Research Log

Jurrien de Jong

4/10/2021

```
codebook <- read.delim("data/codebook.csv", sep = ",")
kable(codebook, caption = "Table 1 : Codebook")
```

Table 1: Table 1 : Codebook

Name	Full.Name	Data.Type	Unit	Description.
age	age	int	years	age in years;
Anaemia	aneamia	boolean	0 (no) or 1 (yes)	decrease of red blood cells or hemoglobin;
HBP	high blood pressure	boolean	0 (no) or 1 (yes)	if the patient has hypertension;
CPK	creatinine phosphokinase	mcg/l (int)	level of the CPK enzyme in the blood;	
diabetes	diabetes	boolean	0 (no) or 1 (yes)	if the patient has diabetes;
ejection fraction	ejection fraction	int	percentage	percentage of blood leaving the heart at each contraction;
platelets	platelets	double	kiloplatelets/mL	platelets in the blood;
sex	sex	boolean	N/A	woman or man;
serum creatinine	serum creatinine	double	mg/dL	level of serum creatinine in the blood;
serum sodium	serum sodium	int	mEq/L	level of serum sodium in the blood;
smoking	smoking	boolean	0 (no) or 1 (yes)	if the patient smokes or not;
time	time	int	time in days	follow-up period;
death event	death event	boolean	0 (no) or 1 (yes)	if the patient deceased during the follow-up period;

Intro

The dataset analyzed for this research contains 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 consisted of 105 women and 194 men, and their ages range between 40 and 95 years old. All patients had left ventricular systolic dysfunction which puts them on a higher risk of death, this needs to be kept in mind when looking at the data.

Heart failure is quite common, and thus affects a lot of people each year. The condition is the leading cause of hospitalization in people over age 65. One solution might be to prevent heart failure from happening by examining large datasets full of data which are known to relate (closely) to heart disease and/or heart failure. This research is trying to replicate this solution with a supplied dataset.

The question this research is aiming to give an answer to is: "Can a death event be predicted when blood serum and age data is given using machine learning techniques?"

Cleaning data

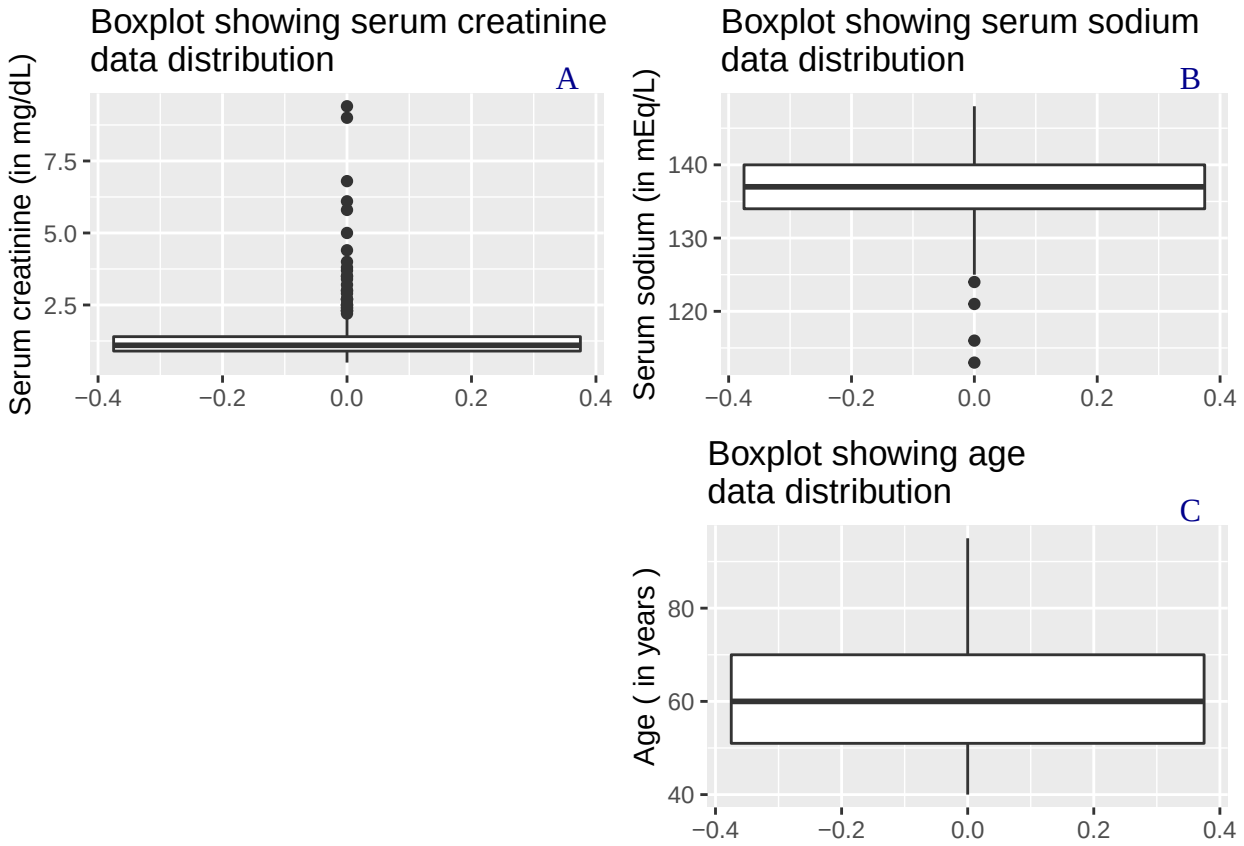
Before the data can be used by a Machine Learning algorithm, the data needs to be cleaned:

The datatypes, readability and reproducibility are very important in an EDA. This is why the function 'mutate' comes in handy. The 0,1 structure will be replaced by "False/True" to make the readability better and also increase the reproducibility. Finally, tibble will be used to give the 'head' of the data is table. As seen in the table below, the data has labels and no missing values at all.

```
## # A tibble: 299 x 13
##      age anaemia creatinine_phosphokinase diabetes ejection_fracti~ high_blood_pres~
##    <int> <fct>                      <int> <fct>                <int> <fct>
##  1    75 False                        582 False                   20 True
##  2    55 False                       7861 False                   38 False
##  3    65 False                        146 False                   20 False
##  4    50 True                         111 False                   20 False
##  5    65 True                         160 True                    20 False
##  6    90 True                          47 False                   40 True
##  7    75 True                         246 False                   15 False
##  8    60 True                         315 True                    60 False
##  9    65 False                        157 False                   65 False
## 10    80 True                         123 False                   35 True
## # ... with 289 more rows, and 7 more variables: platelets <dbl>,
## #   serum_creatinine <dbl>, serum_sodium <int>, sex <fct>, smoking <fct>,
## #   time <int>, DEATH_EVENT <fct>
```

Viewing the data

It is very important to check if the data contains major outliers or maybe even typos. This could be dramatic to the results/conclusions of the research. By creating boxplots, outliers can be visible outside the 'box' as dots:

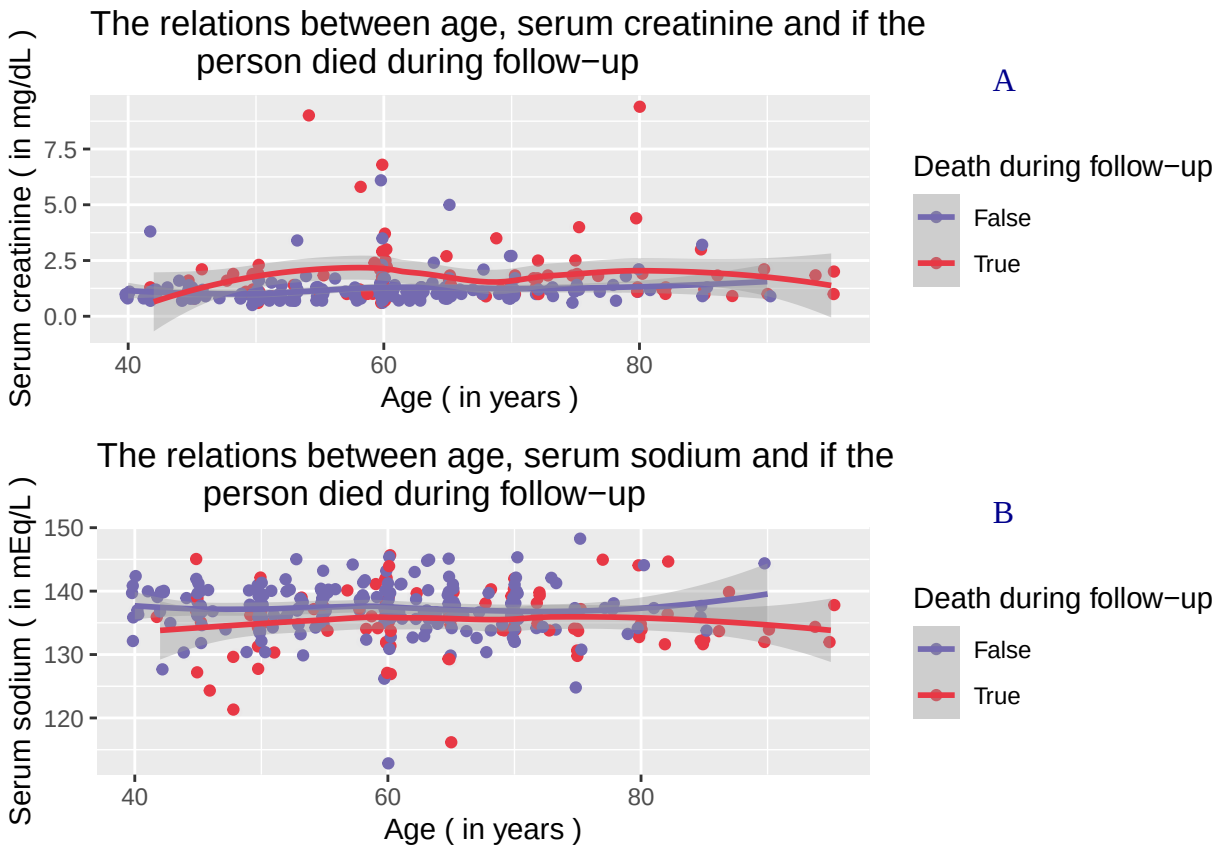


As seen in figure A above, the data contains many outliers. A normal serum creatinine value ranges between 0.59 and 1.35 mg/dL while the data contains values which are close to 10. Taken into account that the patients who have taken part in this research have underlying heart/renal conditions, these values are acceptable. This trend is also visible in figure B, while the range of the outliers is much smaller than figure A. The normal serum sodium level ranges between 135 and 145 milliequivalents per liter (mEq/L). As seen in figure B some outliers have much lower sodium values, this could be quite harmful to the heart because decreased sodium can cause muscle dysfunction and ultimately heart failure. The age boxplot seen in figure C shows normally distributed data. This can be supported by the fact that people get examined more as they get older because key body functions, like muscle- and renal function, will on average deteriorate over time.

Relationship between attributes

Dotplot

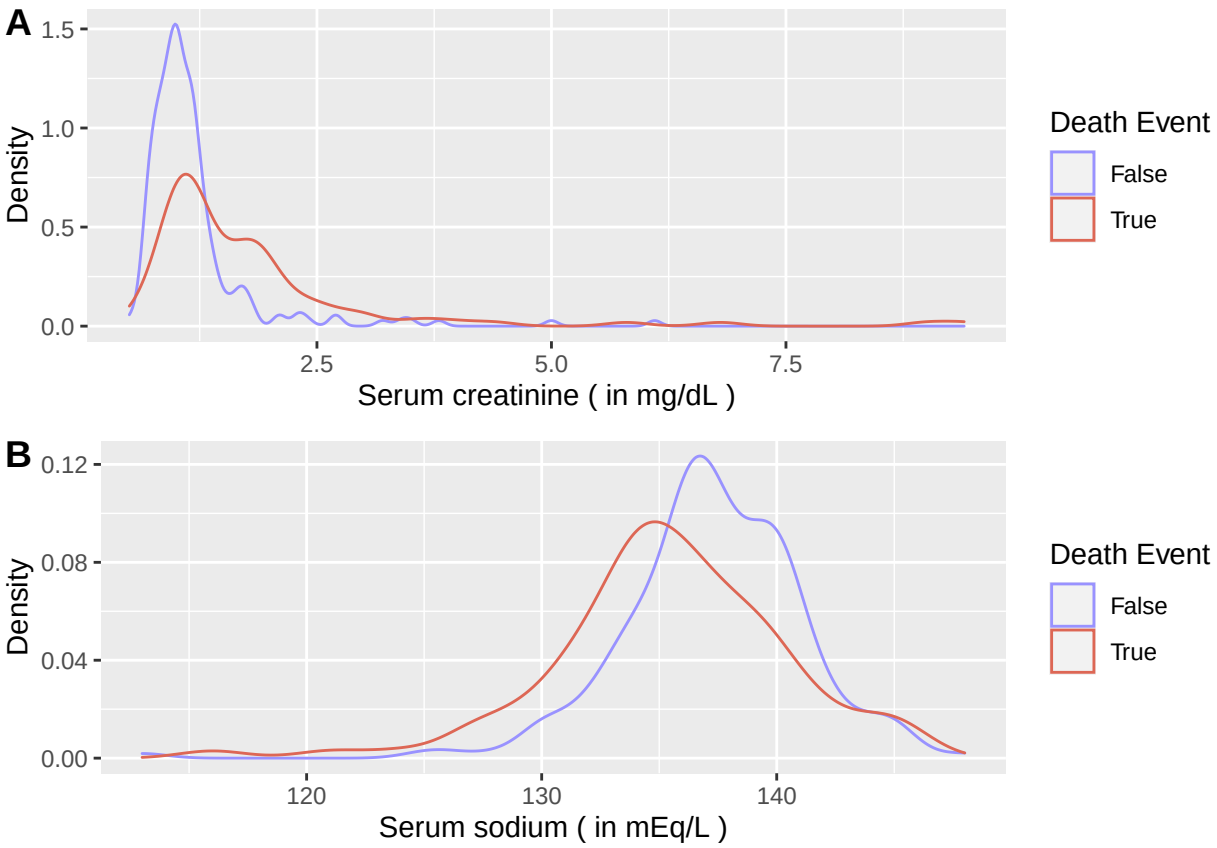
It is important to get to know what kind of relationship there is between the variables. Let's first 'cut out' the part of the data which will be used for the research. The serum and age data will be used, which are from columns 1,8 and 9 because the age, serum creatinine and serum sodium values have a great impact on heart disease and renal dysfunction, as seen in "Survival analysis of heart failure patients: a case study". Below the relations are plotted in graphs:



As mentioned before, a higher serum creatinine count or a lower serum sodium count than normal can result in heart failure. The literature found was correct as seen in the plot above. Figure A shows an increased death chance if the creatinine value is higher than 0.125 mg/dL and figure B shows an increased death chance if the sodium drops below 135 mEq/L.

Density

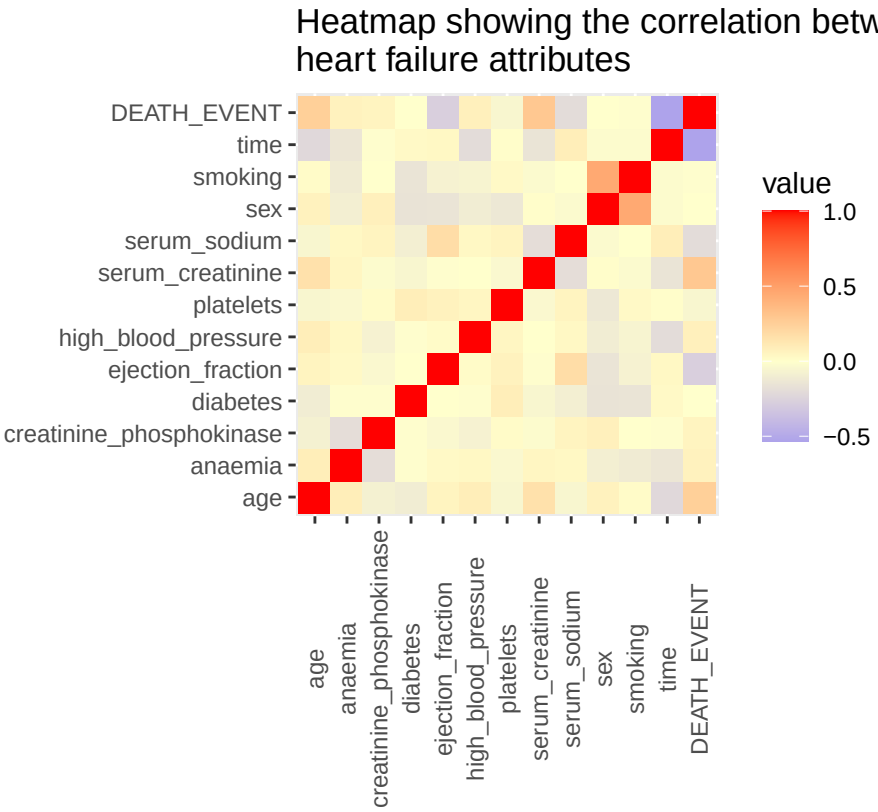
A density plot can be used to help display where values are concentrated over the interval. For this instance the interval is the amount of creatinine or sodium in the blood serum. To illustrate which concentration seems fatal, two density plots have been created, and colored based on the outcome of the follow-up period:



Just like the dotplot before, the density plot shows that a lower serum sodium value seems more fatal, as is a slightly higher serum creatinine value. The difference in frequency of figure B does seem more convincing than figure A.

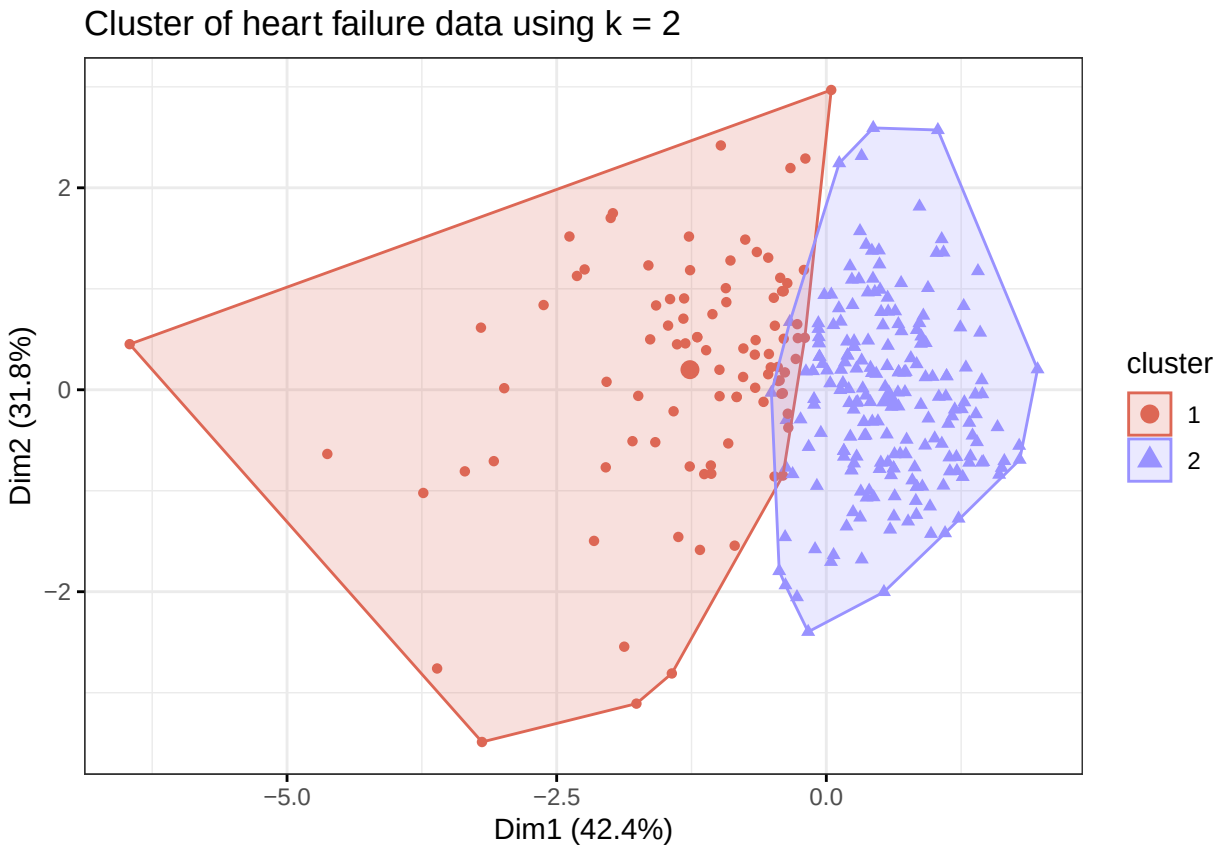
Correlation

The correlation is an extremely important factor in machine learning because ML algorithms assume that all attributes are independent. When looking at the heatmap below, 3 attributes stand out. At first, the follow-up time has a very negative correlation to the class attribute. The serum sodium has a somewhat negative correlation while the serum creatinine shows that there is a positive correlation between it and death event. So concluded, both values have differing correlation with the death event attribute which means they are independent.



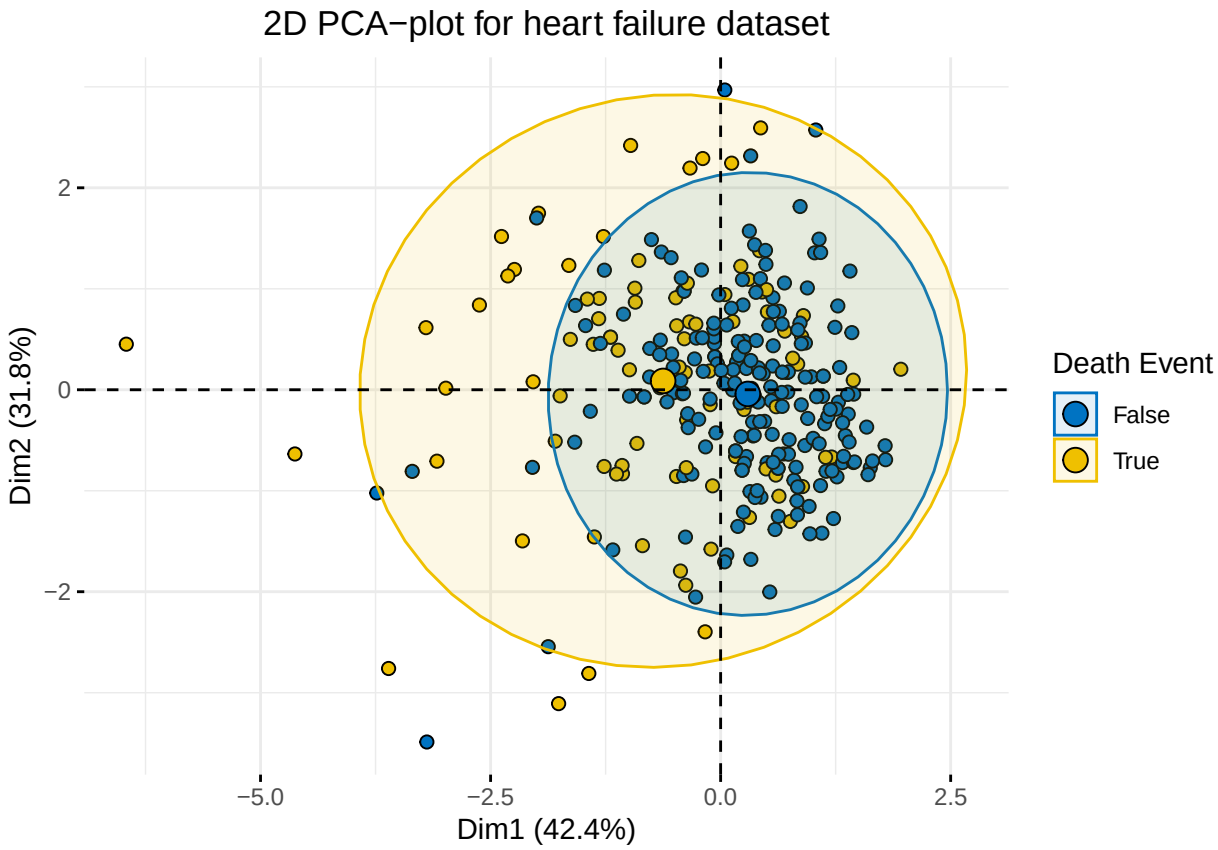
A heatmap visualizing the correlation between all heart value attributes. The red in the diagonal can be ignored because it represents a correlation of one variable instead of two. When the color of a tile is close to white there is no to near minimal correlation between attributes. If the tile is blue colored, there is a negative correlation present, while a red color indicates a positive correlation.

Next we will perform clustering: Clustering can be a great tool to give a better understanding of the data by discovering patterns. Below, a k-means cluster of 2 will be performed. A k of 2 has been chosen because the death event attribute has 2^1 possibilities.



In the plot above there can clearly be seen that two independent clusters have been formed with the given data. The most right cluster in blue represents the patients who survived while the red cluster shows the patients which died during follow-up. There is however a small section visible where the two clusters overlay, which means that it is not certain what the result will be for patients inside that range.

Next a PCA cluster plot will be created. It is quite similar to the k-means cluster, so the result will most likely be the same. It shows clusters of samples based on their similarity, which gives a good indication for independent values. PCA is mostly used to decrease the amount of dimensions of dataset, but in this instance that is not the reason.



The PCA-plot seems to give the same result as the k-means cluster. A 'death' cluster seems to emerge when data points go to the left. There is however overlap between the false and true clusters, so the probability to predict a correct death event might not be very high.

First, the factored data will be saved to use in the program called weka, which will be used to perform all sorts of algorithms on the data.

```
used_data <- factored_data[,c(1,8,9,13)]
write.csv(used_data, "data/data.csv", row.names = FALSE)
```

For every machine learning research, there should be a fine balance between sensitivity and specificity. When evaluating ML algorithm performance, accuracy is the default quality metric. For this occasion, the speed of the algorithm is not that important. But if the dataset contains all the records of patients in a 10 year period, speed is a big factor. A small group of patients was examined for this research, so it might be a good idea to let the algorithm learn overtime when new data is added. This is because 'only' 299 patients does not always give the big picture. The quality metric accuracy might be the most important for this research. It is very important to get an accurate result from a patient because it could predict someone's death chance. For this reason the most accurate yet somewhat speedy algorithm will be chosen.

Performance of ML algorithms

In order to choose which ML algorithm is best, it is a good exercise to look at the results of each algorithm. To start, Zero R and One R will be used to measure baseline performance. Although they do not seem important, they can be used to check if you are over fitting with other algorithms.

Zero R only chooses which occurs the most of the death event class. Most of the people did not die during follow-up, so Zero R chooses false. The percentage of correct guesses was only 67.893% so improvements can be made by more complex algorithms. The confusion matrix contains no true values, because false was chosen by Zero R, and 96 false positives out of 299 instances. One R, on the other hand, did a much better job at predicting a death event. This is possibly because One R takes the best attribute to predict a death event with. 75.5853% of instances were predicted correctly, so this might be one of the better algorithms. But let's first look at the other ones. All confusion matrix values are visible in the table below.

Next up, the decision trees. Because a big part of the data contains numeric values, J48 will be used, which can deal with such values. When creating a tree with a small bucket size, the tree becomes quite overwhelming. To fix this issue, a bucket size of 10 will be used which gives a much smaller tree, which also gives quite an accurate percentage: 77.9264%.

Nearest neighbor seems like a good algorithm because all of the data points close to each other will 'cluster' towards one of the boolean values, true or false. When using k = 1, 100% of the values seem correct. But since it only uses the training data, it might not be useful in the long run.

Naive Bayes makes a percentage based on all attributes. Some attributes might not be very good at predicting a death event, so this algorithm might not give a good prediction. When run, 72.5753% of instances were predicted correctly. This is smaller than of One R so taking all attributes is not the correct decision.

There is no clear linear line visible in the data, so simple logistic might not be a good choice. When run however, 73.2442% of instances were correctly predicted. The only down side is that 69 people got a false positive result, which is a pretty big chunk of patients. This situation should never happen in hospitals so this algorithm also does not function per request.

```
codebook <- read.delim("data/algorithm_data.csv", sep = ",")
kable(codebook, caption = "Table 2a : Algorithm Data")
```

Table 2: Table 2a : Algorithm Data

Algorithm	Accuracy...	Speed.sec.	True.Positive	False.Positive	True.Negative	False.Negative
ZeroR	67.89	0.00	0	0	96	203
OneR	75.59	0.00	30	7	66	196
J48	77.93	0.01	40	10	56	193
IKB	99.67	0.03	96	1	0	202
Naïve Bayes	72.58	0.01	30	16	66	187
Simple Logistic	0.14	73.24	27	11	69	192
SMO	68.56	0.00	4	2	92	201
Random Forest	99.66	0.01	96	1	0	202

```
codebook <- read.delim("data/algorithm_data_validated.csv", sep = ",")
kable(codebook, caption = "Table 2b : Validated Algorithm Data")
```

Table 3: Table 2b : Validated Algorithm Data

Algorithm	Accuracy..	Speed.sec	True.Positive	False.Positive	True.Negative	False.Negative
ZeroR	67.83	0.00	0	0	96	203
OneR	73.57	0.01	35	18	61	185
J48	72.57	0.02	37	23	59	180

Algorithm	Accuracy..	Speed.sec	True.Positive	False.Positive	True.Negative	False.Negative
IBK(k=3)	67.22	0.00	40	42	56	161
Naive Bayes	71.90	0.01	29	17	67	186
Simple Logistic	72.90	0.17	28	13	68	190
SMO	68.56	0.03	4	2	92	201
Random Forest	73.24	0.15	51	35	45	168

Ensemble Learning

Next up, meta learners will be used to further validate the model, and also run statistical tests. There are multiple meta learners available, but for this research, 3 of them will be used: Stacking, Bagging and Boosting. With stacking, multiple algorithms will all perform a task, whichever algorithm does best is chosen. Bagging can help to reduce variation, and prevent overfitting. It uses a bootstrapping method which samples with replacement, so the accuracy might be pretty high. Boosting works on a 'weighing' system. Each time it does a iteration, all mistakes are weight more, so those are filtered out for the next iteration. This process continues for the max iterations provided. Below, a table can be seen which represents the data given by these meta learners. For each learner, zeroR, OneR and J48 are used as algorithms with a minimal bag size of 10. For the statistical t-test, a p-value of 0.05 is chosen, mostly because it is the default p-value. The standard deviation is also visible in the table to illustrate what the range of deviation is between all iterations.

```
codebook <- read.delim("data/meta-learner_data.csv", sep = ",")
knitr::kable(codebook, escape = TRUE)
```

Algorithm	Accuracy	SD
ZeroR	67.90%	1.53
OneR	73.41%	6.35
J48	73.06%	6.55
IKB	67.84%	7.29
Naive Bayes	71.90%	6.03
Simple Logistic	72.55%	5.68
SMO	68.67%	2.83
Random Forest	72.20%	7.12

Weeks 5 till 7;

Creating a Java application which takes known heart data and creates a J48 model based on the model chosen for this research. When a batch file of unknown instances, or a single instance is fed to the file, it will classify all of the instances. So true or false (dead or alive). The wrapper can be found with the following link: https://github.com/JurrienDeJong/JavaWrapper/tree/main/src/main/java/nl/bioinf/wrapper