Liver disease prediction

Moschos Evangelos-Jason

Abstract

In this document, we are presenting our analysis on a liver disease classifier, based on the indian liver disease dataset found on Kaggle. The project is conducted for the Harvard Data science professional program.

Contents

Executive summary

The prevalence of liver diseases has increased tremendously the past few years, due to alcohol abuse, illegal substances, poisoned or contamined food etc. It is often characterized as a burden on both health sciences and economies and has a moderate mortality rate, amongst chronic diseases.

As part of the Harvard data science program, we will be attempting to create a classifier for detecting liver disease, based on a dataset for Indian patients. The dataset we are using is found at https://www.kaggle.com/uciml/indian-liver-patient-records. The aim of the project is to be able to predict (based on some characteristics) the existance of a liver disease in people. These characteristics include some general demographic characteristics and some blood-test results. Our dataset is slightly imbalanced, consisting of 72% liver patients and 28% healthy individuals.

In our exploratory analysis, we firstly examined the 10 variables (Age, Gender, Total Bilirubin, Direct Billirubin, Alkaline Phospotase, Alamine Aminotransferase, Aspartate Aminotransfease, Total Proteins, Albumin and Albumin to Globulin ratio) seperately. Our results indicate that age and gender do not appear to have substantially different distributions on the prevalence of liver disease, although women were slightly less prone to it. This statement is also in-line with the literature online, which states that liver disease occur throughout the world, irregardless of sex, region and age.

For the other 8 variables, due to lack of domain knowledge, we examined the distribution of each variable for both groups (healthy and sick) seperately, as well as the correlation between the variables. We identified that while extreme values in test results are correlated with the existence of the disease, it is hard to distinguish between a healthy and a sick individual for values within the normal range. As several variables were highly correlated with each other, we decided to remove the Direct Bilirubin, the Alamine aminotranferase and the Albumin and use all the remaining variables as features.

Additionally, we created a new variable, indicating the total number of extreme values (i.e. blood test results outside the normal range, defined solely on the train set). Finally, we also decided to preprocess our dataset by scaling and centering all the variables.

Based on our EDA, we have developed nine models that consider these variables. The best performing model we developed was a Random forest algorithm that achieved an:

- **Accuracy of 81.03% in the test-set**

To evaluate the feature engineering, we have re-trained a random forest model (10th model), on the unprocessed dataset; the accuracy of the second Random forest algorithm was **75.86%**, and thus the features selected were beneficial to the overall modelling process.

The analysis shows significant improvement over simpler methods (13% improvement), but several other factors can be incorporated to further improve the accuracy of the model.

1 Introduction

As part of the Harvard data science professional program, we will attempt to create a classifier for detecting liver disease in individuals. In the next chapters, our analysis is structured as follows:

- Chapter 2: Dataset loading/generation
- Chapter 3: Exploratory Data Analysis
- Chapter 4: Feature selection
- Chapter 5: Analysis: Model building & Testing
- Chapter 6: Conclusion

In chapter 2, the generation of the dataset is explained, and some basic dimensions of the data are explored.

In chapter 3, an Exploratory data analysis is conducted, to identify interesting features in the dataset, and help us select the appropriate variables in our models.

In chapter 4, the features of the models are selected based on the EDA. Some pre-processing is also performed.

In chapter 5, the basic models are built, based on the characteristics identified in chapter 4. All the models are also evaluated.

In chapter 6, an overview of the results is presented, limitations of the project are discussed, and direction for future research is provided.

2 Dataset generation/loading

In this chapter, we will describe the dataset generation process, some useful (general) statistics for the dataset and some first insights we have.

The entire dataset is found at https://www.kaggle.com/uciml/indian-liver-patient-records, but it is automatically downloaded in our code through https://github.com/jmoschos/Liver_disease/blob/master/indian_liver_patient.csv.

2.1 Dataset loading and basic manipulations

We begin by examining the different variables:

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

There are 11 variables:

- Age:
- Gender:
- Total Bilirubin
- Direct Bilirubin
- Alkaline Phosphotase
- Alamine Aminotransferase
- Aspartate Aminotransferase
- Total Proteins
- Albumin
- Albumin-to-Globulin ratio
- Dataset: Target Variable

Based on the data, we see that the majority of the variables are continuous with the exception of the target variable and the gender variable. A summary of the variable types is also illustrated in the following figure:

```
'data.frame':   583 obs. of  11 variables:
 $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
 $ Gender                    : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 2 1 1 2 2 ...
 $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...
 $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...
 $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202 290 ...
 $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
 $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
 $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...
 $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...
 $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...
 $ Dataset                   : int  1 1 1 1 1 1 1 1 2 1 ...
```

Additionally, based on the documentation, we see that the target variable is 1 if the person is sick and 2 if he is healthy. We are going to change the convention to 1 if the person is sick and 0 if he is healthy. For the gender variable, we are also going to change it to a binary variable with 0 representing male and 1 female. Finally, we are going to convert these 2 variables into factors, as a preparation for our machine learning models.

```
[1] "Number of healthy individuals:  167"
```

```
[1] "Number of sick individuals:  416"
```

We are also going to check for missing values in our dataset:

	x
Age	0
Gender	0
Total_Bilirubin	0
Direct_Bilirubin	0
Alkaline_Phosphotase	0
Alamine_Aminotransferase	0
Aspartate_Aminotransferase	0
Total_Protiens	0
Albumin	0
Albumin_and_Globulin_Ratio	4
y	0

Based on the previous table, only one variable has (4) missing values. Due to the low number of missing values, we could consider removing these records, but in our implementation, we have chosen to replace these values with the average of all other values for that variable.

2.2 Dataset generation

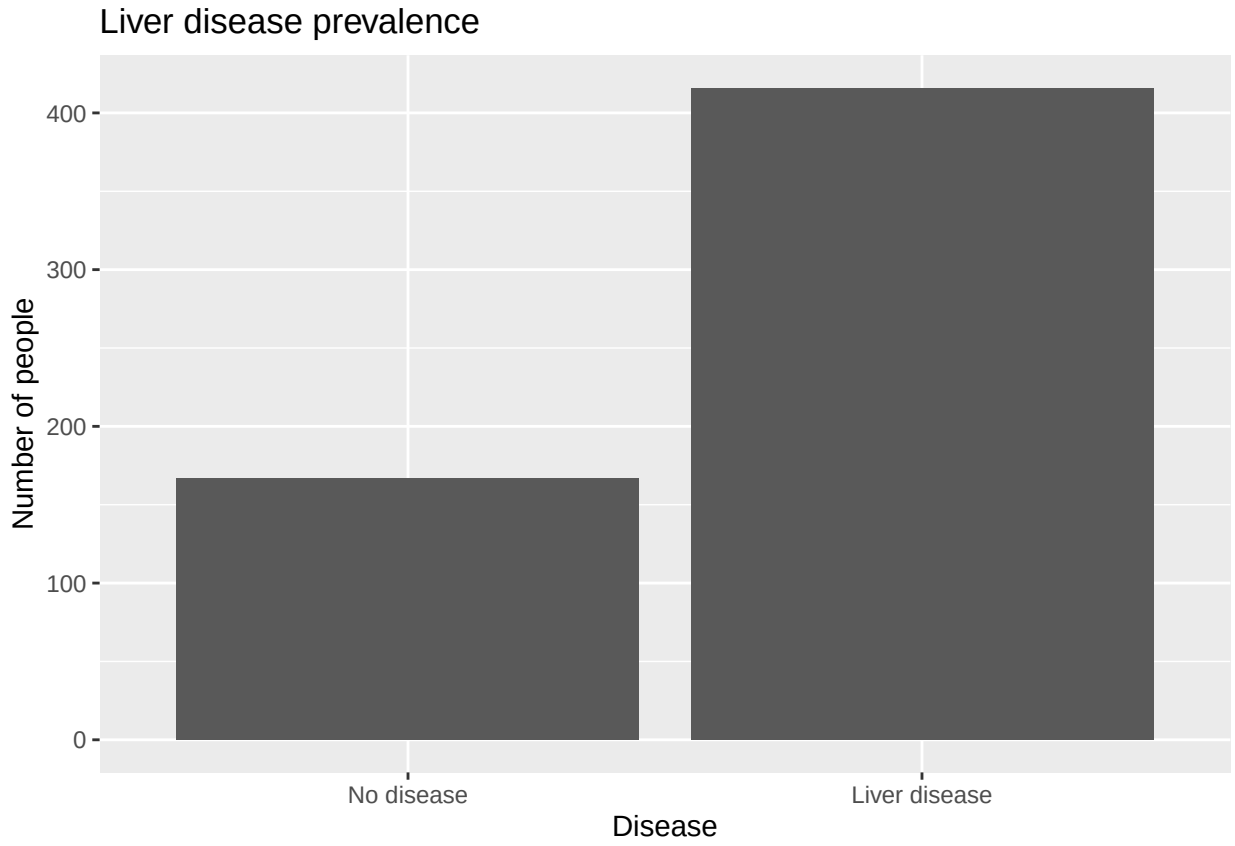
For model building purposes, we are going to split the initial dataset into a train set and a test set. We have decided to use a 80/20 split rule for these sets. For the sake of reproducability, we have added a seed to both the set generation and all subsequent algorithms, where randomness plays a role.

3 Exploratory Data analysis

In this section, we are presenting our exploratory data analysis (EDA). From the variables in our dataset, described in the previous section, we will examine all of them seperately and then calculate the correlation between them.

3.1 Target Variable: y

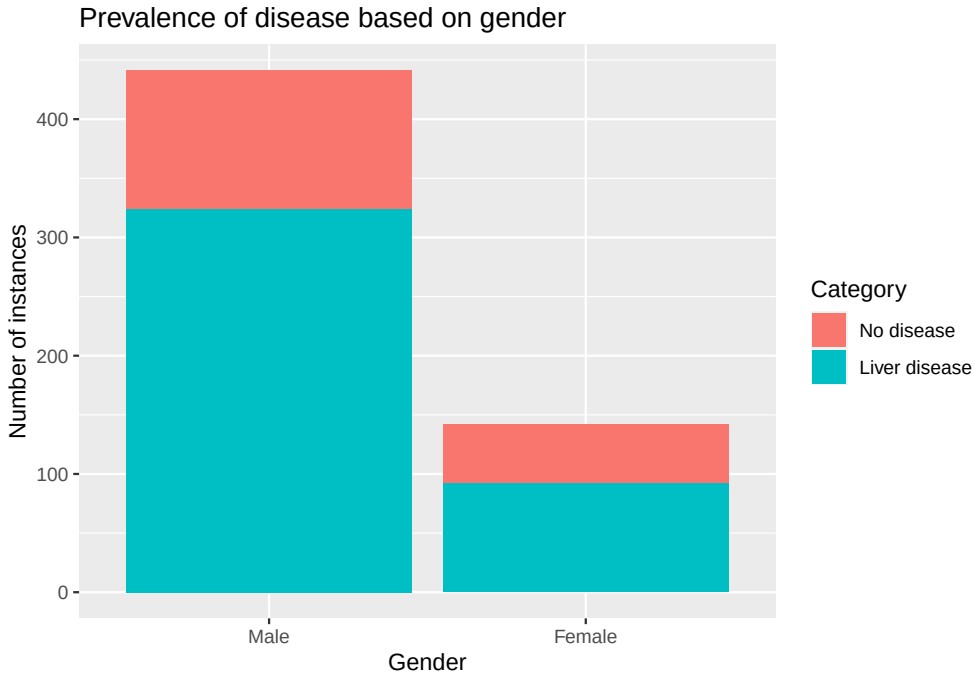
We will start by examining the distribution of healthy vs. sick individuals in our dataset.



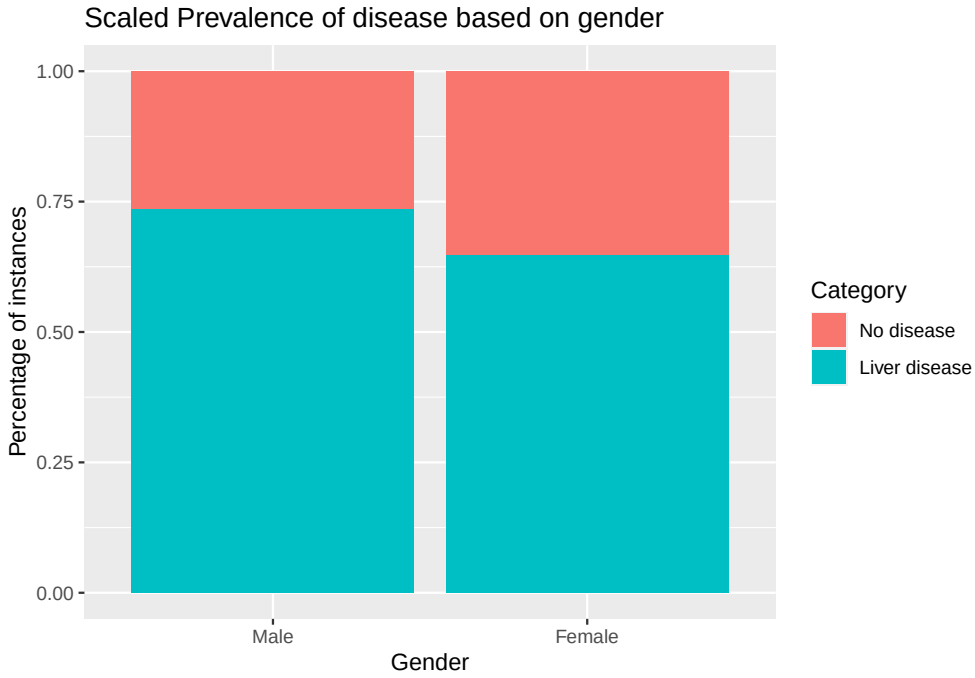
It appears that our dataset is imbalanced with only 28% healthy individuals and 72% sick.

3.2 Gender analysis

Literature suggests that the disease appears in both genders . To examine this statement, we are going to plot the number of instances of healthy vs. unhealthy individuals for two genders.



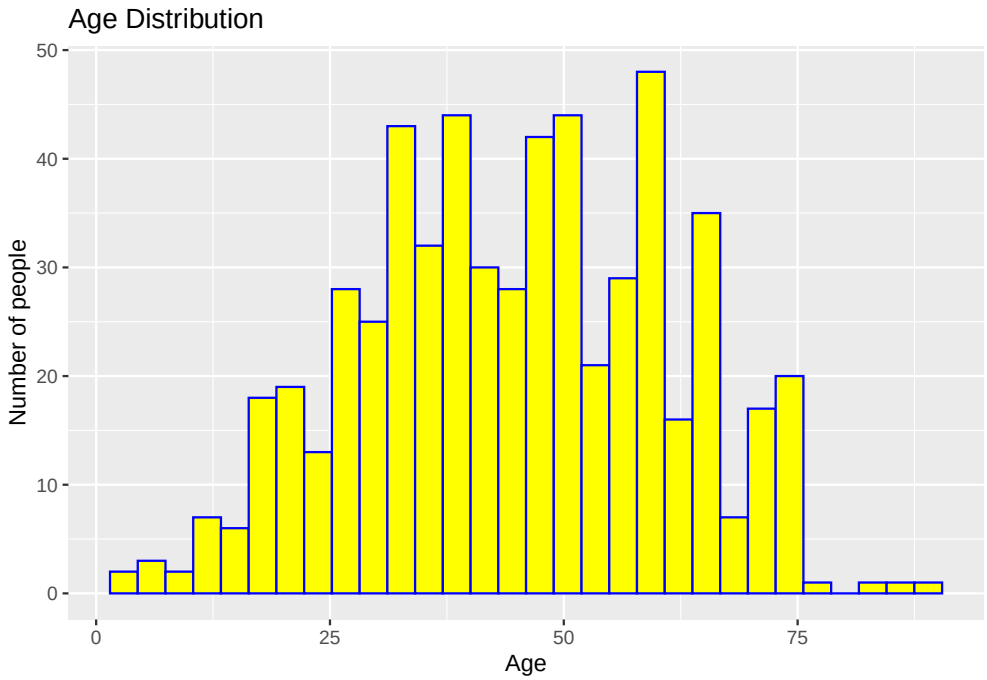
Based on the the plot, we can see that the dataset contains mostly male data, but its hard to compare the prevalence of the disease between the two genders. Therefore, we are going to scale our data to be able to do the comparison.



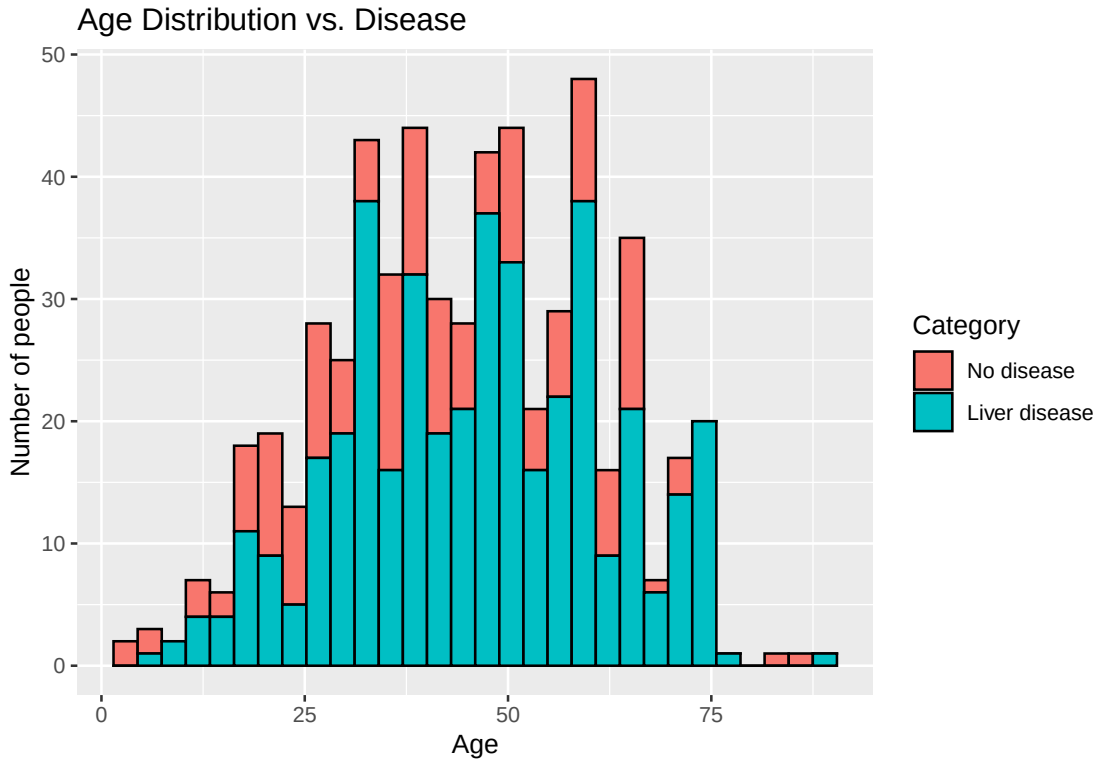
The graph indicates that both genders behave similarly, although females have a slightly lower rate of disease.

3.3 Age analysis

Continuing our EDA, we are going to explore the age distribution across our dataset:



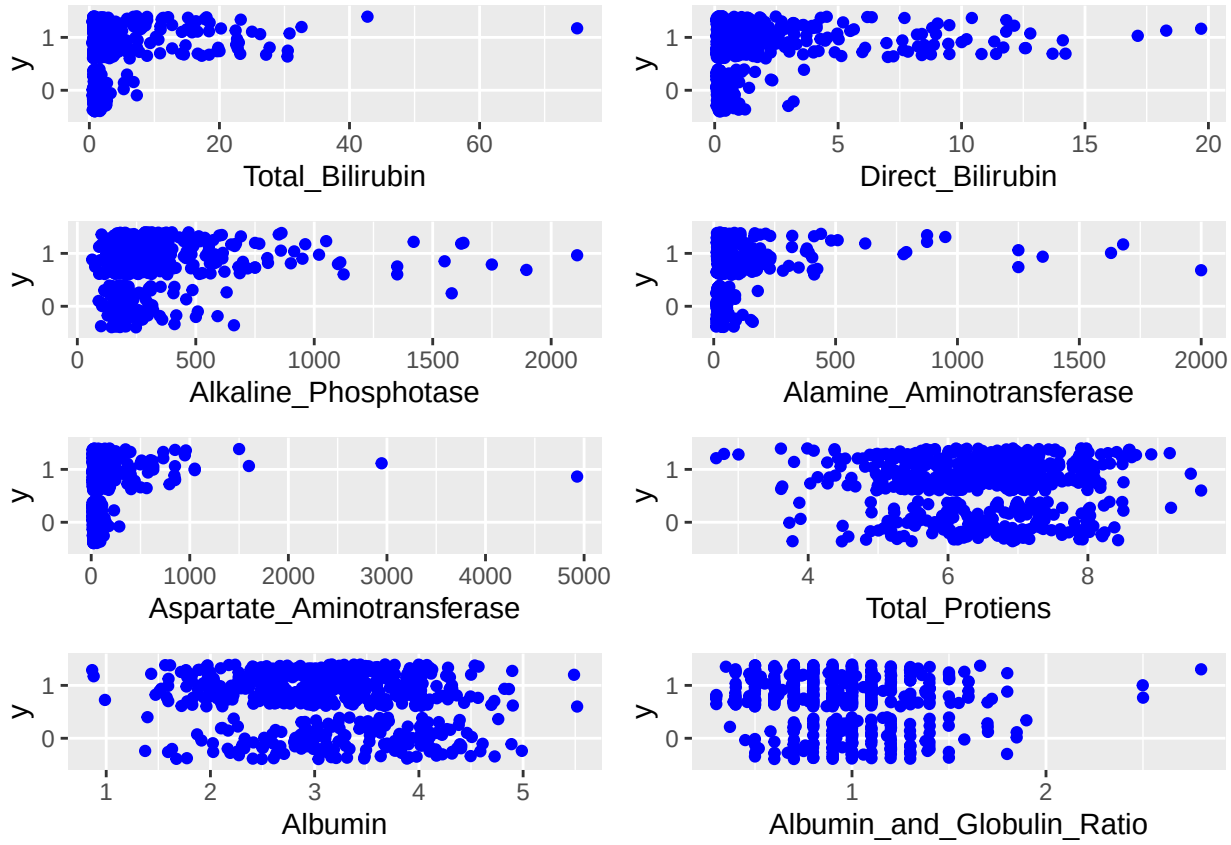
The graph shows a wide distribution, across nearly all ages that is centered around the middle. We are now going to plot the histogram of ages but also indicate the target variable y.



It appears that both healthy and sick individuals follow similar age distribution.

3.4 Blood test results

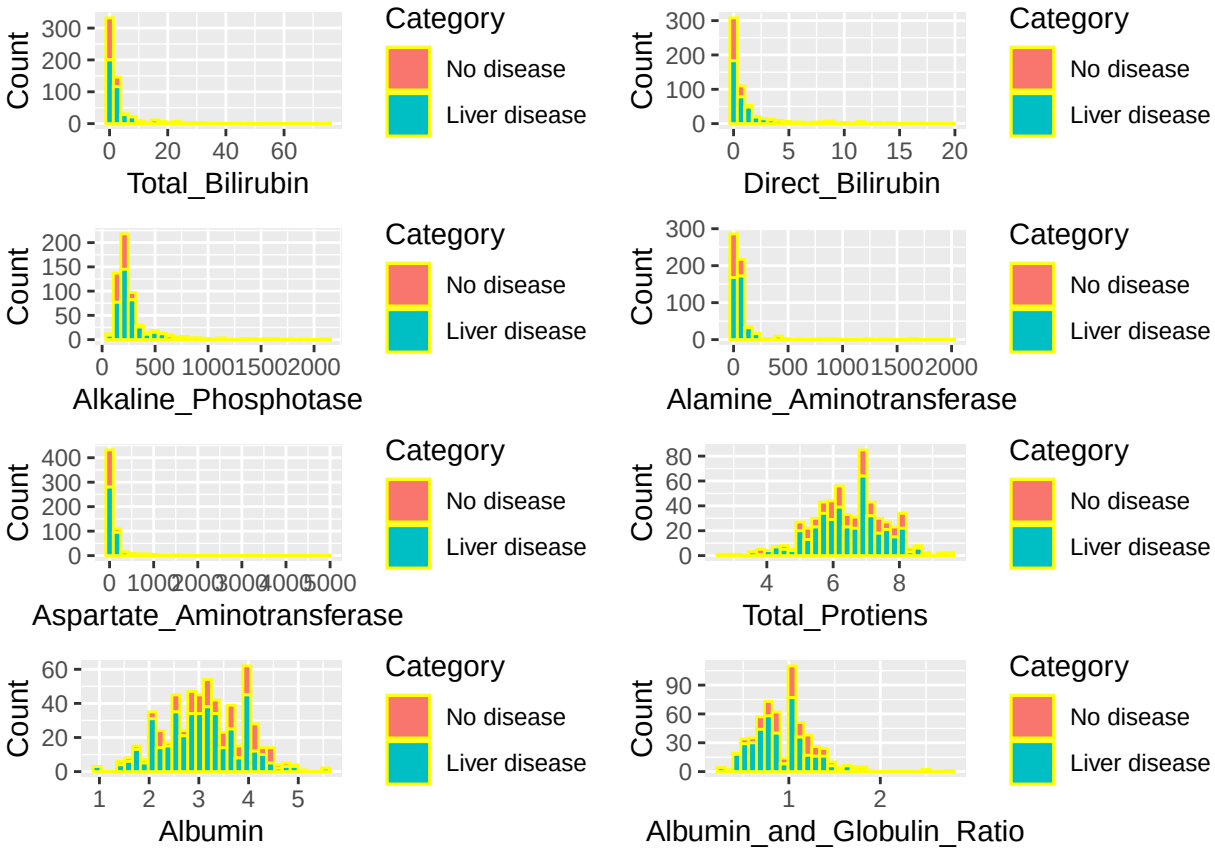
Due to lack of domain knowledge, exploring the remaining 8 variables is not a straightforward process. We are going to start by plotting scatter plots of the individual variables, but also add jitter in our plots, in order for the individual dots to be seperated.



From these 8 plots, we can see the following:

- **Total Bilirubin and Direct Bilirubin have similar behaviour**: For healthy invididuals, both metrics have small values, and for sick individuals we have a wide range of values (including the range of healthy people).
- **Alkaline phosphotase has a similar behaviour to the Bilirubins but to a smaller extent**: While for the bilirubins we can see up to 20 times higher values, in Phospotase we see up to 4 times higher values in liver patients.
- **Total Proteins, Albumin and Albumin-to-Globulin ratio have similar graphs**, where both healthy and sick people are centered around a value but the range for sick people is (slightly) larger.
- **The two aminotransferases also have the same behaviour as the bilirubins and the alkaline** but with fewer outliers.

To continue the exploration we are going to plot the histograms for the same variables:

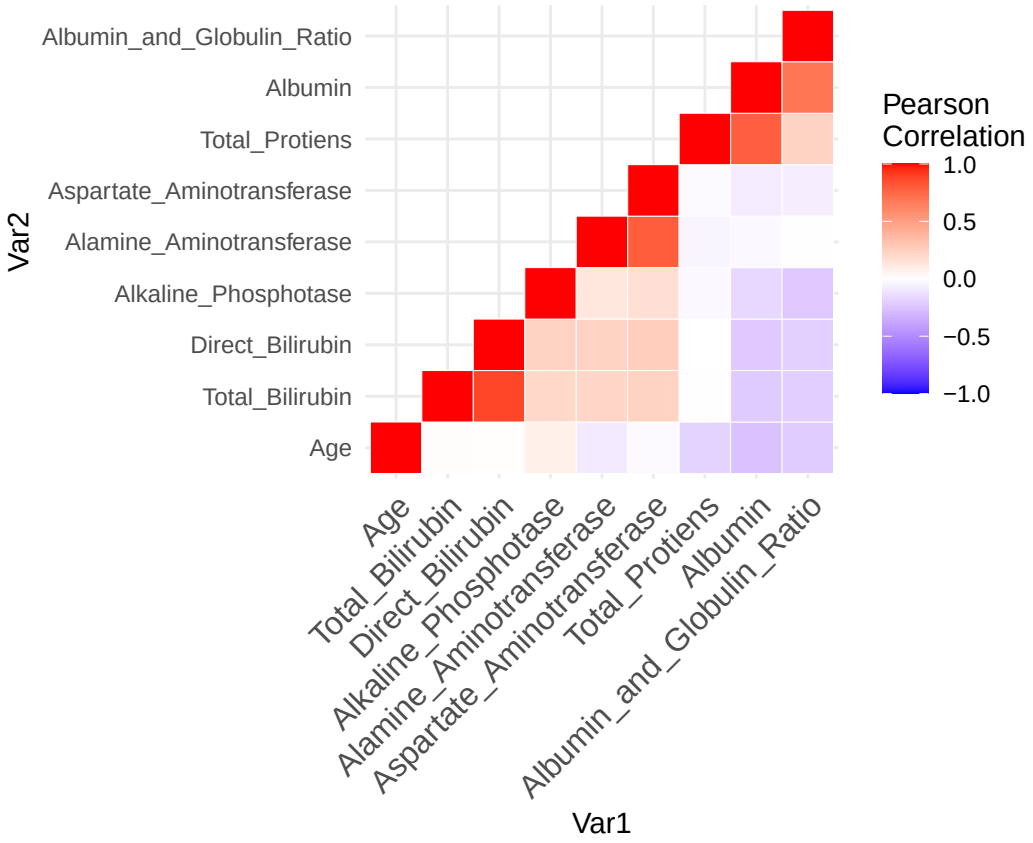


From the histograms, we observe that:

- Albumin, Albumin-to-globulin ratio and Total proteins follow similar distributions.
- The other 5 variables have very similar distributions. Alkaline Phosphotase is slightly different than the rest, as it has very few observations at 0.
- In all the variables, it seems like having a liver disease has a bigger range of values, compared to not having the disease and the range of values for sick individuals includes the range of values for healthy individuals.
- There are patients with "normal" indicators that have the disease.

3.5 Correlation between variables

Based on the histograms and scatterplots, it appears that some of the variables have similar characteristics. To verify that, we are going to calculate the correlation matrix for all continuous variables (and exclude the gender).



Similar to our observations:

- Albumin is highly correlated with both Albumin-to-globulin ration and Total proteins.
- Direct Bilirubin is highly correlated with Total Bilirubin.
- Alamine aminotransferace is highly correlated with Aspartate aminotransferace.

4 Feature selection

Based on our EDA, we are now going to select the features that will be used in our models.

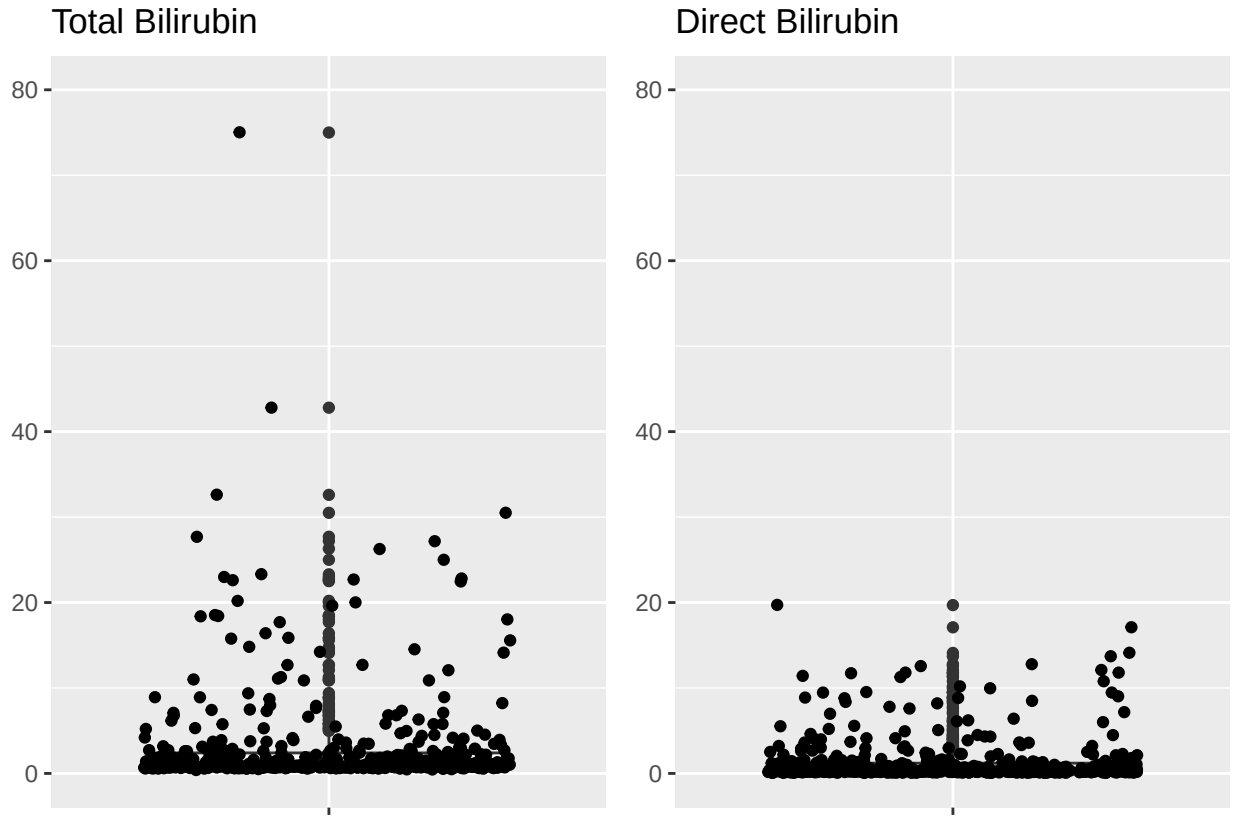
To Test the effectiveness of the steps performed in this section, we are going to save the original train and test datasets and re-train the best performing model on those. Our comparison will be between:

- The best performing model with the feature-engineered dataset [case 1]
- The same model as case 1, but with the original (no changes made) dataset [case 2]

4.1 Variable removal

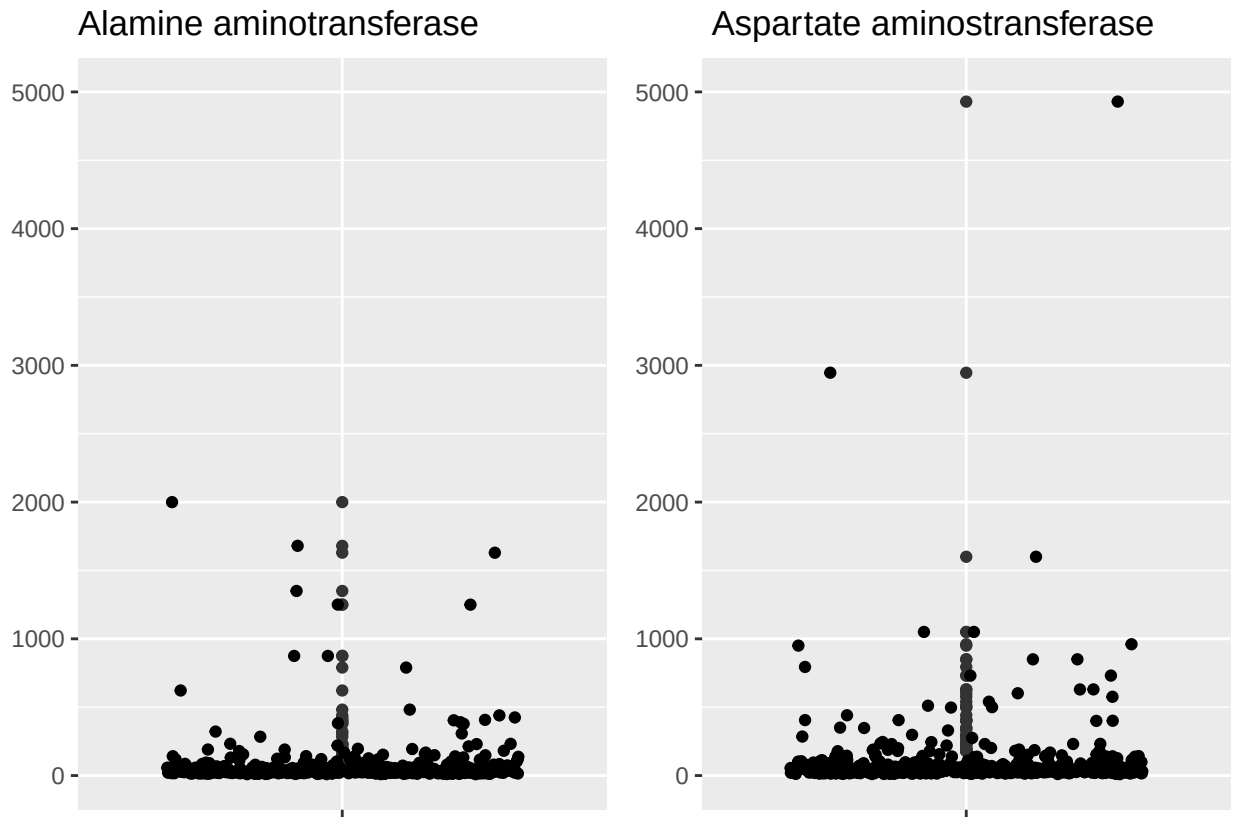
In the correlation matrix, we observed that some variables are highly correlated with each other. We are going to remove some of the variables that convey the same type of information.

Firstly, we are going to look at Bilirubin (Total vs. Direct).



Based on this graph, Total Bilirubin has a higher variation, and thus we are going to remove Direct Bilirubin from our dataset.On a sidenote, Total Bilirubin is the sum of Direct and Indirect bilirubin, which is why there is such a high correlation between the two.

We are now going to check Alamine aminotransferase and Aspartate aminotransferase:



Similarly to the bilirubins, we observe that Aspartate has a much higher variation, and therefore we are going to remove Alamine.

Finally, for albumin, since its highly correlated with both Albumin to Globulin ratio and Total proteins, we are going to remove it, to solve both problems at once.

4.2 New variable creation: Number of extremes

Besides the existing variables in our dataset, we believe it might be of interest to add a new variable: Number of extreme blood test result values. The motivation behind this choice is that most healthy patients will not have any extreme values in their blood test results, while sick individuals will have at least some indication of a liver disease.

To create this variable we will perform the following steps:

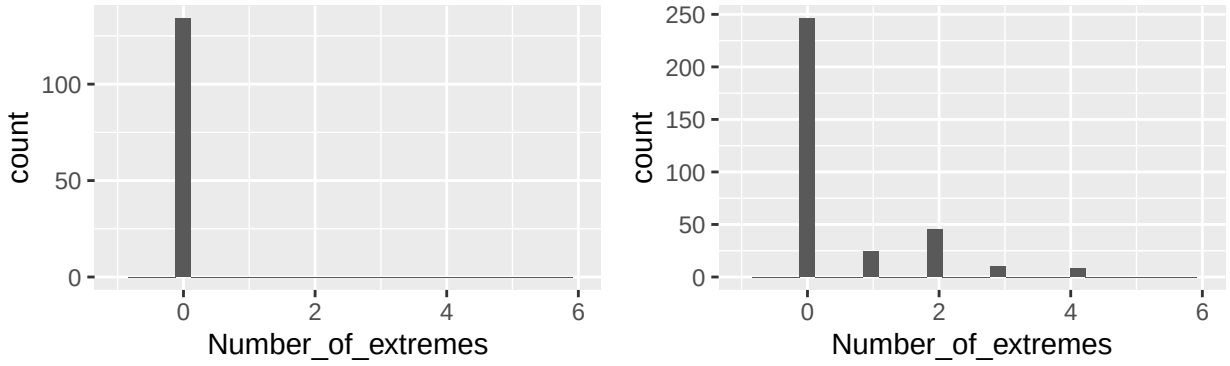
- First calculate the normal range of values on the **train set only** for the **healthy individuals**. In a real scenario we would not have the test set output (sick or healthy), and therefore we cannot base our analysis on that set.
- Create a dummy variable for each blood test result (8 variables)
- Fill the dummy variable with 1 if the blood test result of that individual is outside the limits calculated (below lower limit, or above upper limit).
- Sum the rows

The final vector contains for each individual the total number of blood test result values that is outside the "normal range" (which again is calculated **ONLY on the train set**). Our hyptothesis is that people with

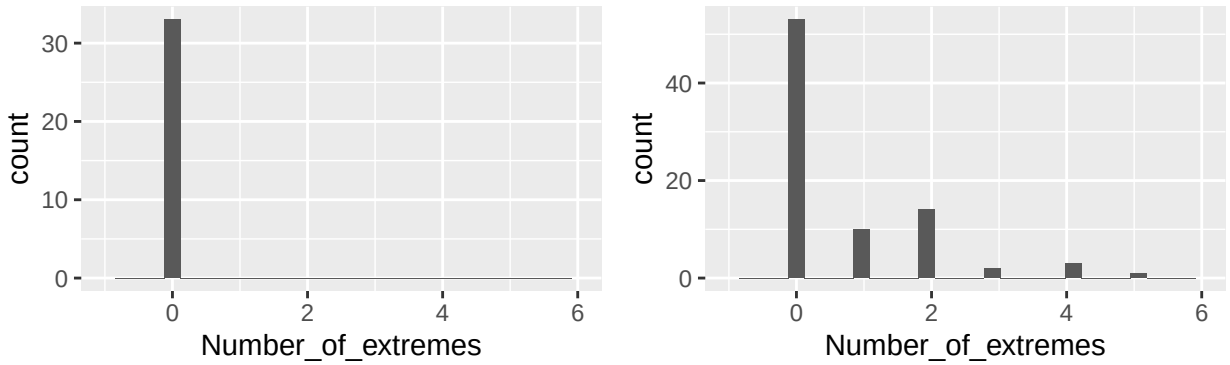
a large number of extreme values are sick, while people with lower number (or even 0) are more likely to be healthy.

The two histograms showing the distribution of Extreme values for healthy and sick individuals are depicted in the following figures:

For the train set:



For the test set:



As can be seen for the train set, we have, as expected, only 0 values for the healthy individuals, since the bounds we used were created from this set. On the other hand, for individuals with liver disease, we see a different distribution (but also a lot of 0 values i.e. all blood test values within what we defined as normal range).

On the test set, we observe similar results, as the train set, but we should underline the fact that there was a chance to observe a different distribution (not all 0s) for healthy individuals, as the values for the bounds were created based on the train set and were agnostic of any "extreme" values in the test set.

4.3 Features of our models

Based on our feature engineering, the final features for our models are:

- Age
- Gender
- Total Bilirubin
- Alkaline Phosphotase
- Aspartate Aminotransferase
- Total Proteins
- Albumin to Blobulin ratio
- Number of extremes

5 Analysis: Model building and testing

In this section, we are presenting the different models developed and their performance. We are also discussing the metric used for evaluation and the validation method for tuning.

5.1 Data pre-processing

Before implementing any ML algorithms, we are going to scale and center the data with the help of the **Preprocess** function on caret.

In the following table, a sample of our scaled data is presented:

Age	Gender	Total_Bilirubin	Alkaline_Phosphotase	Aspartate_Aminotransferase	Total_Protiens	Albumin_and_Globulin_Ratio	y	Number_of_extremes
1.2202607	1	-0.4020882	-0.4244791	-0.2923834	0.3125290	-0.1682740	1	-0.4313849
1.0403422	0	1.2061970	1.5984148	-0.0243186	0.9558901	-0.6623522	1	1.8578965
1.0403422	0	0.6385669	0.7726632	-0.1289293	0.4963465	-0.1991538	1	0.7132558
0.8004509	0	-0.3547857	-0.4442339	-0.2858452	0.3125290	0.1405250	1	-0.4313849
1.6400704	0	0.1024719	-0.3928714	-0.1583510	0.7720726	-1.7122685	1	-0.4313849

5.2 Metric for performance

Typically, in medical applications false negatives are far more important than false positives and therefore, metrics such as F1 score are utilized. In our implementation, we are going to train our models with the **F1 score** as the metric, but also track the overall **accuracy**.

Due to the fact that our dataset is imbalanced with the sick individuals being more than 70%, it is likely that some models might classify all people as sick and therefore the F1 score cannot be calculated.

5.3 Validation method and tuning

For tuning our algorithms, we have selected **10-fold cross-validation** for all methods. The final reported performance (accuracy, F1) is based on the test set, after selecting the optimal parameters (from cross validation on the train set).

5.4 Model 1: Naive model

For the first model, which we will use as benchmark, we are going to implement a naive-approach: classify every patient as having the disease. Since our dataset is imbalanced (favoring disease), our accuracy will be high (the f1 score is not defined in this case).

The accuracy of this method is:

```
 Accuracy
0.7155172
```

Finally, the confusion matrix is:

	0	1
0	0	0
1	33	83

5.5 Model 2: Classification tree

In the second model, we are implementing a classification tree and we are experimenting on the complexity parameter (values between 0 and 1 with 0.001 step).

The accuracy of this model is:

```
 Accuracy
0.7155172
```

The F1 score is:

```
[1] "Not defined"
```

The classification tree appears to produce the same results, as our naive method and is therefore not an improvement.

Finally, the confusion matrix is:

	0	1
0	0	0
1	33	83

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA

5.6 Model 3: Neural network

In this model, we are building a neural network.

The accuracy of this model is:

```
 Accuracy
0.7586207
```

The F1 score is:

```
[1] "0.263157894736842"
```

Finally, the confusion matrix is:

	0	1
0	5	0
1	28	83

The Neural network, similar to the CART correctly classifies all individuals with liver disease but also 5 individuals who do not have a liver disease and is therefore an improvement over both Naive method and CART.

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579

5.7 Model 4: Adaptive boosting (adaboost)

For our fourth model, we are implementing an adaptive boosting algorithm

The accuracy of this model is:

```
 Accuracy
0.7844828
```

The F1 score is:

```
[1] "0.489795918367347"
```

The adaboost algorithm has an impressive improvement in accuracy, but has also missclassified 4 patients. This can be seen in the confusion matrix:

	0	1
0	12	4
1	21	79

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959

5.8 Model 5: Gradient boosting

In this model, we are implementing a gradient boosting algorithm and tuning 3 out of the 4 parameters.

The accuracy of this model is:

```
 Accuracy
0.7241379
```

The F1 score is:

```
[1] "0.304347826086957"
```

Unfortunately, this algorithm is slightly better than the naive method and does not provide any further improvement.

The confusion matrix is shown in the following table:

	0	1
0	7	6
1	26	77

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959
GBM	0.7241379	0.3043478

5.9 Model 6: Support vector machine

For this model, we are implementing an SVM method with a Radial kernel.

The accuracy of this model is:

```
 Accuracy
0.7155172
```

The F1 score is:

```
[1] "Not defined"
```

The SVM algorithm produces exactly the same results as our naive method, and is therefore not an improvement.

The confusion matrix is shown in the following table:

	0	1
0	0	0
1	33	83

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959
GBM	0.7241379	0.3043478
SVM	0.7155172	NA

5.10 Model 7: K nearest neighbors

For the K-nearest neighbors, we will be experimenting with the value k (number of neighbors selected).

The accuracy of this model is:

```
 Accuracy
0.7155172
```

The F1 score is:

```
[1] "0.108108108108108"
```

The confusion matrix is shown in the following table:

	0	1
0	2	2
1	31	81

The KNN algorithm has the same accuracy as our naive method, but does so by classifying two healthy individuals correctly and missclassifying two sick individuals as healthy. Therefore, it is worse in terms of F1.

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959
GBM	0.7241379	0.3043478
SVM	0.7155172	NA
KNN	0.7155172	0.1081081

5.11 Model 8: Random Forest

For the random forest algorithm, we are tuning the mtry parameter.

The accuracy of this model is:

```
 Accuracy
0.8103448
```

The F1 score is:

```
[1] "0.56"
```

The confusion matrix is shown in the following table:

	0	1
0	14	3
1	19	80

The RF algorithm has a big improvement in terms of accuracy, but 3 patients are also missclassified as healthy.

The results so far:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959
GBM	0.7241379	0.3043478
SVM	0.7155172	NA
KNN	0.7155172	0.1081081
Random forest	0.8103448	0.5600000

5.12 Model 9: Extreme Gradiant boosting (Xgboost)

In this section, we are going to implement an extreme gradient boosting algorithm. Due to its complexity and computing time, we are only tuning some of the parameters and keeping the rest fixed.

The accuracy of this model is:

```
 Accuracy
0.7327586
```

The F1 score is:

```
[1] "0.205128205128205"
```

The confusion matrix is shown in the following table:

	0	1
0	4	2
1	29	81

The algorithm is a slight improvement over our naive method, in terms of accuracy, but not nearly as good as the Random forest algorithm.

The results so far are:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959
GBM	0.7241379	0.3043478
SVM	0.7155172	NA
KNN	0.7155172	0.1081081
Random forest	0.8103448	0.5600000
XGBoost	0.7327586	0.2051282

5.13 Model 10: Random forest with the original unprocessed dataset

In this final model, we are going to re-train our Random Forest algorithm (best performing one) but this time we are going to use the unprocessed dataset with all the original variables, and excluding the number of extreme values we created.

The accuracy of this model is:

```
 Accuracy
0.7586207
```

The F1 score is:

```
[1] "0.416666666666667"
```

The confusion matrix is shown in the following table:

	0	1
0	10	5
1	23	78

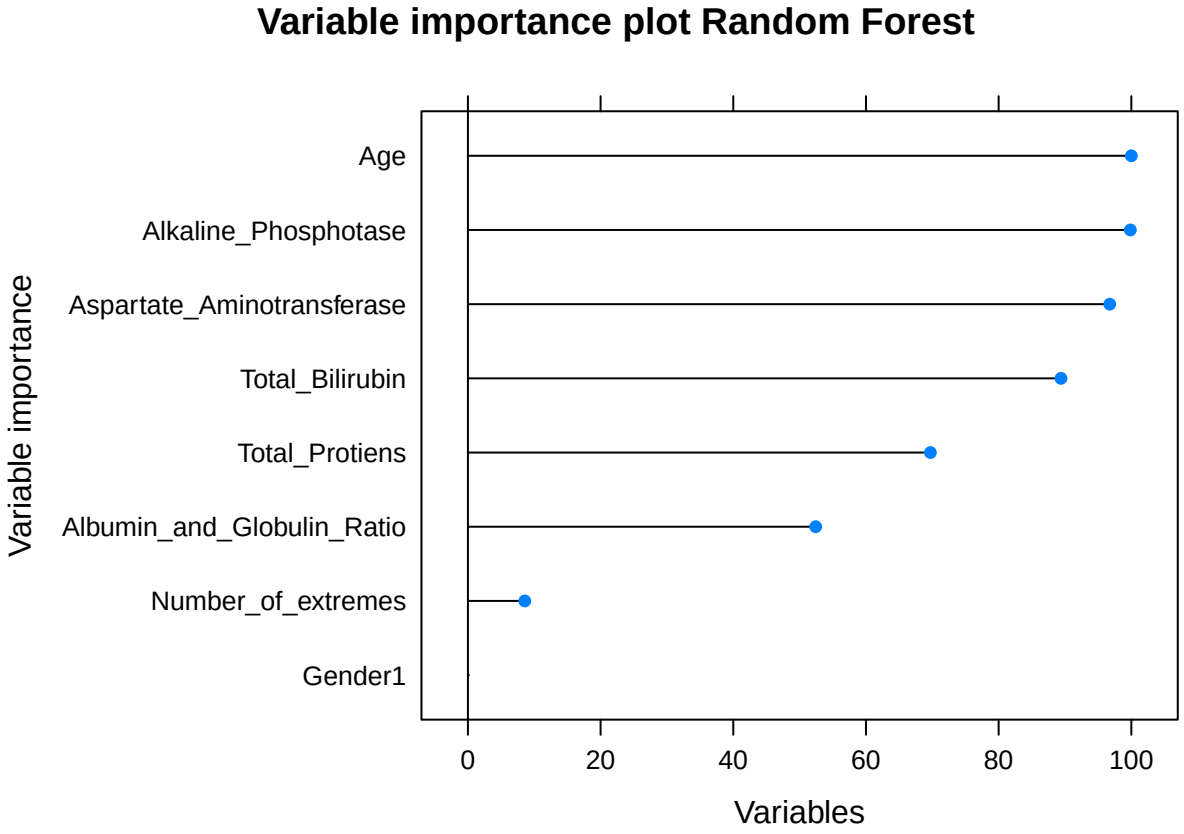
The RF algorithm with the unprocessed dataset is not nearly as good as our implementation, proving that the steps we took in feature engineering were beneficial in the model development phase.

The final results are:

method	Accuracy	F1
Naive	0.7155172	NA
Classification tree	0.7155172	NA
Neural network	0.7586207	0.2631579
Ada boost	0.7844828	0.4897959
GBM	0.7241379	0.3043478
SVM	0.7155172	NA
KNN	0.7155172	0.1081081
Random forest	0.8103448	0.5600000
XGBoost	0.7327586	0.2051282

5.14 Variable importance in the Random forest model (feature engineered dataset)

Before concluding this session, we are going to plot the variable importance of our final algorithm.



6 Conclusion

In this chapter, we will be discussing our methods, models and results, explain some of the limitations of our research and conclude this project with some directions for future research.

6.1 Discussion of results

In chapter 5, we have developed ten machine learning models (9 original models + 1 model for testing feature engineering steps). From those models, the best performing one was the Random forest, with the second best model being 2.5% less accurate (adaboost). In terms of missclassifying sick individuals as healthy, our model managed to missclassify only 3 out of the 83 people with a liver disease, which is quite accurate. Compared to our naive approach (classify all people as sick), we see a trade-off: the increase in accuracy vs. the incorectly classified people. Given the fact that it is a medical application and false negatives are more important than false positives, our naive approach is only accurate 71% of the times (mainly due to the imbalanced dataset) and therefore we suggest that the RF classifier is an improvement over the naive method.

From our analysis, we can also see that the variable, which had the biggest impact in the model improvement, was the age. This finding is counter-intuitive as our EDA did not show a strong relationship between the age and the existence of a liver disease. Additionally, the variable we have created has low importance (8%) and the gender appears to play no role in detecting the liver disease (in line with literature). Interestingly, using all the variables as features proved to be less valueable than using only some of them (conveying different type of information).

Overall, the best performing model had a substantial improvement (13%) over our first benchmark model (Naive method) and predicts with high accuracy the existence of a liver disease in the unseen test set (Accuracy = 81.03%).

6.2 Limitations

The algorithm and research presented, yielded some interesting results but are not without limitations. To begin with, for some of the algorithms, due to the computation time, only a small subset of parameters was studied; however, we could potentially use virtual machines for estimating a larger range of parameters, which could lead to further improvements. Furthermore, as part of the research, we only implemented a small number of algorithms. In that regard, an alternative algorithm might prove to be more robust in predicting the existence of liver disease in people. Finally, given that the dataset is based on the records of Indian patients, the generelizability of the methodology presented needs to be tested with other (international) patients.

6.3 Further research

In this project, several methods for creating a liver disease classifier have been studied, with decent results. In future research, new methods for estimating the existence of liver disease should be examined and benchmarked against the original models. Additionally, we propose a more in-depth investigation of the age effect, to identify some evidence on the importance of this feature. Finally, despite the low importance of the number of extreme blood test results that was developed, some tuning on what should be considered "normal range of values" should also be examined.