ML project

Pietro & Marita

Two aricles where these data come from:

1. Chicco, D., Jurman, G. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Med Inform Decis Mak 20, 16 (2020). DOI
2. Ahmad T, Munir A, Bhatti SH, Aftab M, Raza MA (2017) Survival analysis of heart failure patients: A case study. PLoS ONE 12(7): e0181001. DOI

These are the data we have about the survavial from

Gender, *Smoking*, *Diabetes*, *BP* and *Anaemia* are factors that have been saved as numerical values. To start, we decided to transform them into categorical variables

DESCRIPTION OF THE DATA

... IMPORTANT: explain difference btw usual data with only classification of the event and this problem, that requires survival models.

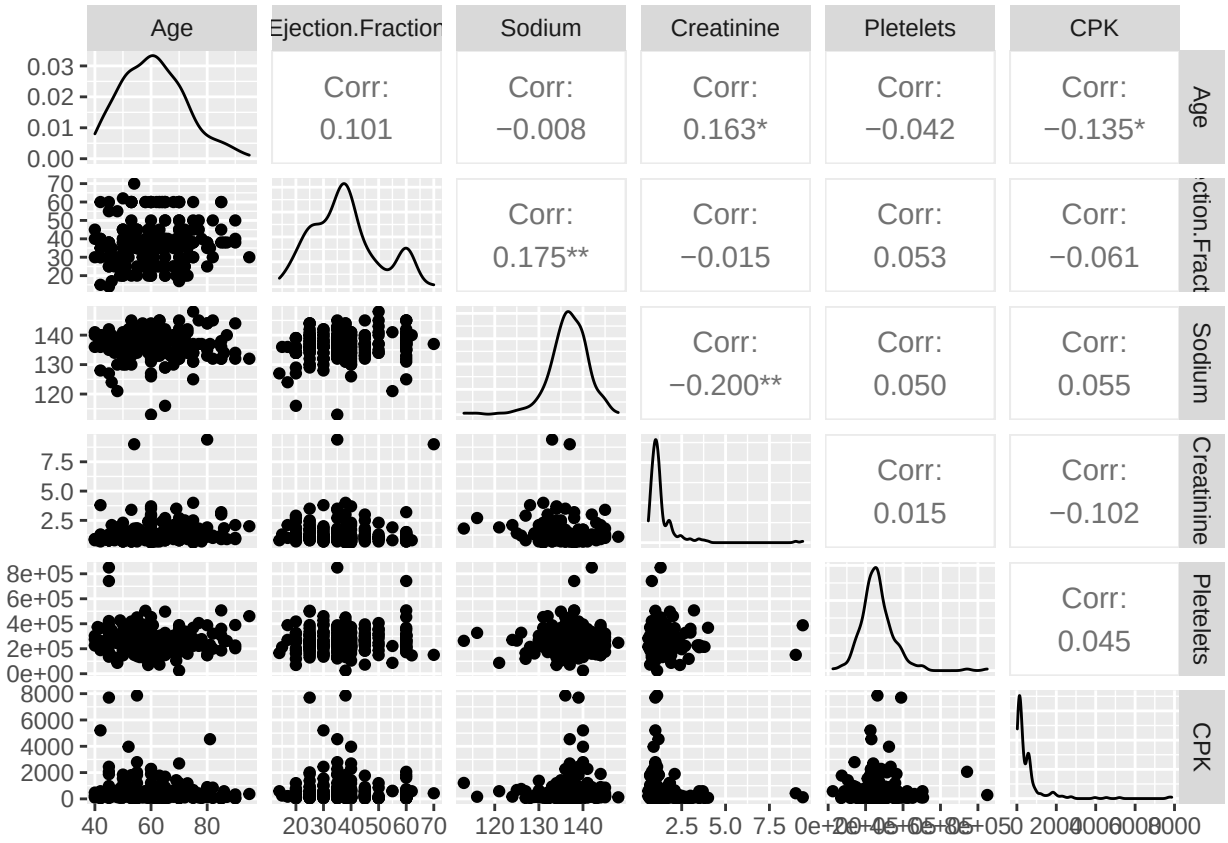
Before starting looking at the data we randomly split it in one set for training and one for testing. We are splitting this two sets of data with an 80-20 ratio. We want to use as much data as possible for training since we have few instances, while being able to get a good estimation of the best model at the end of this analysis. We also make this split stratified, which means keeping the proportion of positive cases in the subsamples as it appears in the full data set. We want our models to learn the data as it is in reality, quite unbalanced. Splitting data in train and test will assure that the observations we will do during the explorative analysis and the decisions that will be made on them will not be biased by the knowledge acquired from the test set. What follows is based on the training set.

EXPLORATIVE ANALYSIS

We started with the explorative analysis of the data. The aim of this first section is to reach a general understanding of the data we have, their distribution, the correlation between variables and in general all the aspects that concern the descriptive analysis of the observations. Only after a rigorous analysis we can start to model the data, since before that we will not have a complete comprehension of variables and their relationships.

Pairs of continous variables

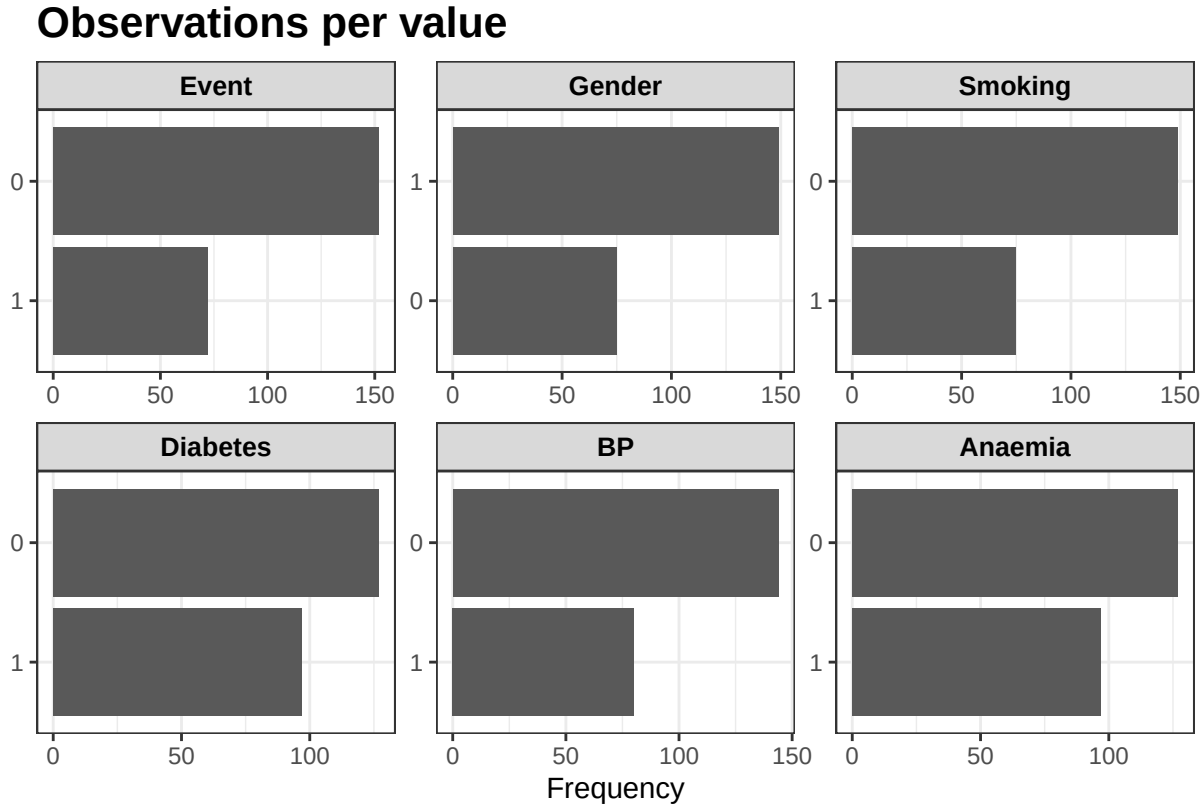
First, we are interested to see the joint and disjoint distribution of all the pairs of continous variables.



The correlation is significantly different from zero only for the couples *Creatinine-Age*, *Creatinine-Sodium* and *Sodium-Ejection.Fraction*, and anyways very low.

Factors

The, we plotted the binomial variables in order to see if their distribution is balanced.



These ditribution plots show that the proportion of statistical units with *Event*=1 is only `{r} sum(data$Event==1)/n`. Even *Gender*, *Smoking* and *BP* have unbalanced distributions, but the number of elements in the two levels for all these variables are enough to say that we are not in presence of rare classes.

!!!!!!! Should we add here this?

In the BMC article cited above, the conclusion is drawn that the variables *Ejection.Fraction* and *Creatinine* are the best performers when training the models. Therefore, we want to put special emphasis on the correlation between these two predictors and the response variable *Event*. Since these are numerical and the response variable is continuous, we must perform a correlation test according to these characteristics: Point-Biserial Correlation.

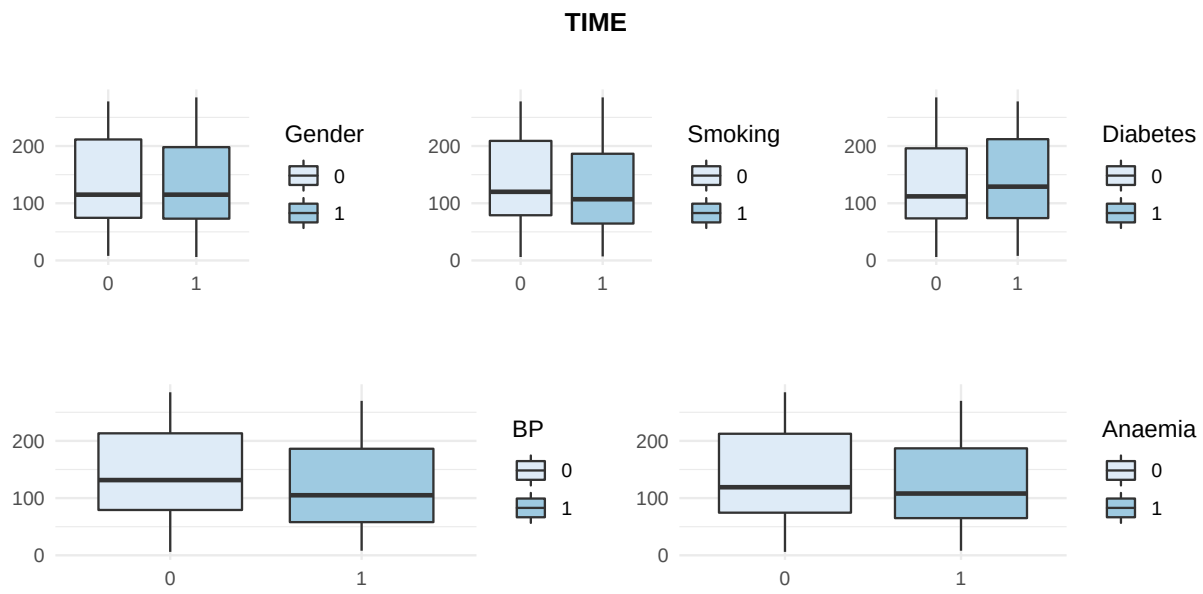
```
## Biserial Correlation between Ejection.Fraction and Event is:  0.2855952
```

```
## Biserial Correlation between Ejection.Fraction and Event is:  -0.3282622
```

The fact that the two best predictors according to BMC article have a correlation with the response variable of approximately 0.3 will make our predictions rather mediocre. This will be a clear limitation in the project, because not only we have few instances available but now we encountered low correlations.

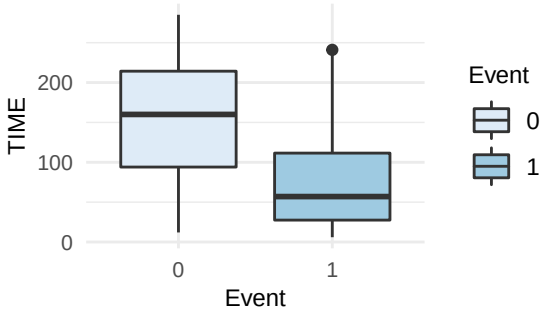
Time and factors

An important aspect to consider is how the time under observation is related to the other variables. The following boxplots show that there is not relation between the time and the binomial variables.



This result is interesting, since it tell us that the experiment from which the data has been collected is unbiased in respect of differences between the persons under observation. In other words, the selected people were kept under observation indipendently from the collected variables that described themselves. It is important because it means that we can rely the way data has been collected and use it to provide results that will not be biased.

We have to consider aside the boxplot about *TIME* in respect to the *Event*. It is clear that the time under observation for people that have presented an hart attack is lower than the the one of the ones that did not. As expected, this indicates that the heart attack looks to reduce the time under observation. The opposite would have been unexpected. It can seems obvious, but not having this result would have lead us to conclude that the data were not able to explain the relation between the explicative variables and the answer variable.



Countinous variables vs Event or cathegorical variables

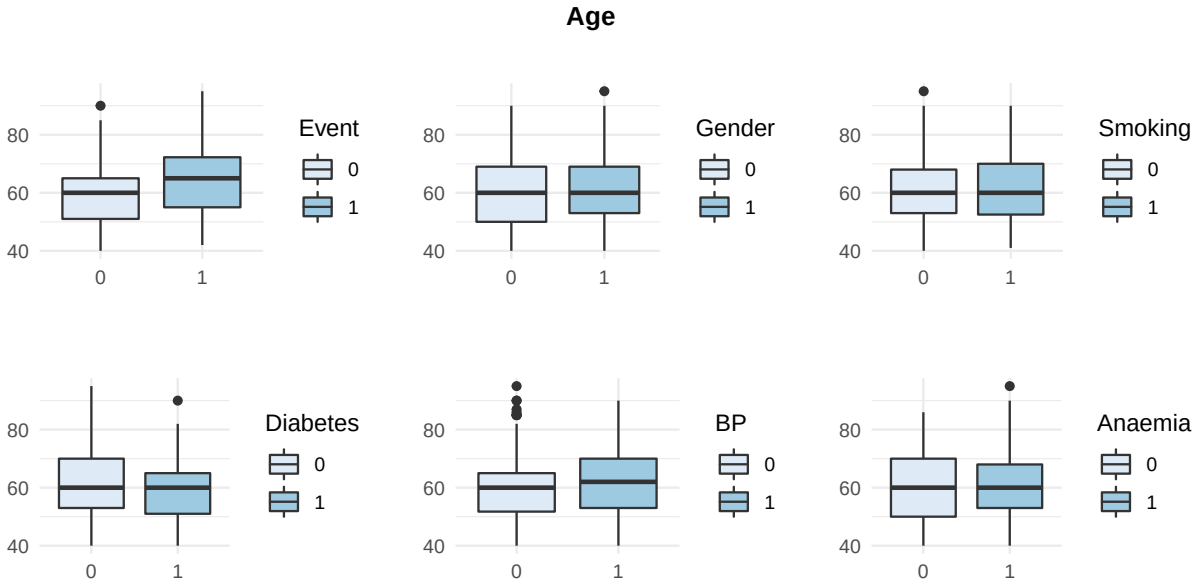
Then, we looked at the relation between the pairs of categorical and continuous variables, included the *Event.*

The scope of the following analysis is double:

a) to understand if the event of having an heart attack is related to different distributions of the continuous variables, i.e. if the population of people having *Event*=1 is the same of the one having *Event*=0

b) to understand if and how the categorical variables are related to the continuous ones.

Age vs factors

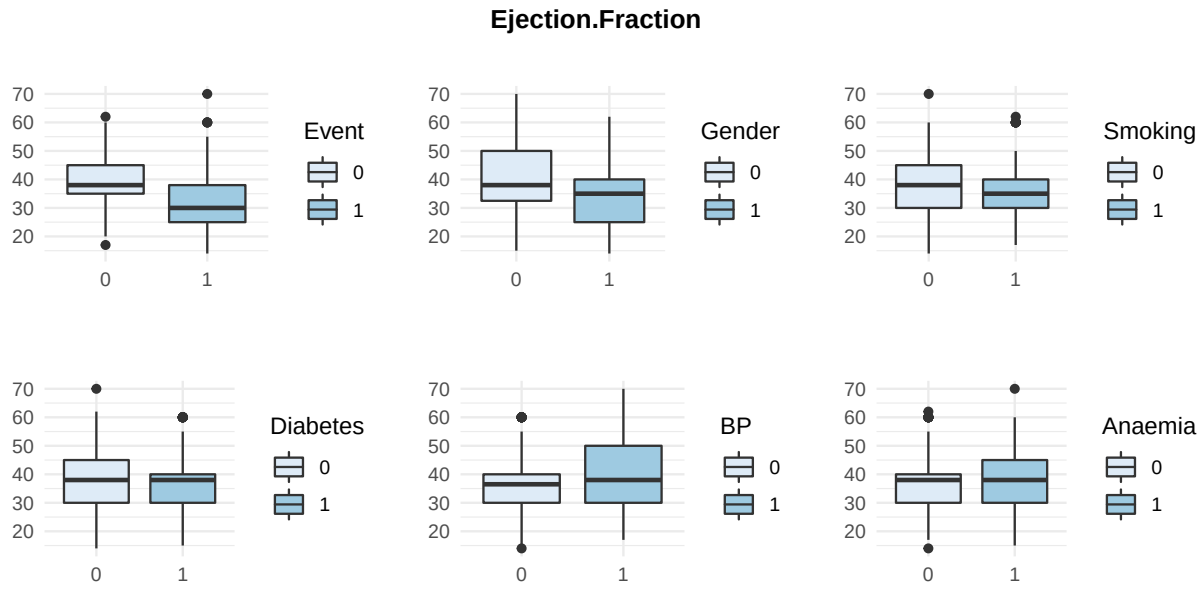


It seems that the population on which the event has occurred is older than the one that did not observed it. It is reasonable, since heart attack are generally more frequent in older people. For all the other variables, there is basically no difference between the groups.

Some points are more than 1.5 times the interquartile distance far from the median, but thay are few and there is not enough evidence to consider them as outliers. Anyways, it is osmething that we should keep in mind.

As expected, the youngest observed age is fourty.

Ejection Fraction vs factors

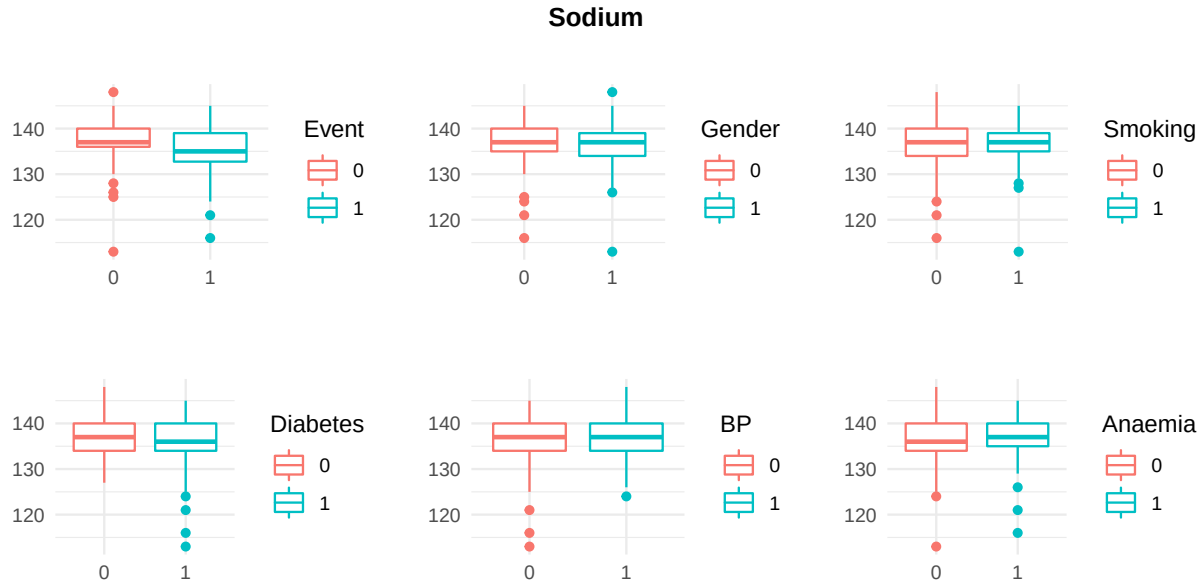


It seems that the population on which the event has occurred has lower values of *Ejection fraction*. That means that this variable can be useful to describe the distribution of the vent over the population.

About the relation between this variable and the categorical ones, we can say something similar to what we said for the *Age*.

For men, the *Ejection fraction* looks to be lower as for persons with *Diabetes*. The remaining ones are not really related to this continuous variable.

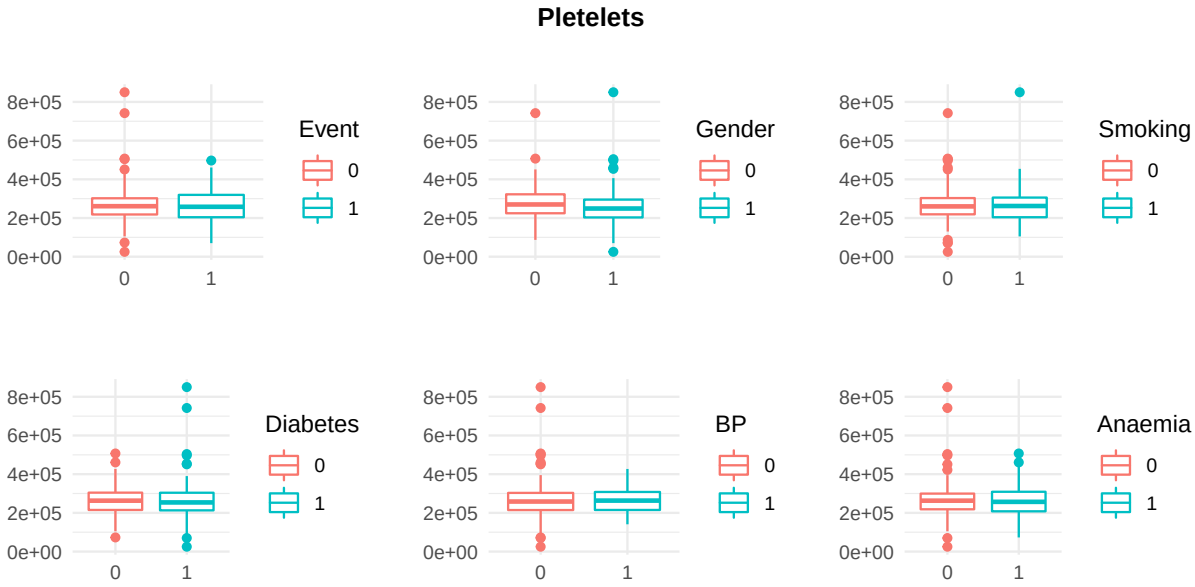
Sodium vs factors



In this situation data look to be less regular. There are multiple points that are enough far from the median to allow us to say that a deeper analysis of outliers could be considered. In particular, some data present an unusual low value of *Sodium*.

Talking about differences between the distribution of the amount of *Sodium* in the two groups, there is not difference to be noticed. On the other hand, this variable does not look very useful in term of explaining the distribution of the *Event*, since the level of *Sodium* is almost the same for both the groups of people having had an heart attack while being under observation and people who didn't.

Pletelets vs factors



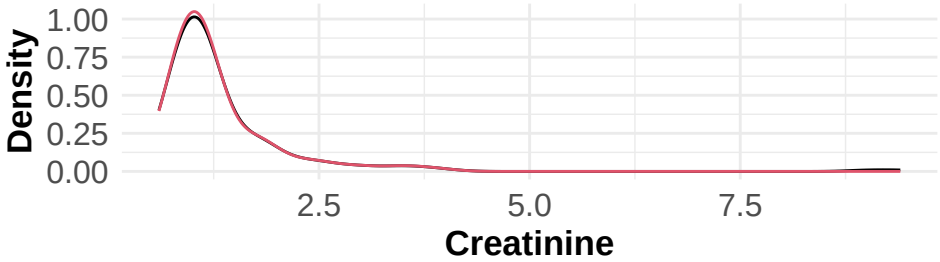
The *Pletelets'* value has not distribution changes between the people having had the heart attack and the others. This suggest that this variable is not really useful for our scopes.

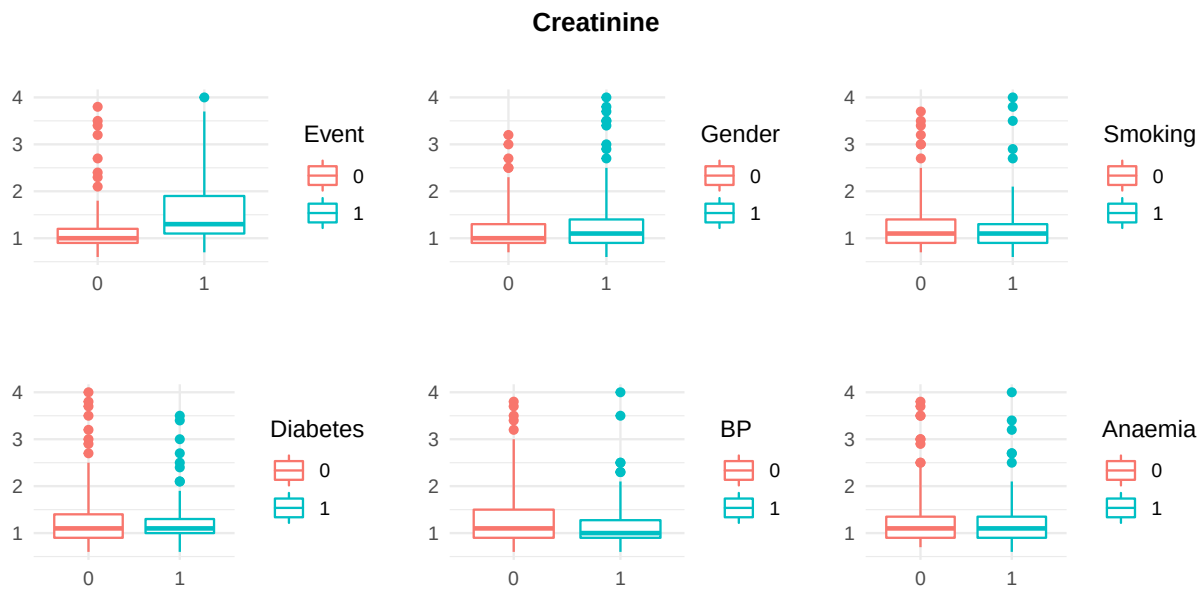
Furthermore, we can notice that there is not difference in distribution between this continuous variable and all the others.

Cratininine vs factors

At the beginning of our analysis we noticed that its distribution has a long right tail. In order to provide a sensefull visualization, we removed those very high value. More specifically, we removed from the visualization observation with a level of Creatinine higher than 5.

#ADD LEGEND TO THE GRAPH



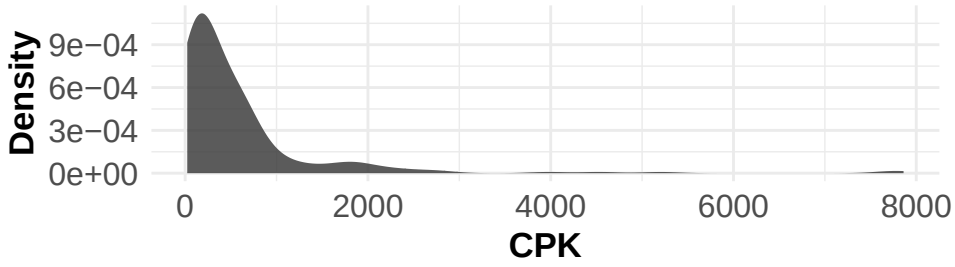


As saw before, the distribution of the *Creatinine* variable is uneven. Anyways, it seems that this variable can be considered between the ones useful to forecast the event, since people having observed the event during the observation time seems to have higher value of *Creatinine*.

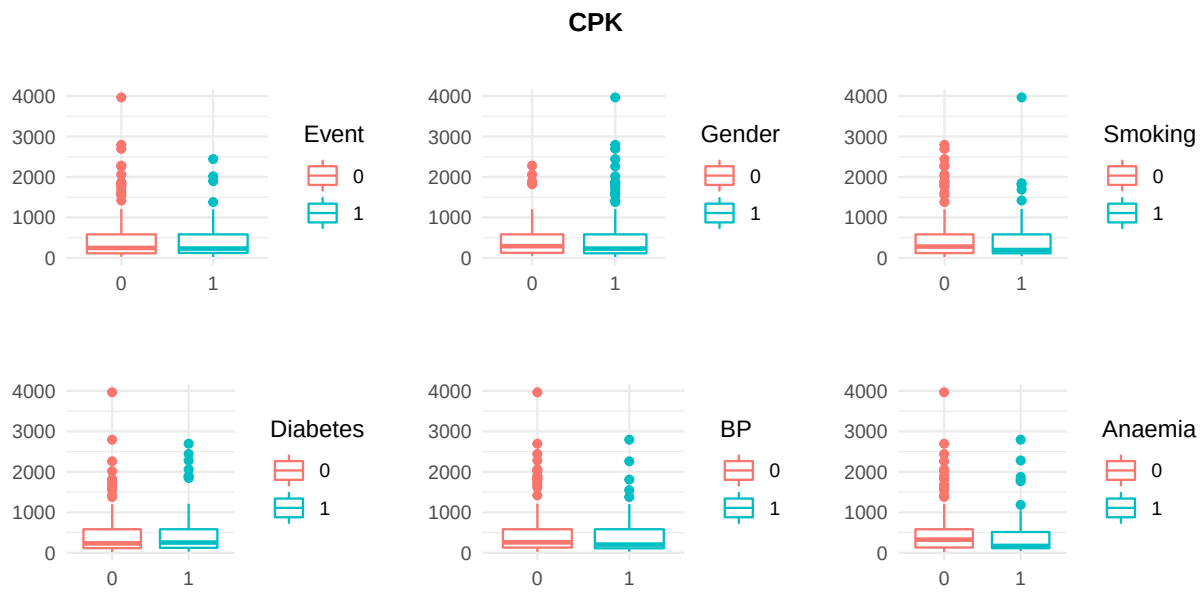
On the other hand, the other categorical variables do not present differences in distribution in the two groups identify by the two levels of each categorical variable.

CPK vs factors

Before analyzing the level of enzyme *CPK*, we had to apply the same process we applyed to the *Creatinine* variable. In fact, its distribution is unbalanced:



This curve is very similar to the distribution of the *Creatinine* variable. This relation between the two of them can also been seen throgh the following graphs:



Even the distribution of CPK is unbalanced to the rigth, with a lot of big values. Unfortunately, the *CPK*/*Event* boxplot shows that the former variable is probably not really useful to explain the latter one. The relation between the level of CPK and the other factors is the same of the *Creatinine* variable.

From this latter graphs, we can conclude that the relation between these two variables (*Creatinine* and *CPK*) is very strong. The difference between the two of them is only in the effect that they have on the distribution of the *Event*. While the *Creatinine* level looks to have an effect on the heart attacks, the *CPK* does not.

We should take into account this observation while performing variable selection.

Pairs of continuous variables and Event

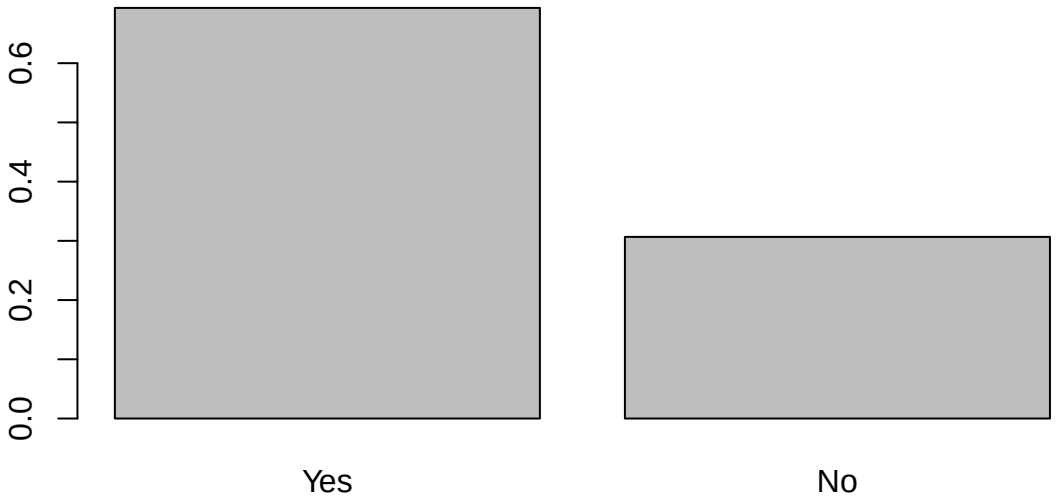
The next step is to look for patterns between pairs of continuous variables and the fact that the event was observed (or not).

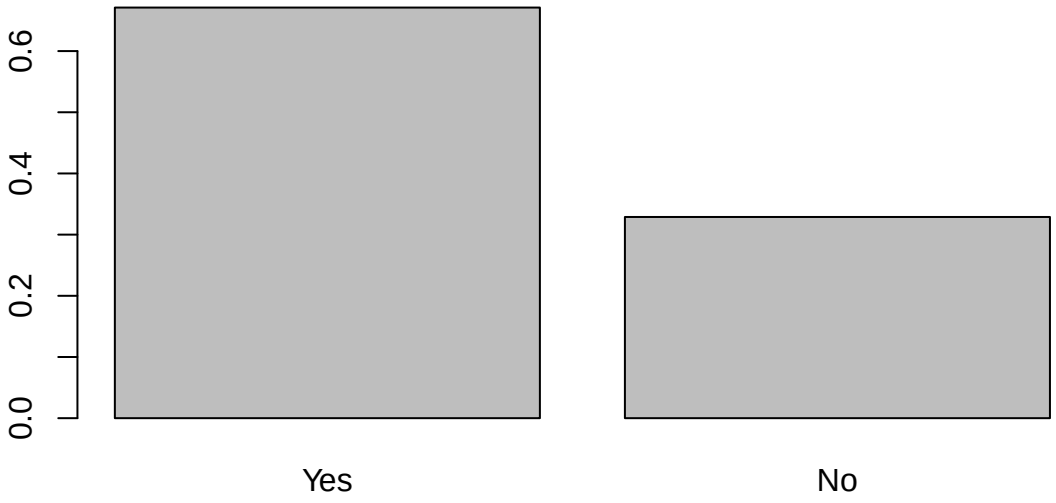
There are not natural clusters defined by these composition of variables.

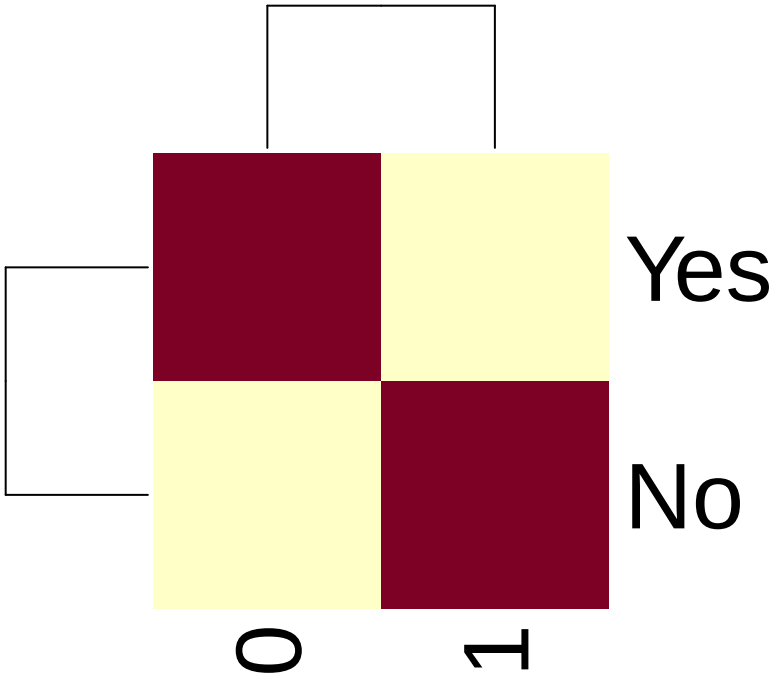
Chategorical variables and Event

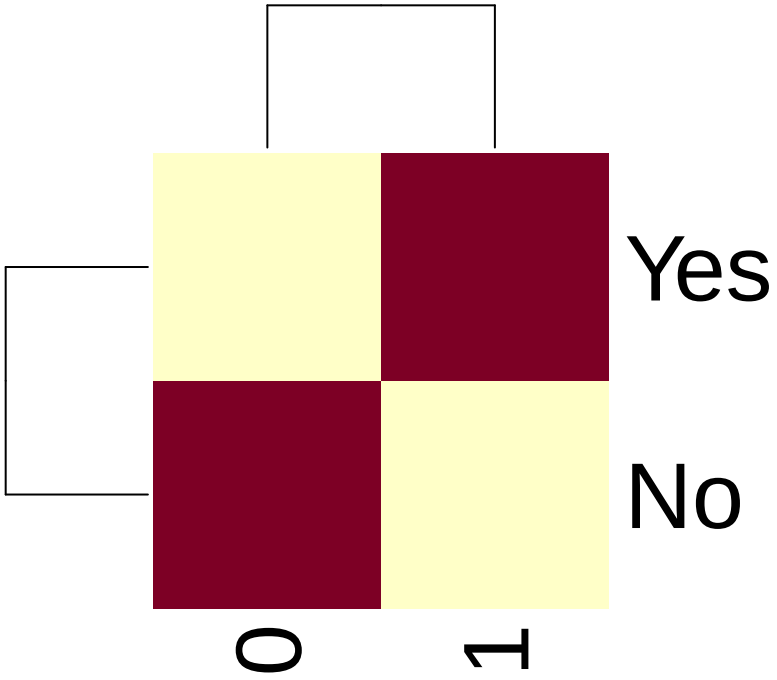
An other interesting thing to see is if the binomial variables have impact over the Event. To perform this check we dived in two groups defined by the levels of the factors, the people that had an heart attack while being under observation. The results follow:

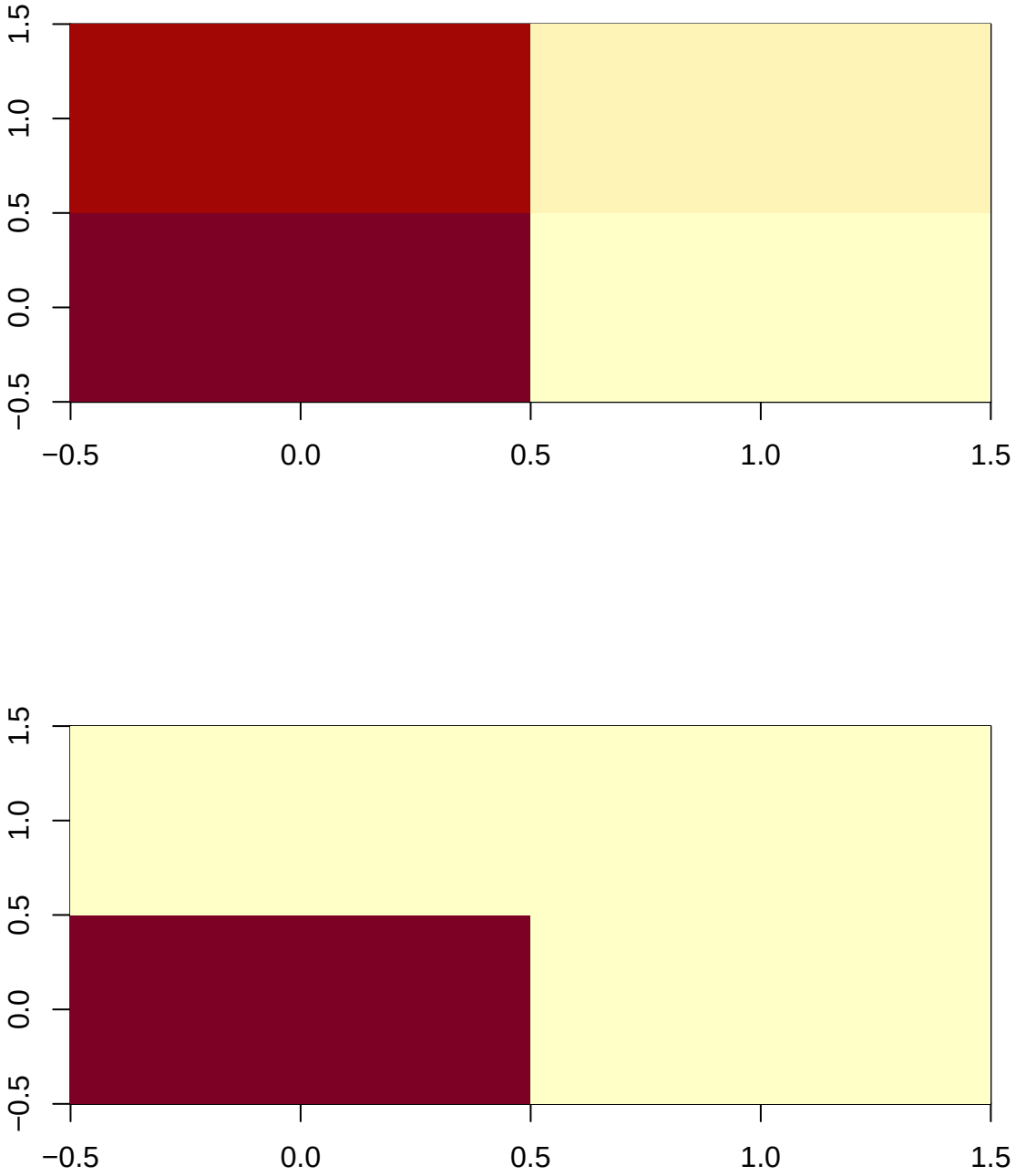
ADD HEATMAPS or HISROGRAMS






Yes
No
0
1


Yes
No
0
1



These graph represent the proportions of people that have or have not experienced the *Event*. The results show that the distribution of this variable is not influenced by the levels of the other categorical ones.

Conclusions of the explorative analysis

To summarize, we have seen that:

- there is a strong relation between variables *CPK* and *Creatinine*;
- the level of *Creatinine* looks to have an high impact in the *Event*;
- the level of *Sodium*, the *Age* and the *Ejection Fraction* look to have a low impact on the Event variable;
- the number of *Pletelets* in the blood and the level of *CPK* have no impact on the Event variable.
- there are not natural clusters defined by pairs of continuous variables
- the categorical variables do not seem to influence the *Event*.

All these aspect are fundamental to have a first overview of the information contained in the data. It does not mean that these will certainly be the variable that will be used in the classification models, but it will help us in better understanding the nature of the case-study.

WHAT ABOUT OUTLIERS?

SURVIVAL ANALYSIS

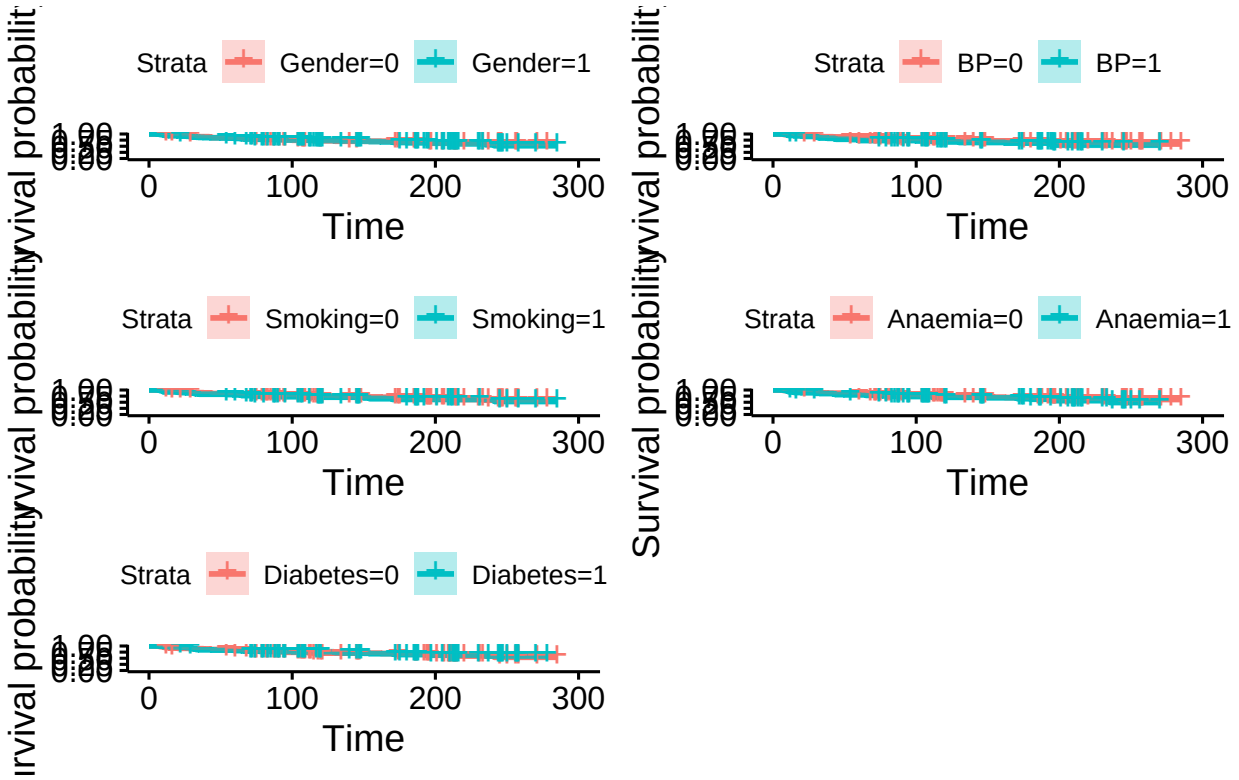
As said in the introduction to the data, this dataset has a specificity regarding the way we the variable *Event* has been collected. More precisely, each patient had been under observation for a certain number of days (variable *TIME*). That means that the *Event* could be observed only under the period of observation of each statistical unit. This considerations leaded us to use the traditional approach for data of this type, i.e. survival analysis. Our main idea is that this kind of approach will better explain the behavior of the data, leading us to a deeper comprehension of the relation between the variables and the *Event* and, most of all, to a beter choiche of the features to include in our classification models.

Non-parametric analysis

In the survival context, the first thing to do is to look at the survival curves using the Kaplain-Meyer method (more details at Goel, Manish Kumar et al. "Understanding survival analysis: Kaplan-Meier estimate." International journal of Ayurveda research vol. 1,4 (2010): 274-8. doi:10.4103/0974-7788.76794).

Individual effect of the factors

First, we evaluated the dichotomous variables.



Each pair of survival curves defined by different levels of the categorical variables have confidence intervals that overlaps, as it can be seen throw the fact that the red area and the light blue one always overlap. This means that, from a descriptive point of view, these variables do not have effects on the answer. This is in line with what we saw in the previous explorative analysis.

Joint effect of the factors

We can perform a general test to see if the combination of all these factors, taken together, has significant effect on the survival curve.

Previously, we saw the comparison between survival curves of groups of people individuated by each of the factors, taken individually. From those graphs, we could notice that the lines generally does not intersect. On this base, we can reasonably use a *logarithmic rank test* for the hypothesis that all the survival curves defined by all the combination of the possible values of the binomial variables are equal. The overall p-value of this test is `{r} pchisq(ss$chisq, df, lower.tail=F)`. That means that, even if individually the variables does not look to be very useful, *their combination can be used to explain the behavior of the heart attacks among the population of interest* .

To conclude, we can say that each of the categorical variable does not seem to have impact on the event if individually considered, the combination of all the categorical variables seems to have impact on the event.

These results imply that, while doing variables selection and, in general, building our models, we will have to pay attention in finding the set of variables which the joints effect is explicative, even if the individual ones are not.

Survival models

We can now apply survival regression models to better understand the utility of the different variables in explaining the survival ratio. In other words, we will make some assumption over the data, apply a

parametric model that can be build on the top of those assumptions, check the bounty of this model and, if the measures are indicating that it well fit the data, extract conclusion from its output.

It should be noticed that we are still working in a descriptive paradigm. Our scope is not to provide previsions on future observations, but only to understand, using the training set, which are the useful variables.

We applied a Weibull model.

WEIBULL MODEL

Assumption:

- the residuals follows a Weibull distribution (https://en.wikipedia.org/wiki/Weibull_distribution)
- the risks of observing the event for different populations are proportional
- the hazard ratio is constant (i.e., the risk for a person with certain levels of the explicative variables divided by the risk for a person with different levels of those variables is constant)

Model: $S(time) = e^{-\lambda time^{\alpha}}$

First, we trained the model with all the variables. The optimal starting model would be the one with all the possible interactions since we noticed in the previous non parametric analysis that some variables are ineffective if individually considered but could be important if jointly taken into account. Unfortunately, the small amount of data we have prevent us to apply the complete model. We will then manually adapt some forward selection to individuate the useful features and discard the others. The selection is semiauthomatized based on the Akaike Information Criteria.

```
##                   Value Std. Error     z    p
## (Intercept)        7.55       0.95  7.95 0.00
## Gender             0.23       0.31  0.74 0.46
## Smoking           -0.10       0.29 -0.35 0.73
## Diabetes          -0.13       0.26 -0.50 0.62
## BP                -0.33       0.26 -1.30 0.19
## Anaemia           -0.31       0.24 -1.26 0.21
## Age               -0.05       0.01 -4.11 0.00
## Ejection.Fraction  0.06       0.01  4.63 0.00
## Creatinine        -0.41       0.08 -5.34 0.00
## Pletelets          0.00       0.00  0.14 0.89
## Log(scale)         0.01       0.10  0.06 0.96
```

The variables *Gender*, *Smoking*, _Diabetes_and *Pletelets* are not significant for this model. Before discard them we tried, for each of them, to see if their interactions with the other variables are to be kept in the model.

```
##                   Value Std. Error     z    p
## (Intercept)        6.63       0.83  8.02 0.00
## Gender             0.22       0.30  0.72 0.47
## Age               -0.04       0.01 -3.15 0.00
## Ejection.Fraction  0.06       0.01  4.64 0.00
## Creatinine        -0.42       0.07 -5.78 0.00
## Gender:Smoking     2.34       0.93  2.51 0.01
## Age:Smoking       -0.04       0.01 -2.76 0.01
## Log(scale)        -0.01       0.10 -0.08 0.93
```

The only significant ones are *Smoking:Gender* and *Smoking:Age*.

```
##                   Value Std. Error     z    p
## (Intercept)        6.76       0.82  8.28 0.00
## Age               -0.04       0.01 -3.06 0.00
## Ejection.Fraction  0.06       0.01  4.62 0.00
## Creatinine        -0.42       0.07 -5.74 0.00
## Smoking:Gender     2.50       0.91  2.75 0.01
## Age:Smoking       -0.04       0.01 -2.85 0.00
## Log(scale)        -0.01       0.10 -0.07 0.95
```

Given the interaction with *Pletelets*, *Gender* is now significant. It is then kept in the model.

The same process has been applied for all the variables, obtaining the following model:

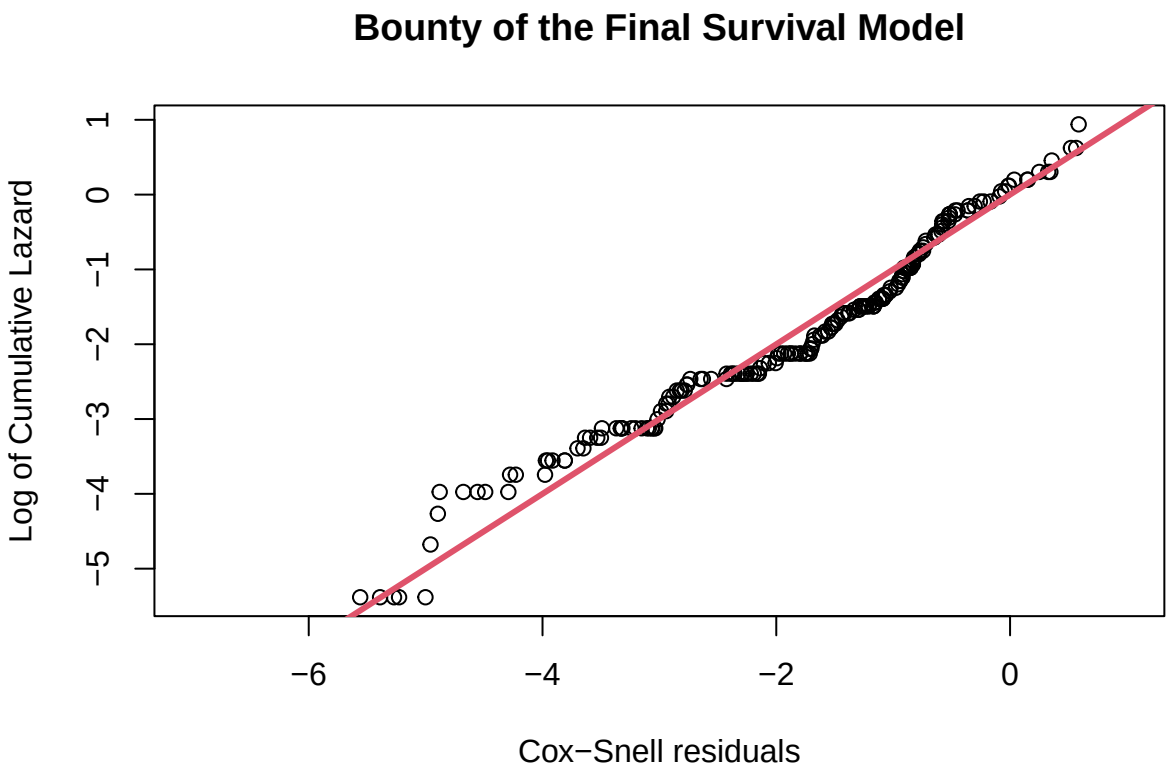
```
##  [1] "(Intercept)"                  "Pletelets"
##  [3] "Age"                          "Diabetes"
##  [5] "BP"                           "Ejection.Fraction"
##  [7] "Gender"                       "Smoking:Gender"
##  [9] "Smoking:Age"                  "Gender:Pletelets"
## [11] "Age:Ejection.Fraction"        "Ejection.Fraction:Diabetes"
## [13] "Ejection.Fraction:Creatinine" "Log(scale)"
```

It has to be noticed that the scale parameter is not significantly far from 0. It means that the (easier) exponential model would be a better choice for this data. Based on this observation, we then decided to use it: $S(time) = e^{-\lambda time}$. The results in terms of variables to be kept are the same.

We then have the following model:

```
##                               Value    p
## (Intercept)                   -4.35 0.09
## Pletelets                      0.00 0.31
## Age                            0.09 0.01
## Diabetes                       2.71 0.00
## BP                            -0.32 0.22
## Ejection.Fraction              0.38 0.00
## Gender                         1.47 0.16
## Smoking:Gender                 2.65 0.01
## Smoking:Age                   -0.04 0.00
## Gender:Pletelets               0.00 0.16
## Age:Ejection.Fraction          0.00 0.00
## Ejection.Fraction:Diabetes    -0.09 0.00
## Ejection.Fraction:Creatinine -0.01 0.00
```

Usual methodologies to asses the bounty of models are not appropriate for these family of models. In fact, usually there is no the concept of time under observation to be taken into account. This exponential model, on the other hand, deal with that time. For this reason, we performed the assessment of the fit of the final model using the *Cox-Snell residuals*. The approach allowed us to verify about correctness of the distribution assumptions. The idea is that the relation between these residuals and the logaritm of the cumulative hazard ratio estimated using the Fleming-Harrington method is linear. We then plotted these two variables obtaining a confirm about this linear relation. It means the the assumption of exponential distribution is quite fine.



The process through which we selected these variables was, as already mentioned, purely based on the AIC. It leaded to a model were there are iterations of terms that are not individually present. In order to increment the interpretability of the results, we decided to add those variables anyways.

CHECK IF MY IDEAS MAKE SENSE

Meaning of these interactions

- *Smoking* and *Gender*: the data is collected in Pakistan. It could be reasonable to think that there are not collected variables that could distinguish women that smoke from the ones that does not. For example, it could be that women that smoke are from specific social subgroups where other risky behaviors are more frequent than in the average men population (e.g., the women that smoke are western women with different habits from Pakistani people).
- *Gender* and *Pletelets*: see Gender-based differences in platelet function and platelet reactivity to P2Y12 inhibitors Ranucci M, Aloisio T, Di Dedda U, Menicanti L, de Vincentiis C, et al. (2019) Gender-based differences in platelet function and platelet reactivity to P2Y12 inhibitors. PLOS ONE 14(11): e0225771. DOI
- *Age* and *Ejection Fraction*: the increment of the ejection fraction leads to a decrease of the survival ratio (of -0.003977) for each more year of age. This result in completely aligned with experimental results (Chuang, Michael L et al. "Association of age with left ventricular volumes, ejection fraction and concentricity: the Framingham heart study." Journal of Cardiovascular Magnetic Resonance vol. 15,Suppl 1 P264. 30 Jan. 2013, DOI.

- *Ejection Fraction* and *Diabetes*: the increment of the ejection fraction leads to a decrease of the survival ratio (of -0.0854973) for people with diabetes. This is out of the range of our knoweldge and after some research we were able to say that this is still an open issue (Ehl NF, Kühne M, Brinkert M, Müller-Brand J, Zellweger MJ. Diabetes reduces left ventricular ejection fraction–irrespective of presence and extent of coronary artery disease. Eur J Endocrinol. 2011 Dec;165(6):945-51. doi Epub 2011 Sep 8. PMID: 21903896.)
- *Ejection Fraction* and *Creatinine*: the same as before.

CPK and Creatinine

1) In the previous analysis we saw that *CPK* and *Creatinine* are highly related. On the other hand, the correlation between the two is very low: $Cor_{Creatinine,CPK} =$ -0.102.
2) The explorative analysis showed that *CPK* doesn't seem useful to classify the *Event*.
3) In addition, we just saw that it is not even founded to be used neither using the Kaplain-Meyer methor nor
4) the Exponential models.
5) Furthermore, we applied a generalized linear model with only *CPK* and *Creatinine* to explain the *Event*.

The scope was to answer the question if both variables were useful if jointly used to (linearly) classify the Event. To look at this kind of linear dependence we have generated two models:

- $logit(Event) = \beta_0 + \beta_1 CPK + \beta_2 Creatinine + \epsilon$
- $logit(Event) = \beta_0 + \beta_1 CPK + \epsilon$

The significances of the parameters of the first model are, respectively: 0.23, 0.

Before definitely discharging CPK as a variable for the classification of the level of Event, we performed two other tests to see even over the linear dependency. In order to do that, we built the following model: $logit(Event) = \beta_0 + \beta_1 CPK + \beta_2 CPK^2 + \beta_3 CPK^3 + \epsilon$
. All these coefficients where not significantly different from 0. In fact, their p-values are respectively: 0.55, 0.25, 0.19.

On the basis of these 5 different approaches the gave all the same result we decided to drop *CPK* can be dropped for our dataset. In our context, it means that CPK is not considered useful in inferring the *Event* of interest if combined with *Creatinine.*

Conclusions

In conclusion, it can be said that the variables that significantly directly impact the mortality curves are *Pletelets*, *Age*, *Diabetes*, *BP*, *Ejection.Fraction* and *Gender*. *Smoking1* and *Creatinine* have impact only if combined with some of the others. Relying of these results, we can create some new features on the top of the original ones. More precisely, we are generating new features representing the iterations that were founded as significant by the Exponential model.

CLASSIFICATION

MLR package

We will be using R package *mlr* as a framework. It follows the following pattern:

1. Define task: what will be predicted and any consideration given the data to be used (e.g. partitioning)
2. Create learner: which ML algorithm will be used, as well as hyperparameters, preprocessing, etc.
3. Determine resample and measure: which validation strategy will be followed and which metric will be used.
4. Train the model
5. Predict and obtain model accuracy

In our case, since there are many models that we want to apply, we will be doing a benchmark before actually training. This will give us a glimpse of the accuracies that these methods would get. After that, we will be tuning hyperparameters in the best performing models.

Creating the task and Validation partitioning

k-Fold-Cross-Validation

If the sample size is small, it is recommended to use repeated k-Fold-Cross-Validation, as it achieves a good bias-variance balance and, given that there are not many observations, the computational cost is not excessive. (In other words, you could even do a high number of folds because it will take little time). For instance, in our benchmark we will use 5 folds with 10 reps, which is not equivalent to 50-folds-cross-validation. The issue with partitioning data with few instances is that results may depend on luck. That's why we are considering 10 different partitions with 5 folds, so results will not depend on chance, but the average will be closer to reality.

Machine Learning Analysis

CHANGE THIS !!!

Why mlrCPO Reference

Basically you define a pipeline using the mlrCPO pipeop %»%. Every pipeop you put before the learner will be applied directly before the training but after the train test split.

Classification Analysis

We will not consider the variable *Time* from now on because it would not be information that we would obtain from a new patient, it is only a control variable.

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
## Starting parallelization in mode=socket with cpus=2.
```

```
## Exporting objects to slaves for mode socket: .mlr.slave.options
```

```
## Mapping in parallel: mode = socket; level = mlr.benchmark; cpus = 2; elements = 9.
```

```
##        task.id            learner.id tpr.test.mean timetrain.test.mean
## 1 HeartFailure       glm.range.scale     1.0000000               0.014
## 2 HeartFailure     naive.range.scale     0.9833333               0.011
## 3 HeartFailure       knn.range.scale     0.8222222               0.012
## 4 HeartFailure     rpart.range.scale     1.0000000               0.015
## 5 HeartFailure        rf.range.scale     1.0000000               0.056
## 6 HeartFailure    ranger.range.scale     1.0000000               0.034
## 7 HeartFailure       SVM.range.scale     0.9777778               0.016
## 8 HeartFailure neuralnet.range.scale     1.0000000               0.031
## 9 HeartFailure   xgboost.range.scale     1.0000000               0.020
```

```
## Stopped parallelization. All cleaned up.
```

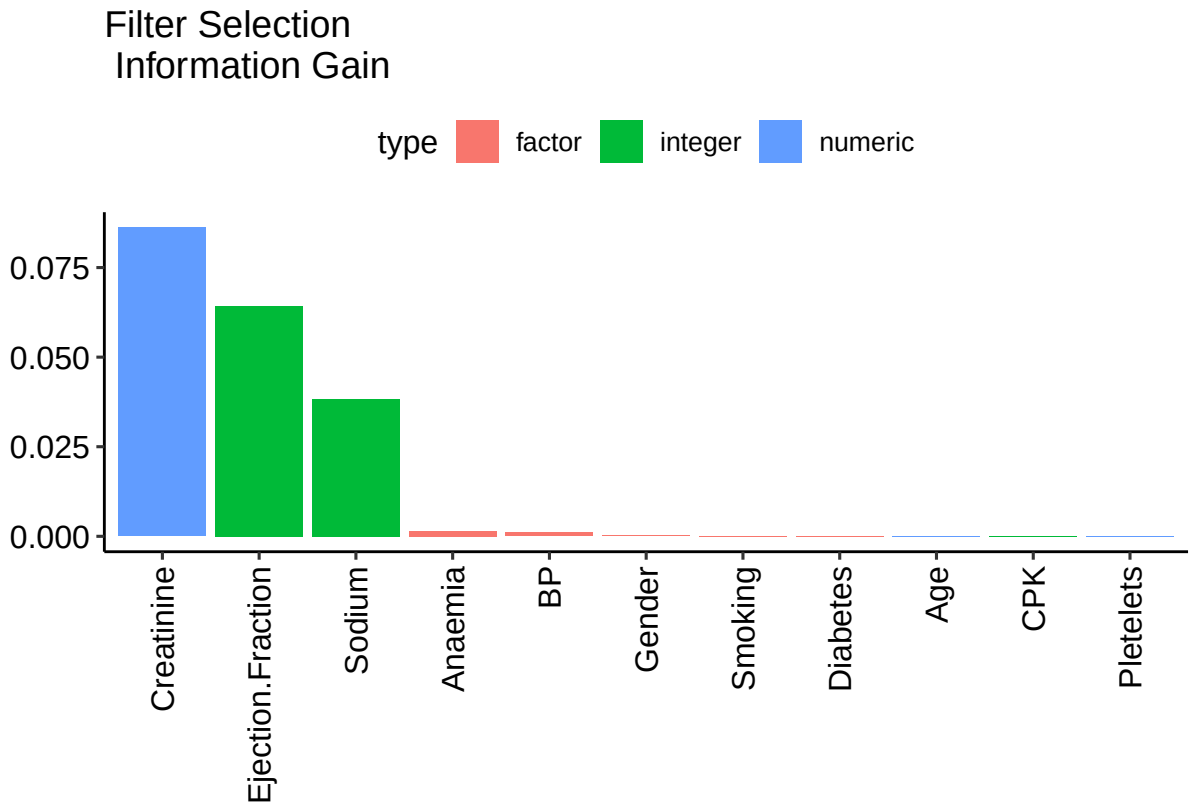
Filter Selection

In order to decide which features would be selected to see if they help the models get better predictions, we will perform filter selection. Unlike feature selection, this method does not require of a learner to reach a conclusion. We have checked that depending on the learner selected, results change drastically. Filter selection was found to be a more impartial methodology.

The metric used for this filter selection is *Information Gain*. It measures the reduction in entropy (or surprise) by splitting a dataset according to a given value of a random variable. A larger information gain suggests a lower entropy group or groups of samples, and hence less surprise. Reference

Entropy quantifies how much information there is in a random variable, or more specifically its probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy.

```
## Warning: package 'FSelectorRcpp' was built under R version 4.1.3
```



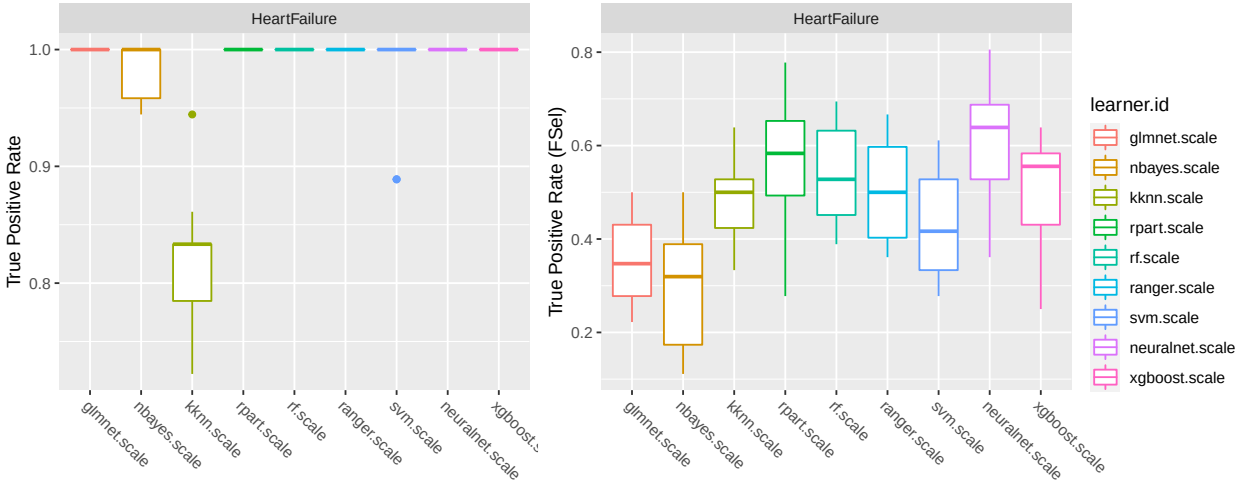

```
##                 name    type                          filter        value
##  1:        Creatinine numeric FSelectorRcpp_information.gain 8.614224e-02
##  2: Ejection.Fraction integer FSelectorRcpp_information.gain 6.418660e-02
##  3:            Sodium integer FSelectorRcpp_information.gain 3.834260e-02
##  4:           Anaemia  factor FSelectorRcpp_information.gain 1.476998e-03
##  5:                BP  factor FSelectorRcpp_information.gain 1.033179e-03
##  6:            Gender  factor FSelectorRcpp_information.gain 2.525059e-04
##  7:           Smoking  factor FSelectorRcpp_information.gain 1.630264e-04
##  8:          Diabetes  factor FSelectorRcpp_information.gain 5.935304e-06
##  9:               Age numeric FSelectorRcpp_information.gain 0.000000e+00
## 10:         Pletelets numeric FSelectorRcpp_information.gain 0.000000e+00
## 11:               CPK integer FSelectorRcpp_information.gain 0.000000e+00
```

We saw in the BMC article that one of their conclusions was: "Our results of these two-feature models show not only that *serum creatinine* and *ejection fraction* are sufficient to predict survival of heart failure patients from medical records, but also that using these two features alone can lead to more accurate predictions than using the original data set features in its entirety."

Therefore, considering the results obtained, the conclusions of the previous analysis, and that the correlation between the response variable and Sodium is approximately 0.2, we have decided to use only *serum creatinine* and *ejection fraction* for the comparison of results.

```
##       task.id           learner.id tpr.test.mean timetrain.test.mean
## 1 HeartFailure       glm.range.scale     0.3555556                0.014
## 2 HeartFailure     naive.range.scale     0.2916667                0.006
## 3 HeartFailure       knn.range.scale     0.4861111                0.008
```

```
## 4 HeartFailure     rpart.range.scale     0.5527778              0.008
## 5 HeartFailure        rf.range.scale     0.5416667              0.047
## 6 HeartFailure    ranger.range.scale     0.5083333              0.038
## 7 HeartFailure       SVM.range.scale     0.4305556              0.022
## 8 HeartFailure neuralnet.range.scale     0.6111111              0.258
## 9 HeartFailure   xgboost.range.scale     0.5000000              0.018
```



Here, our aim is to maximize TPR, because in the medical area it is better to have false alarms (overestimating positives) than not predicting real positives. Some models may have a good accuracy (even better than other models), but that is not the priority in healthcare analysis (such as predicting cancer and other diseases). For instance, a trivial model (e.g. predicting all patients as class 0), would give very high accuracies, not taking into account any of the data provided. That's why we will decide which model is better based on TPR (true positive rate).

In this stage, we would like to decide whether we use the whole dataset or the subset obtained selecting some features. From the results we can comment on three main points:

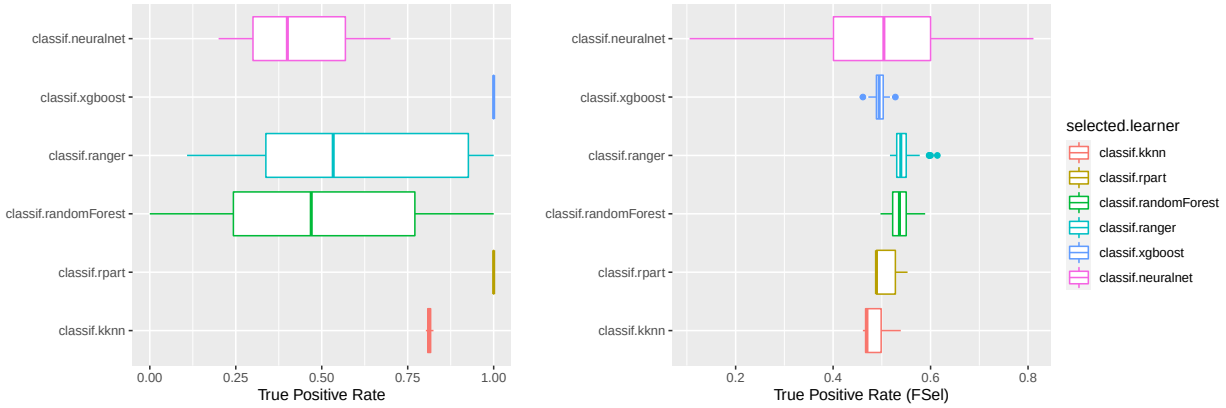
- Weak learners perform visibly worse, some of them even reaching a tpr of around 0.2
- K-nearest neighbor performs surprisingly better when selecting few features. This can be explained because the algorithm does not work well with high dimensions. Performance changed from 0.3 to 0.6, with a default of 7 neighbors. Support Vector Machine also benefits from less features.
- Lastly, we see that overall models perform slightly better with the feature selection. However, results vary much more depending on the partition. In view of the results, we decide to perform tuning for both datasets using a subset of the best performing learners. We will be tuning mainly "strong learners" (i.e. excluding *glm*, *naive bayes*) as well as SVM, because we believe they will be able to discard useless features and reach better performances while being consistent, not fluctuating much.

For now, we should mention that *rpart* (tree) is accomplishing good results, and would be convenient for this project since it is easy to explain.

Tuning models

In this section, we are going to tune different hyperparameters for the following models: kknn, rpart, random forest, ranger, extreme gradient boosting, and neural network. Below we will explain the reasoning behind selecting each parameter. The same validation method as for the previous analysis will be used, i.e. Repeated

Cross Validation with 2 folds and 5 iterations. For the hyperparameter optimization, random search will be executed. It is shown to be more efficient in a paper by Bergstra, J. and Bengio, Y. (2012).Reference. A total of 200 iterations will be splitted among the 7 learners chosen for tuning.



Finally, based on these graphs we can conclude that selecting few features is essential for this study. Excluding *neuralnet*, that clearly fails for this data set, all models are consistent in the right-side graph. The two bagging models (*ranger* and *randomForest*) get equivalent results, so one of these two should be used in the final model.

```
## Warning: package 'reshape' was built under R version 4.1.3

##
## Attaching package: 'reshape'

## The following object is masked from 'package:Matrix':
##
##     expand

## The following object is masked from 'package:dplyr':
##
##     rename

## The following objects are masked from 'package:tidyr':
##
##     expand, smiths

## Using selected.learner, error.message as id variables

## Joining, by = "selected.learner"

##        selected.learner       tpr parameter      value
## 1          classif.kknn 0.5388889         k  5.0000000
## 2         classif.rpart 0.5527778  minsplit 20.0000000
## 3         classif.rpart 0.5527778  maxdepth  6.0000000
## 4  classif.randomForest 0.5888889     ntree 92.0000000
## 5  classif.randomForest 0.5888889      mtry  1.0000000
## 6  classif.randomForest 0.5888889  nodesize 30.0000000
## 7        classif.ranger 0.6138889      mtry  2.0000000
## 8        classif.ranger 0.6138889     trees 27.0000000
```

```
## 9        classif.ranger 0.6138889      size 42.0000000
## 10      classif.xgboost 0.5277778       eta  0.4950905
## 11      classif.xgboost 0.5277778   nrounds  6.0000000
## 12      classif.xgboost 0.5277778 max_depth  6.0000000
## 13    classif.neuralnet 0.8111111    hidden  7.0000000
## 14    classif.neuralnet 0.8111111 threshold  3.5596021
## 15    classif.neuralnet 0.8111111   stepmax  2.0224134
```

Explanation of parameters

- RPART For the model *rpart* we are tuning two of the "stopping" parameters in the algorithm, that tells the tree when to stop growing (a way of pruning)
  - *minsplit*: minimum number of observations that must exist in a node in order for a split to be attempted. If it is too small, the tree will keep growing and probably lead to overfitting, if it is too big the accuracy may decrease.
  - *maxdepth*: Set the maximum depth of any node of the final tree, with the root node counted as depth 0. Again, if we let the tree make too many splits, it will lead to overfitting. A tree that is too small may generalize too much.
- RANDOM FOREST
  - *ntree*: this should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
  - *mtry*: number of variables to possibly split at in each node. (Cannot be bigger than the number of variables).
  - *nodesize*: Setting this number larger causes smaller trees to be grown (and thus preventing from overfitting and takes less time)
- RANGER Same parameters as in random forest, since it's the same algorithm, just optimized.
- SVM
  - *cost*: is the penalty parameter, which represents misclassification or error term. If it's smaller, the decision boundary will have a big margin, resulting in more missclassified data. If it's bigger, the penalty is higher and the algorithm tries to minimize missclassifications.
  - *gamma*: large gamma values will probably lead to overfitting. Small values will generalize too much.
- ADA
  - *nu*: This parameter is provided to shrink the contribution of each classifier, sometimes called learning rate. Reducing this parameter will mean the weights will be increased or decreased to a small degree, forcing the model train slower, although sometimes results in better performance scores.
  - *iter*: number of boosting iterations to perform (total number of weak learners).
- XGBOOST
  - *eta*: control the learning rate. Used to prevent overfitting by making the boosting process more conservative.
  - *nrounds*: max number of boosting iterations.
  - *max_depth*: maximum depth of a tree
- NEURALNET

- *hidden*: hidden neurons, for each layer (only one layer in our case since it will make it faster obtaining very good results) To help the algorithm converge when enlarging *hidden*, we need to make the *threshold* and *stepmax* bigger as well.
- *threshold*: threshold for the partial derivatives of the error function as stopping criteria.
- *stepmax*: maximum steps for the training of the neural network, if we make it bigger, we let more time for the algorithm to converge.

Test result

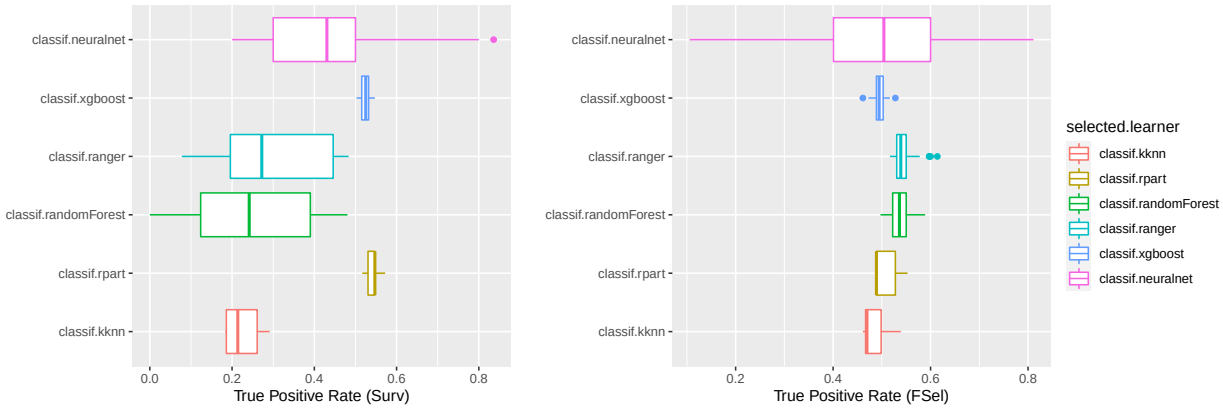
```
## Resampling: holdout

## Measures:             tpr

## [Resample] iter 1:    0.4166667

##

## Aggregated Result: tpr.test.mean=0.4166667

##
```

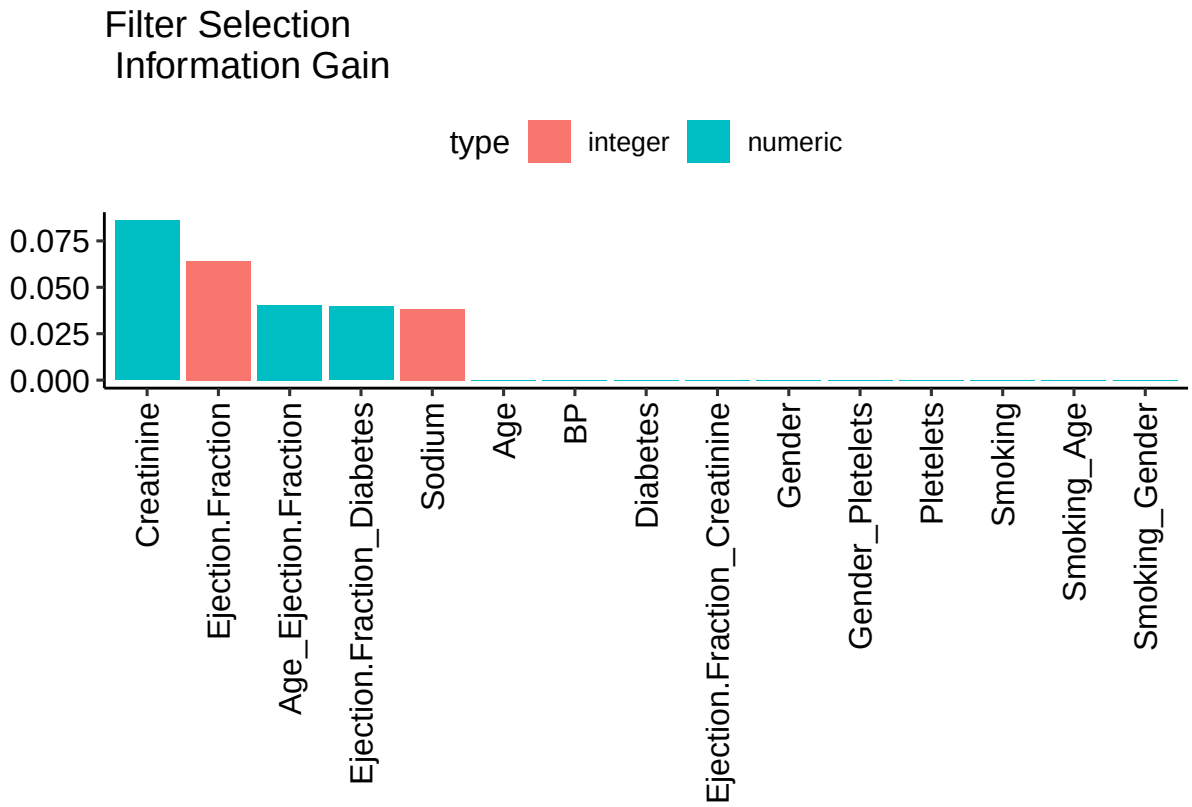
SUrvival Analysis

In this section we want to do a comparison of two different approaches, statistic-based and generic classification. To do so, we have independently chosen the most important features given by each approach, and we will be performing the same machine learning analysis to see which gets a higher true positive rate. Every parameter (e.g. validation methodology, search space) will be the same as the previous analysis.

Tuning models

Comparing results





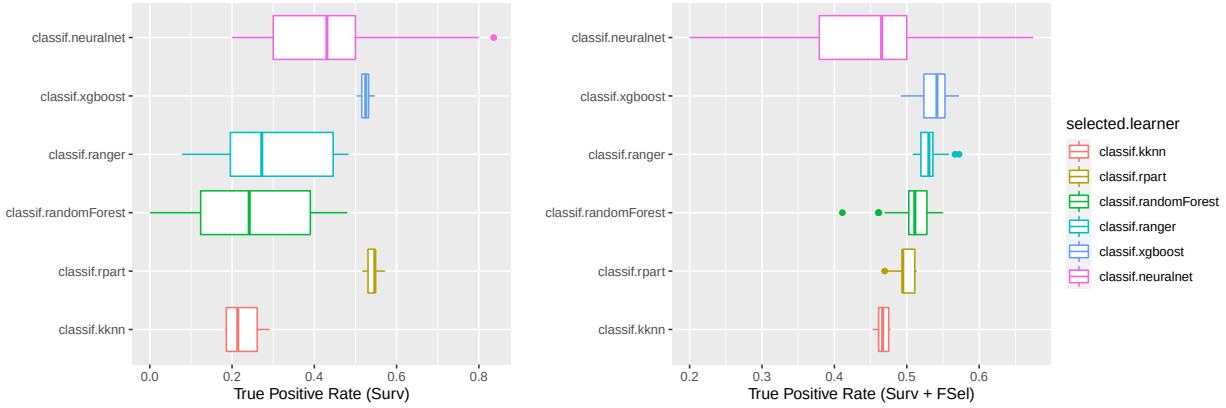

```
##                              name    type                          filter
##  1:                     Creatinine numeric FSelectorRcpp_information.gain
##  2:               Ejection.Fraction integer FSelectorRcpp_information.gain
##  3:          Age_Ejection.Fraction numeric FSelectorRcpp_information.gain
##  4:     Ejection.Fraction_Diabetes numeric FSelectorRcpp_information.gain
##  5:                         Sodium integer FSelectorRcpp_information.gain
##  6:                         Gender numeric FSelectorRcpp_information.gain
##  7:                        Smoking numeric FSelectorRcpp_information.gain
##  8:                       Diabetes numeric FSelectorRcpp_information.gain
##  9:                             BP numeric FSelectorRcpp_information.gain
## 10:                            Age numeric FSelectorRcpp_information.gain
## 11:                      Pletelets numeric FSelectorRcpp_information.gain
## 12:                  Smoking_Gender numeric FSelectorRcpp_information.gain
## 13:                     Smoking_Age numeric FSelectorRcpp_information.gain
## 14:                Gender_Pletelets numeric FSelectorRcpp_information.gain
## 15: Ejection.Fraction_Creatinine numeric FSelectorRcpp_information.gain
##           value
##  1: 0.08614224
##  2: 0.06418660
##  3: 0.04054012
##  4: 0.03948337
##  5: 0.03834260
##  6: 0.00000000
##  7: 0.00000000
##  8: 0.00000000
##  9: 0.00000000
```

```
## 10: 0.00000000
## 11: 0.00000000
## 12: 0.00000000
## 13: 0.00000000
## 14: 0.00000000
## 15: 0.00000000
```

Tuning models

```
## Joining, by = "selected.learner"
```

```
## # A tibble: 6 x 5
##   selected.learner     MedianAllFeat MedianFSel MeanAllFeat MeanFSel
##   <fct>                        <dbl>      <dbl>       <dbl>    <dbl>
## 1 classif.kknn                 0.214      0.467       0.226    0.466
## 2 classif.rpart                0.547      0.494       0.542    0.496
## 3 classif.randomForest         0.242      0.511       0.253    0.510
## 4 classif.ranger               0.272      0.531       0.299    0.531
## 5 classif.xgboost              0.525      0.542       0.525    0.537
## 6 classif.neuralnet            0.431      0.465       0.437    0.450
```




```
## Using selected.learner, error.message as id variables
```

```
## Joining, by = "selected.learner"
```

```
##        selected.learner       tpr parameter       value
## 1          classif.kknn 0.4777778         k   9.0000000
## 2         classif.rpart 0.5138889  minsplit  16.0000000
## 3         classif.rpart 0.5138889  maxdepth   7.0000000
## 4  classif.randomForest 0.5500000     ntree 240.0000000
## 5  classif.randomForest 0.5500000      mtry   2.0000000
## 6  classif.randomForest 0.5500000  nodesize  33.0000000
## 7        classif.ranger 0.5722222      mtry   2.0000000
## 8        classif.ranger 0.5722222     trees 234.0000000
## 9        classif.ranger 0.5722222      size  39.0000000
## 10      classif.xgboost 0.5722222       eta   0.3874785
## 11      classif.xgboost 0.5722222   nrounds   7.0000000
## 12      classif.xgboost 0.5722222 max_depth   8.0000000
```

```
## 13    classif.neuralnet 0.6750000    hidden  10.0000000
## 14    classif.neuralnet 0.6750000 threshold   3.0412305
## 15    classif.neuralnet 0.6750000   stepmax   2.1558356
```

Test result

```
## Warning in makeTask(type = type, data = data, weights = weights, blocking =
## blocking, : Provided data is not a pure data.frame but from class tbl_df, hence
## it will be converted.
```

```
## Resampling: holdout
```

```
## Measures:             tpr
```

```
## [Resample] iter 1:    0.4166667
```

```
##
```

```
## Aggregated Result: tpr.test.mean=0.4166667
```

```
##
```

NOTES Explicability of the models

Stratification (no need for rebalancing because of is not toooo unbalanced. Expain why we stratified.)

Algorithms like naive bayes benefit from gaussian data

show.info = FALSE

PIETRO

plot_bar try color it fix heatmap