Exploratory Data Analysis (for) Stroke Data Set

Pascal Visser

2022-09-14

Contents

1. Intro **2**

2. Dataset **2**

3 Dataset modifying **4**

4. Exploratory Data analysis **7**

4.1 Summary 7

4.2 Distribution of classifier 8

4.3 Age and gender distribution 9

4.4 health related variables 9

4.4.1 Smoke 9

4.4.2 BMI and Glucose levels 10

4.4.3 Cardiovascular diseases 12

4.5 Lifestyle varibles 12

4.5.1 Marriage status 13

4.5.2 Type of work 14

4.5.2 Residence type 14

5. Comparing variables **15**

5.1 health **16**

5.2 lifestyle 18

6. EDA conclusion **20**

7. Machine learning **20**
7.1 Dataset changing . . . 21
7.1.1 Methods . . . 21
7.1.2 SMOTE . . . 22
7.2 Testing algorithms . . . 23
7.3 Choosing Alogrithm . . . 25
7.3.1 Baseline . . . 25
7.3.2 Advanced algorithms . . . 26

Loading libraries

```
# load libraries
library(tidyverse)
library(naniar)
library(readr)
library(ggplot2)
library(lemon)
library(knitr)
library(pander)
library(cowplot)
library(dplyr)
library(DMwR)
```

1. Intro

A stroke is a life-threatening medical condition, where there is no or poor blood flow to (parts of) the brain. This causes parts of the brain to stop functioning and die off in minutes time. A stroke or brain attack is labelled as the second leading cause of death worldwide and is responsible for an annual mortality of about 5.5 million.

This dataset consists of people that had a stroke and people that have not. The variables are values that indicate their lifestyle, health and environment factors. The goal of this research is to calculate the chance of a stroke based of their lifestyle. This can be accomplished by machine learning. This gives the following research question:

Is it possible to produce an accurate algorithm with machine learning, to calculate the risk of a stroke based on lifestyle variables of people with stroke history and people that never had a stroke?

2. Dataset

The data is as said, lifestyle information of people with stroke history or not. In this section the data will be talked trough and the variables explained. First the data is loaded in:

```
# load in the data
strokedata <- read.csv("Data/Stroke_dataset.csv", header = T)
kable(strokedata[1:6, 1:6], format = 'simple', caption = "Head of data")
```

Table 1: Head of data

id	gender	age	hypertension	heart_disease	ever_married
9046	Male	67	0	1	Yes
51676	Female	61	0	0	Yes
31112	Male	80	0	1	Yes
60182	Female	49	0	0	Yes
1665	Female	79	1	0	Yes
56669	Male	81	0	0	Yes

```
cat("The dataset is", dim(strokedata)[1], "rows long, and has", dim(strokedata)[2], "columns")
```

```
## The dataset is 5110 rows long, and has 12 columns
```

Above, the first six rows of the data are shown. Also is given how many records there are. The codebook will clarify the variables and their values. The codebook will be loaded in:

```
# load codebook
codebook <- read.table("Data/codebook.txt", header = T, sep = ';')
kable(codebook, format = 'pipe', caption = "Codebook of stroke data")
```

Table 2: Codebook of stroke data

Column	Description	value	datatype
id	unique identifier of patient	number	int
gender	male of female	sex	chr
age	age of patient	number	dbl
hypertension	suffering from hypertension?	binairy	int
heart_disease	suffering from heart disease?	binary	int
ever_married	married or married in the past?	binary	chr
work_type	sort of work	string	chr
Residence_type	type of residence	string	chr
avg_glucose_level	average level of glucose	number	dbl
bmi	body mass index	number	chr
smoking_status	smoking history	string	chr
stroke	stroke history	binary	int

The codebook gives a detailed description the columns content about the value/datatypes. This is necessary to understand the data.

Now that the data is loaded in, there can be looked at the negatives of the data. The biggest flaws are the datatypes of the columns that are not equal. For example: the columns age is the type double instead of integer. The column BMI is character, while a double datatype is much more logical. Also, the columns hypertension and hearth disease are a yes or no question, in these columns the yes or no is represented by 0 or 1. The columns of marriage history is a yes or no question too, but here yes or no is represented by text. This also needs modifying.

The character variables are can be converted to factor classes.

Additionally, it's good to know how many NA's are present in the dataset. These missing values can screw the data. Also, values like 'unknown' can be declared as a missing value

```r
# check unique values of character values

cat("Gender:", unique(strokedata$gender),"\n",
    "Married:", unique(strokedata$ever_married), "\n",
    "Work type:", unique(strokedata$work_type), "\n",
    "Residence type:", unique(strokedata$Residence_type), "\n",
    "Smoking:", unique(strokedata$smoking_status))
```

```
## Gender: Male Female Other
##  Married: Yes No
##  Work type: Private Self-employed Govt_job children Never_worked
##  Residence type: Urban Rural
##  Smoking: formerly smoked never smoked smokes Unknown
```

Above reveals the Smoking status variable has a value "Unknown". This need to be addressed in the modifying section. Also, the column Gender has one other record. For the N/A the scan function of the 'naniar' package is used.

```r
# scan for missing values
nas <- miss_scan_count(data = strokedata, search = list("N/A", "Unknown"))
kable(nas, format = 'pipe', caption = "Number of missing values per variable")
```

Table 3: Number of missing values per variable

Variable	n
id	0
gender	0
age	0
hypertension	0
heart_disease	0
ever_married	0
work_type	0
Residence_type	0
avg_glucose_level	0
bmi	201
smoking_status	1544
stroke	0

The column BMI is responsible for 201 missing values, these missing values are likely responsible for the column to be parsed as character datatype. the column smoking_status has 1544 missing values. 1544 values is a big number and need to be dealt with in the modifying.

3 Dataset modifying

As talked in the previous section, some column types makes no sense, these types need to be changed to more sensible types. The columns that will be converted are:

- age, dbl -> int
- bmi, chr -> dbl

And additionally, the contents of the column ever_married will be converted to binary, and the smoking_status unknown will be cleared.

First the column age will be cast to integer, because a double type with decimals makes no sense in age, secondly, the column bmi will be cast to double, to keep the decimal precision:

```
# cast columns to another type
strokedata <- transform(strokedata, age = as.integer(age))
strokedata <- transform(strokedata, bmi = as.double(bmi))
```

Now the ever_married column need to be changed to binary and a integer datatype:

```
# replace yes and no by 1 or 0
strokedata$ever_married[strokedata$ever_married == "Yes"] <- "1"
strokedata$ever_married[strokedata$ever_married == "No"] <- "0"
strokedata <- transform(strokedata, ever_married = as.integer(ever_married))
```

The character classes will be cast to factor:

```
strokedata$Residence_type <- as.factor(strokedata$Residence_type)
strokedata$gender <- as.factor(strokedata$gender)
strokedata$work_type <- as.factor(strokedata$work_type)
```

The record with gender 'Other' will be replaced with an gender value

```
# seek the most present value inside the column
table(strokedata$gender)
```

```
##
## Female   Male  Other
##   2994   2115      1
```

```
# remove record where gender is other
strokedata$gender[strokedata$gender == 'Other'] <- "Male"
```

It has been chosen to replace the other value with male, because of the balance inside the variable.

Now what is left are the missing values, BMI has N/A and smoking status has "unknown". For BMI the Na's will be replaced with the mean of the column:

```
# replace the missing values with the column mean
strokedata$bmi[is.na(strokedata$bmi)] <- mean(strokedata$bmi, na.rm = T)

# keep the digit precision as before
strokedata$bmi <- round(strokedata$bmi, digits = 1)
```

The smoking status 'unknown' is harder to replace because the variables are character, first we need to know how many records there are of the column:

```
# Count the unique variables in the gender column
table(strokedata$smoking_status)
```

```
##
## formerly smoked    never smoked          smokes         Unknown
##             885            1892             789            1544
```

Their are 885, 1892 and 789 records of not unknowns. Based on these numbers, we can calculate the probability of occurring in the smoking status variable.

```
# Calculate the probability of formerly smoker, current smokers and non-smokers given that there's only

#prop.table?
prob.Formerly <- 885 / (885 + 1892 + 789)
prob.Never <- 1892 / (885 + 1892 + 789)
prob.Smoke <- 789 / (885 + 1892 + 789)
```

With the probabilities, we can replace the unknowns based on there weight in the column.

```
# Replacing 'Unknown' in smoking_status by the other 3 variables according to their weightage
strokedata$rand <- runif(nrow(strokedata))

strokedata <- strokedata%>%mutate(Probability = ifelse(rand <= prob.Formerly, "formerly smoked", ifelse

strokedata <- strokedata%>%mutate(smoking.status = ifelse(smoking_status == "Unknown", Probability, smo

# View the new Smoking Status column's unique values and their counts
table(strokedata$smoking.status)
```

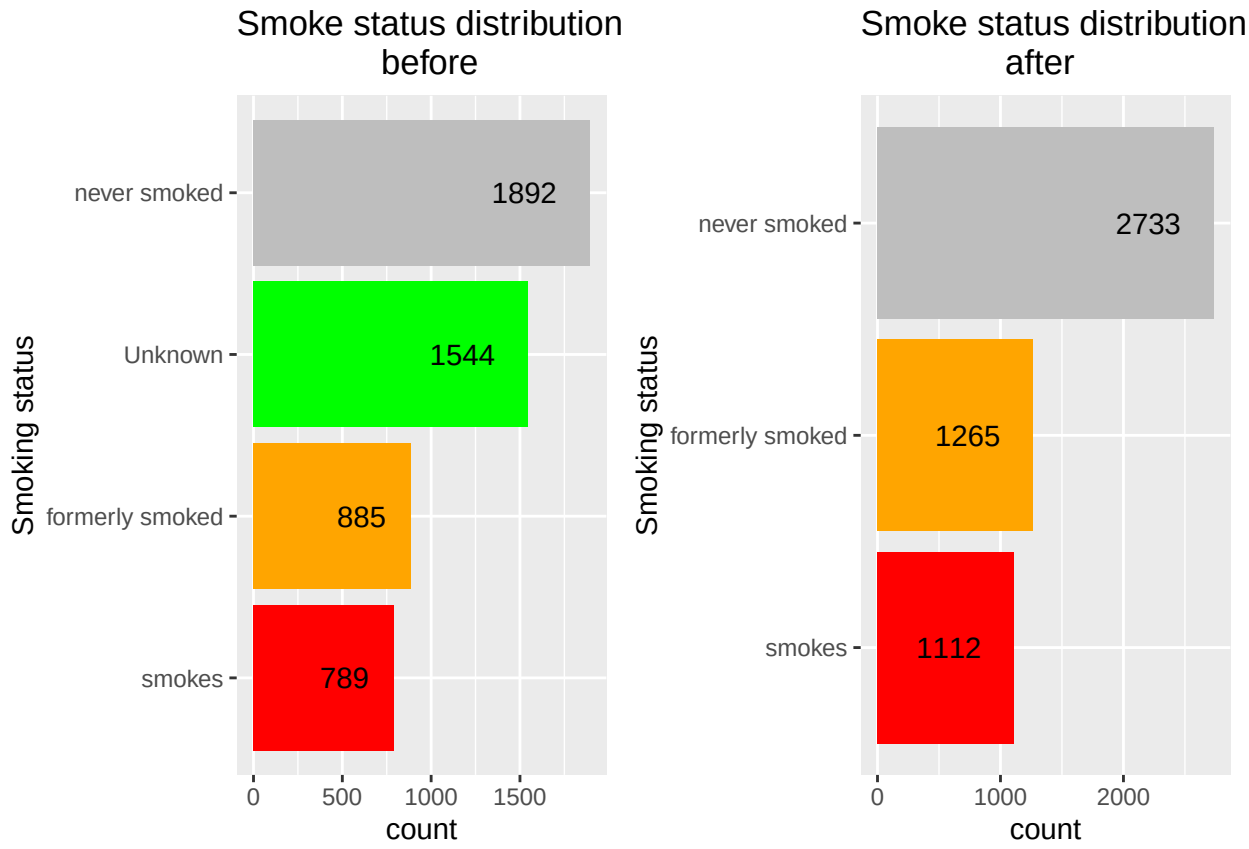
```
##
## formerly smoked    never smoked          smokes
##            1265            2733            1112
```

```
# Remove columns that are not needed
health <- subset(strokedata, select = -c(rand,Probability,smoking_status))

# revise the column name of smoking status
colnames(health)[12] <- "smoking_status"

# 'health' is the final modified dataset which will be used for the EDA section below.
```

Now we have a column without the unknowns, but with increased other variables based on there weight. Which is shown in the plots below:



In the plots, the effect is shown of replacing the unknown value by other values based on weight. With this method, the distribution of values is not affected.

4. Exploratory Data analysis

In this part, the modified data is visualised into useful graphs. These graphs will tell about the distribution of variables and the relations to each other.

4.1 Summary

First the summary of the values numeric values, we have three columns with important numeric values, age, glucose level and BMI. The summary of these variables tells about the distribution of values and possible outliers

```
#summary of subset data
pander(summary(health[c(3,9:10)],), caption = "Summary of numeric values")
```

Table 4: Summary of numeric values

age	avg_glucose_level	bmi
Min. : 0.00	Min. : 55.12	Min. :10.30
1st Qu.:25.00	1st Qu.: 77.25	1st Qu.:23.80

age	avg_glucose_level	bmi
Median :45.00	Median : 91.89	Median :28.40
Mean :43.22	Mean :106.15	Mean :28.89
3rd Qu.:61.00	3rd Qu.:114.09	3rd Qu.:32.80
Max. :82.00	Max. :271.74	Max. :97.60

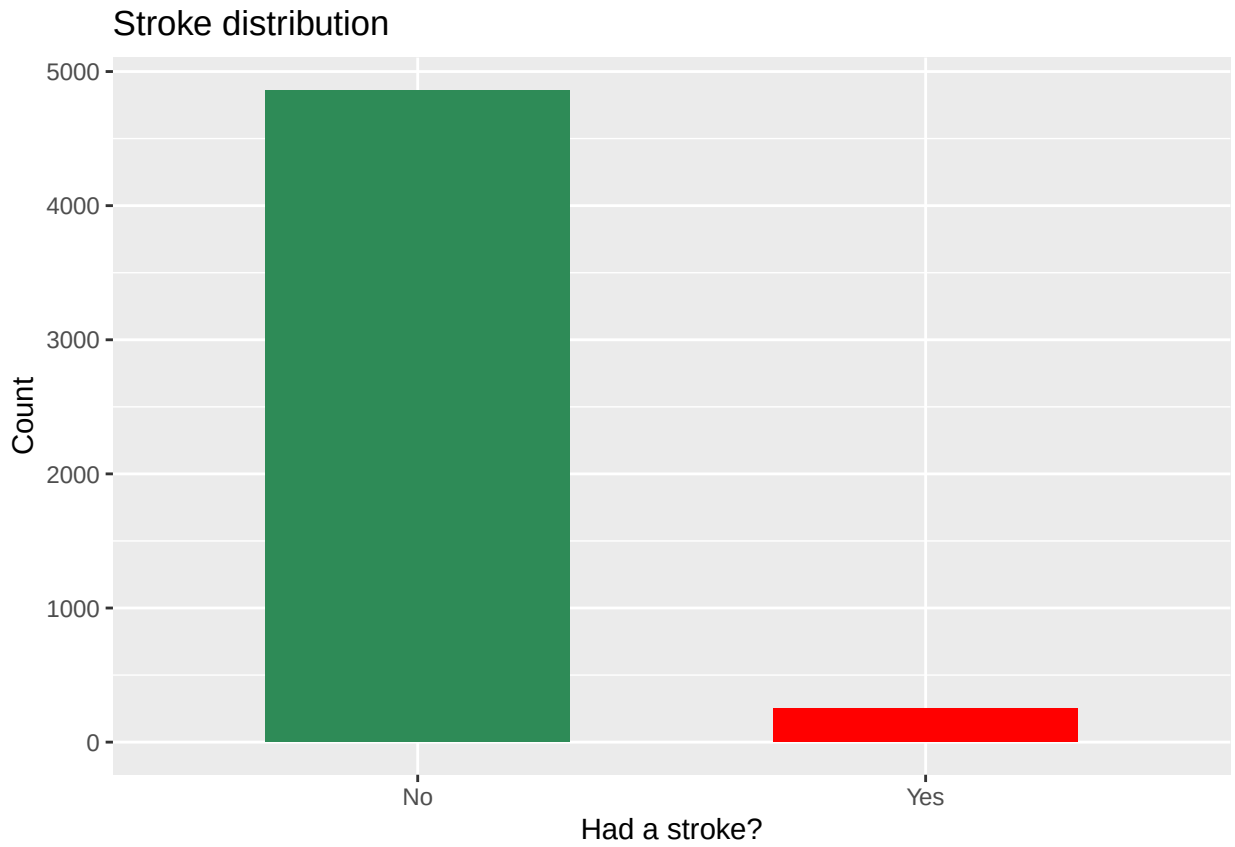
Looking at age, the minimum value is 0, which is a bit odd. Maybe this is an unknown value. Glucose looks normal, but BMI has a very low minimum, which is also odd.

4.2 Distribution of classifier

The research is about classifying patients chances of stroke based on there lifestyle and health. The figure below is the distribution of the stroke incidents in the data.

```
# Plot stroke distribution
fig1 <- ggplot(health, aes(x=factor(stroke))) +
  geom_bar(width = 0.6, fill = c("seagreen", "red")) +
  labs(title = "Stroke distribution", x = "Had a stroke?", y  = "Count")

# plot the graph
fig1 + scale_x_discrete(breaks=c("0", "1"), labels=c("No", "Yes"))
```

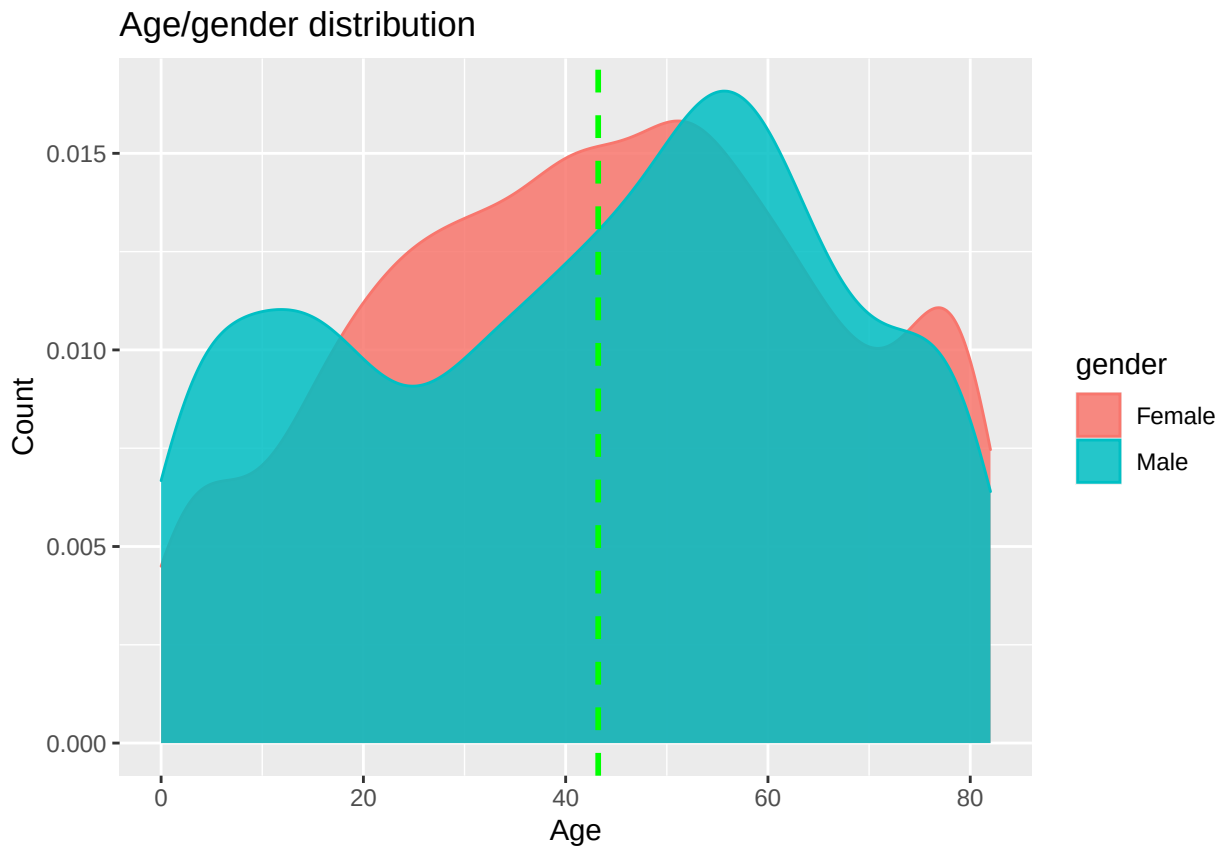


The number of stroke in the dataset is very low. The vast majority of the data didn't have a stroke. This needs to be balanced later.

4.3 Age and gender distribution

There are 5110 people in the dataset from which there is data, the gender and age distribution is as follows:

```
# make age and gender plot
ggplot(health, aes(x=age, fill=gender, color=gender)) +
    geom_density(alpha =.85) +
    labs(title = "Age/gender distribution", x = "Age", y  = "Count") +
    geom_vline(aes(xintercept=mean(age)), color = 'green', lwd = 1, linetype = 'dashed')
```



Most people are around 40 - 60 years old, with some more females round the 20- 40 mark.

4.4 health related variables

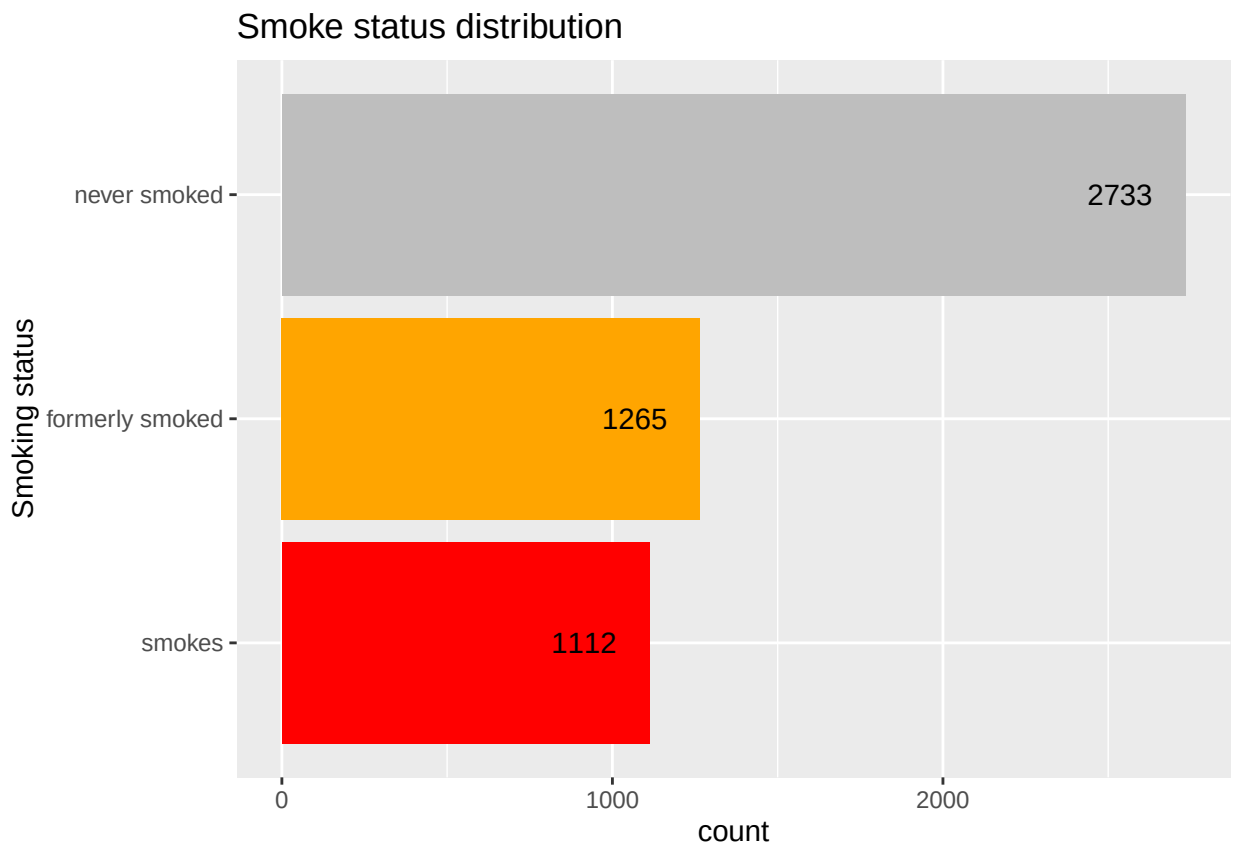
This section tells about the heath related variables such as, smoking habits, BMI, hypertension, heart disease and glucose levels

4.4.1 Smoke

Smoking can affect your health in a bad way, therefore it is good to know how many people (formerly) are smoking.

```
health %>%
  group_by(smoking_status) %>%
    summarise(count = length(smoking_status)) %>%
```

```
      mutate(smoking_status = factor(smoking_status)) %>%
        ggplot(aes(x = fct_reorder(smoking_status, count), y = count)) +
        geom_col(fill = c("Orange", "gray", "red")) +
        geom_text(aes(label = count, x = smoking_status, y = count), size = 4, hjust = 1.5) +
        coord_flip() +
        labs(x = "Smoking status", title = "Smoke status distribution")
```



Most of the people never smoked, but a fairly amount of the people smokes or formerly smoked. This can indicate a bad overall health, which possible can contribute towards strokes. The term smokes is a bit too general, because it says nothing about how much someone smokes on a daily basis. The same applies to formerly smoking people.

4.4.2 BMI and Glucose levels

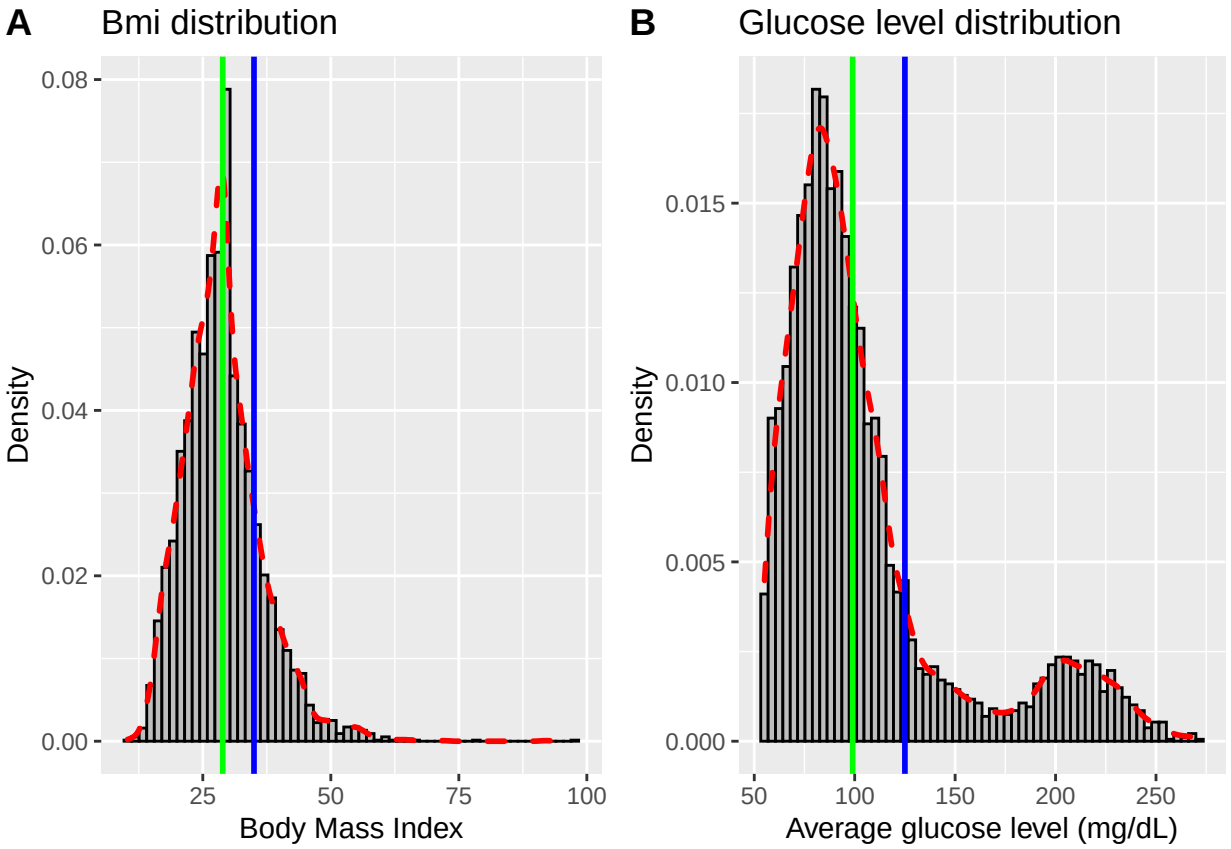
Body mass index (BMI) and glucose levels are also a good indication of general health. High BMI and glucose levels are considered bad for health.

```
fig2 <- ggplot(health, aes(x = bmi)) +
  geom_histogram(aes(y = ..density..), bins = 60, color = 1, fill = "gray") +
  geom_density(lwd = 1, linetype = 2, color = 'red') +
  labs(title = "Bmi distribution", x = "Body Mass Index", y = "Density") +
  geom_vline(aes(xintercept=mean(bmi)), color = 'green', lwd = 1) +
  geom_vline(xintercept = 35, color = 'blue', lwd = 1)

fig3 <- ggplot(health, aes(x = avg_glucose_level)) +
```

```
  geom_histogram(aes(y = ..density..), bins = 60, color = 1, fill = "gray") +
  geom_density(lwd = 1, linetype = 2, color = 'red') +
  labs(title = "Glucose level distribution", x = "Average glucose level (mg/dL)", y = "Density") +
  geom_vline(xintercept = 99, color = 'green', lwd = 1) +
  geom_vline(xintercept = 125, color = 'blue', lwd = 1)


plot_grid(fig2, fig3, labels = "AUTO", rel_widths = c(1,1))
```



BMI has a sort of normal distribution, with a mean of ~ 28. A BMI between 25 - 30 means overweight and 30+ means obese. Their is a fairy amount of people with a BMI higher than 35 (the blue line). This means that a reasonable amount of people in the dataset is serious overweight.

For glucose level, the distribution is a little bit right skewed with a bulge toward the end of the x axis. The mean of 106 is high. 99 mg/dL or lower is normal, 100 to 125 mg/dL indicates you have pre-diabetes, and 126 mg/dL or higher indicates you have diabetes. The green line represents 99 mg/dL and the blue line 125 mg/dL. These borders shows that, according to the above information. All the people left of the green line are healthy and all the people right of the blue line have diabetes. So, their are some serious unhealty people according to these numbers.

```
length(health$id[health$bmi > 35 & health$avg_glucose_level > 125])
```

```
## [1] 265
```

It tells that with the above information 265 people are serious unhealthy.

4.4.3 Cardiovascular diseases

There are two cardiovascular diseases in the data, which are heart disease and hypertension.

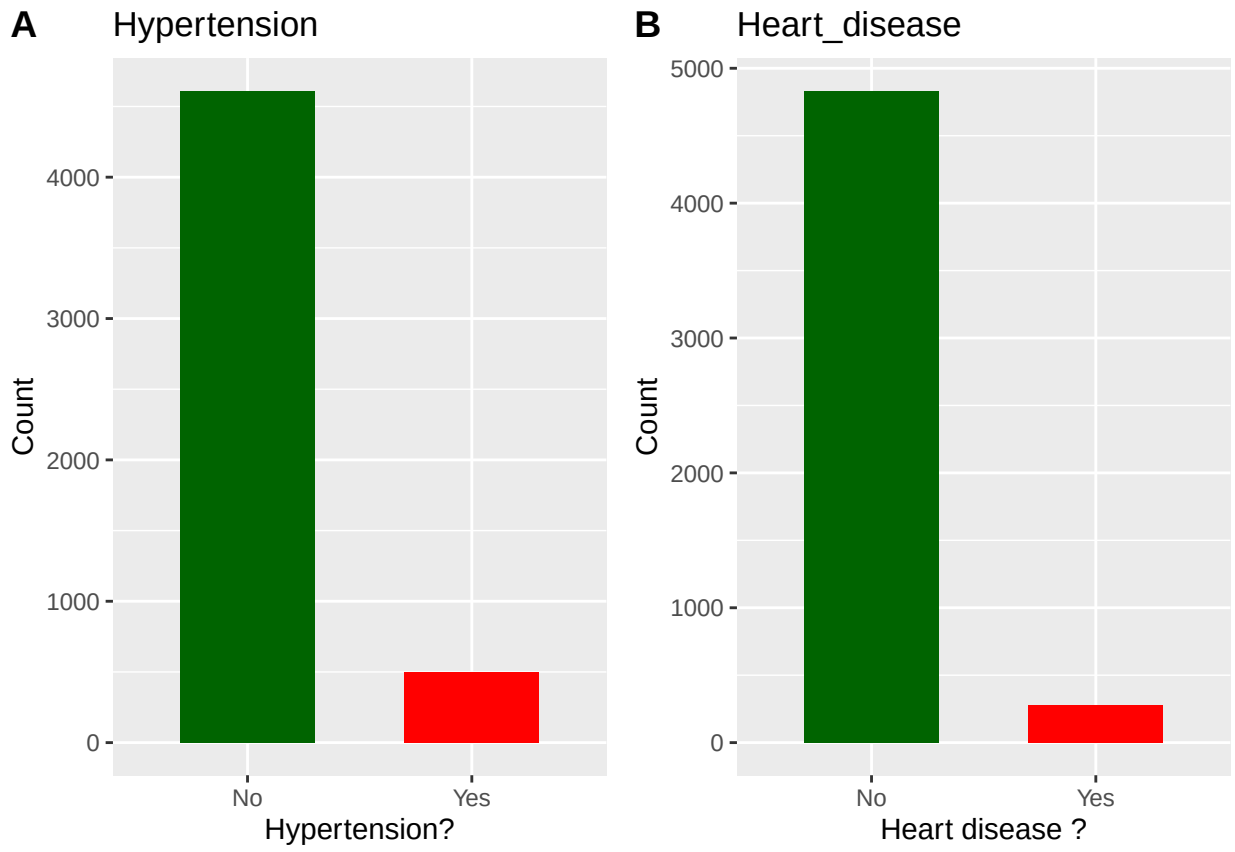
```
fig4 <- ggplot(health, aes(x=factor(hypertension))) +
  geom_bar(width = 0.6, fill = c("darkgreen", "red")) +
  labs(title = "Hypertension", x = "Hypertension?", y  = "Count")

fig4 <- fig4 + scale_x_discrete(breaks=c("0", "1"), labels=c("No", "Yes"))

fig5 <- ggplot(health, aes(x=factor(heart_disease))) +
  geom_bar(width = 0.6, fill = c("darkgreen", "red")) +
  labs(title = "Heart_disease", x = "Heart disease ?", y  = "Count")

fig5 <- fig5 + scale_x_discrete(breaks=c("0", "1"), labels=c("No", "Yes"))


plot_grid(fig4, fig5, labels = "AUTO")
```



The cardiovascular diseases are both overwhelming no, this is also imbalanced.

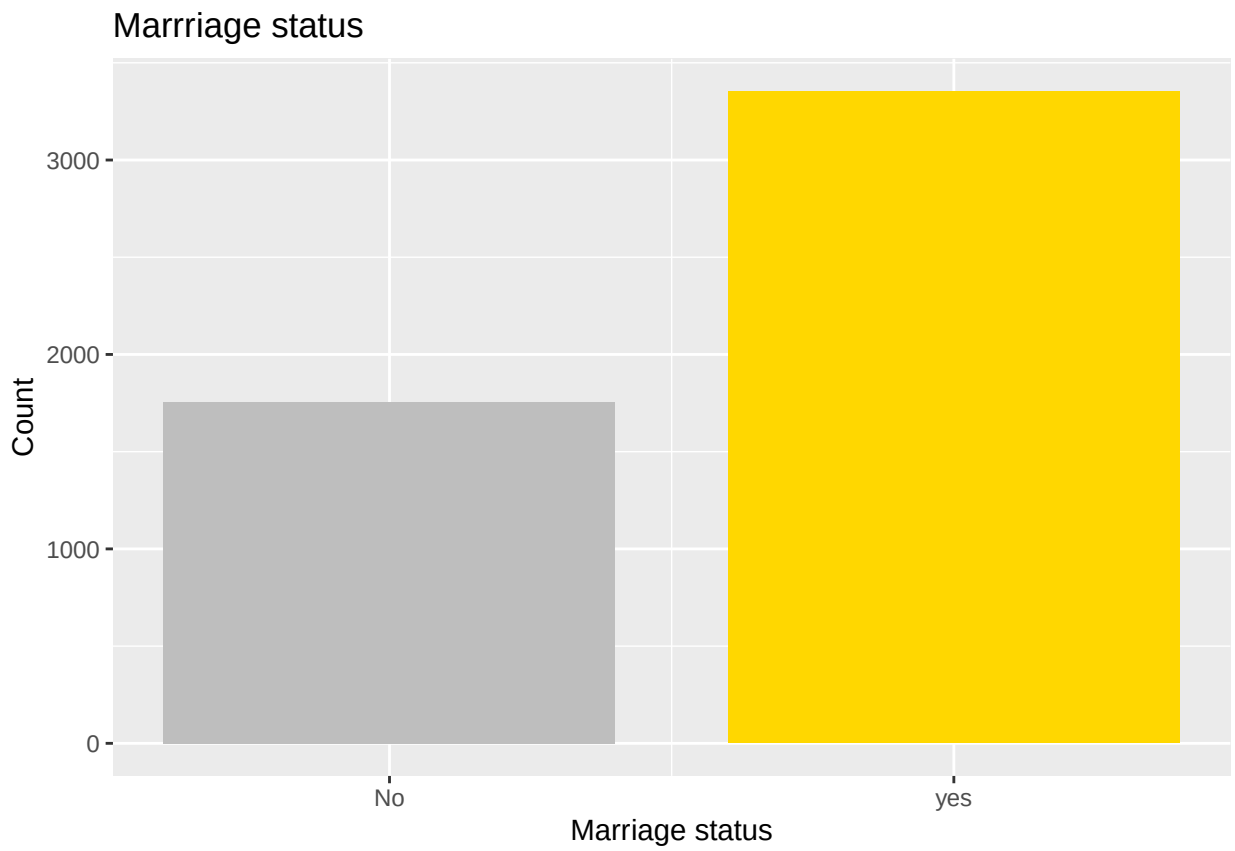
4.5 Lifestyle varibles

There are three lifestyle variables: married?, work type and residence type. These variables tells about the persons possible stress level. All these factors can contribute toward stress and possible health affection.

4.5.1 Marriage status

The below plot shows the distribution of marriage. A married life could be stressful, even as a divorce. The 'yes' in ever married tells not the whole story, because a person could be married in the past, but is now divorced. These previously married people also are categorised into married. Which makes the 'yes' records a bit iffy.

```
ggplot(health, aes(x = ever_married)) +
  geom_bar(width = 0.8, fill = c("gray", "gold")) +
  scale_x_continuous(breaks = c(0,1), labels = c("No", "yes")) +
  labs(x = "Marriage status", y = "Count", title = "Marrriage status")
```



As is shown, most people have been married at some point in their lives. The not-married group is a sizeable group with people that have not been married. This maybe could be young people, which would be logic. The code below tests this hypothesis:

```
cat("Mean age of not married: ", round(mean(health$age[health$ever_married == "0"]), 2), '\n')
```

```
## Mean age of not married:  21.98
```

```
cat("Mean age of ever married: ", round(mean(health$age[health$ever_married == "1"]), 2))
```

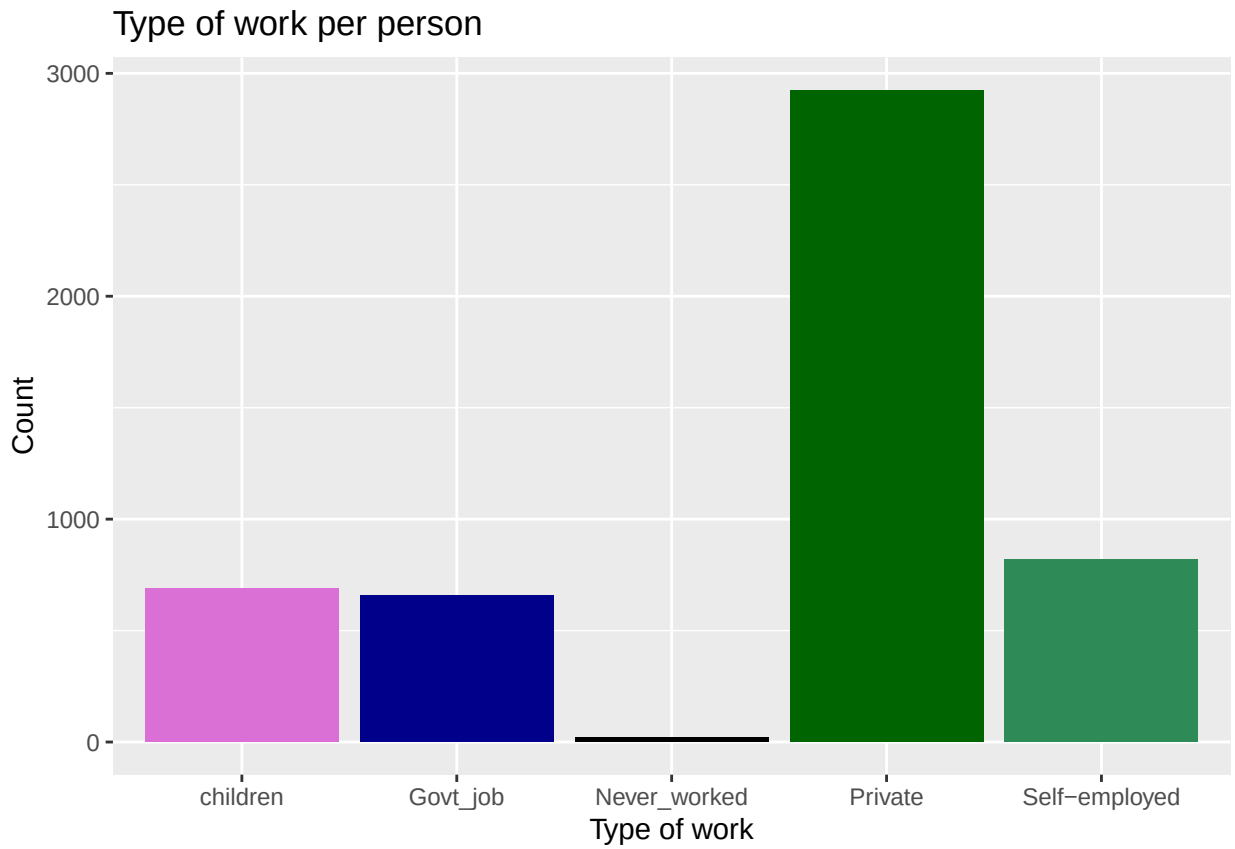
```
## Mean age of ever married:  54.34
```

As thought, the mean age is ~ 22 years old. Which means that this group mostly are young people.

4.5.2 Type of work

The type of work someone preforms can contribute toward higher stress levels. The plot beneath displays the distribution of work types:

```
ggplot(health, aes(x = work_type)) +
  geom_bar(position = 'identity', fill = c("orchid", "darkblue", "black", "darkgreen","seagreen")) +
  labs(x = "Type of work", y = "Count", title = "Type of work per person")
```



looking at the graph, most people work for a private company. Children, government job and self employed are somewhat balanced. Children would be mostly women is the guess.

```
table(health$gender[health$work_type == 'children'])
```

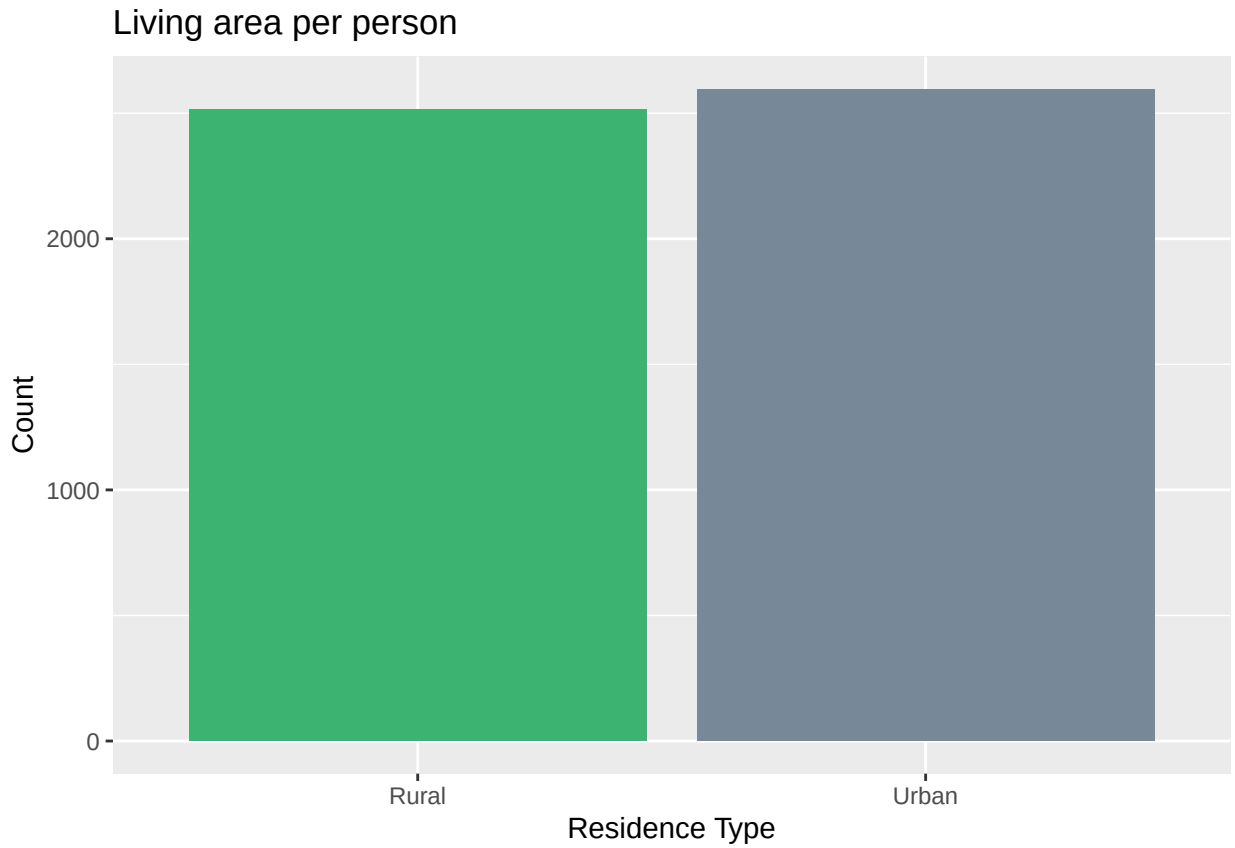
```
##
## Female   Male  Other
##    326    361      0
```

Surprisingly, children is balanced between male and female. so the thought of extra stress on women with children (which possibly can lead to higher stroke chances) can be rejected.

4.5.2 Residence type

The type of living area can have influence on someone's life, living in a busy urban area maybe contributes to stress. The plot of residence type below shows the spread of living areas.

```
ggplot(health, aes(x =  Residence_type)) +
  geom_bar(position = 'identity', fill = c("mediumseagreen", "lightslategray")) +
    labs(x = "Residence Type", y = "Count", title = "Living area per person")
```



The bar plot of residence type shows a balance between the two types. There can be assumed that mostly young people live in urban areas and elderly people leave the busy areas.

```
cat("Mean age of urban: ", round(mean(health$age[health$Residence_type == "Urban"]), 2), '\n')
```

```
## Mean age of urban:  43.53
```

```
cat("Mean age of rural: ", round(mean(health$age[health$Residence_type == "Rural"]), 2))
```

```
## Mean age of rural:  42.89
```

The mean ages did not show any significant age difference between living rural or urban

5. Comparing variables

Now that all variables are shown, as standalone. It is time to compare variables with each other. Looking at the variables, their are two types, lifestyle and health. Obviously, health variables is the first thing to look at, because stroke is a cardiovascular disease. For this reason, health variables will be compared against stroke events.

5.1 health

In the dataset there are multiple people with heart disease and hypertension, could this combination be related towards stroke events? We take a look at hypertension at stroke and heart disease at stroke, also we look at heart disease and hypertension together.

```
# total cases of heart diseases and hypertension
cat(" Number of cases with stroke: ", sum(health$stroke),"\n","Number of cases with heart disease: ", su
```

```
##  Number of cases with stroke:  249
##  Number of cases with heart disease:  276
##  Number of cases with hypertension:  498
```

Inspecting the numbers above, their are more heart disease and hypertension cases than stroke events. So, not all heart disease and hypertension cases could be linked to stroke events.

So, how many stroke cases did also have underlying conditions such as heart disease and hypertension?

```
# stroke/heart disease
cat("The number of case with stroke and heart disease are: ",sum(health$stroke[health$heart_disease == 1
```

```
## The number of case with stroke and heart disease are:  47
```

```
# stroke/hypertension
cat("The number of case with stroke and hypertension are: ",sum(health$stroke[health$hypertension == 1])
```

```
## The number of case with stroke and hypertension are:  66
```

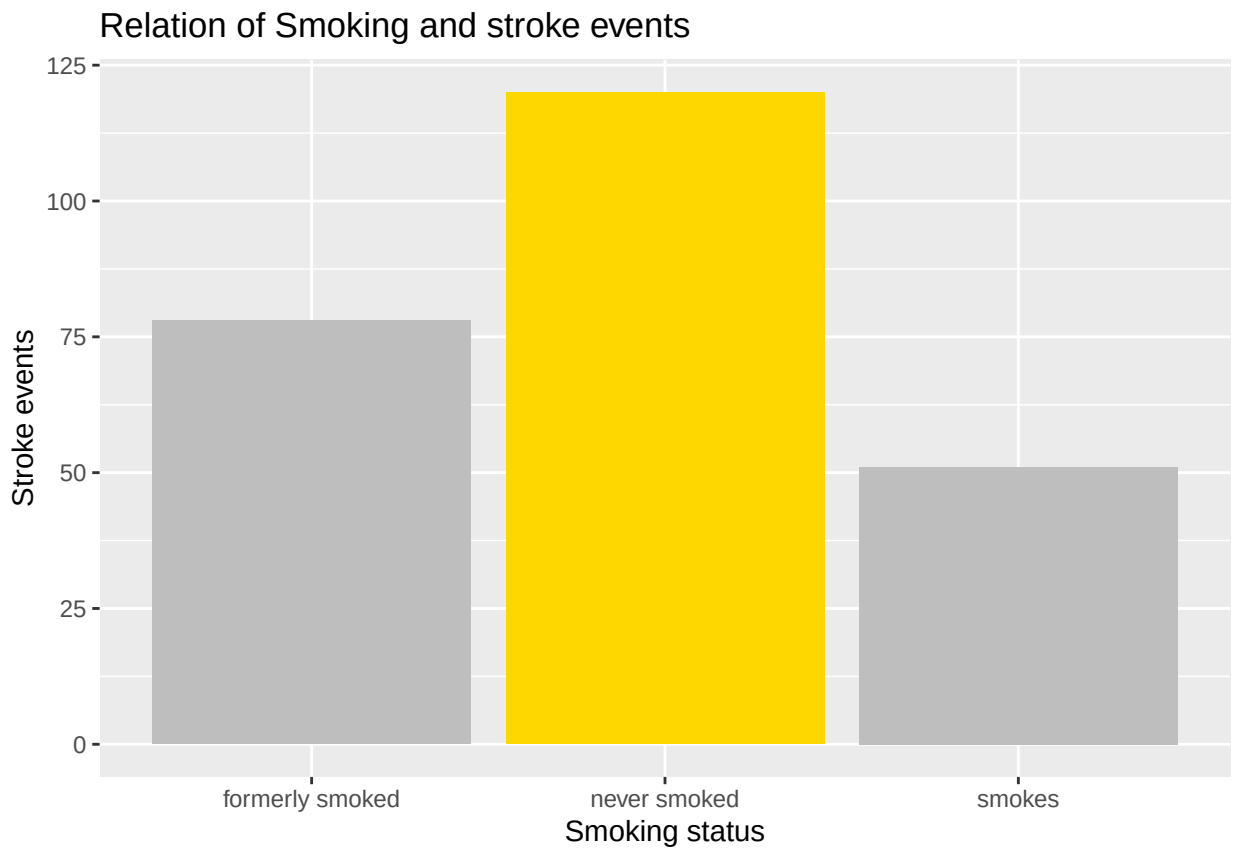
```
# stroke/ heart disease and hypertension
cat("The number of case with stroke and heart disease & hypertension are: ",sum(health$stroke[health$hyp
```

```
## The number of case with stroke and heart disease & hypertension are:  13
```

The relationship between stroke and heart disease |& hypertension is not big. Only 13 cases of the 249 stroke events have two other cardiovascular diseases. This means that, according to this dataset, the relation between hypertension/heart disease and stroke is not great.

Next, looking at smoke. There is a expectation that smoking is related to cardiovascular diseases. So, expected is that the most stroke events occurred with smokers or formerly smokers.

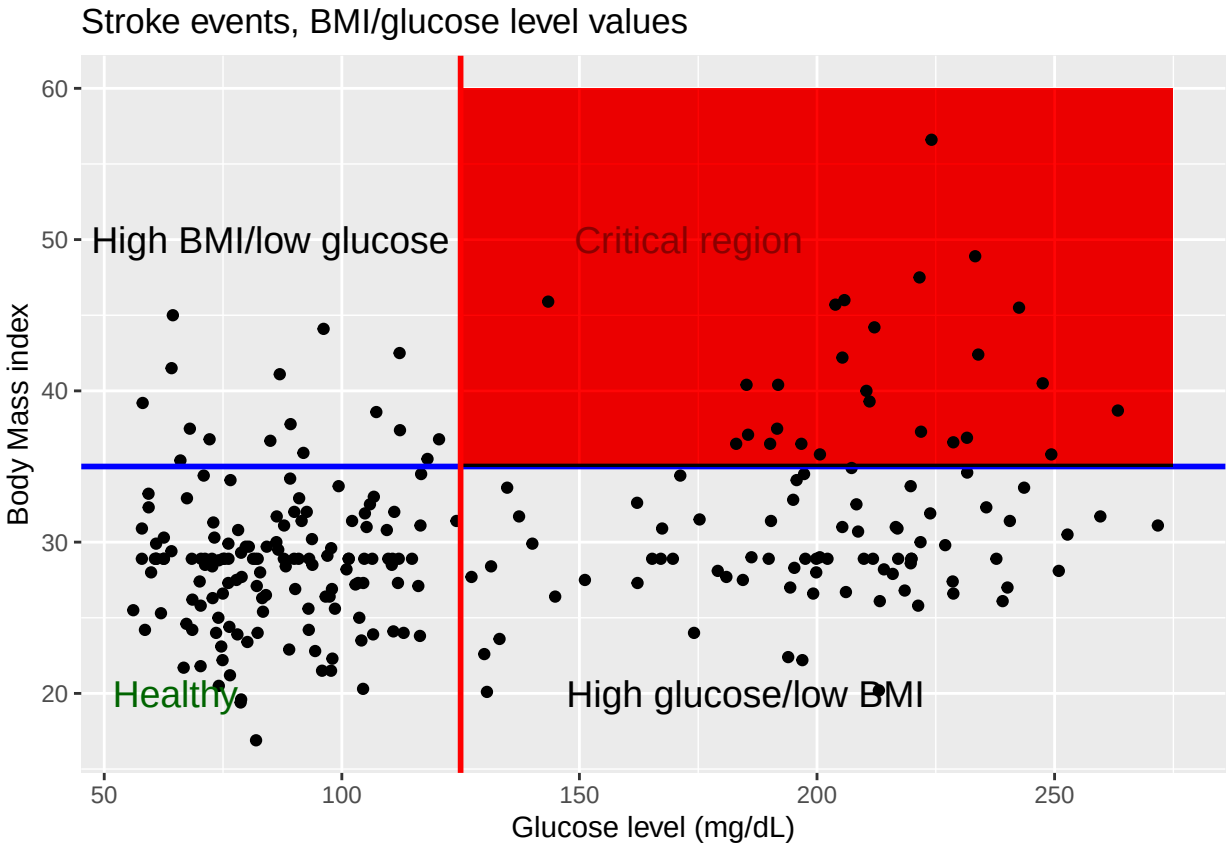
```
ggplot(health[health$stroke == T,], aes(x=smoking_status)) + geom_bar(position = 'dodge', fill = c("gray
  labs(x = "Smoking status", y = "Stroke events", title = "Relation of Smoking and stroke events")
```



The graph above shows that the Group with the most stroke events are the non-smokers. Something that wasn't expected, but take into mind that the non smokers group is also the biggest group in the dataset. Outnumbering the other smoking groups by at least 1000 records.

high BMI and glucose levels can also contribute towards cardiovascular diseases. The graph beneath checks if it is true that high BMI and glucose levels increase the chance of a stroke.

```
ggplot(health[health$stroke == T,], aes(x = avg_glucose_level, y = bmi), ) + geom_point(show.legend = T
  labs(x = "Glucose level (mg/dL)", y = "Body Mass index", title = "Stroke events, BMI/glucose level va
  geom_hline(yintercept = 35, color = "blue", lwd = 1) +
  geom_vline(xintercept = 125, color = "red", lwd = 1) +
  annotate("text", x= 173, y = 50, label="Critical region", color= "darkred", lwd = 5) +
  annotate("text", x= 185, y = 20, label="High glucose/low BMI", color= "black", lwd = 5) +
  annotate("text", x= 85, y = 50, label="High BMI/low glucose", color= "black", lwd = 5) +
  annotate("text", x= 65, y = 20, label="Healthy", color= "darkgreen", lwd = 5) +
  geom_rect(aes(xmin= 125, xmax = 275, ymin = 35, ymax = 60), fill = "red", alpha = 0.002)
```



All the records in the graph above are people that had a stroke. Remembering the section about BMI en glucose levels, a BMI of 35 is considered to high and a glucose level of 125 mg/dl is also to high. The region in red is called the 'critical region'. This region is where there are records of people with a BMI high than 35 and a glucose level high than 125 mg/dL. So, the critical region is where the very unhealthy people are. The expectation is that the most stroke events happens by these people.

But, the most of the records are outside of the critical region, most people reside in the 'healthy' region. So the assumption of the contribution of high BMI and high glucose levels towards strokes can be rejected based on this dataset.

5.2 lifestyle

The lifestyle variables tells about the living, working and marriage status of a person. A successfully stress free life would reduce the chances of stroke, won't it? To test this hypothesis the lifestyle variables are compared against the classifier stroke.

The first look is at Work type, the type of work someone does

```
round(prop.table(table(health$work_type[health$stroke == T])) * 100, 1)
```

```
##
##      children      Govt_job  Never_worked       Private Self-employed
##           0.8          13.3           0.0          59.8          26.1
```

The table overhead tells that the most stroke events happens by people that are working for a private company 59,8%. To recall the plot of work types earlier, the most people work at private companies, so it is not unexpected that the most stroke events happens by people that work for a private company.

Let's take a look a living area. The living area's are rural or urban. Earlier is said that the mean age of living is not that different. So, the assumption that young people, that mostly live in urban regions, decreases the percentage of strokes events in that living region can be rejected.

```
round(prop.table(table(health$Residence_type[health$stroke == T])) * 100, 1)
```

```
##
## Rural Urban
##  45.8  54.2
```

Viewing the percentages above, the most stroke events happens at urban region. With not a big difference between the percentages. Which leave marriage status as the last variable.

The variable ever married indicate if a person is married of have been married at some point in their life. The percentages beneath shows the stroke events by married status.

```
round(prop.table(table(health$ever_married[health$stroke == T]))* 100, 1)
```

```
##
##    0    1
## 11.6 88.4
```

Interesting is that The majority of the stroke events happens by people that are or have been married, 88,4%. But, take into mind that the most people that are married are older then the not married people. Speaking of age, what is the age where most stroke events take place?

```
ggplot(health[health$stroke == T,], aes(x = age, fill = gender)) +
  geom_density(aes(y = stat(count)), alpha=0.6) +
  geom_vline(xintercept = mean(health$age[health$stroke == T]), linetype = 'dashed', color = 'seagreen'
  labs(x = "Age", y = "Number of stroke events", title = "Age and gender in relation to stroke events"
```

In the distribution above is clearly seen that strokes mostly affects older people, which is expected. The mean age of stroke is 67 years old. It is also more frequent with female than with male.

6. EDA conclusion

The data is in a good condition with several variables to work with for machine learning. The removal of the Na's made the quality of the data better, mainly trough the conversion of the term 'unknown' by smoking status. Also the casting of types helps al lot. The variables does not show in the ED a very big difference in relation with stroke. But the variables age and marriage status shows some hope. But the data is still having one big flaw.

The plots shows that the distribution of certain variables skewed is, namely: stroke, hypertension and heart_disease. These variable have a very big imbalance inside. This imbalance can cause problems with machine learning because saying 'No' at these variables can give you a 95%+ accuracy. This imbalance need to be tackled before moving on into machine learning. By doing this, the dataset can be workable for machine learning.

7. Machine learning

The Goal in the machine learning project is to predict a chance of a stroke event based of the health and lifestyle variables. This part covers the cleaning of the data for machine learning, determine quality metrics and investigate performance of machine learning algorithms.

For the machine learning (ML) a program named Weka will be used. Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. In this analysis Weka version 3.9.6 is used.

In the section below the word accuracy is many times being said. In this experiment *accuracy means the precision of the ML algorithm in which it can rightly predict the chance of a stroke label.* So, an accuracy of 70% means that ,based on the inputted values of that person, there is a 70% chance that a persons stroke history is right. In other words if the output is true, you're likely to get a stroke in the near future.

7.1 Dataset changing

As talked over in the EDA, the dataset is okay but infected with imbalanced data. Namely the variables hypertension, heart disease and the classifier stroke. By leaving the imbalance in the data, the ML algorithm will bow towards the 'No', because this automatic will give a 95% accuracy. Any sort of algorithm will tent to high accuracy, if that high percentage can be achieved by taking the majority, it will. So, it is necessary to tackle this imbalance.

7.1.1 Methods

With imbalanced data sets, an algorithm doesn't get the necessary information about the minority class to make an accurate prediction. Which will lead to misleading accuracies.

There are a few ways to fix imbalanced dataset:

1. Undersampling
2. Oversampling
3. Synthetic Data Generation
4. Cost Sensitive Learning

Undersampling is reducing the number of observations from the majority class to achieve a balance. Undersampling can be split into two types: random and informative. Random undersampling means, taking random records from the majority and eliminate these to achieve a balance. Informative undersampling means eliminate records based on a pre-defined condition. undersampling seems good, but the danger lies in deleting (unwanted) important information for the algorithms. Not the best choice.

Oversampling is replicating records from the minority class. This can also be random or informative. the advantage with this method is that there is no information loss. But, with extra records from the minority, it can lead to overfitting.

Synthetic Data Generation is a method to, in stead of replication, add observations to the minority. It is a type of oversampling. These new observations are artificial generated. The prefect method for this is SMOTE (synthetic minority oversampling technique). The SMOTE algorithm creates artificial data based on feature space. similarities from minority samples. It creates a random set of minority class observations to shift the classifier learning bias toward the minority class. This method looks really good from this dataset.

Cost Sensitive Learning is the last technique. It puts weight on misclassified observations. There are penalties applied at false positives and false negatives. By doing this, the algorithm knows which classifications are 'heavier' to classify. It look good to use on the dataset.

Inspecting the four methods above for fixing imbalanced dataset. Synthetic Data Generation with SMOTE looks the best. This method generates new data for the minority class without losing important information or oversample the data with duplicate observations.

7.1.2 SMOTE

The SMOTE technique creates a new dataset by oversampling observations from the minority class. Which is excellent for this imbalanced dataset. The easiest way is to use the SMOTE() function from the 'DMwR' package. But, there is a problem with using SMOTE, there are categorical variables and numeric variables. SMOTE works best with numeric data. Luckily, the categorical variables are not big. The biggest is work_type with four labels. The code below convert the variables to numeric, there are four variables that need changing:

```
# gender
health$gender <- as.character(health$gender)

health$gender[health$gender == "Male"] <- 1
health$gender[health$gender == "Female"] <- 0
names(health)[names(health) == "gender"] <- "is_Male?"

health$`is_Male?` <- as.integer(health$`is_Male?`)

#work_type
health$work_type <- as.character(health$work_type)

health$work_type[health$work_type == "Never_worked"] <- 0
health$work_type[health$work_type == "children"] <- 1
health$work_type[health$work_type == "Self-employed"] <- 2
health$work_type[health$work_type == "Govt_job"] <- 3
health$work_type[health$work_type == "Private"] <- 4

health$work_type <- as.integer(health$work_type)

#Residence_type
health$Residence_type <- as.character(health$Residence_type)

health$Residence_type[health$Residence_type == "Rural"] <- 0
health$Residence_type[health$Residence_type == "Urban"] <- 1
names(health)[names(health) == "Residence_type"] <- "living_urban?"

health$`living_urban?` <- as.integer(health$`living_urban?`)

#smoking_status

health$smoking_status <- as.character(health$smoking_status)

health$smoking_status[health$smoking_status == "never smoked"] <- 0
health$smoking_status[health$smoking_status == "formerly smoked"] <- 1
health$smoking_status[health$smoking_status == "smokes"] <- 2

health$smoking_status <- as.integer(health$smoking_status)

# relocate classifier
health <- health %>% relocate(stroke, .after = last_col())

head(health)
```

```
##      id is_Male? age hypertension heart_disease ever_married work_type
```

```
## 1  9046        1  67           0            1           1         4
## 2 51676        0  61           0            0           1         2
## 3 31112        1  80           0            1           1         4
## 4 60182        0  49           0            0           1         4
## 5  1665        0  79           1            0           1         2
## 6 56669        1  81           0            0           1         4
##   living_urban? avg_glucose_level  bmi smoking_status stroke
## 1             1            228.69 36.6              1      1
## 2             0            202.21 28.9              0      1
## 3             0            105.92 32.5              0      1
## 4             1            171.23 34.4              2      1
## 5             0            174.12 24.0              0      1
## 6             1            186.21 29.0              1      1
```

Now all the variables are numeric and the classifier is relocated toward the last column. The code below preforms the SMOTE balancing on the dataset. note that SMOTE tent to calculate decimal values of variables with 1 - 4. therefore smoking_status, work_type, is_male?, living_urban and ever_married is converted to factor to tackle this problem. Also the classifier is slightly modified to grow more towards SMOTE's preferences.

```
# changes certain variables to factor to prevent problems
health$smoking_status <- as.factor(health$smoking_status)
health$work_type <- as.factor(health$work_type)
health$`is_Male?` <- as.factor(health$`is_Male?`)
health$`living_urban?` <- as.factor(health$`living_urban?`)
health$ever_married <- as.factor(health$ever_married)

# change classifier to factor with boolean labels
health$stroke <- as.numeric(health$stroke)
health$stroke[health$stroke == 0] <- "FALSE"
health$stroke[health$stroke == 1] <- "TRUE"
health$stroke <- as.factor(health$stroke)

# write pre-SMOTE data to csv
write.csv(health, "Data/health.csv")


# preform SMOTE on data
x <- SMOTE(form = stroke ~ ., data = health, perc.over = 3000, perc.under = 150)

# write SMOTE data to csv.
write.csv(x = x, file = "Data/SMOTEdata.csv")
```

The result are two dataset, one without SMOTE and one with SMOTE applied. The SMOTE data has a perc.over of 3000 and a perc.under of 150. Which means that the minority class is synthetic oversampled by 3000% and the majority class synthetic oversampled is by 150%. This creates a ratio of 60/40 instead of 95/5

7.2 Testing algorithms

We ended up with two datasets.One with the imbalance and one with the imbalanced SMOTE'd out. These two dataset will be tested on accuracy to determine which dataset is most suitable to proceed with. Also, With a too high accuracy overfitting might occur. Overfitting is getting a very high accuracy($> 95\%$) form

an algorithm which is too precise. For example, we speak of overfitting when a tree algorithm creates a tree of 457 nodes and gets a accuracy of 98,9%. It works excellent on the handed data, and only on that data. So, a extremely high accuracy on the data in not necessarily good.

Weka self offers a handful of filters and tools. One of this tools is 'ClassBalancer'. It reweights the instances in the data so that each class has the same total weight. This form of balancing will be applied on the imbalanced data, Which makes the datasets to test a total of three. Normal, Normal with classbalancer and SMOTE modified data.

there will be eight algorithms tested:

- ZeroR
- OneR
- J48
- Random Forest
- NaiveBayes
- Simple Logistic
- Logistic
- Sequential minimal optimization(SMO)

The results are tabled below accuracies, all algorithms are executed on default options.

```
algo_accuracies <- read.table("Data/accuracies.txt", sep = ",", header = T)
colnames(algo_accuracies) <- c("Type of algorithm", "Normal", "Normal (classBalancer)", "SMOTE")
kable(algo_accuracies, format = "simple", caption = "Accuracies of different algorithms")
```

Table 5: Accuracies of different algorithms

Type of algorithm	Normal	Normal (classBalancer)	SMOTE
ZeroR	95.1	49.8	59.2
OneR	95.1	-	96.2
J48	95.1	60.7	96.1
RandomForest	94.4	54.6	96.2
NaiveBayes	89.1	76.2	74.2
SimpleLogistic	95.1	75.1	77.8
Logistic	95.1	76.4	77.7
SMO	95.1	76.3	78.3

In the table is seen that the normal imbalanced dataset almost always has an accuracy of 95%. This is because the imbalance in the dataset is equal to 95/5. So, the algorithm deals with this by saying that almost everything is false. In 95% of the cases, that is correct. As earlier concluded, the dataset with the imbalance is not good.

Looking at the dataset with the ClassBalance filter. The accuracies are reasonable balanced, with the bayes and function algorithms getting +70% accuracies. But the way how the ClassBalance filter acts is questionable. To put it simply, records form the minority are duplicated to achieve a 50/50 ratio and random duplication is not desirable.

The SMOTE dataset has a couple of overfitted accuracies, but the logistics and bayes preforms well. The way that created the extra records is also more desirable that the ClassBalance filter. So, The SMOTE dataset is the one to go with in the next section of the ML phase.

7.3 Choosing Alogrithm

Choosing the right algorithm is a crucial aspect to proceed. Beforehand it is important to know, what makes algorithm 'good'? What are the characteristics of a desirable algorithm. For this question some quality metrics are determined.

Example of different quality metrics:

- accuracy
- speed
- size of tree*
- Confusion matrix

Obviously a high accuracy is desired by all means. Then the question is, how accurate is good enough? As base, a accuracy of 90% is considered good. with diseases like strokes, 80% is not good enough. Which means that one in five is classified wrong, which can have serious consequences. Also, How is this accuracy achieved? Look out for overfitting.

Speed of the algorithm is important by big usage. Implementing the model in a local environment, 1 to 2 minutes is okay. As it would not calculate more than 10 times a day. But in big scale usage, an fast algorithm is a demand.

In case of a tree, the size of a Decision tree is a important factor. With a large tree with many leave, overfitting could be possible. As the algorithm is to specific on the trainings set. A small, but accurate tree is desirable.

A confusion matrix, is a specific table that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. This indicates also the rate of true positives, true negatives, false positives and false negatives. The errors in the matrix will weight heavy on the model performance.

In conclusion: accuracy, confusion matrix and size of tree will weight the heaviest in selecting a good algorithm.

7.3.1 Baseline

With a proper dataset and determined quality metrics,the 'SMOTEdata.csv' file is loaded into Weka. The first two rows are removed, the blank attribute and ID. In the edit viewer, the Attributes are quickly visual scanned, to see if all the labels says numeric,except the classifier.

In The test options, cross-validation is set to 10 Folds (default). First a baseline performance is set (ZeroR).

TP = True positive rate FP = False positive rate

```
baseline <- read.csv("Data/zeroR_weka.csv", sep = ";", header = TRUE)
colnames(baseline)[1] <- "Algorithm ->"
pander(baseline, caption = "ZeroR mesurement")
```

Table 6: ZeroR mesurement

Algorithm ->	ZeroR
Accuracy (%)	59,2
Speed (sec)	0,01
T TP (%)	0
T FP (%)	0

Algorithm ->	ZeroR
F TP (%)	100
F FP (%)	100

ZeroR is a very simple algorithm that looks only at the classifier, in this case there are 11205 instances False, and 7719 True. ZeroR takes the majority and uses that for its accuracy, in this case 59,2%. SO, it is known that the baseline is 59,2%. This is the Zero measurement.

7.3.2 Advanced algorithms

Now that the baseline is determined, it is time for testing more complex algorithms. Earlier for comparison between normal and SMOTE data a couple of algorithms already were tested on default settings. This time speed, TP and FP also added.

```
advanced_algo <- read.csv("Data/advanced_weka.csv", sep = ";", header = T)
colnames(advanced_algo)[1] <- "Algorithms ->"
kable(advanced_algo, caption = "Advanced algorithms")
```

Table 7: Advanced algorithms

Algorithms ->	OneR	J48	RandomForest	Naive.Bayes	Simple.logistic	Logistic	SMO
Accuracy (%)	96,3	96,2	96	74	77,9	77,5	78,3
Speed (sec)	0.03	0.35	4.43	0.04	1.34	0.19	15.94
T TP (%)	91,1	90,7	93,5	65,6	74,3	73,7	78,4
T FP (%)	0	0	2,2	20,2	19,6	19,8	21,7
F TP (%)	100	100	97,8	79,8	80,4	80,2	78,3
F FP (%)	8,9	9,3	6,5	34,4	0,25	26,3	21,6
Size of tree	.	177	.	.	.	.	.

// parameters

// significant?

// final model

// wrapper