266, & \text{if } AN = 1 \end{cases}$$

Then we define the quantities

- $p_{i1} = P(Y_i = 1|AN = 1)$
- $p_{i0} = P(Y_i = 1|AN = 0)$

At this point the estimated coefficient for *anaemia* can be rewritten as

$$\begin{aligned} 0.266 &= logit(p_{i1}) - logit(p_{i0}) \\ &= ln(\frac{p_{i1}}{1-p_{i1}}) - ln(\frac{p_{i0}}{1-p_{i0}}) \\ &= ln(\frac{\frac{p_{i1}}{1-p_{i1}}}{\frac{p_{i0}}{1-p_{i0}}}) \end{aligned}$$

The obtained quantity is known as **logodds ratio**. It is a useful measure to compare probability of successful response under two situation, in this case defined by occurrence of *anaemia* in a patience.

3.2.2 ANOVA and AIC: Model Selection

Given multiple models available, it is important to have some criterion to choose between them.

For the selection part, first we use **ANOVA** test (Analysis of Variance), which allows comparison of nested models. In R this is done using `anova(smaller model, bigger model,test="Chisq")`. The null of the test is defined as: $H_0 : \beta' = 0$ with β' equals the vector of coefficients referring to the variables not in common between the two models.

We start comparing the *small* and *big*

```
print(anova(small, big, test="Chisq"))
```

```
## Analysis of Deviance Table
##
## Model 1: DEATH_EVENT ~ ejection_fraction + serum_creatinine
## Model 2: DEATH_EVENT ~ age + anaemia + ejection_fraction + serum_creatinine +
##     high_blood_pressure
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       296     324.32
## 2       293     302.19  3    22.13 6.129e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Looking at the very small *p-value* respect to significance level, we can safely **reject the null** of taking the small model instead of the bigger one.
Performing the test again for the remaining pairs, we conclude that we have no evidence to reject the *medium*, that we then prefer.

Another way to do selection is using **Akaike Information Criterion**, *AIC* for brevity. In R we exploit `AIC()` function. In particular, AIC of a model is computed as follow:

$$AIC = 2(p+1) - 2ln(\hat{L})$$

Where $\hat{L}$ is the maximized likelihood function and p is the number of predictors. We see that using this criterion larger p implies larger penalization: since models are a (limited) explaination of the world, then this means that we should tend to prefer simpler ones. Overall, we select those with smaller Akaike's.

```r
as.matrix(AIC(small, medium, big))
```

```
##        df      AIC
## small   3 330.3211
## medium  4 313.2827
## big     6 314.1911
```

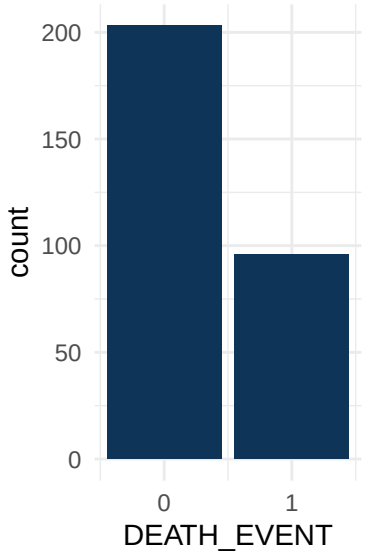
As it was with *ANOVA*, also *AIC* justifies the use of *age* predictor in our models. In this case, instead, it is hard to have a clear preference between *medium* and *big*.

3.2.3 K-Fold Cross Validation

Another way of comparing models is by testing their performances. For example this can be done via K-fold Cross Validation: given a dataset of cardinality n, we define K test subsets of cardinality $\frac{n}{K}$. In turn each partition is used as validation set while the others are used as training set. This method is feasible in particular for small datasets, otherwise it results being very time consuming.

In order to evaluate classifiers' performance, we need to define some metrics, but before doing that, we also want to visualize the occurrency count of our binary response variable

```r
ggplot(data= hf.df)+
    geom_bar(aes(DEATH_EVENT), fill = "#0E3557") +
    xlab("DEATH_EVENT")+ylab("count")+
    theme_minimal(base_size = 11)+
  scale_x_continuous(breaks = c(0, 1))
```



There is a quite evident unbalancement, since 0 value frequency (survival of the patience) is almost double that of 1 event. In a situation of unbalanced response, accuracy score might be unreliable: in our case for example, an algorithm predicting always 0, would have a decent score of roughly 66%. Therefore we decide to use F1 metric instead of accuracy. To measure the misclassification error, we use MAE (Mean Absolute Error) cost function.

- F1: $2\frac{precision \cdot recall}{precision + recall}$

- MAE: $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$, with n the cardinality of the dataset

At this point, we can define a cross validation pipeline for our logistic regression models. Remember that what we see reported from K-fold CV is the arithmetic average of the mean scores from each of the K subsets. Fixing K = 5, we obtain the following results

```
##    Model        F1       MAE
## 1  small 0.4579950 0.2575251
## 2 medium 0.5603630 0.2374582
## 3    big 0.5301655 0.2508361
```

Considering separately the 2 metrics, we see that F1 score of the medium model is quite higher then *big*'s value, while both clearly ourperforms the small model. The situation is not equally clear for MAE, where medium model results the best performing, while *big* is very marginally better than *small*.

At the end of this validation analysis, we decide to retain only the medium model, which according to these tests revealed to be the best available classifier among those we tried.

4 SVM

In this final section, we define an SVM model for discriminating between binary response events. This will allow us to explore solutions outside the regression domain and to have a comparison with our logistic regression model.

4.1 Theory Introduction

Consider a set of n sample data $\{x_i, y_i\}$, $i = 1, ..., n$ where $x_i \in R^p$ is a realization of random predictor vector, and $y_i \in \{\pm 1\}$ a realization of the response. In the *hard-margin* problem we want to find a **separating hyper-plane** with **largest possible margin**, where the latter is proportional to the minimum of the possible distances between the projections of 2 sample points of opposite class on the direction normal to the hyper-plane. The points in space defining the margin are called support vectors, which we denote x_+ and x_-. Mathematically, this becomes

$$margin = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Since the goal is to find the hyper-plane maximizing the margin, this is done by solving the following minimization problem

$$min_w \mathbf{w}^T\mathbf{w}$$

S.t.

$$y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1,\ i = 1, ..., n$$

which is necessary condition for correct linear classification.

In case of non linearly separable samples, we introduce *soft-margin* SVM, where optimality condition is relaxed by an extra-term. The minimization problem becomes

$$min_w \mathbf{w}^T\mathbf{w} + C\sum_{j=0}^{p}\xi_j$$

S.t.

$$y_i(\mathbf{x}^T\mathbf{w} + b) \geq 1 - \xi_i$$
$$\xi_i > 0,\ i = 1, ..., n$$

C is a regularization hyper-parameter to trade off between error allowed and margin minimization.

4.2 Comparing Logistic Regression and SVM

We train a soft SVM linear classifier from our data to separate between death and survival events. We arbitrarly set C = 10.

```
svmfit <- svm(as.factor(DEATH_EVENT) ~ age + serum_creatinine + ejection_fraction,
              data = hf.df,
              cost = 10,
              kernel = "linear",
              scale = FALSE)
```

Now we can artificially define a new sample x_f, over which make predictions using both our logistic regession model and SVM.

```
newdata <- data.frame("ejection_fraction" = 60,
                      "age" = 40,
                      "serum_creatinine" = 6.1)

ypred.svm <- predict(svmfit, newdata)
ypred.glm <- predict(medium, newdata, type = "response")

if(ypred.glm > 0.5) {
  ypred.glm <- 1
} else {
  ypred.glm <- 0
}
cat(paste0("\nSVM prediction: ",ypred.svm[1]),
    paste0("\nLogistic Regression prediction: ",ypred.glm[1]))
```

```
##
## SVM prediction: 1
## Logistic Regression prediction: 0
```

For this test instance the 2 classifiers return a different prediction on the death event of the patience. Now, using coefficients estimates, we can display the fitted logistic regression model (*serum_creatinine(SC)*, *ejection_fraction (EF)* and *age* predictors)

$$\hat{logit}(p) = -2.35 + 0.05 \cdot age - 0.07 \cdot EF + 0.67 \cdot SC$$

which is linear in the logit response: positive coefficients will increase the logodds, and thus increase the probability of 1 response (death), while negative coefficients decrease the logodds along with the event probability. In this sense, older patiences are more at risk than younger ones, whereas those with smaller EF (*i.e.* which heart pumps less blood in the body) and renal disfunction (higher serum creatinine) are more in danger. As we can see, logistic regression allows for good interpretability, which is postive in case we want our decision not to be driven by a black box.

Looking at our example, where new sample is

```
print(as.matrix(newdata))
```

```
##      ejection_fraction age serum_creatinine
## [1,]                60  40              6.1
```

we have a reasonable understanding that age and EF have prevailed in defining the prediction.
Moreover, to study our model we can use some measures for uncertainty quantification over estimated parameters: as we already shown, we can derive confidence intervals for the coefficients with a certain level of statistical significance.

```
suppressMessages(round(confint(medium),6))
```

```
##                        2.5 %    97.5 %
## (Intercept)       -4.031292 -0.728235
## age                0.028113  0.076551
## ejection_fraction -0.099062 -0.043107
## serum_creatinine   0.377757  1.015338
```

If instead we are interested in analyzing the decision process in SVM, this can be easily done only with a small number of predictors. For example when number of attributes is 2, we can plot the decision boundaries defined by the hyper-plane. But when this number gets larger, this is not possible anymore, and it becomes less meaningful if we decide to use a non-linear kernel. Additionally, SVM as other models does not aim at providing any insights on the uncertainty characterizing coefficients, since the focus is mostly put on the predictions.

5 Conclusion

Our work showed that SVM and logistic regression can be used effectively for classification tasks, such as ours concerning binary discrimination from health records of patients with cardiovascular hearth diseases. We tried to cover a complete pipeline from data exploration to model selection, focusing both on theoretical insights and on practical results. It can also be concluded that growing age, renal dysfunction and lower values of ejection fraction are key factors leading to increased risk of mortality among heart failure patients.

As a limitation of the present study, we report the small size of the dataset, consisting only of 299 records: a larger cardinality would have permitted us to obtain more reliable results.

6 Appendix

6.1 Appendix A

To confirm the intuition that time is not a relevant attribute, but only a hint towards the right response, we can build a naive model using only time as predictor, and display its F1 and MAE scores using K-fold cross validation, with K = 5.

```
time.model <- glm(DEATH_EVENT ~ time, data =hf.df, family = "binomial")

accuracy <- function(y, yhat) {
  tp = tn = fp = fn <- 0
  for (i in 1:length(y)) {

    if(yhat[i] < 0.5) {
      yhat[i] = 0
    }
    else {
      yhat[i] = 1
```

```
    }
    if((y[i] == yhat[i]) & (yhat[i] == 1)) {
      tp <- tp+ 1
    }

    if((y[i] == yhat[i]) & (yhat[i] == 0)) {
      tn <- tn+1
    }

    if((y[i] == 1) & (yhat[i] == 0)){
      fn <- fn+1
    }

    if((y[i] == 0) & (yhat[i] == 1)){
      fp <- fp+1
    }
  }

  accuracy <- (tp+tn)/(tp+tn+fp+fn)

  return(accuracy)

}

f1_score <- cv.glm(hf.df, time.model, f1.score, 5)$delta[1]
mae <- cv.glm(hf.df, time.model, mae.score, 5)$delta[1]

cat(paste0("Naive time model F1-score: ", f1_score),
    paste0("\nNaive time model MAE: ", mae))
```

```
## Naive time model F1-score: 0.722018490699939
## Naive time model MAE: 0.167224080267559
```

If we compare this score with those given by all other regression models, keeping in mind that time plays no role on the health state of the patience, we can see how our hypothesis is confirmed.

References

Ahmad, T. 2017. ""Survival Analysis of Heart Failure Patients: A Case Study"." *PLoS ONE*, July. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0181001#sec010.

Chicco, Davide. 2020. "UCI Machine Learning Repository." University of California, Irvine, School of Information and Computer Sciences. https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records.

Chicco, and Jurman. 2020. "Machine Learning Can Predict Survival of Patients with Heart Failure from Serum Creatinine and Ejection Fraction Alone." *BMC Medical Informatics and Decision Making*, February. https://doi.org/10.1186/s12911-020-1023-5.

Wikipedia. 2020. "Heart Failure — Wikipedia, the Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Heart_failure&oldid=977540669.