Predicting Heart Disease

Mark van de Streek

2023-11-12

1 Practical assignment bioinformatics - predicting heart diseases

```
# Copyright (c) 2023 <Mark van de Streek>.
# Licensed under GPLv3. See gpl.md
```

Loading all the packages that are necessary for this research.

```
# Loading the packages
library(pander)
library(ggplot2)
library(GGally)
library(ggcorrplot)
library(gridExtra)
library(dplyr)
library(data.table)
library(tidyverse)
```

2 Introduction

This study examines a dataset [1] on predicting heart disease based on clinical variables.

The ultimate goal of this research is to create a machine learning model that predicts whether a patient has heart disease.

The dataset contains data from people with and without a heart disease. The last column shows whether the patient has heart disease (presence) or not (absence). The logbook often refers to the classification column. It is then referred to as "presence" and "absence" of heart disease.

research question of the research: Which medical properties are important to be so accurate possible to predict a heart condition/abnormality?

2.1 Details of the data

The dataset contains data from 270 patients. There are 14 independent predictive variables for each patient. These variables are described in further detail in the codebook.

The dataset comes from the UCI repository. The Machine Learning Repository (UCI Repository) is a collection of databases and domains used by the machine learning community to create machine learning algorithms.

The dataset was created and merged by the following 4 agencies:

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

The data was all collected in medical institutions. Good to remember that there can be a difference between medical equipment. However, these are of course not cheap devices that are found in the living room at home, for example. So the deviation will not be very large

2.2 What is heart disease?

Many people have heart disease. Heart disease can vary considerably. A heart condition is defined as [2] *(Heart disease - symptoms and causes - Mayo Clinic, 2022)*:

1. Blood vessel disease
2. Irregular heartbeats
3. Congenital heart defects
4. Disease of the heart muscle
5. Heart valve disease

3 Exploration of the data

In the very first part of this logbook, the data will be examined exploratively. I.e., for example, the distribution of values, correlations, characteristics, etc. is looked at. This is all explained below.

3.1 Data structure and Codebook

3.1.1 Reading the data

Loading the data from the .csv file.

```
data <- read.csv("Heart_Disease_Prediction.csv", header = T, sep = ",")
data <- data.frame(data)
```

An overview of the data will be stored inside a codebook, so it's easy to change/translate. After that's done, it's easier to show all the columns' datatypes and description.

```
codebook <- data.frame(
  attributes = c("index", "Age", "Sex", "Chest.pain.type",
                 "BP", "Cholesterol", "FBS.over.120",
                 "EKG.results","Max.HR", "Exercise.angina",
                 "ST.depression", "Slope.of.ST", "Number.of.vessels.fluro",
                 "Thallium", "Heart.Disease"),
  unit = c("-", "years", "-", "-", "mm/Hg", "mg/dl","-", "-", "Beats/minute",
           "-", "-", "-", "-", "-", "-"),
  dtype = c("int","int","factor","factor","int","int","factor","factor",
            "int","factor","int","factor","factor","factor","factor"),
  description = c("Patient number", "The age of the patient",
                  "The gender of the patient",
                  "The type of chest pain experienced by the patient",
                  "The blood pressure level of the patient",
                  "The cholesterol level of the patient",
                  "The fasting blood sugar test results over 120 mg/dl",
                  "The electrocardiogram results of the patient",
                  "The maximum heart rate levels achieved during exercise testing",
                  "The angina experienced during exercise testing",
                  "The ST depression on a Electrocardiogram",
                  "The slope of ST segment electrocardiogram readings",
                  "The amount vessels seen in Fluoroscopy images",
                  "Thallium Stress test findings",
                  "Whether or not the patient has been diagnosed with Heart Disease"))
```

3.1.2 Description of the attributes

With the codebook we can print a overview of all the variables.

```
# Print an overview of the columns with right datatypes and description
pander(codebook)
```

attributes	unit	dtype	description
index	-	int	Patient number
Age	years	int	The age of the patient
Sex	-	factor	The gender of the patient
Chest.pain.type	-	factor	The type of chest pain experienced by the patient
BP	mm/Hg	int	The blood pressure level of the patient
Cholesterol	mg/dl	int	The cholesterol level of the patient
FBS.over.120	-	factor	The fasting blood sugar test results over 120 mg/dl
EKG.results	-	factor	The electrocardiogram results of the patient
Max.HR	Beats/minute	int	The maximum heart rate levels achieved during exercise testing
Exercise.angina	-	factor	The angina experienced during exercise testing
ST.depression	-	int	The ST depression on a Electrocardiogram
Slope.of.ST	-	factor	The slope of ST segment electrocardiogram readings
Number.of.vessels.fluro	-	factor	The amount vessels seen in Fluoroscopy images
Thallium	-	factor	Thallium Stress test findings
Heart.Disease	-	factor	Whether or not the patient has been diagnosed with Heart Disease

As shown above, the dataset contains some 'factor' attributes. These columns need some extra declaration to understand.

Additional description of nomial attriutes in the dataset:

1. sex
   - 1 = Male
   - 0 = Female
2. Chest pain type
   - Value 1: Typical angina
   - Value 2: Atypical angina
   - Value 3: Non-anginal pain (pain without disease)
   - Value 4: Asymptomatic
3. EKG results
   - Value 0: Normal

- Value 1: ST-T wave abnormality
- Value 2: Probable or definite LVH (thickening of the walls of the left ventricle, main chamber)

4. FBS over 120 (Blood glucose level measured after fasting for at least 8 hours)
    - Value 1: True
    - Value 0: False
5. Slope of ST Depression (Segment in EKG that may indicates a disease)
    - Value 1: Up sloping
    - Value 2: Flat
    - Value 3: Down sloping
6. Exercise induced angina (Narrowed blood vessels that deliver oxygen/nutrients to heart)
    - Value 1 = Yes
    - Value 0 = No
7. Number of major vessels coloured by fluoroscopy
    - range from 1 to 3
8. Thallium (how much blood is reaching different parts of your heart)
    - Value 3: Normal
    - Value 6: Fixed defect
    - Value 7: Reversible defect

With all the right datatypes given, it is possible to change all the columns to the right one. R does a good job by automatically choosing the right datatype for the int/dbl columns. For that reason, you only have to change the nominal datatypes. This is done below.

```
for (i in 1:ncol(data)) {
  if (codebook[,3][i] == "factor") {
    data[, codebook[,1][i]] <- sapply(data[, codebook[,1][i]], as.factor)
  }
}
```

Now the data is all in the right datatypes, we can look at the dimensions of the data.

```
dims <- dim(data)
sprintf("Amount of rows: %.f, amount of columns: %.f", dims[1], dims[2])
```

```
## [1] "Amount of rows: 270, amount of columns: 15"
```

The data contains 270 rows. That means there is data from 270 patients. This corresponds to expectations. For all these patients there are 14 values that say something about that pearson. Not 15 because the first column is the index.

3.1.3 Missing values

As the name says, this section will look at the missing values in the dataset. Missing values can have major influence on the model. To make sure there aren't a lot of missing values in the dataset, we will check this with a simple loop.

```
cat("Missing values:", any(is.na(data)))
```

```
## Missing values: FALSE
```

As you can see above, there are no missing values in the dataset. beyond that, there is little to say about it.

3.2 Univariate analysis

This section examines each variable in the dataset separately. The distribution of numerical values, outliers of numerical values and variation of ordinal values will be examined

3.2.1 Variation of the data

There are a lot of numeric columns in the dataset. This means that the distribution shows us a good first impression of the data.

We will look at the nominal/ordinal values as well, so we're going to filter these first.

```
numeric.columns <- c(2,5,6,9,11)
nominal.columns <- c(3,4,7,8,10,12,13,14,15)
```

3.2.1.1 Variation of numeric attributes We made sure there aren't missing values in the dataset. Let's look at the range of columns.

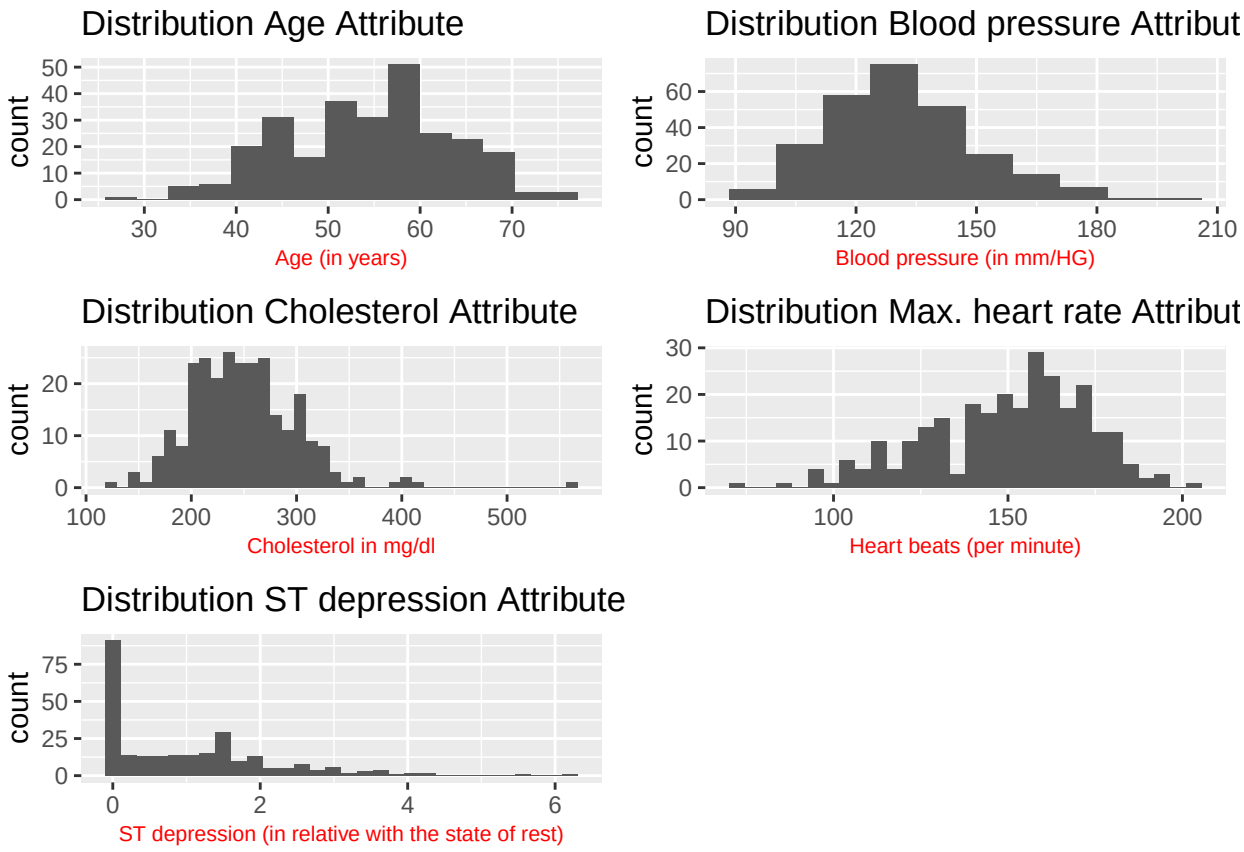
```
pander(summary(data[numeric.columns]))
```

Age	BP	Cholesterol	Max.HR	ST.depression
Min. :29.00	Min. : 94.0	Min. :126.0	Min. : 71.0	Min. :0.00
1st Qu.:48.00	1st Qu.:120.0	1st Qu.:213.0	1st Qu.:133.0	1st Qu.:0.00
Median :55.00	Median :130.0	Median :245.0	Median :153.5	Median :0.80
Mean :54.43	Mean :131.3	Mean :249.7	Mean :149.7	Mean :1.05
3rd Qu.:61.00	3rd Qu.:140.0	3rd Qu.:280.0	3rd Qu.:166.0	3rd Qu.:1.60
Max. :77.00	Max. :200.0	Max. :564.0	Max. :202.0	Max. :6.20

Looking at the ST.depression, thit column has a wide range between minimum and maximum. The median and mean value is also far from the maximum. These could be outlier values, but it could also just be how the value is constructed.

Now let's look at the distribution. This may give us a slightly clearer picture

```
c1 <- ggplot(data, aes(x=Age)) + geom_histogram(bins = 15) +
  labs(title = "Distribution Age Attribute") + xlab("Age (in years)") +
  theme(axis.title.x = element_text(colour = "red", size = 8))

c2 <- ggplot(data, aes(x=BP)) + geom_histogram(bins = 10) +
  labs(title = "Distribution Blood pressure Attribute") +
  xlab("Blood pressure (in mm/HG)") + theme(axis.title.x =
                                  element_text(colour = "red", size = 8))

c3 <- ggplot(data, aes(x = Cholesterol)) + geom_histogram(bins = 40) +
  labs(title = "Distribution Cholesterol Attribute") +
  xlab("Cholesterol in mg/dl") +
  theme(axis.title.x = element_text(colour = "red", size = 8))

c4 <- ggplot(data, aes(x=Max.HR)) + geom_histogram(bins = 30) +
  labs(title = "Distribution Max. heart rate Attribute") +
  xlab("Heart beats (per minute)") + theme(axis.title.x =
                                    element_text(colour = "red", size = 8))

c5 <- ggplot(data, aes(x=ST.depression)) + geom_histogram(bins = 30) +
  labs(title = "Distribution ST depression Attribute") +
  xlab("ST depression (in relative with the state of rest)") +
  theme(axis.title.x = element_text(colour = "red", size = 8))

grid.arrange(c1,c2,c3,c4,c5, ncol = 2)
```

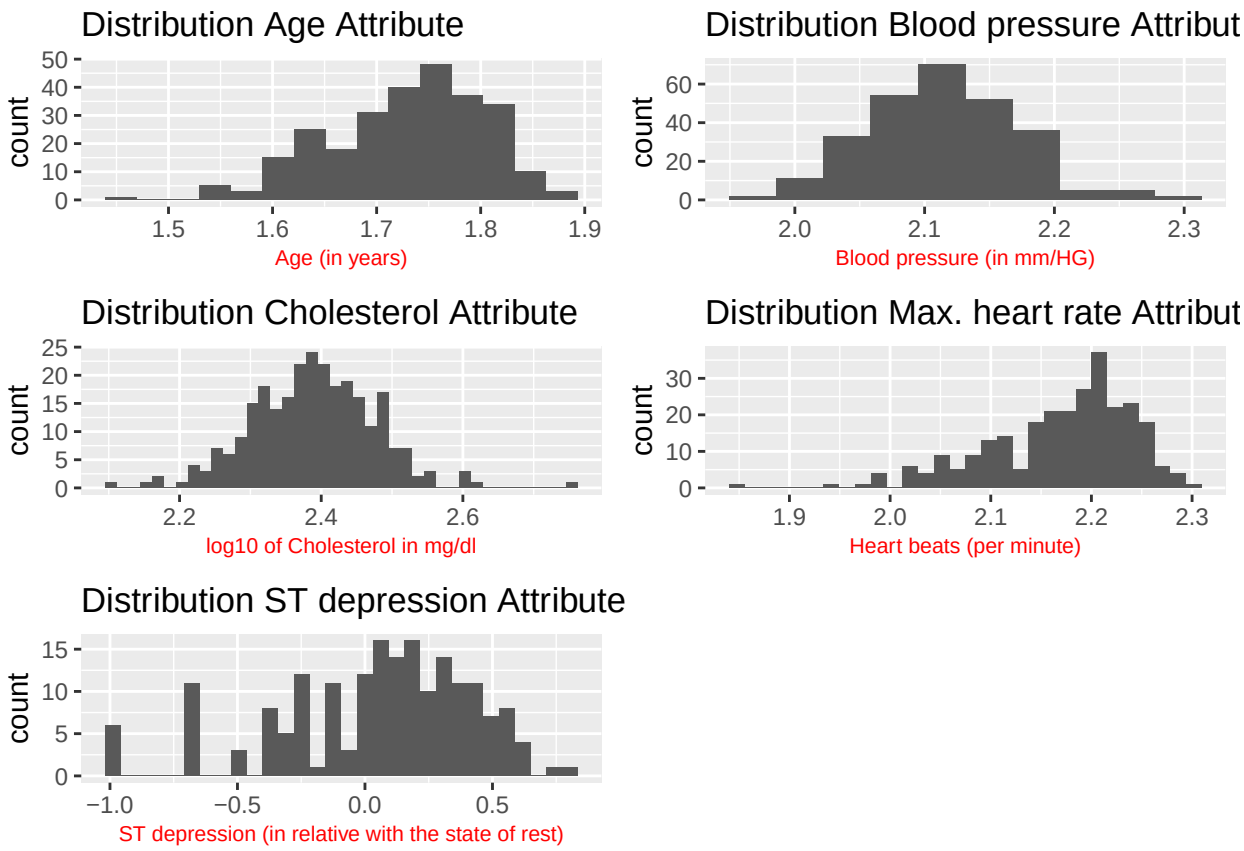


The above graphs tells us that Cholesterol is quite well normal distributed. The same can be said for BP. Max.HR and Age appears to be split more towards the right. ST.depression shows a very high peak at the first bin.

Let's transform the data with log10 and see if the distribution is better/more observable.

```
c1 <- ggplot(data, aes(x=log10(Age))) + geom_histogram(bins = 15) +
  labs(title = "Distribution Age Attribute") + xlab("Age (in years)") +
  theme(axis.title.x = element_text(colour = "red", size = 8))

c2 <- ggplot(data, aes(x=log10(BP))) + geom_histogram(bins = 10) +
  labs(title = "Distribution Blood pressure Attribute") +
  xlab("Blood pressure (in mm/HG)") + theme(axis.title.x =
                                  element_text(colour = "red", size = 8))

c3 <- ggplot(data, aes(x = log10(Cholesterol))) + geom_histogram(bins = 40) +
  labs(title = "Distribution Cholesterol Attribute") +
  xlab("log10 of Cholesterol in mg/dl") +
  theme(axis.title.x = element_text(colour = "red", size = 8))

c4 <- ggplot(data, aes(x=log10(Max.HR))) + geom_histogram(bins = 30) +
  labs(title = "Distribution Max. heart rate Attribute") +
  xlab("Heart beats (per minute)") + theme(axis.title.x =
                                    element_text(colour = "red", size = 8))

c5 <- ggplot(data, aes(x=log10(ST.depression))) + geom_histogram(bins = 30) +
  labs(title = "Distribution ST depression Attribute") +
```

```
  xlab("ST depression (in relative with the state of rest)") +
  theme(axis.title.x = element_text(colour = "red", size = 8))

grid.arrange(c1,c2,c3,c4,c5, ncol = 2)
```

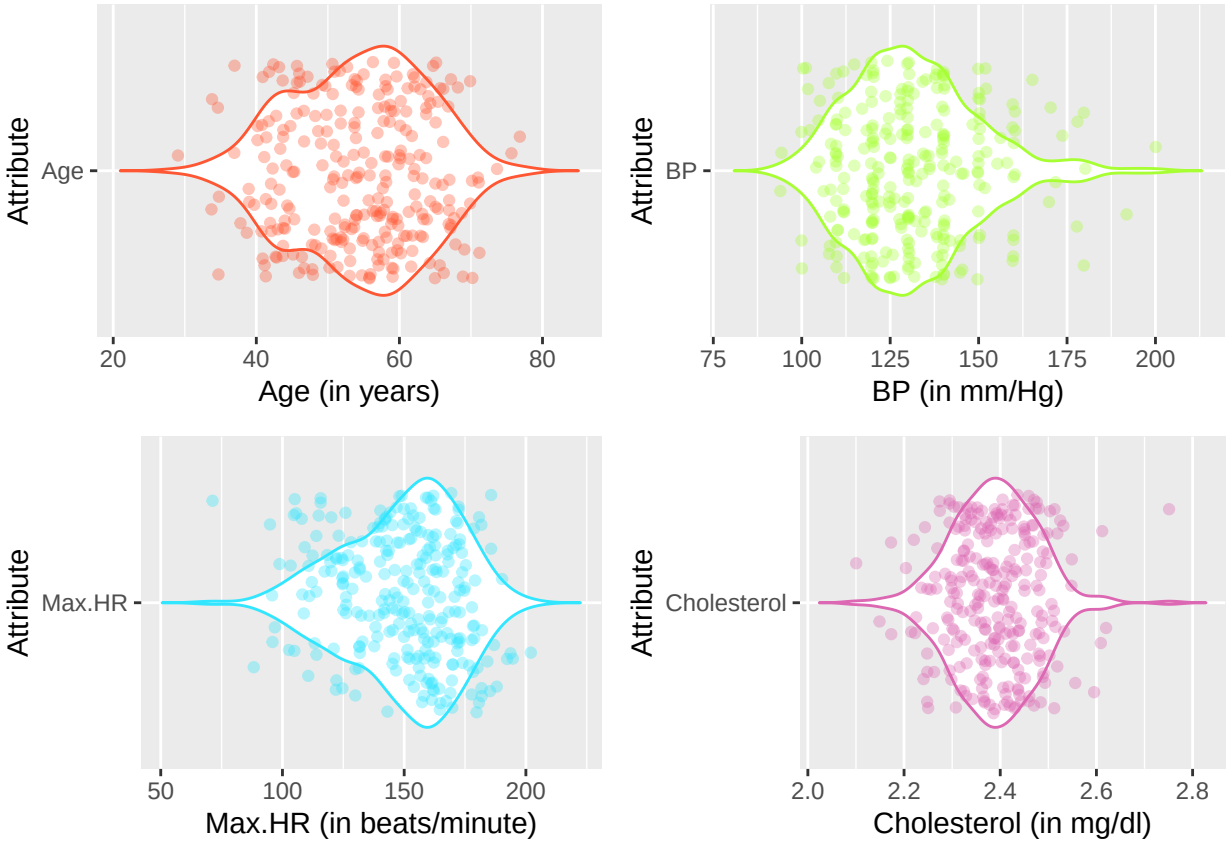


The cholesteral attribute is good normal distributed now. ST.depression is not good normal distributed but is more readable. However, in ST.depression there is a high peak at - infinity. The attribute is still not normally distributed, so it is left behind for the transformation.

```
data$Cholesterol <- log10(data$Cholesterol)
```

With the distributions displayed, we can then begin to identify *possible* outliers. For this we use a violin plot. That's because a violin plot allows you to examine the distribution of the data and spot some outliers. It can also be immediately seen whether the outliers have a strong influence on the distribution.

```
numeric.frame <- data.frame(
  Age = rep("Age", length(data$Age)),
  BP = rep("BP", length(data$BP)),
  Max.HR = rep("Max.HR", length(data$Max.HR)),
  Cholesterol = rep("Cholesterol", length(data$Cholesterol)),
  Age.value = data$Age,
  BP.value = data$BP,
  Max.HR.value = data$Max.HR,
  Cholesterol.value = data$Cholesterol)

p1 <- ggplot(numeric.frame, aes(x=Age, y=Age.value, colour = Age)) +
  theme(legend.position = "top") + coord_flip() +
  geom_violin(trim = F, alpha = 1, color = "#FF5733") +
  geom_jitter(alpha = 0.35, color = "#FF5733") +
  ylab("Age (in years)") + xlab("Attribute")

p2 <- ggplot(numeric.frame, aes(x=BP, y=BP.value)) +
  theme(legend.position = "top") + coord_flip() +
  geom_violin(trim = F, alpha = 1, color = "#A7FF33") +
  geom_jitter(alpha = 0.35, color = "#A7FF33") +
  ylab("BP (in mm/Hg)") + xlab("Attribute")

p3 <- ggplot(numeric.frame, aes(x=Max.HR, y=Max.HR.value)) +
  theme(legend.position = "top") + coord_flip() +
  geom_violin(trim = F, alpha = 1, color = "#33E6FF") +
  geom_jitter(alpha = 0.35, color = "#33E6FF") +
  ylab("Max.HR (in beats/minute)") + xlab("Attribute")

p4 <- ggplot(numeric.frame, aes(x=Cholesterol, y=Cholesterol.value)) +
  theme(legend.position = "top") + coord_flip() +
  geom_violin(trim = F, alpha = 1, color = "#DC67B2") +
  geom_jitter(alpha = 0.35, color = "#DC67B2") +
  ylab("Cholesterol (in mg/dl)") + xlab("Attribute")

grid.arrange(p1,p2,p3,p4, ncol = 2)
```



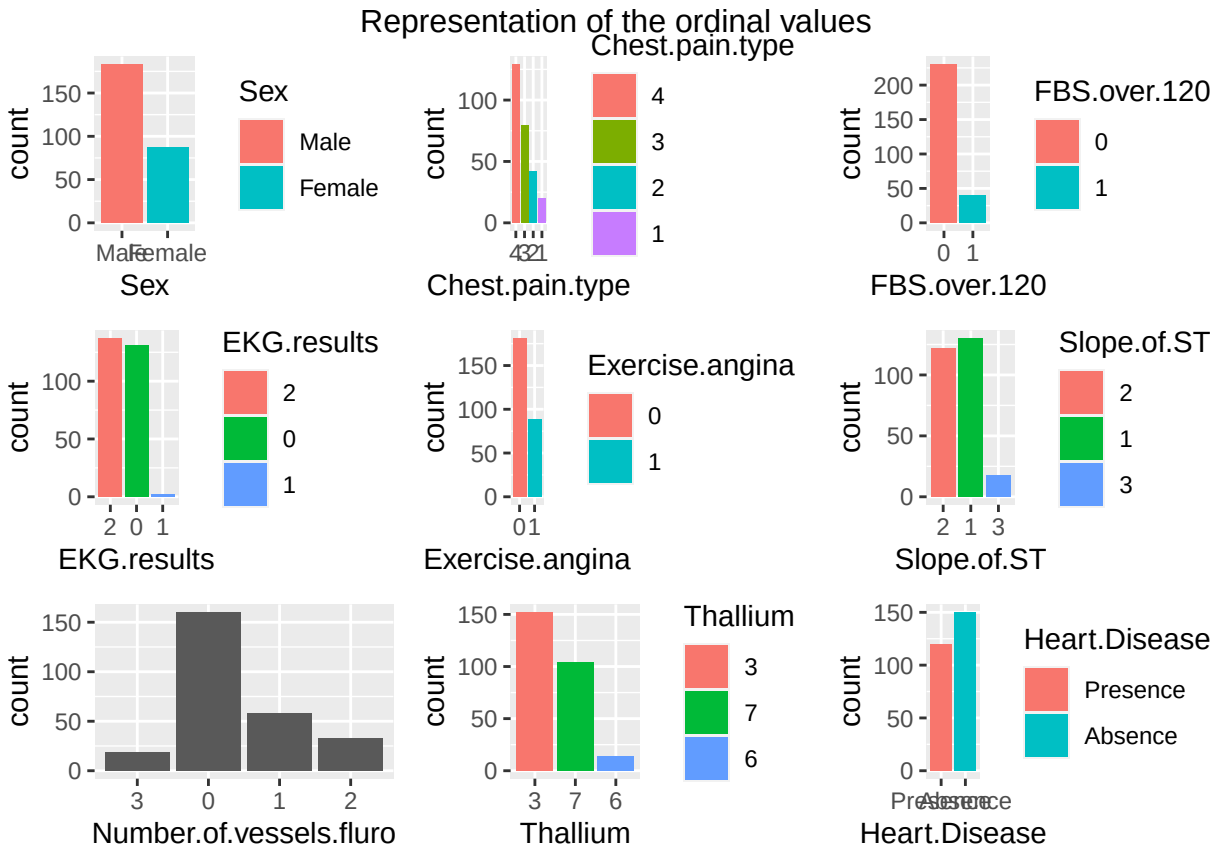
The data contains few outliers. BP and Max.HR has a number of points that differ slightly. But looking at the distribution, it still looks quite well normal. We can conclude that:

- There are no major deviations in the distribution of the data
- The numerical data contain no notable outliers

3.2.1.2 Variation of nominal/ordinal attributes

Let's look at the nominal values new. We are going to plot how often these occur. This is to be able to see whether all classes are represented.

```
levels(data$Sex) <- c("Male", "Female")
p1 <- ggplot(data, aes(x = Sex, fill = Sex)) + geom_bar()
p2 <- ggplot(data, aes(x = Chest.pain.type, fill = Chest.pain.type)) + geom_bar()
p3 <- ggplot(data, aes(x = FBS.over.120, fill = FBS.over.120)) + geom_bar()
p4 <- ggplot(data, aes(x = EKG.results, fill = EKG.results)) + geom_bar()
p5 <- ggplot(data, aes(x = Exercise.angina, fill = Exercise.angina)) + geom_bar()
p6 <- ggplot(data, aes(x = Slope.of.ST, fill = Slope.of.ST)) + geom_bar()
p7 <- ggplot(data, aes(x = Number.of.vessels.fluro)) + geom_bar()
p8 <- ggplot(data, aes(x = Thallium, fill = Thallium)) + geom_bar()
p9 <- ggplot(data, aes(x = Heart.Disease, fill = Heart.Disease)) + geom_bar()

grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9, ncol = 3,
             top = "Representation of the ordinal values")
```



The dataset contains more man than women. Most values lean towards 'healthy' people. There are also more people who do not have a condition than do. So this difference partly comes from here. However, there are also many people who do have a condition, but have few or no complaints, for example, see figure Chest.pain.type.

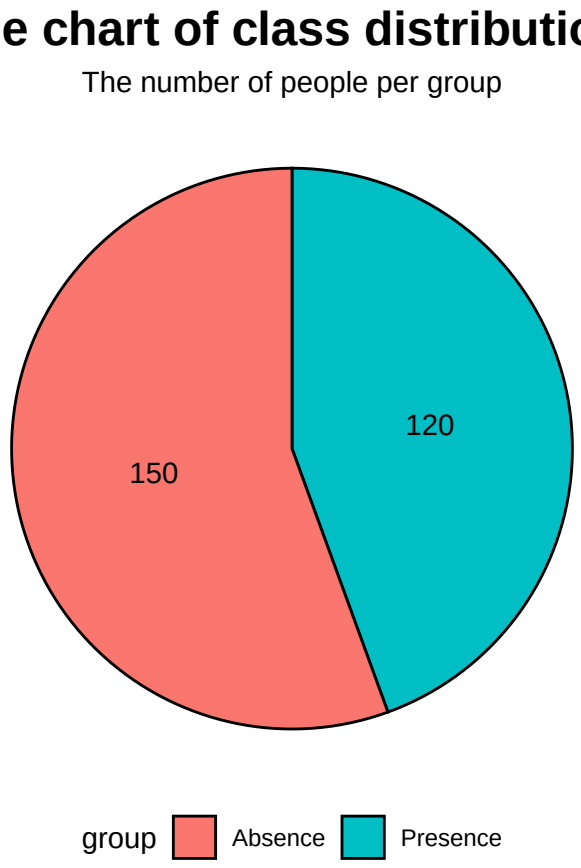
conclusions:

- Comparatively few people have an FBS of over 120
- There seems to be enough distribution in these attributes to train a good model (this says nothing about the usability of all attributes)

3.2.2 Class distribution

For our model it's important to know how the classes are distributed. We can have a big dataset, but if it's only "absence" people, we can't say much about it.

```
df <- data.frame(
  group = c("Presence", "Absence"),
  value = c(sum(data[15] == "Presence"), sum(data[15] == "Absence"))
)

ggplot(df, aes(x = "", y = value, fill = group)) +
  geom_col(color = "black") +
  geom_text(aes(label = value),
            position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") + labs(title = "Pie chart of class distribution",
  subtitle = "The number of people per group") + theme_void() +
  theme(legend.position = "bottom", plot.title = element_text(face = "bold",
      size = 18, hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```



As you can see, the classes are not exactly evenly represented. But it certainly has no major deviation.

3.3 Bivariate analysis

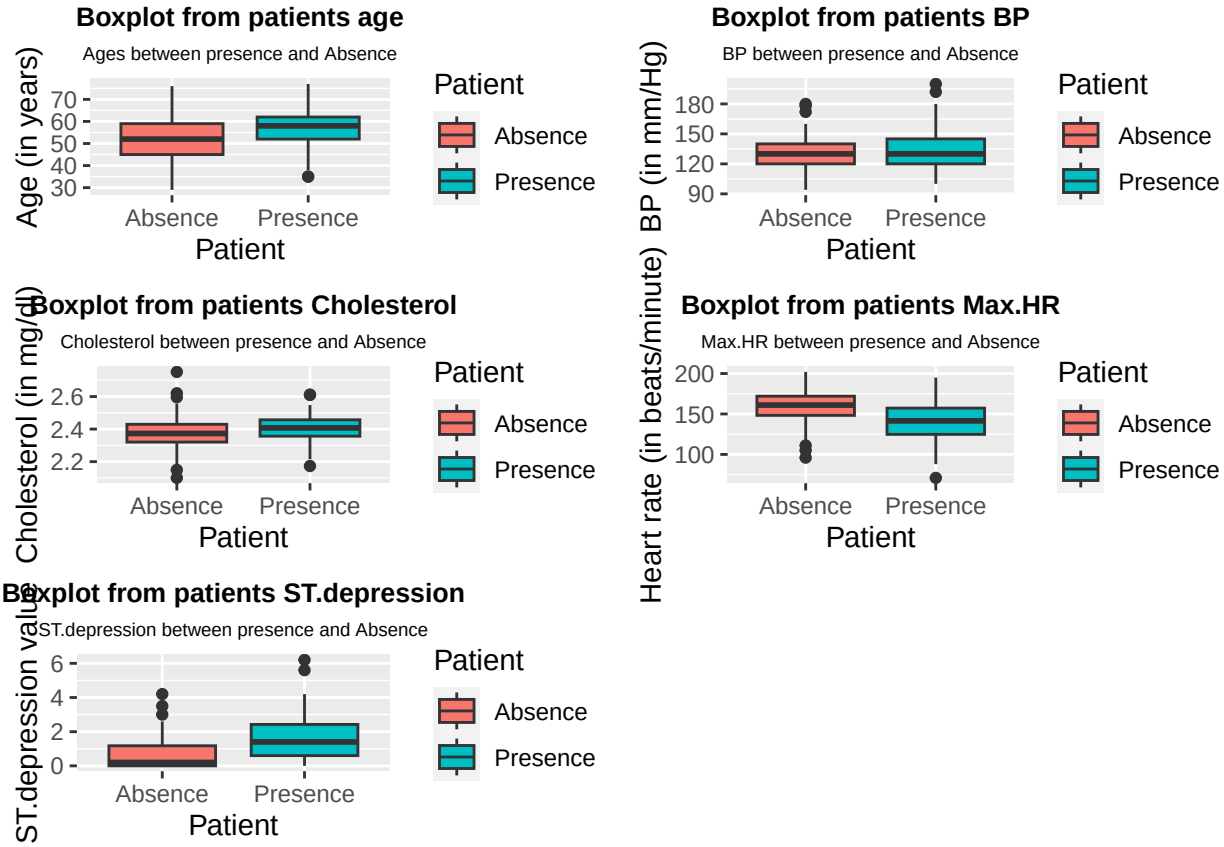
In this sector, not one, but several variables are considered. By this we mean that we will look at each other. For example, how much correlation certain attributes have. First, we will compare age between groups

3.3.1 Comparing the numeric attributes with classification attribute

Let's take a closer look at the patients and numeric values. Perhaps we spot some initial differences.

```
age.frame <- data.frame(
  Patient = c(rep("Presence", 120), rep("Absence", 150)),
  Age = c(data$Age[data$Heart.Disease == "Presence"],
            data$Age[data$Heart.Disease == "Absence"]),
  BP = c(data$BP[data$Heart.Disease == "Presence"],
         data$BP[data$Heart.Disease == "Absence"]),
  Cholesterol = c(data$Cholesterol[data$Heart.Disease == "Presence"],
                  data$Cholesterol[data$Heart.Disease == "Absence"]),
  Max.HR = c(data$Max.HR[data$Heart.Disease == "Presence"],
             data$Max.HR[data$Heart.Disease == "Absence"]),
  ST.depression = c(data$ST.depression[data$Heart.Disease == "Presence"],
                    data$ST.depression[data$Heart.Disease == "Absence"]))

b1 <- ggplot(age.frame, aes(x = Patient, y = Age, fill=Patient)) +
  geom_boxplot() + labs(title = "Boxplot from patients age",
                        subtitle = "Ages between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 10),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) +
  ylab("Age (in years)")

b2 <- ggplot(age.frame, aes(x = Patient, y = BP, fill=Patient)) +
  geom_boxplot() + labs(title = "Boxplot from patients BP",
                        subtitle = "BP between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 10),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) +
  ylab("BP (in mm/Hg)")

b3 <- ggplot(age.frame, aes(x = Patient, y = Cholesterol, fill=Patient)) +
  geom_boxplot() + labs(title = "Boxplot from patients Cholesterol",
                        subtitle = "Cholesterol between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 10),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) +
  ylab("Cholesterol (in mg/dl)")

b4 <- ggplot(age.frame, aes(x = Patient, y = Max.HR, fill=Patient)) +
  geom_boxplot() + labs(title = "Boxplot from patients Max.HR",
                        subtitle = "Max.HR between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 10),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) +
  ylab("Heart rate (in beats/minute)")

b5 <- ggplot(age.frame, aes(x = Patient, y = ST.depression, fill=Patient)) +
  geom_boxplot() + labs(title = "Boxplot from patients ST.depression",
                        subtitle = "ST.depression between presence and Absence") +
```

```
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 10),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("ST.depression value")

grid.arrange(b1,b2,b3,b4,b5, ncol = 2)
```



The results show that the age is slightly higher in the presence group. It can be concluded that patients with a presence-condition are often slightly older. The Max.HR is also higher in the absence group. And ST.depression is higher in the Presence group.

We can confirm the differneces with statistical tests.

```
age.test <- t.test(data$Age[data$Heart.Disease == "Presence"],
       data$Age[data$Heart.Disease == "Absence"])$p.value

bp.test <- t.test(data$BP[data$Heart.Disease == "Presence"],
                  data$BP[data$Heart.Disease == "Absence"])$p.value

cholesterol.test <- t.test(data$Cholesterol[data$Heart.Disease == "Presence"],
                  data$Cholesterol[data$Heart.Disease == "Absence"])$p.value

max.hr.test <- t.test(data$Max.HR[data$Heart.Disease == "Presence"],
                      data$Max.HR[data$Heart.Disease == "Absence"])$p.value

st.depression.test <- t.test(data$ST.depression[data$Heart.Disease == "Presence"],
                  data$ST.depression[data$Heart.Disease == "Absence"])$p.value

t.test.results <- data.frame(
  Attributes = c("Age", "BP", "Cholesterol", "Max.HR", "ST depression"),
  P.Value = c(age.test, bp.test, cholesterol.test,
              max.hr.test, st.depression.test))

pander(t.test.results)
```

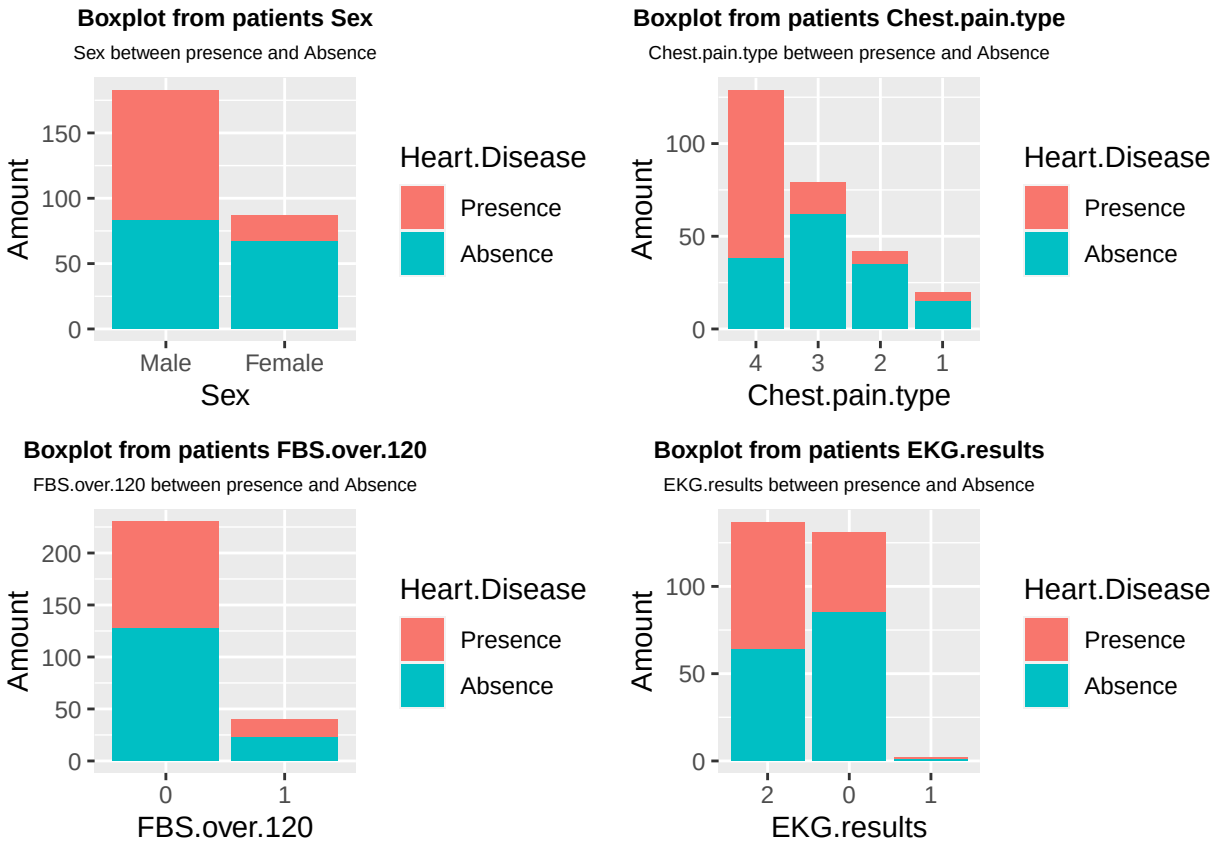
Attributes	P.Value
Age	0.0003526
BP	0.01196
Cholesterol	0.02946
Max.HR	2.604e-12
ST depression	1.601e-11

The cholesterol p-value is less than 0.05, but the value is just below 0.05. The difference could therefore be characterized as small. The rest of the values have a significant difference between both groups

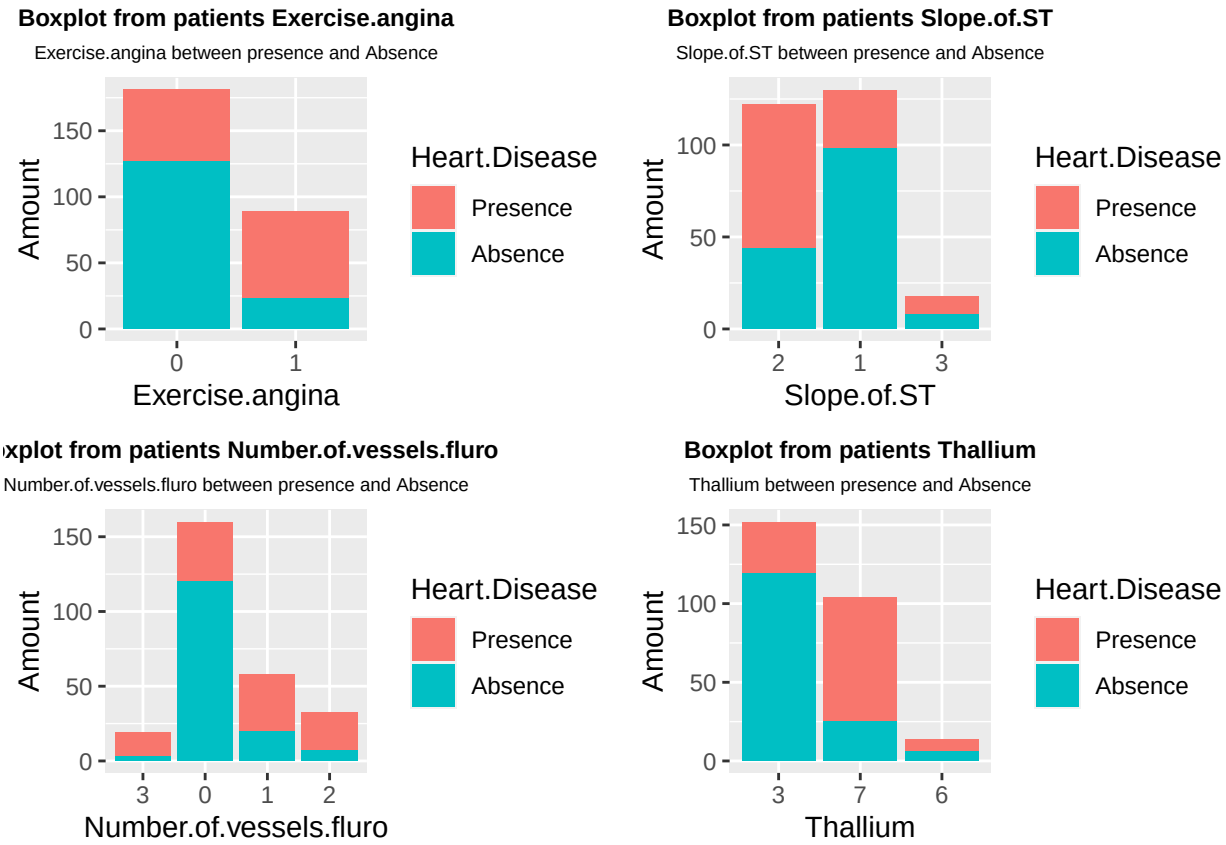
3.3.2 Comparing the Categorial attributes with classification attribute

Now let's do the same as above, but then for categorial data.

```
b1 <- ggplot(data, aes(x = Sex, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients Sex",
                        subtitle = "Sex between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b2 <- ggplot(data, aes(x = Chest.pain.type, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients Chest.pain.type",
                    subtitle = "Chest.pain.type between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b3 <- ggplot(data, aes(x = FBS.over.120, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients FBS.over.120",
                      subtitle = "FBS.over.120 between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b4 <- ggplot(data, aes(x = EKG.results, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients EKG.results",
                        subtitle = "EKG.results between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b5 <- ggplot(data, aes(x = Exercise.angina, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients Exercise.angina",
                    subtitle = "Exercise.angina between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b6 <- ggplot(data, aes(x = Slope.of.ST, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients Slope.of.ST",
                        subtitle = "Slope.of.ST between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b7 <- ggplot(data, aes(x = Number.of.vessels.fluro, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients Number.of.vessels.fluro",
          subtitle = "Number.of.vessels.fluro between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

b8 <- ggplot(data, aes(x = Thallium, fill=Heart.Disease)) +
  geom_bar() + labs(title = "Boxplot from patients Thallium",
                        subtitle = "Thallium between presence and Absence") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 8.5),
        plot.subtitle = element_text(hjust = 0.5, size = 7)) + ylab("Amount")

grid.arrange(b1, b2, b3, b4, ncol = 2)
```




```
grid.arrange(b5, b6, b7, b8, ncol = 2)
```



There are many more men than women with a heart disease. Most people with a condition experience no pain on the chest. The ratio between sick and not sick between FBS over 120 is the same. There are many more people without FBS over 120. There are few people with exercise angina and no condition. Thallium 7 is most common in people with a condition.

Let's do statistical tests for this as well.

```
sex.test <- chisq.test(data$Sex, data$Heart.Disease,
                       correct = FALSE)$p.value

chest.pain.type.test <- chisq.test(data$Chest.pain.type, data$Heart.Disease,
                                   correct = FALSE)$p.value

FBS.test <- chisq.test(data$FBS.over.120, data$Heart.Disease,
                       correct = FALSE)$p.value

EKG.test <- chisq.test(data$EKG.results, data$Heart.Disease,
                       correct = FALSE)$p.value

Exercise.test <- chisq.test(data$Exercise.angina, data$Heart.Disease,
                            correct = FALSE)$p.value

Slope.test <- chisq.test(data$Slope.of.ST,
                         data$Heart.Disease, correct = FALSE)$p.value
Number.test <- chisq.test(data$Number.of.vessels.fluro,
                          data$Heart.Disease, correct = FALSE)$p.value
Thallium.test <- chisq.test(data$Thallium,
                            data$Heart.Disease, correct = FALSE)$p.value

chisq.results <- data.frame(
  Attributes = c(
    "Sex", "Chest Pain Type", "FBS over 120", "EKG results",
    "Exercise angina", "Slope of ST", "Number of vessels", "Thallium"),
  P.Value = c(sex.test, chest.pain.type.test, FBS.test, EKG.test,
              Exercise.test, Slope.test, Number.test, Thallium.test))

pander(chisq.results)
```

Attributes	P.Value
Sex	9.979e-07
Chest Pain Type	8.561e-15
FBS over 120	0.7886
EKG results	0.01122
Exercise angina	5.585e-12
Slope of ST	1.713e-09
Number of vessels	1.437e-13
Thallium	6.419e-17

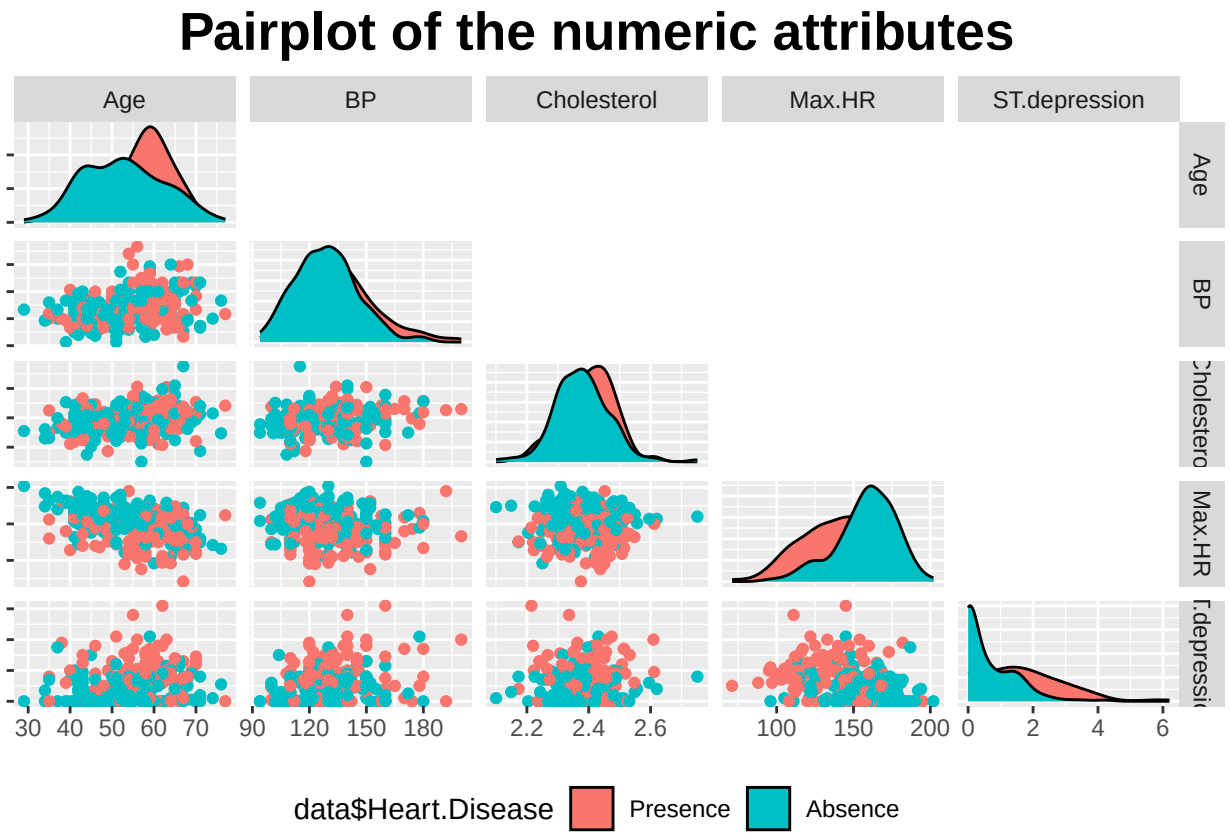
All p-values except FBS over 120 have a p-value lower than 0.05. This means that all these values are related to the classification column (Presence/Absence). The lowest p-values are of course the most correlated.

3.3.3 Corrolation between attributes

As mentioned earlier, the main goal of this research was to develop a machine learning model. For this model, it is useful to see which attributes may be important. We will compare all numeric attributes with each other. The goal is to demonstrate similarities or differences.

Scatterplots are plotted between all attributes. This makes it possible to see how far apart the groups presence/absence are. Or just how close they are to each other.

```
ggpairs( data[numeric.columns], ggplot2::aes(colour=data$Heart.Disease),
         progress = F, upper = "blank", legend = 1) +
  labs(title = "Pairplot of the numeric attributes") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
        axis.text.y = element_blank(), legend.position = "bottom")
```

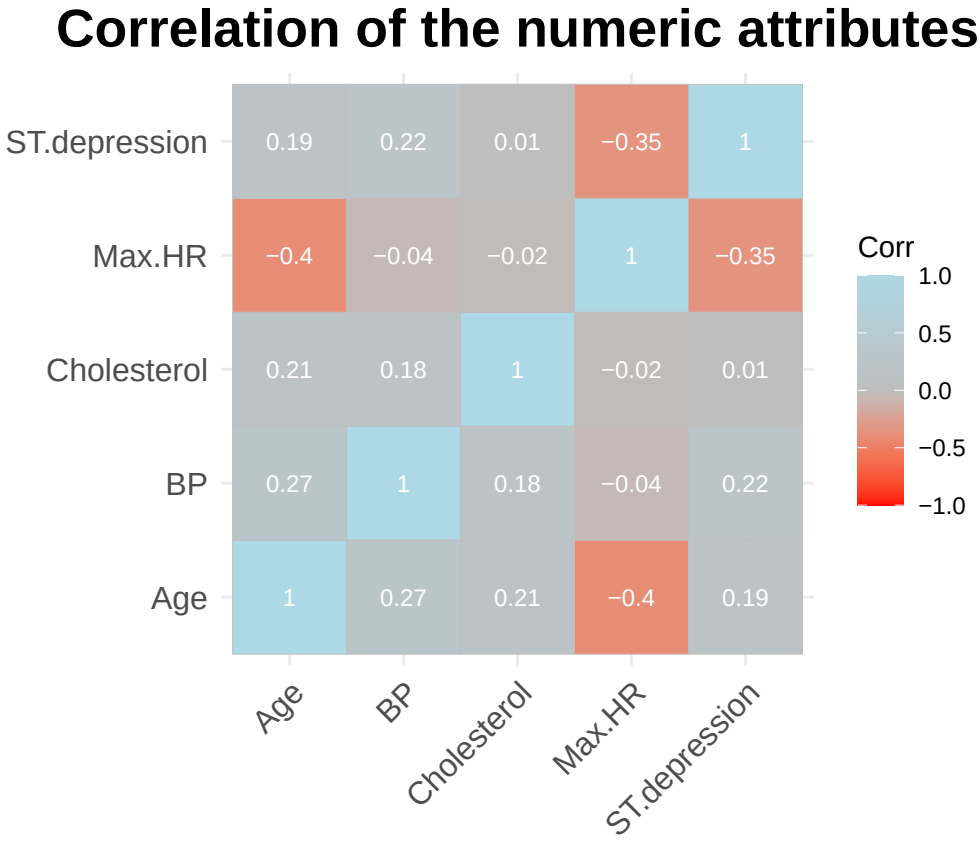


The figure above shows a number of things. It shows the similarities between all variables. However, it is of course much clearer to simply attach a number to the similarities. This happens below with a correlation heat map.

Conclusions pairplot:

- There is a small similaritie in the age attribute
- The maximum heart rate appears to be lower in people who have a condition

```r
colMA <- function(n = 3) {
  colorRampPalette(c("red", "grey", "lightblue"), space = "rgb")(n)
}


# plotting corr heatmap
ggcorrplot(cor(data[numeric.columns]), colors = colMA(),
           lab = T, lab_col = "white", lab_size = 3) +
  labs(title = "Correlation of the numeric attributes") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 20))
```



The output tells us that there are some values that are slightly to moderately correlated. Blood pressure and age have the highest correlation. Furthermore, there are no values that stand out. ST depression and maximum heart rate have the least correlation

Conclusion:

- BP and Age have the highest correlation
- There are no very high or low correlations, the values complement each other well

3.4 Conclusions of the tests sorted

Below is a table in which all p-values are sorted.

```
combined <- bind_rows(chisq.results, t.test.results)
pander(combined[order(combined$P.Value),])
```

	Attributes	P.Value
8	Thallium	6.419e-17
2	Chest Pain Type	8.561e-15
7	Number of vessels	1.437e-13
12	Max.HR	2.604e-12
5	Exercise angina	5.585e-12
13	ST depression	1.601e-11
6	Slope of ST	1.713e-09
1	Sex	9.979e-07
9	Age	0.0003526
4	EKG results	0.01122
10	BP	0.01196
11	Cholesterol	0.02946
3	FBS over 120	0.7886

The attribute thallium is most related to disease. This value is going to be important for the model. The list if followed by chest pain type. The FBS value over 120 will be the least useful

4 Clean Dataset

As told earlier, some columns have been log10 transformed. We are now going to make a 'clean' dataset. I.e., a dataset with usable columns and with the right range.

Let's delete the index column. This column is unnecessary and we don't need it for our machine learning model later on.

```
data <- data[ -c(1) ]
```

For the clean dataset it is also easy to read a number of values more quickly. Numbers cannot be read quickly, so we will convert (simple) values into words.

```
levels(data$Sex) <- c("Male", "Female")
levels(data$FBS.over.120) <- c(FALSE, TRUE)
levels(data$Exercise.angina) <- c("No", "Yes")
levels(data$Chest.pain.type) <- c("Asymptomatic", "Non-anginal pain (pain without disease)",
                                  "Atypical angina", "Typical angina")
levels(data$EKG.results) <- c("Probable or definite LVH", "Normal", "ST-T wave abnormality")
levels(data$Slope.of.ST) <- c("Flat", "Up sloping", "Down sloping")
levels(data$Number.of.vessels.fluro) <- c("3", "0", "1", "2")
levels(data$Thallium) <- c("Normal", "Reversible defect", "Fixed defect")
```

We are now creating three other datasets to test algorithms in weka. First, a dataset is created that contains only the attributes of the top 5 lowest p-values. Then another data set is saved in which the remaining numeric columns are transformed. Finally, a dataset will be created, containing only the simple attributes. These attributes can easily be taken with a simple test.

```
data.top5 <- data[c("Thallium", "Chest.pain.type",
                    "Number.of.vessels.fluro", "Max.HR",
                    "Exercise.angina", "Heart.Disease")]

data.allTransformed <- data
data.allTransformed$Max.HR <- log10(data.allTransformed$Max.HR)
data.allTransformed$BP <- log10(data.allTransformed$BP)

data.simple <- data[c("Age", "Sex", "Cholesterol", "BP", "Heart.Disease")]
```

There are now clean data sets. We are going to save these in a CSV file so that we can use it in Weka.

```
write.csv(data, "ExperimenterTestSets/01-Clean_dataset.csv", row.names = F)
write.csv(data.top5, "ExperimenterTestSets/02-top5_p_values.csv", row.names = F)
write.csv(data.allTransformed, "ExperimenterTestSets/03-Numeric_transformed.csv",
          row.names = F)
write.csv(data.simple, "ExperimenterTestSets/04-Simple.csv", row.names = F)
```

5 Machine learning model

Now that the datasets are stored, the best model can be created. We create a new Weka experiment with different algorithms and datasets.

5.1 Algorithms and Settings

In weka there is a option to compare all datasets and concepts with eachother. This is done at the "experimenter" tab. Create a "new" experimenter and add the files that you've just created above. On the right pannel, add the algorithms that we want the run. The algorithms are described below.

1. ZeroR
2. OneR
    1. Bucket size 6
    2. Bucket size 15
3. J48
    1. Minimum number of objects: 2, Backward Pruning: 0.25
    2. Minimum number of objects: 2, Unpruned
    3. Minimum number of objects: 15, Backward Pruning: 0.25
    4. Minimum number of objects: 35, Backward Pruning: 0.25
    5. Minimum number of objects 10, Backward Pruning: 0.025
4. RandomForest - standard settings
5. RandomTree - standard settings
6. K-nears neighbours
    1. KNN: 10
    2. KNN: 35 (35 was the best result from CVParamaterSelection)
    3. KNN: 35, Manhattan distance
7. SMO - standard settings
8. Logistic - standard settings
9. NaieveBayes
10. ClassificationViaClustering - standard settings

After you've add all the algorithms, we can now run the experiment. This can be done at the "Run" tab. In the "Analyse" tab, you can retrieve the experiment results by pressing "Experiment". It's now possible to perform tests. These tests can be used to determine which concept scores the best or highest.

By selecting a comparison field, you can get the right information about the experiment. We will look at Percent_correct, Area_under_ROC, False_negative_rate, False_pisitive_rate, User-CPU_CPU_Time_Training and F-score. After the right comparison field is selected, press "Perform test". A table will now be displayed with the results. **"V"** means siginificantly higher and **"*"** means significantly lower.

The columns and rows can be reversed by pressing "sawp". Sometimes you have to select a comparison field again, after pressing swap.

It is possible to save the experimenter's results. This can be done in the "setup" tab. A path and file name can be selected under the heading "Results destination". The results are saved after pressing "Run".

The next section looks at the experimenter's results with the above datasets and algorithms.

5.2 Experimenter analysis

```
weka.results <- read.csv("WEKA_RESULTS.csv")
weka.results <- data.frame(weka.results)
```

We have to group the results because there are a lot of values. With grouping it is possible to see what values are per dataset or per concept. This way we don't have to go through the complete results and create groups/new data frames with results.

```
Results <- weka.results %>%
 group_by(Key_Dataset, Key_Scheme, Key_Scheme_options) %>%
  summarise(mean = mean(Percent_correct), .groups = "keep")
```

5.2.1 Accuracy

Let's look at the accuracy first.

```
# Dataset                    (1) 01-Clean | (2) 02-to (3) 03-Nu (4) 04-Si
# ------------------------------------------------------------------------
# rules.ZeroR '' 4805554146(100)   55.56 |   55.56     55.56     55.56
# rules.OneR '-B 6' -345942(100)   71.63 |   73.07     71.63     51.41 *
# rules.OneR '-B 15' -34594(100)   71.63 |   73.07     71.63     60.81 *
# trees.J48 '-C 0.25 -M 2' (100)   76.41 |   80.96 v   76.44     65.44 *
# trees.J48 '-U -M 2' -2177(100)   75.81 |   81.07 v   75.93     65.22 *
# trees.J48 '-C 0.25 -M 15'(100)   73.81 |   73.00     73.81     63.07 *
# trees.J48 '-C 0.25 -M 35'(100)   72.44 |   73.00     72.44     64.63 *
# trees.J48 '-C 0.025 -M 10(100)   73.89 |   76.52     73.89     63.22 *
# trees.RandomForest '-P 10(100)   81.33 |   79.33     82.22     64.96 *
# trees.RandomTree '-K 0 -M(100)   73.00 |   74.59     73.89     62.48 *
# lazy.IBk '-K 10 -W 0 -A \(100)   80.85 |   80.89     80.93     64.22 *
# lazy.IBk '-K 35 -W 0 -A \(100)   83.67 |   83.37     83.30     67.26 *
# lazy.IBk '-K 35 -W 0 -A \(100)   83.44 |   83.19     83.41     67.04 *
# functions.SMO '-C 1.0 -L (100)   84.07 |   83.70     84.04     62.67 *
# functions.Logistic '-R 1.(100)   83.96 |   83.81     83.85     66.04 *
# bayes.NaiveBayes '' 59952(100)   84.89 |   83.44     84.15     68.26 *
# meta.ClassificationViaClu(100)   78.07 |   80.81     78.15     62.11 *
# ------------------------------------------------------------------------
#                                (v/ /*) |  (2/15/0)  (0/17/0)  (0/1/16)
```

NaiveBayes has the highest accuracy in the "clean" dataset. Logistic scores the highest in the "top5" dataset.

We can also look at mean accuracy in the four datasets.

```
Results.4.datasets <- weka.results %>%
 group_by(Key_Scheme, Key_Scheme_options) %>%
  summarise(mean = mean(Percent_correct), .groups = "keep")

pander(head(Results[c(2, 4)]))
```

Key_Scheme	mean
weka.classifiers.bayes.NaiveBayes	84.89
weka.classifiers.functions.Logistic	83.96
weka.classifiers.functions.SMO	84.07
weka.classifiers.lazy.IBk	80.85
weka.classifiers.lazy.IBk	83.67
weka.classifiers.lazy.IBk	83.44

We can conclude that:

- NaiveBayes scores the highest
- Logistic scores seccond highest

5.2.2 Area under ROC

The results of the area under ROC comparison field are showed in the table below.

```
# Dataset                    (1) 01-Clea | (2) 02-t (3) 03-N (4) 04-S
# ------------------------------------------------------------------
# rules.ZeroR '' 4805554146(100)   0.50 |   0.50     0.50     0.50
# rules.OneR '-B 6' -345942(100)   0.71 |   0.73     0.71     0.50 *
# rules.OneR '-B 15' -34594(100)   0.71 |   0.73     0.71     0.60 *
# trees.J48 '-C 0.25 -M 2' (100)   0.76 |   0.83 v   0.76     0.70
# trees.J48 '-U -M 2' -2177(100)   0.76 |   0.82     0.76     0.70
# trees.J48 '-C 0.25 -M 15'(100)   0.78 |   0.78     0.78     0.68 *
# trees.J48 '-C 0.25 -M 35'(100)   0.74 |   0.74     0.74     0.68
# trees.J48 '-C 0.025 -M 10(100)   0.76 |   0.79     0.76     0.66 *
# trees.RandomForest '-P 10(100)   0.89 |   0.84 *   0.89     0.71 *
# trees.RandomTree '-K 0 -M(100)   0.73 |   0.75     0.74     0.62 *
# lazy.IBk '-K 10 -W 0 -A \(100)   0.89 |   0.88     0.90     0.71 *
# lazy.IBk '-K 35 -W 0 -A \(100)   0.91 |   0.89     0.90     0.73 *
# lazy.IBk '-K 35 -W 0 -A \(100)   0.90 |   0.89     0.90     0.74 *
# functions.SMO '-C 1.0 -L (100)   0.83 |   0.83     0.83     0.63 *
# functions.Logistic '-R 1.(100)   0.90 |   0.90     0.90     0.72 *
# bayes.NaiveBayes '' 59952(100)   0.91 |   0.90     0.91     0.72 *
# meta.ClassificationViaClu(100)   0.78 |   0.80     0.78     0.64 *
# ------------------------------------------------------------------
#                                (v/ /*) | (1/15/1) (0/17/0) (0/4/13)
```

RandomTree, K-nearest neighbours (KNN 35, euclideanDistance) and NaieveBayes are the highest here. There are very close to eachother, but NaiveBayes is at top followed by K-nearest neighbours.

- NaieveBayes
- K-neares neighbours (-K 35, EuclideanDistance)
- RandomTree

5.2.3 False negative rate and False positive rate

We want a concept with a low false negative rate and a low false positive rate. Below we first show a table with the false negative rate.

```
# Dataset                    (1) 01-Clea | (2) 02-t (3) 03-N (4) 04-S
# ------------------------------------------------------------------
# rules.ZeroR '' 4805554146(100)   1.00 |   1.00     1.00     1.00
# rules.OneR '-B 6' -345942(100)   0.32 |   0.30     0.32     0.60 v
# rules.OneR '-B 15' -34594(100)   0.32 |   0.30     0.32     0.47 v
# trees.J48 '-C 0.25 -M 2' (100)   0.30 |   0.25     0.30     0.46 v
# trees.J48 '-U -M 2' -2177(100)   0.29 |   0.24     0.28     0.47 v
# trees.J48 '-C 0.25 -M 15'(100)   0.30 |   0.34     0.30     0.50 v
# trees.J48 '-C 0.25 -M 35'(100)   0.29 |   0.26     0.29     0.52 v
# trees.J48 '-C 0.025 -M 10(100)   0.26 |   0.24     0.26     0.50 v
# trees.RandomForest '-P 10(100)   0.25 |   0.27     0.23     0.39 v
# trees.RandomTree '-K 0 -M(100)   0.33 |   0.31     0.30     0.41
# lazy.IBk '-K 10 -W 0 -A \(100)   0.20 |   0.21     0.20     0.33 v
# lazy.IBk '-K 35 -W 0 -A \(100)   0.22 |   0.25     0.22     0.43 v
# lazy.IBk '-K 35 -W 0 -A \(100)   0.23 |   0.26     0.23     0.44 v
# functions.SMO '-C 1.0 -L (100)   0.23 |   0.23     0.23     0.30
# functions.Logistic '-R 1.(100)   0.22 |   0.23     0.22     0.38 v
# bayes.NaiveBayes '' 59952(100)   0.19 |   0.22     0.21     0.38 v
# meta.ClassificationViaClu(100)   0.24 |   0.26     0.23     0.24
# ------------------------------------------------------------------
#                                (v/ /*) | (0/17/0) (0/17/0) (13/4/0)

# Key:
# (1) 01-Clean_dataset
# (2) 02-top5_p_values
# (3) 03-Numeric_transformed
# (4) 04-Simple
```

NaieveBayes has the lowest false negative rate. This score has been achieved in the clean dataset. K-nearest neighbours (IBK) has a solid score as well. It has the lowest FNR in the second (top5) and third (numeric transformed) dataset.

- NaieveBayes scores the best
- K-nearest neighbours also scores well (-K 10, EuclideanDistance)

The false positive rate is now discussed below, first the results will be showed.

```
# Dataset                    (1) 01-Clea | (2) 02-t (3) 03-N (4) 04-S
# ------------------------------------------------------------------
# rules.ZeroR '' 4805554146(100)   0.00 |   0.00     0.00     0.00
# rules.OneR '-B 6' -345942(100)   0.26 |   0.24     0.26     0.40 v
# rules.OneR '-B 15' -34594(100)   0.26 |   0.24     0.26     0.33
# trees.J48 '-C 0.25 -M 2' (100)   0.18 |   0.14     0.18     0.25
# trees.J48 '-U -M 2' -2177(100)   0.21 |   0.15     0.21     0.25
# trees.J48 '-C 0.25 -M 15'(100)   0.23 |   0.21     0.23     0.26
# trees.J48 '-C 0.25 -M 35'(100)   0.26 |   0.28     0.26     0.22
# trees.J48 '-C 0.025 -M 10(100)   0.26 |   0.23     0.26     0.26
# trees.RandomForest '-P 10(100)   0.14 |   0.16     0.13     0.32 v
```

```
# trees.RandomTree '-K 0 -M(100)   0.22 |   0.21     0.23     0.35 v
# lazy.IBk '-K 10 -W 0 -A \(100)   0.18 |   0.17     0.18     0.38 v
# lazy.IBk '-K 35 -W 0 -A \(100)   0.12 |   0.10     0.12     0.24 v
# lazy.IBk '-K 35 -W 0 -A \(100)   0.11 |   0.10     0.11     0.24 v
# functions.SMO '-C 1.0 -L (100)   0.11 |   0.11     0.10     0.43 v
# functions.Logistic '-R 1.(100)   0.11 |   0.11     0.11     0.30 v
# bayes.NaiveBayes '' 59952(100)   0.12 |   0.12     0.12     0.27 v
# meta.ClassificationViaClu(100)   0.21 |   0.14     0.21     0.49 v
# ------------------------------------------------------------------
#                                (v/ /*) | (0/17/0) (0/17/0) (10/7/0)
```

For the false positive rate, zeroR scores best. This makes sense, because Zero always choose the attribute with the highest number of occurance. Not looking at ZeroR, K-nearest neighbours and SMO scores the lowest. In the simple dataset, a J48 concept scores the lowest.

Overall findings:

- K-nearest neighbours (-K 35, ManhattanDistance)
- SMO
- Logitisc
- NaiveBayes

5.2.4 UserCPU_CPU_Time_Training.

The UserCPU_CPU_Time_Training for each model is very fast. ZeroR scores the fastest here. But this model will not not be used eventually.

5.2.5 F messure

F-score is quite similar to accuracy, but it is calculated in a different way. For that reason we also look at the F-score.

```
# Dataset                    (1) 01-Clea | (2) 02-t (3) 03-N (4) 04-S
# ------------------------------------------------------------------
# rules.ZeroR '' 4805554146  (0)         |
# rules.OneR '-B 6' -345942(100)   0.68 |   0.69     0.68     0.42 *
# rules.OneR '-B 15' -34594(100)   0.68 |   0.69     0.68     0.53 *
# trees.J48 '-C 0.25 -M 2' (100)   0.72 |   0.78 v   0.72     0.57 *
# trees.J48 '-U -M 2' -2177(100)   0.72 |   0.78     0.72     0.57 *
# trees.J48 '-C 0.25 -M 15'(100)   0.70 |   0.68     0.70     0.53 *
# trees.J48 '-C 0.25 -M 35'(100)   0.69 |   0.71     0.69     0.54 *
# trees.J48 '-C 0.025 -M 10(100)   0.71 |   0.74     0.71     0.54 *
# trees.RandomForest '-P 10(100)   0.78 |   0.76     0.79     0.60 *
# trees.RandomTree '-K 0 -M(100)   0.69 |   0.71     0.70     0.58 *
# lazy.IBk '-K 10 -W 0 -A \(100)   0.79 |   0.78     0.79     0.62 *
# lazy.IBk '-K 35 -W 0 -A \(100)   0.81 |   0.80     0.80     0.60 *
# lazy.IBk '-K 35 -W 0 -A \(100)   0.80 |   0.79     0.80     0.60 *
# functions.SMO '-C 1.0 -L (100)   0.81 |   0.81     0.81     0.63 *
# functions.Logistic '-R 1.(100)   0.81 |   0.81     0.81     0.61 *
# bayes.NaiveBayes '' 59952(100)   0.83 |   0.81     0.82     0.63 *
# meta.ClassificationViaClu(100)   0.75 |   0.77     0.75     0.64 *
# ------------------------------------------------------------------
#                               (v/ /*) | (1/15/0) (0/16/0) (0/0/16)
```

NaiveBayes scores the highest. This is accomplished in the clean dataset. In the second dataset (top5), SMO and Logistic, scores as high as NaiveBayes and in the third dataset (numeric transformed), NaiveBayes scores the highest again.

Overall:

- NaiveBayes
- Logistic
- SMO

5.2.6 Cost sensitive classifier

There is also a model where the cost can be adjusted. We will take a closer look. 'Cost' means that more errors are charged for instances that are done wrong, which are punished more severely. The cost can be customized.

Imagine a concept where 50 people are classified as absence but are presence in fact. This means that people with presence are being overlooked. Therefore, we want a concept with a low False positive.

In the Weka explorer, under functions there is a classifier "CostSensitiveClassifier". After selecting this classifier, open the options and resize the scoring matrix. After resizing, select a score of 100 for the false positives and 10 for the false negative. Select Logistic as classifier and press start (use cross-validations fold 10).

First, let's look at Logistic with default cost:

Table 7: Confusion matrix of default logistic

a	b	<– classified as
95	25	a = Presence
16	134	b = Absence

And now, with the costSensitiveClassifier:

Table 8: Confusion matrix of CostSensitiveClassifier with Logistic

a	b	<– classified as
114	6	a = Presence
76	74	b = Absence

The above accuracy is 69.6296%. That is significantly lower. The number of false positives is a lot less, but the false negatives are a lot higher. Due to the low accuracy and mistakes, we will not consider this for further research.

Let's run again, but than with NaiveBayes as classifier. Add the same costst as above.

Table 9: Confusion matrix of CostSensitiveClassifier with Naive-Bayes

a	b	<– classified as
106	14	a = Presence
46	04	b = Absence

Again, same story as above. The false positives are lower, but the false negatives are higher. The accuracy is actually a bit higher with 77.7778%, but it's still far from the 85+ we achieved earlier.

5.2.7 Conclusions on best concepts

We've now looked at several comparison fields. 5 to be precies. There are a few concepts that keep coming out here. Looking at the overall:

- NaiveBayes : 5
- Logistic : 3
- SMO : 2
- K-nearest neighbours : 3

NaiveBayes came out highest. It is good to mention K-nearest neighbours, but a different concept (different hyperparameters) always emerged as the best. Logistics also scores well. It's difficult to say, this model is really the best. Based on these results, we will take a closer look at NaiveBayes and Logistic.

5.3 Attribute Selection

Now that the best models are known, it is still important to look at the best attributes.

We have already looked at 4 different data sets. Two of the datasets were modified by removing attributes. In weka it is possible to see which attributes are most suitable for your model, instead of just removing instances and seeing if the accuracy has increased. This can be done on multiple ways. We will look at three posible ways:

1. Single attribute evaluation
    1. Correlation
    2. Information Gain
    3. Gain ratio
    4. Accuracy
2. Attribute subset evaluation
    1. CfsSubsetEval
    2. Wrappr (model dependancy)

5.3.1 Single attribute evaluation

Each attribute is going to be compared with the classification label. This will result in a list of the best attributes. We show 4 measures of ranking.

In Weka, open the explorer and open the "clean" dataset (We use the clean dataset, since all attributes are still present here and the labels are the best readable here). Press "attributeSelection". You can now choose a Attribute Evaluator and search method. It may happen that when selecting a specific Attribute Evaluator, a warning message appears. This is because some evaluators only work with a specific search method. therefore press yes.

After selecting a Attribute Evaluator, leave the rest of the settings unchanged en press start. You now see a list of attributes with percentages. We will combine the results in table below.

Table 10: Single attribute evaluation results

Attribute	Accuracy (oneR)	Correlation	Gain ratio	Information Gain
Thallium	76.2963	0.4888	0.171204	0.208556
Chest.pain.type	75.1852	0.3724	0.111511	0.192202
Number.of.vessels.fluro	74.0741	0.4553	0.17015	0.165916
Exercise.angina	71.4815	0.4193	0.142054	0.129915
ST.depression	70	0.418	0.148019	0.119648
Slope.of.ST	68.8889	0.35	0.086432	0.111153
Max.HR	62.2222	0.4185	0.123102	0.12028
Sex	61.8519	0.2977	0.073773	0.066896
Age	59.6296	0.2123	0.056747	0.056726
EKG.results	58.8889	0.1801	0.022886	0.024152
Cholesterol	53.7037	0.1315	0	0
BP	53.7037	0.1554	0	0
FBS.over.120	53.3333	0.0163	0.000318	0.000193

Just as the t-test in the EDA, the Thallium attribute is highest ranked with 76.29%. It has the highest correlation as well. It also has the highest gain ratio and information gain.

Each evaluator also produces a list of the best selected attributes. Each evaluator in the single attribute evaluator comes with the same list, which contains each attribute.

5.3.2 Attribute subset evaluation

In the Attribute subset evalutation, multiple instances are compared with each other. Where the Single attribute evaluation only looked at one instance at a time, this one looks at them all at once.

For each evaluator, we use exhaustive search. Exhaustive search calculates all possibilities and gets the best one. This may take a long time, but it will produce the best results. Larger datasets are not suitable for exhaustive search. In this case it is better to use BestFirst.

With the Attribute subset evaluation, only a subset is produced and not a list of numbers. The following subsets arise:

> CfsSubsetEval: Chest.pain.type, EKG.results, Max.HR, Exercise.angina, ST.depression, Number.of.vessels.fluro and Thallium
>
> Wrapr + NaieveBayes: Sex, Chest.pain.type, BP, Cholesterol, EKG.results, Max.HR, Exercise.angina, ST.depression, Slope.of.ST, Number.of.vessels.fluro, Thallium
>
> Wrapr + Logitic: Age, Sex, Chest.pain.type, BP, FBS.over.120, Max.HR, Exercise.angina, ST.depression, Slope.of.ST, Number.of.vessels.fluro, Thallium

If the attribute selection is done on the "original" file, virtually the same subsets emerge. In the single attribute evaluation, exactly the same subsets appear and in the attribute subset evaluation, there is only a difference in 1 attribute between Wrapr(NaiveBayes) and Wrapr(logistic).

5.3.3 Subset Experimenter

We will now make the subsets that are defined above. These subsets will again be checked in Experimenter to see which concept scores the highest on which datasets. In the Weka Explorer, you can click the check marks and then press remove. The selected attribute will be removed and you can save the new dataset by pressing save in the top right.

The next filenames will be used:

- Set 1 ALl attributes
- Set 2 CfsSubsetEval: Chest.pain.type, EKG.results, Max.HR, Exercise.angina, ST.depression, Number.of.vessels.fluro and Thallium
- Set 3 Wrapr + NaieveBayes:Sex, Chest.pain.type, BP, Cholesterol, EKG.results, Max.HR, Exercise.angina, ST.depression, Slope.of.ST, Number.of.vessels.fluro, Thallium
- Set 4 Wrapr + Logitic: Age, Sex, Chest.pain.type, BP, FBS.over.120, Max.HR, Exercise.angina, ST.depression, Slope.of.ST, Number.of.vessels.fluro, Thallium

In weka, open a new experimenter and add these files. Add NaiveBayes and Logistic and run the new experiment. All the results are combined in a table.

Set	Set 1	Set 2	Set 3	Set 4
NaiveBayes Percent_correct	84.89	85.07	85.52	84.26
Logistic Percent_correct	83.96	83.44	83.78	86.07
NaiveBayes Area_under_ROC	0.91	0.90	0.91	0.91
Logistic Area_under_ROC	0.90	0.89	0.90	0.91
NaiveBayes False_positive_rate	0.12	0.10	0.11	0.13
Logistic False_positive_rate	0.11	0.12	0.11	0.09
NaiveBayes False_negative_rate	0.19	0.21	0.19	0.19
Logistic False_negative_rate	0.22	0.23	0.23	0.19

The highest accuracy is achieved in the 4th set by Logistic. The Area under ROC is almost identical, but looking at average, set 4 scores the best. In false positive rate, set 4 scores best again with Logitic. False negative rate is again, on average done best in set 4.

It's difficult to choose a concept and say, this is the best. But looking at the results, logistic seems to be the best here in combination with set 4. This is because the highest accuracy is achieved, on average the highest area_under_ROC, the lowest false positive rate and on average the lowest false negative rate.

5.4 Meta learners

Now we know our best concept(s) and attributes. Let's look at meta-classifiers. There are methods to classify multiple concepts together. Or in a specific way.

Three learners will be looked at:

1. Voting
2. Stacking
3. Bagging

5.4.1 Voting

This method allows different concepts to vote. In the weka explorer, under the tab "meta", there is an option "Vote". Of course you choose the dataset with the best attributes. In the option of Vote, it's possible to add concepts. Add the best concepts that are found above (NaiveBayes and Logistic).

This method achieves an accuracy of 86.2963%. This is higher than our original Logistic. The confusion matrix is almost exactly the same. The Area under AOC is also the same.

5.4.2 Stacking

This methods "stacks" the concepts and creates a deciding attribute. Again, this method can be found under "meta". Add the best concepts. You can choose a meta Classifier that decides. We will run twice, with Logistic as meta Classiefier and NaieveBayes.

Stacking with Logistic as meta classifier achieves an accuracy of 86.6667%, which again is higher. Stacking with NaiveBayes achieves an accuracy of 87.037%! The False Positve rate is also lower, 0.16 vs 0.19. False negative rate is the same. The Area under ROC is 0.8983. That is a small fraction lower, but certainly not a lot lower.

Stacking with NaiveBayes as a meta classifier is therefore a lot better!

5.4.3 Bagging

This last method uses multiple classifiers that are exactly the same. The data that comes in, is being changed. Again, under meta you can find bagging. Choose logistic.

This method achieves an accuracy of 85.1852% with Logistic as classifer and 83.7037% with NaiveBayes. Area under AOC is 0.90 for both runs.

We have now looked at three meta-learners. In the experimenter we also looked at randomForest, which is also a form of a meta learner.

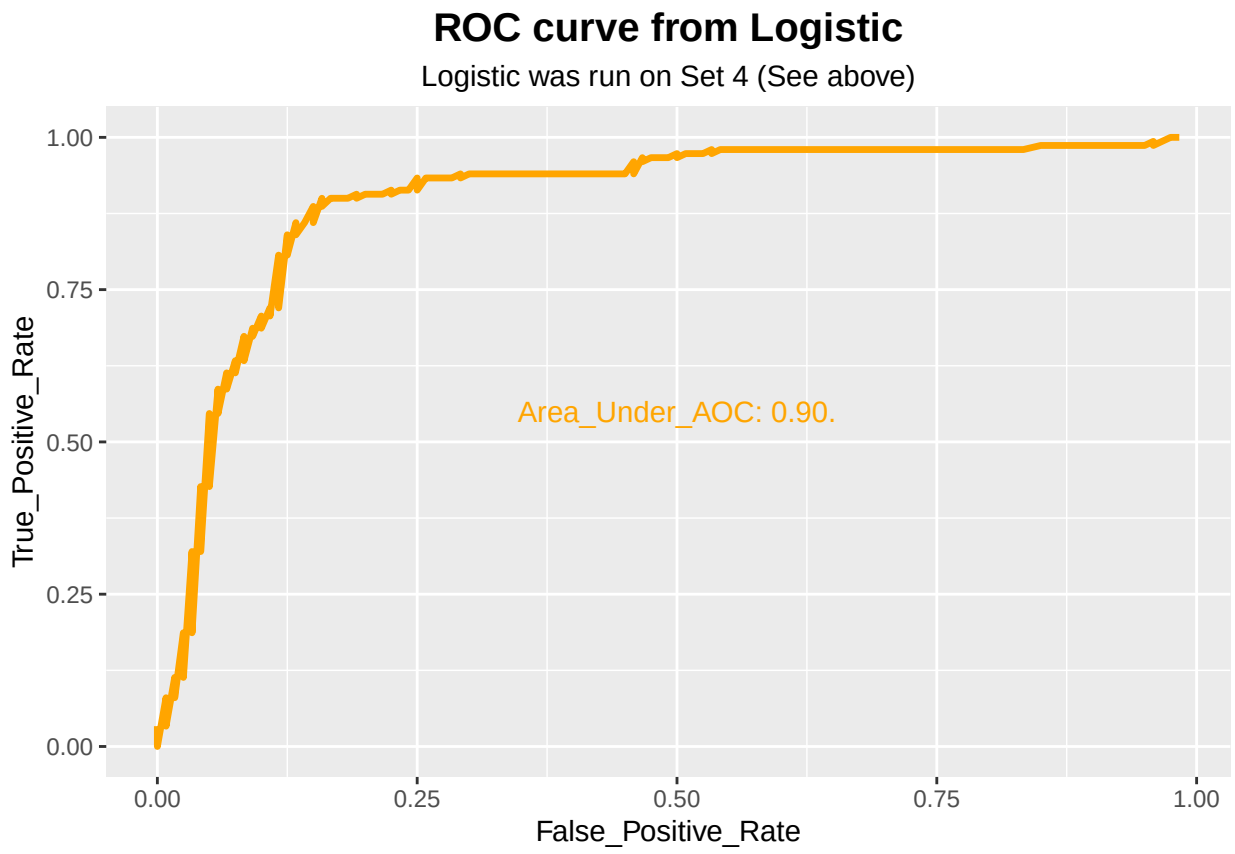
It is now clear that stacking with Logistic and NaiveBayes with NaiveBayes as the meta-classifier achieves the highest accuracy and lowest false positive rate for us.

5.5 ROC curve of model

We will plot the ROC curve of our best model and dataset.

After you've run Stacking with the above settings on set 4 in the Weka explorer, you can right-click on the result (in result list). Press the option: "visualize treshold curve" and then "absence". You can now store the data to make a ROC curve in R. Press save and enter a destination.

```
ROC.data <- data.frame(read.csv("ROCCurve.arff", skip = 18))
colnames(ROC.data) <- c(
  "Instance_number","True_Positives", "False_Negatives", "False_Positives",
  "True_Negatives", "False_Positive_Rate",  "True_Positive_Rate", "Precision",
  "Recall", "Fallout", "FMeasure", "Sample_Size", "Lift", "Threshold")

ggplot(ROC.data, aes(x = False_Positive_Rate, y = True_Positive_Rate)) +
  geom_line(color = "orange", size = 1.2) +
  labs(title = "ROC curve from Logistic",
       subtitle = "Logistic was run on Set 4 (See above)") +
  theme(legend.position = "bottom", plot.title = element_text(face = "bold",
      size = 15, hjust = 0.5), plot.subtitle = element_text(hjust = 0.5)) +
  annotate("text", x=0.50, y=0.55, label= "Area_Under_AOC: 0.90.", color = "orange")
```



The curve has a high deflection to the upper left, which is good for a roc curve. The area under the curve is 0.8983.

6 Java Wrapper

We have now discovered the best concept with the best attributes. We are now going to make our model public by building a Java application around it. This will be done in another repository: https://github.com/MarkStreek/PredictingHeartDiseaseJavaWrapper

References

[1] (2023, 12 januari).: *Predicting heart disease using clinical variables.* <https://www.kaggle.com/datasets/thedevastator/predicting-heart-disease-risk-using-clinical-var>

[2] (2022, 25 augustus).: *Heart disease - symptoms and causes - Mayo Clinic.* Mayo Clinic. <https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118>

[3] Harvard Health. (2021, 21 september).: *Angina: Symptoms, diagnosis and Treatments.* <https://www.health.harvard.edu/heart-health/angina-symptoms-diagnosis-and-treatments>

[4] Perret-Guillaume, C., Joly, L. Bénétos, A. (2009).: *Heart rate as a risk factor for cardiovascular disease. Progress in Cardiovascular Diseases, 52(1), 6–10*