CYO_Project_Stroke Prediction

Christina Papadopoulou

2022-11-20

Introduction

A stroke is a medical condition that affects the arteries leading to the brain. Basically, the blood supply to part of the brain stops. This could happen because a blood vessel which deliver oxygen and nutrients to the brain, either is blocked by a clot or is ruptured and hence, preventing the blood flow to the brain. In both cases, part of the brain cells become damaged or die. As a result, a stroke can cause lasting brain damage, disabilities or even death. According to World Health Organization, stroke was the second leading cause of death globally in 2019. More information about that can be found in The top 10 causes of death. It has been proved that are certain conditions that increase the risk of having a stroke such as diabetes, hypertension, irregular heart beats and others.

In this report a dataset regarding stroke from Kabble website was used. The Stroke Prediction Dataset was used to determine or better predict if a patient is likely to have a stroke based on a number of features. More precisely, the below 12 variables were taken into account where the stroke variable is our response variable with value 1 if the patient had a stroke and 0 if not.

1) id: unique identifier of each observation
2) gender: "Male", "Female" or "Other"
3) age: age of the patient
4) hypertension: 0 if the patient does not have hypertension, 1 if the patient has hypertension
5) heart_disease: 0 if the patient does not have any heart diseases, 1 if the patient has a heart disease
6) ever_married: "No" or "Yes"
7) work_type: "children", "Govt_job" (government jobs), "Never_worked", "Private" or "Self-employed"
8) Residence_type: "Rural" or "Urban"
9) avg_glucose_level: average glucose level in blood
10) bmi: Body mass index (weight in kg/(height in m)^2)
11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
12) stroke: 1 if the patient had a stroke or 0 if not *Note: "Unknown" in smoking_status means that the information is unavailable for this patient

The goal here is to build a machine learning algorithm which will be able to predict whether or not a patient will have a stroke. In the beginning the dataset was loaded, followed by data preprocessing to achieve a tidy form of our data. Visualization of different patterns was executed and finally we proceeded in finding the algorithm that could best predict a stroke.

Methods and analysis

In this section, we are going to explain the techniques and methods used till we get to our final model.

Import data

First of all, we are going to download our data:

```
#Stroke Prediction Dataset
#https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?select=healthcare-dataset-stroke
#https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/download?datasetVersionNumber=1


stroke_temp<-  read.csv("healthcare-dataset-stroke-data.csv")
```

We then proceed in having a first look of their structure:

```
str(stroke_temp)
```

```
## 'data.frame':    5110 obs. of  12 variables:
##  $ id                : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...
##  $ gender            : chr  "Male" "Female" "Male" "Female" ...
##  $ age               : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension      : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease     : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married      : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ work_type         : chr  "Private" "Self-employed" "Private" "Private" ...
##  $ Residence_type    : chr  "Urban" "Rural" "Rural" "Urban" ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi               : chr  "36.6" "N/A" "32.5" "34.4" ...
##  $ smoking_status    : chr  "formerly smoked" "never smoked" "never smoked" "smokes" ...
##  $ stroke            : int  1 1 1 1 1 1 1 1 1 1 ...
```

We have a data frame of 5110 observations (rows) and 12 variables (columns).

The first thing that we notice from the structure of our data is that not all our variables are numerical. Actually, there are six of them that are character variables and six numerical. However, stroke which is the variable that we want to predict, is in fact a categorical variable since it denotes the fact of getting (value 1) or not getting (value 0) a stroke. We will convert the stroke variable to a factor with two levels: positive level ("pos") for those getting a stroke and negative level ("neg") for those not getting a stroke.

```
stroke_temp$stroke<-factor(stroke_temp$stroke, levels=c(0,1),labels=c("neg","pos"))
```

Let us now check if we have duplicated observations:

```
sum(duplicated(stroke_temp))
```

```
## [1] 0
```

There is no duplication, therefore we do not have to decide whether to remove them or not.

Features Selection

Features selection is an important procedure that we should apply before building our algorithm. Basically, if we include insignificant variables in our model this can impact its performance. Therefore, the proper selection of the right features could mean great performance with short training/running times.

id variable As we mentioned before, the id variable is a unique identifier for each observation. Thus, it does not affect at all the prediction of getting or not a stroke.

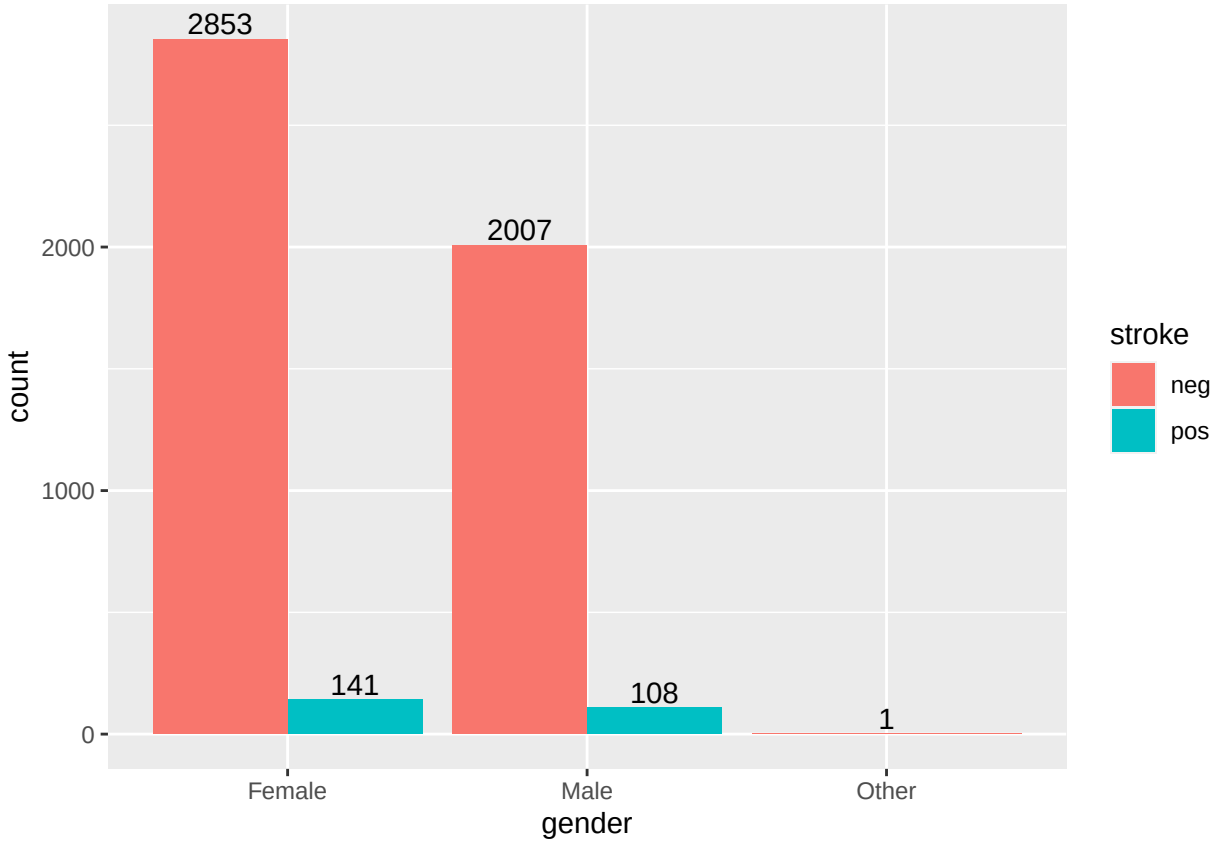
gender variable Gender variable is a categorical variable and we will create the table of frequencies for the gender variable.

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

We observe that there is only one observation defined as "Other". We do not know if this is a data entry error or a natural outlier. For the moment, we will leave it as it is and we will create the table of relative frequencies for the gender variable.

```
##
## Female   Male  Other
##  0.586  0.414  0.000
```

We see that we have more women responders than men in our dataset, but how is the stroke variable affected by this. We continue with the creation of a barplot to check the relation of stroke versus the gender variable.



Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
```

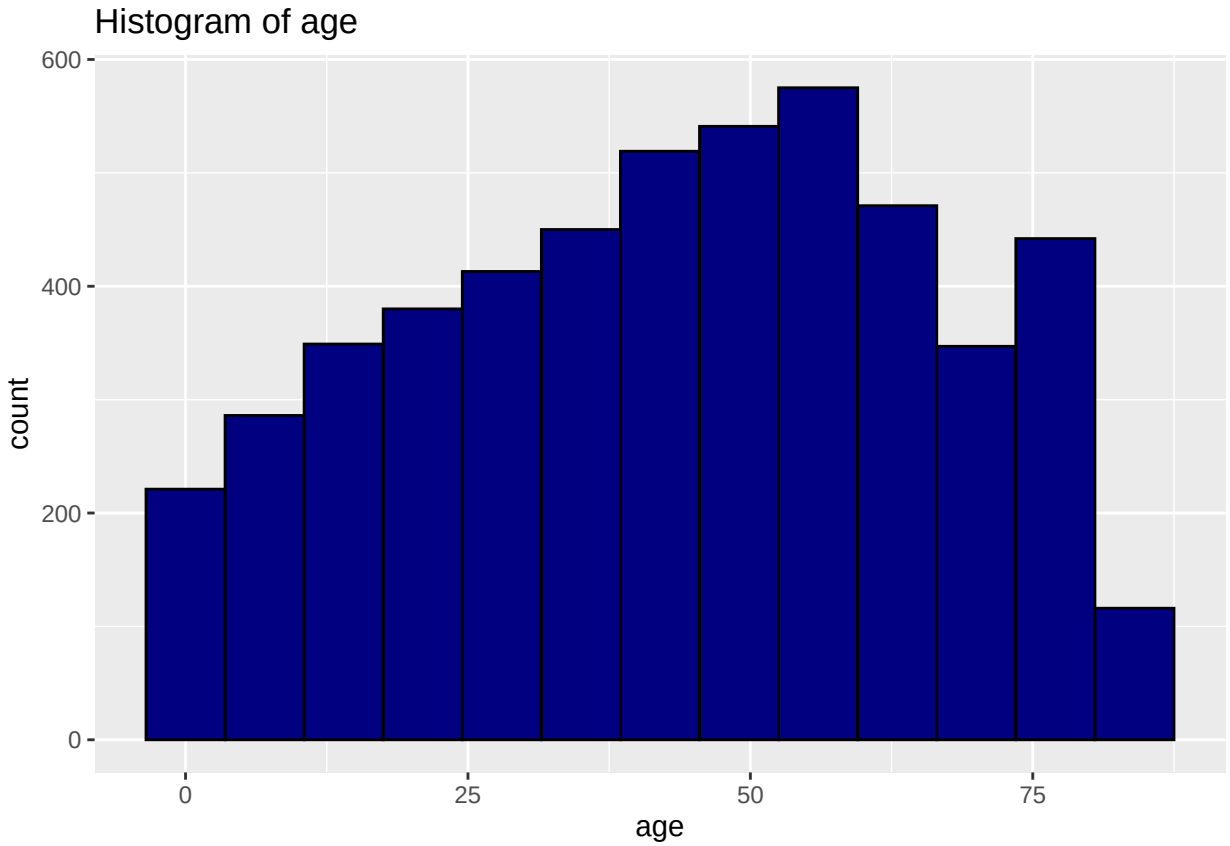
```
##  Pearson's Chi-squared test
##
## data:  stroke_temp$gender and stroke_temp$stroke
## X-squared = 0.47259, df = 2, p-value = 0.7895
```

We get a p-Value more than the significance level of 0.05, hence, we cannot reject the null hypothesis and we conclude that the two variables are independent. However, because we have just one observation reported as "Other", this may affect our chi_square test results. For this reason we will also run a Fisher's exact test, with the same null hypothesis as above, H_0: the two variables are independent.

```
##
##  Fisher's Exact Test for Count Data
##
## data:  stroke_temp$gender and stroke_temp$stroke
## p-value = 0.5746
## alternative hypothesis: two.sided
```

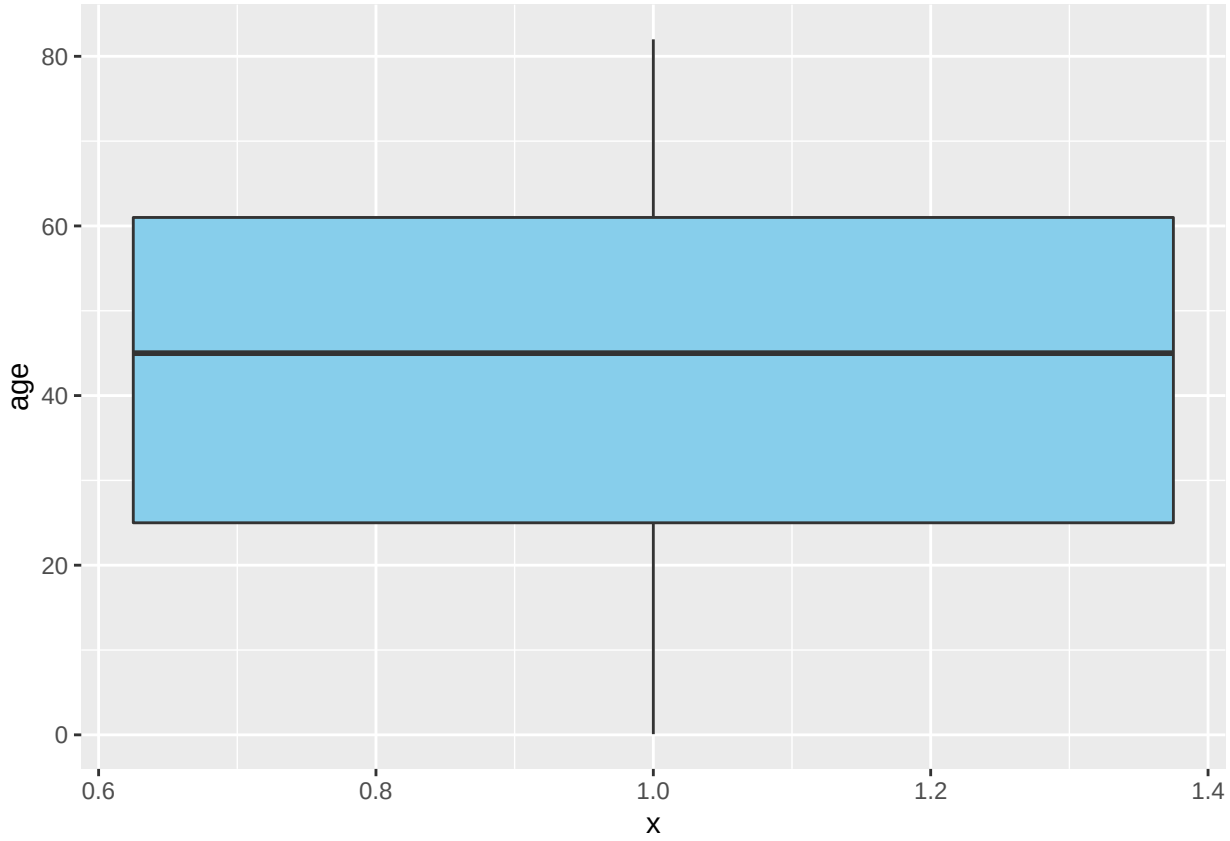
We again get a p-Value more than the significance level of 0.05, hence, we cannot reject the null hypothesis and we can conclude that the two variables are independent.

Age variable Let us now explore the age variable by creating a histogram.



We observe that we have a left skewed histogram. That means that we have more older people in our dataset than younger. We also observe, that we have some really young people (children) in our dataset. Although less common, stroke can happen in children of all ages, even babies.

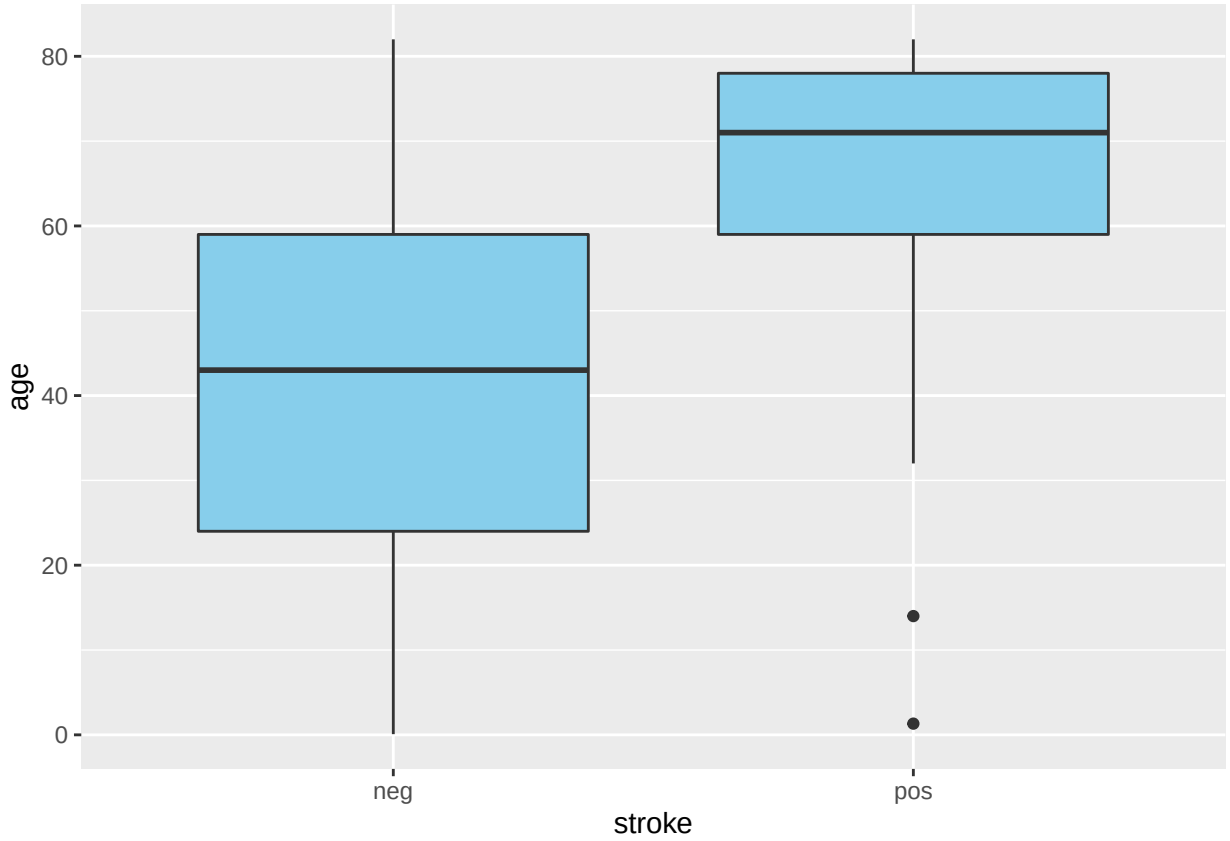
We continue with creating a boxplot and checking for outliers.

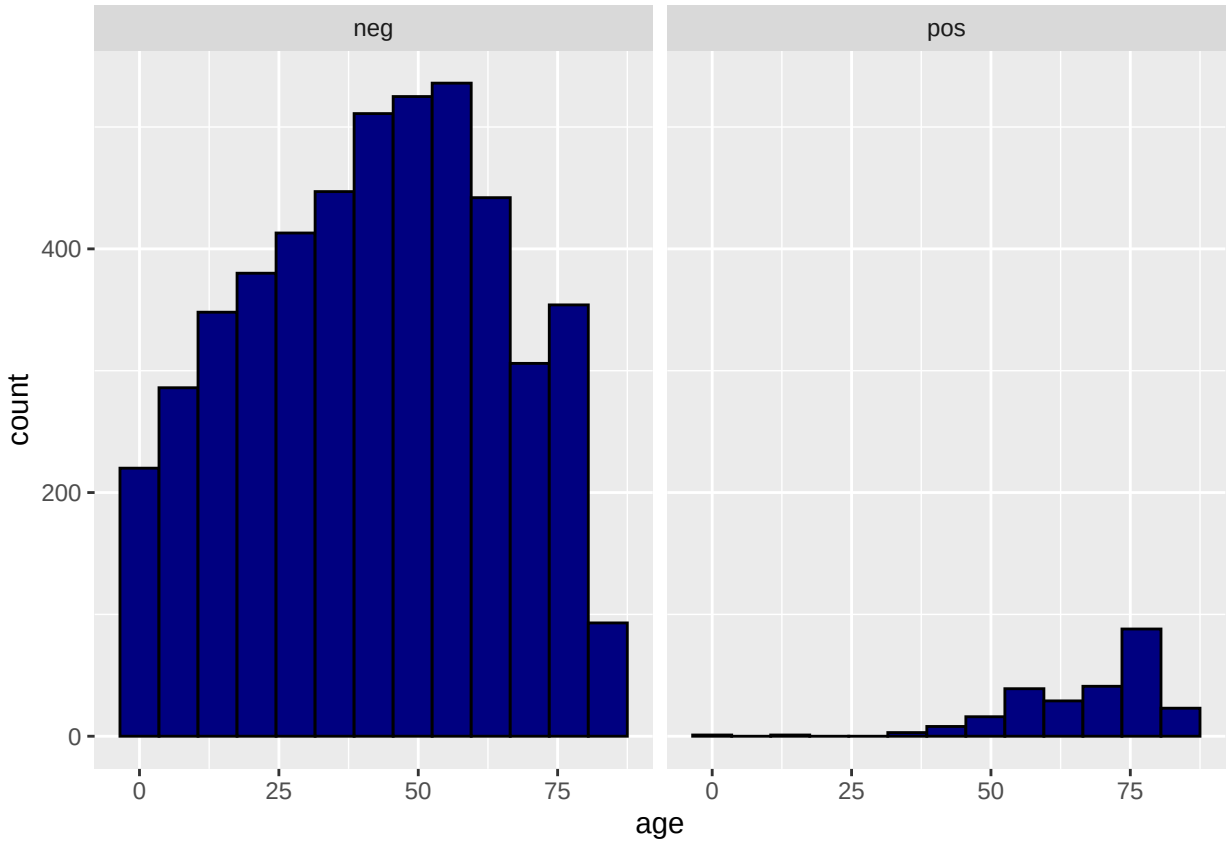



```
boxplot(stroke_temp$age, plot=FALSE)$out
```

```
## numeric(0)
```

We have a median age of 45 and no outliers. Following, we will create a boxplot and an histogram of stroke versus the age.


80
60
40
20
0
age
neg
pos
stroke



From the boxplot, we can clearly understand that the majority of people that are having a stroke are older and from the histogram that both for a positive and a negative case, we have left skewed histograms.

We have seen above that the age distribution is left skewed, hence it does not follow the normal distribution. We can also run a Kolmogorov-Smirnov test to prove this

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  stroke_temp$age
## D = 0.05073, p-value < 2.2e-16
```

We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the the variable is not normally distributed.

As a result from the above, we can use the non parametric Mann Whitney U - Wilcoxon test to check the null hypothesis that people who get a stroke and those that do not get a stroke have on average the same age, therefore the two populations are equal.

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  stroke_temp[stroke_temp$stroke == "pos", "age"] and stroke_temp[stroke_temp$stroke == "neg",
## W = 1010126, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

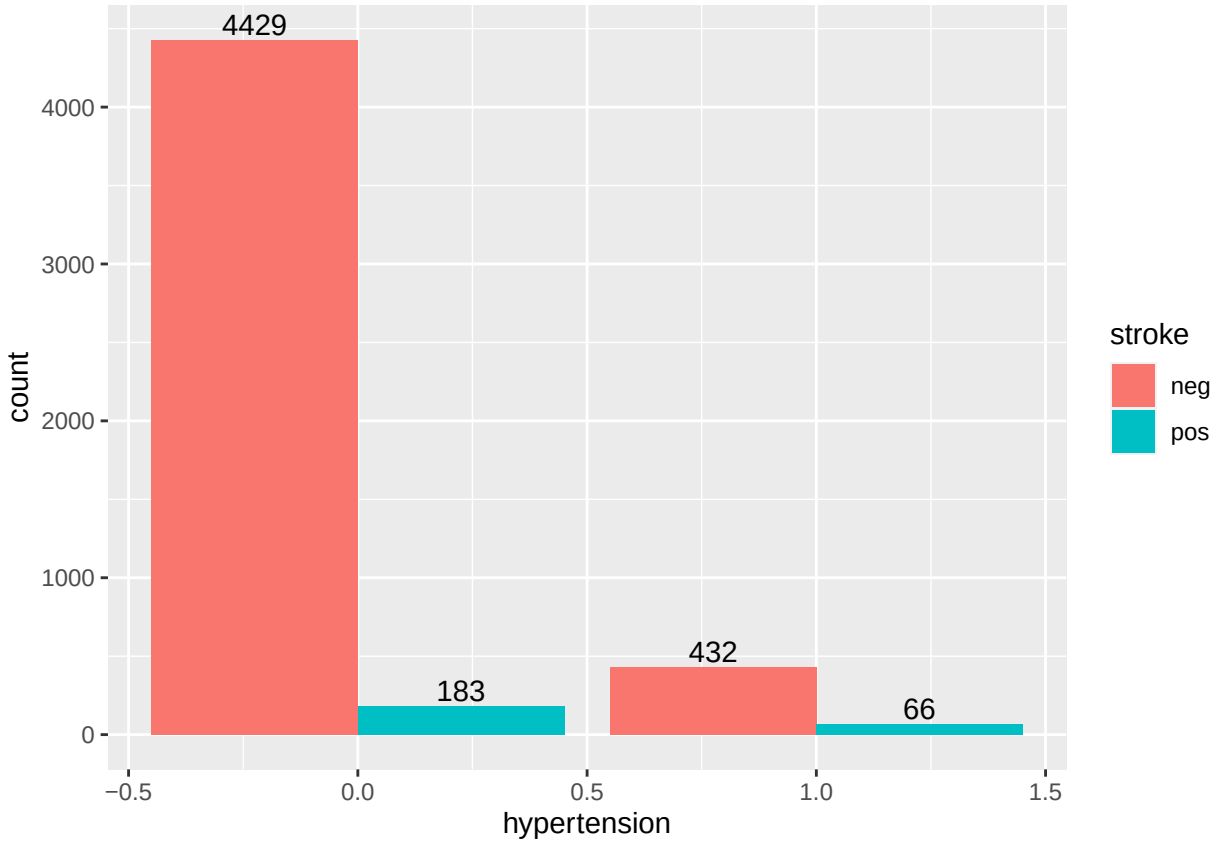
We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the the populations are not equal. Therefore, the age variable appear to be statistically significant.

Hypertension variable As we mentioned in the introduction, the hypertension variable is denoted as an integer where we get 0 if the patient does not have hypertension and 1 if the patient has. Basically, it is a categorical variable for "No" as 0 and "Yes" as 1. Let us create a frequency and a relative table.

```
##
##    0    1
## 4612  498
```

```
##
##     0     1
## 0.903 0.097
```

We observe that more responders do not have hypertension, but how is the stroke variable affected by this. We continue with the creation of a barplot to check the relation of stroke versus the hypertension variable.



Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  stroke_temp$hypertension and stroke_temp$stroke
## X-squared = 81.605, df = 1, p-value < 2.2e-16
```

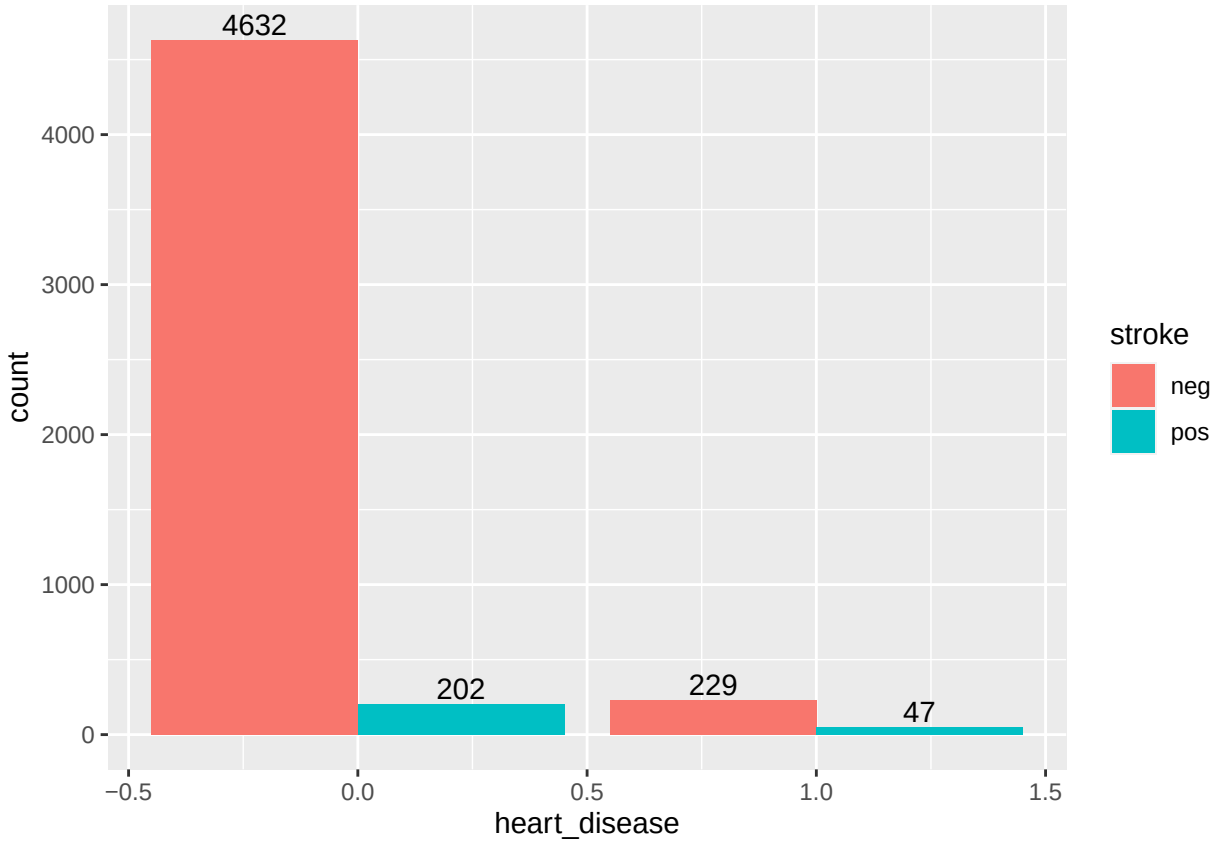
We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the two variables are dependent.

Heart disease The heart_disease variable is denoted as an integer where we get 0 if the patient does not have a heart disease and 1 if the patient has. Basically, it is a categorical variable for "Not having heart disease" as 0 and "Having heart disease" as 1. Let us create a frequency and a relative table.

```
##
##    0    1
## 4834  276
```

```
##
##     0     1
## 0.946 0.054
```

We observe that more responders do not have a heart disease, but how is the stroke variable affected by this. We continue with the creation of a barplot to check the relation of stroke versus the heart_disease variable.



Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  stroke_temp$heart_disease and stroke_temp$stroke
## X-squared = 90.26, df = 1, p-value < 2.2e-16
```

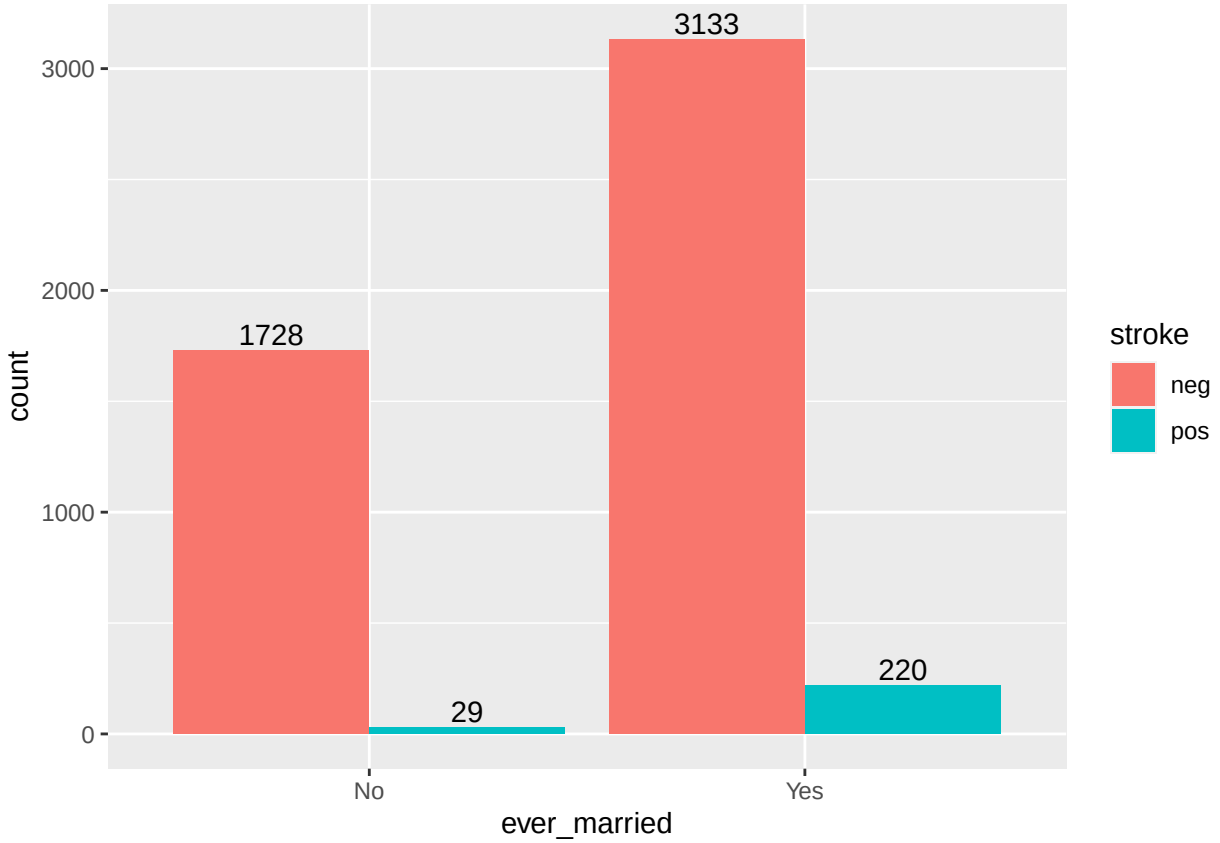
We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the two variables are dependent.

Ever-married variable Here we have a categorical variable with "Yes" denoting that they ar married and "No" that they are not. Let us create a frequency and a relative table.

```
##
##   No  Yes
## 1757 3353
```

```
##
##    No   Yes
## 0.344 0.656
```

We observe that more responders are married, but how is the stroke variable affected by this. We continue with the creation of a barplot to check the relation of stroke versus the ever_married variable.



Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  stroke_temp$ever_married and stroke_temp$stroke
## X-squared = 58.924, df = 1, p-value = 1.639e-14
```

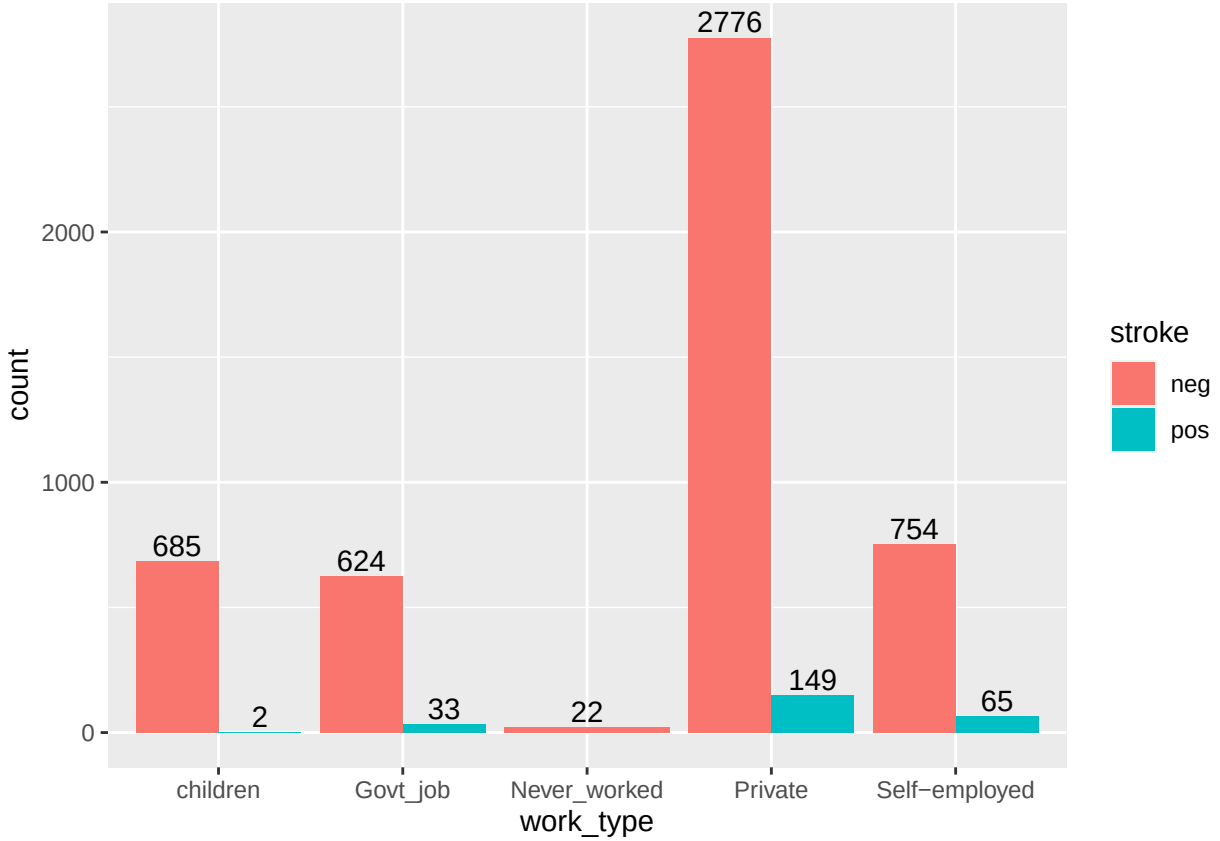
We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the two variables are dependent.

Work-type variable Work-type is a categorical variable. Let us create a frequency and a relative table of the work_type variable and check its categories.

```
##
##      children      Govt_job  Never_worked       Private Self-employed
##           687           657            22          2925           819
```

```
##
##      children      Govt_job  Never_worked       Private Self-employed
##         0.134         0.129         0.004         0.572         0.160
```

We observe that in our dataset we have responders in five categories: "children" for the children, "Govt_job" for those working in government jobs, "Never_worked" for those that they have never worked, "Private" for those working in private companies and finally, the "Self-employed" category. However, how is the stroke variable affected by the work_type variable? We continue with the creation of a barplot to check the relation of stroke versus the work_type variable.



Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
##  Pearson's Chi-squared test
##
## data:  stroke_temp$work_type and stroke_temp$stroke
## X-squared = 49.164, df = 4, p-value = 5.398e-10
```

We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the two variables are dependent. However, because in some cases, like "children that had a stroke", we have very few observations, this may affect our chi_square test results. For this reason we will also run a Fisher's exact test, with the same null hypothesis as above, H_0: the two variables are independent.

```
##
##  Fisher's Exact Test for Count Data with simulated p-value (based on
##  2000 replicates)
##
## data:  stroke_temp$work_type and stroke_temp$stroke
## p-value = 0.0004998
## alternative hypothesis: two.sided
```

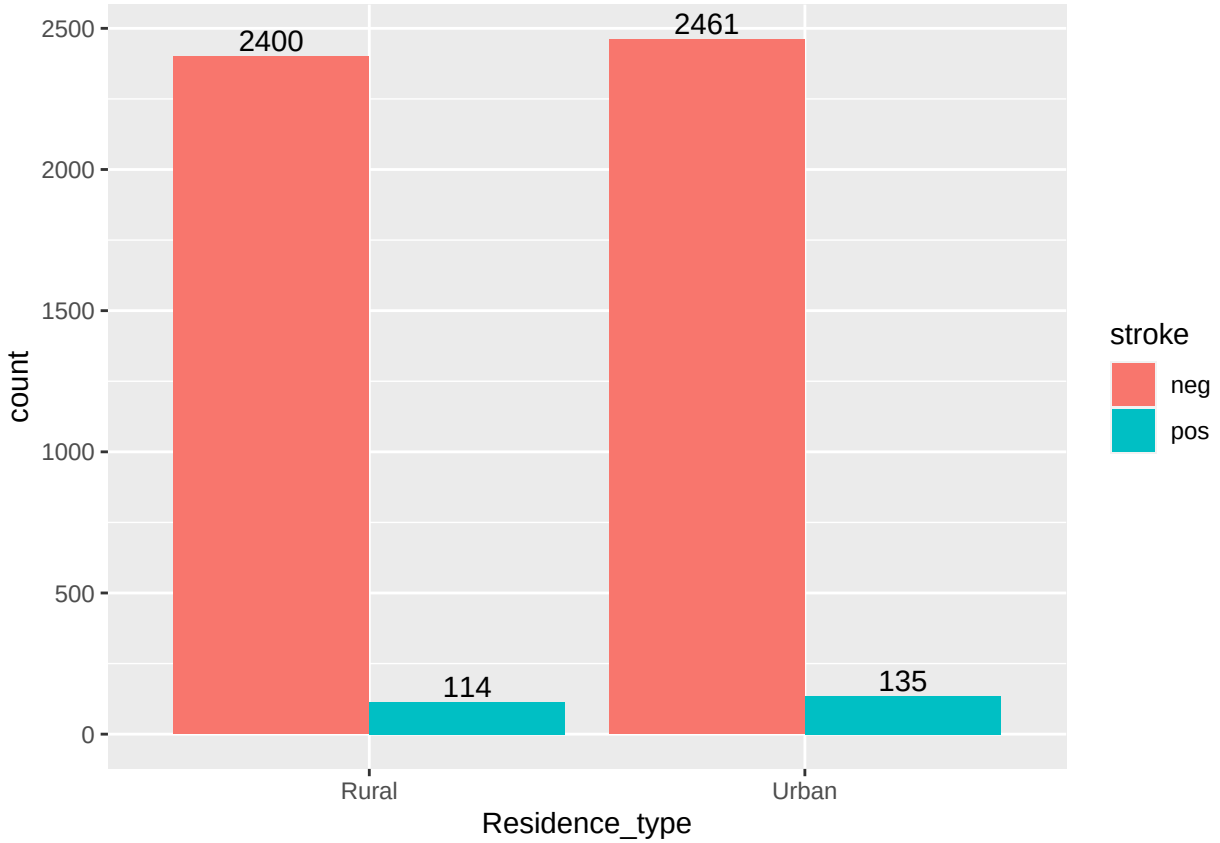
We again get a p-Value more than the significance level of 0.05, hence, we cannot reject the null hypothesis and we can conclude that the two variables are independent.

Residence type variable We are dealing again with a categorical variable with categories: "Rural" to denote those living in rural areas and "Urban" to denote those living in urban areas. Let us create a frequency and a relative table.

```
##
## Rural Urban
##  2514  2596
```

```
##
## Rural Urban
## 0.492 0.508
```

We observe that those that live in rural area are almost the same with those that are not. However, how is the stroke variable affected by this? We continue with the creation of a barplot to check the relation of stroke versus the Residence_type variable.



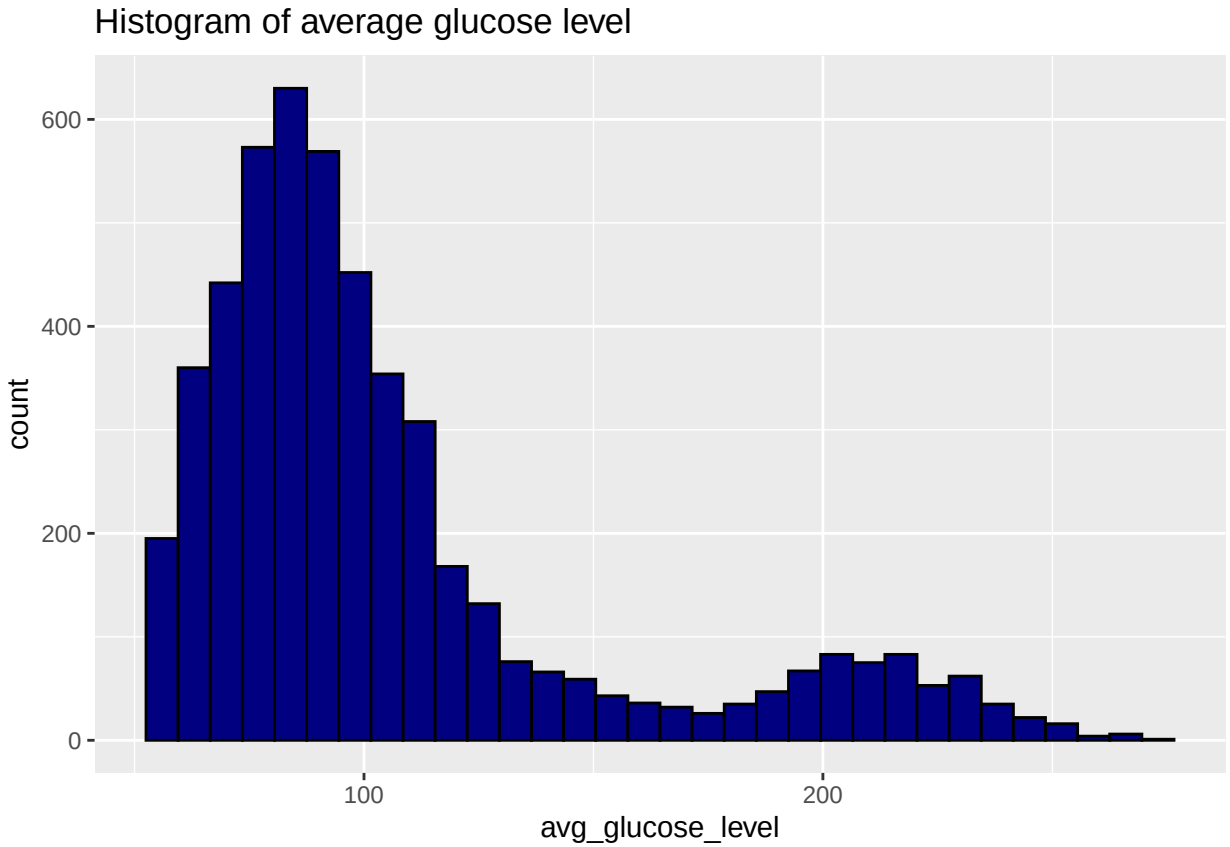
Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  stroke_temp$Residence_type and stroke_temp$stroke
## X-squared = 1.0816, df = 1, p-value = 0.2983
```

We get a p-Value more than the significance level of 0.05, hence, we cannot reject the null hypothesis and conclude that the two variables are independent.

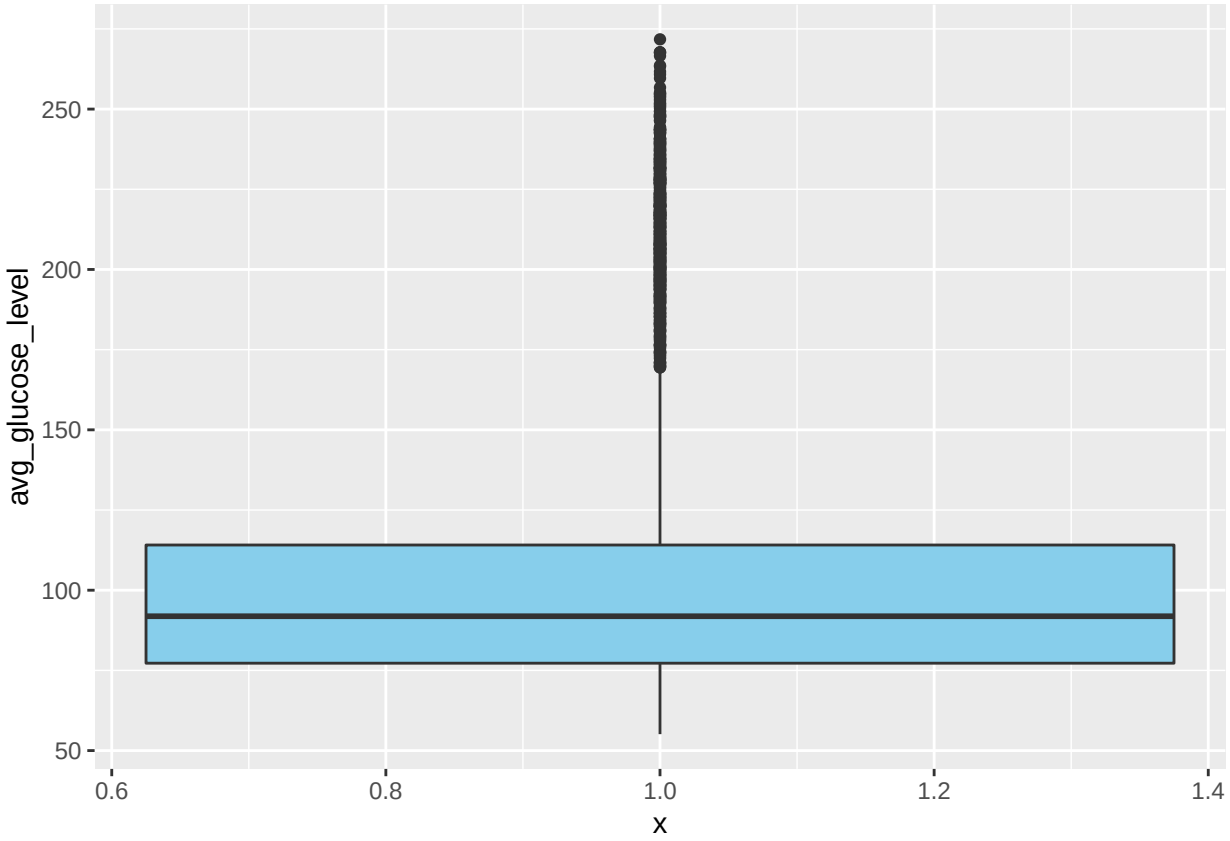
Average glucose level variable With the variable average glucose level, basically we measure the blood sugar. Blood glucose is one of the basic forms of energy used by the cells and make up the muscles and tissues of our body to function properly. Glucose comes from a large part from what we eat, from our diet. The usual procedure is that, after a meal, the blood sugar levels rise and our body responds to this increase in sugar by secreting insulin which get the levels of sugar back to normal by ensuring that glucose is absorbed by the cells and tissues of our body. However, sometimes our body does not make enough insulin or does not use insulin well and when this happens, glucose stay in our blood and does not reach the cells.

Let us now explore the avg_glucose_level variable by creating a histogram.



We observe that we have a right skewed histogram. That means that we have more people with normal glucose level in our dataset than people that they do not.

We continue with creating a boxplot and checking for outliers.




```
boxplot(stroke_temp$avg_glucose_level, plot=FALSE)$out%>%head(15)
```

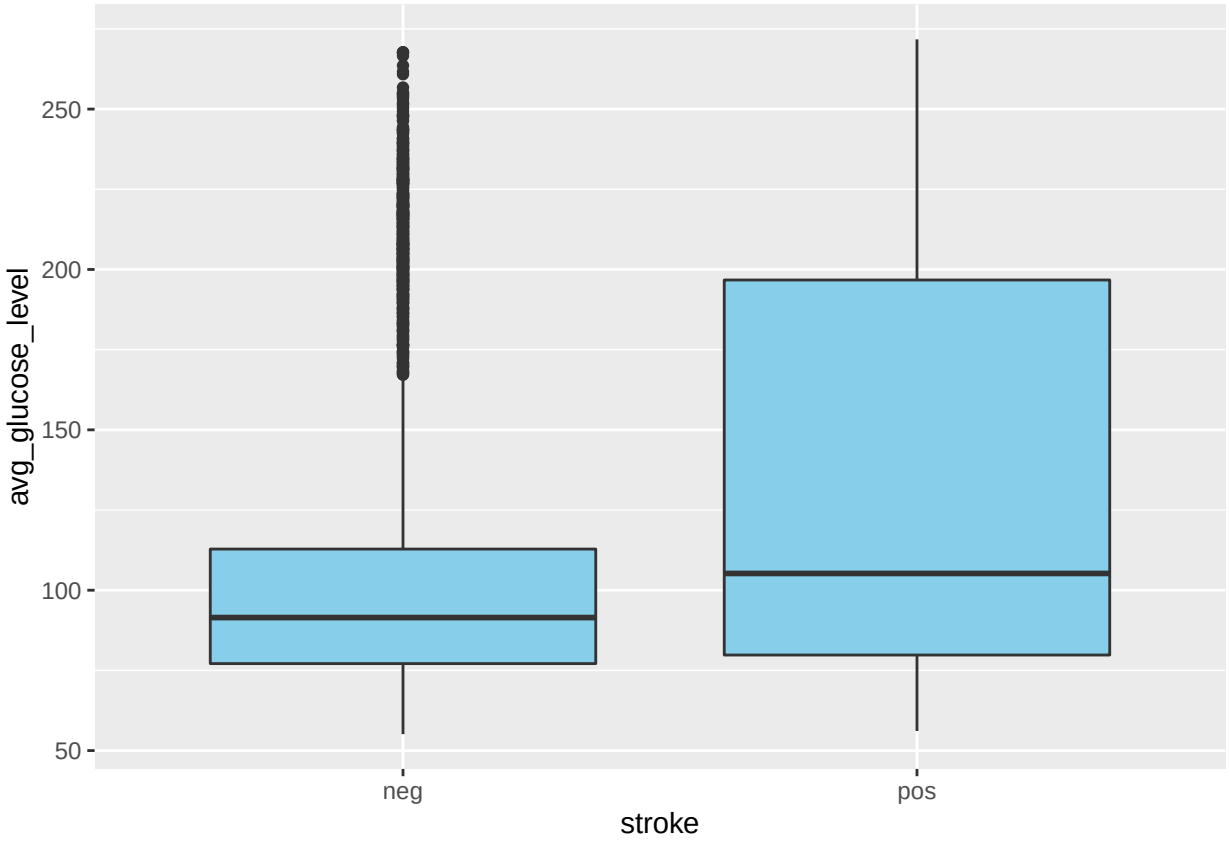
```
##  [1] 228.69 202.21 171.23 174.12 186.21 219.84 214.09 191.61 221.29 217.08
## [11] 193.94 233.29 228.70 208.30 189.84
```

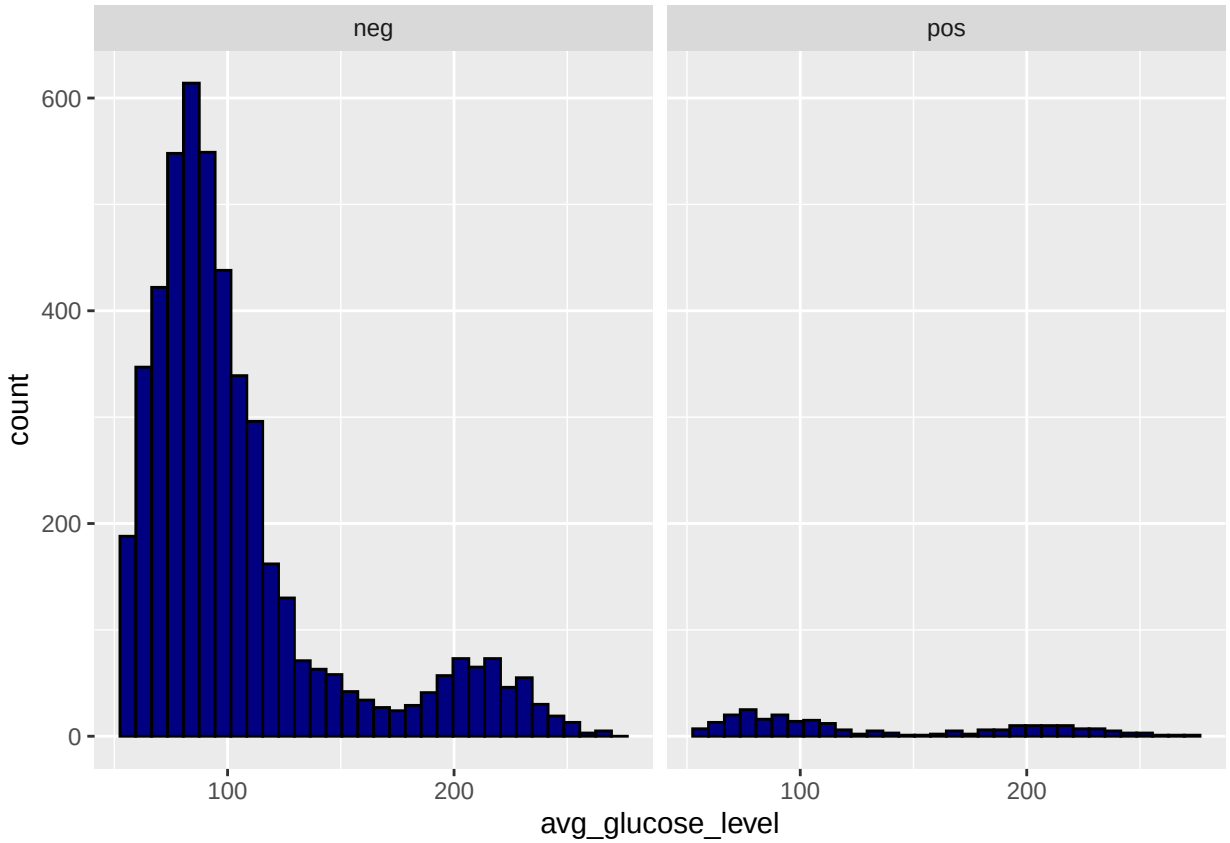
```
max(stroke_temp$avg_glucose_level)
```

```
## [1] 271.74
```

We have a median glucose level of around 92 mg/dL and many natural outliers. These high levels of blood glucose could happen to people that they have just eaten and they have measured their blood glucose. Moreover, people that have prediabetes or diabetes could have high levels of blood sugar.

Following, we will create a boxplot and an histogram of stroke versus the avg_glucose_level.


avg_glucose_level
250
200
150
100
50
neg
pos
stroke



From the boxplot, we can clearly understand that the majority of people that are having a stroke do have higher glucose levels and from the histogram that both for a positive and a negative case, we have right skewed histograms.

We have seen above that the avg_glucose_level distribution is right skewed, hence it does not follow the normal distribution. We can also run a Kolmogorov-Smirnov test to prove this

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  stroke_temp$avg_glucose_level
## D = 0.18284, p-value < 2.2e-16
```

We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the the variable is not normally distributed.

As a result from the above, we can use the non parametric Mann Whitney U - Wilcoxon test to check the null hypothesis that people who get a stroke and those that do not get a stroke have on average the same glucose levels, therefore the two populations are equal.

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  stroke_temp[stroke_temp$stroke == "pos", "avg_glucose_level"] and stroke_temp[stroke_temp$stro
## W = 739150, p-value = 3.64e-09
## alternative hypothesis: true location shift is not equal to 0
```

We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the the populations are not equal. Therefore, the avg_glucose_level variable appear to be statistically significant.

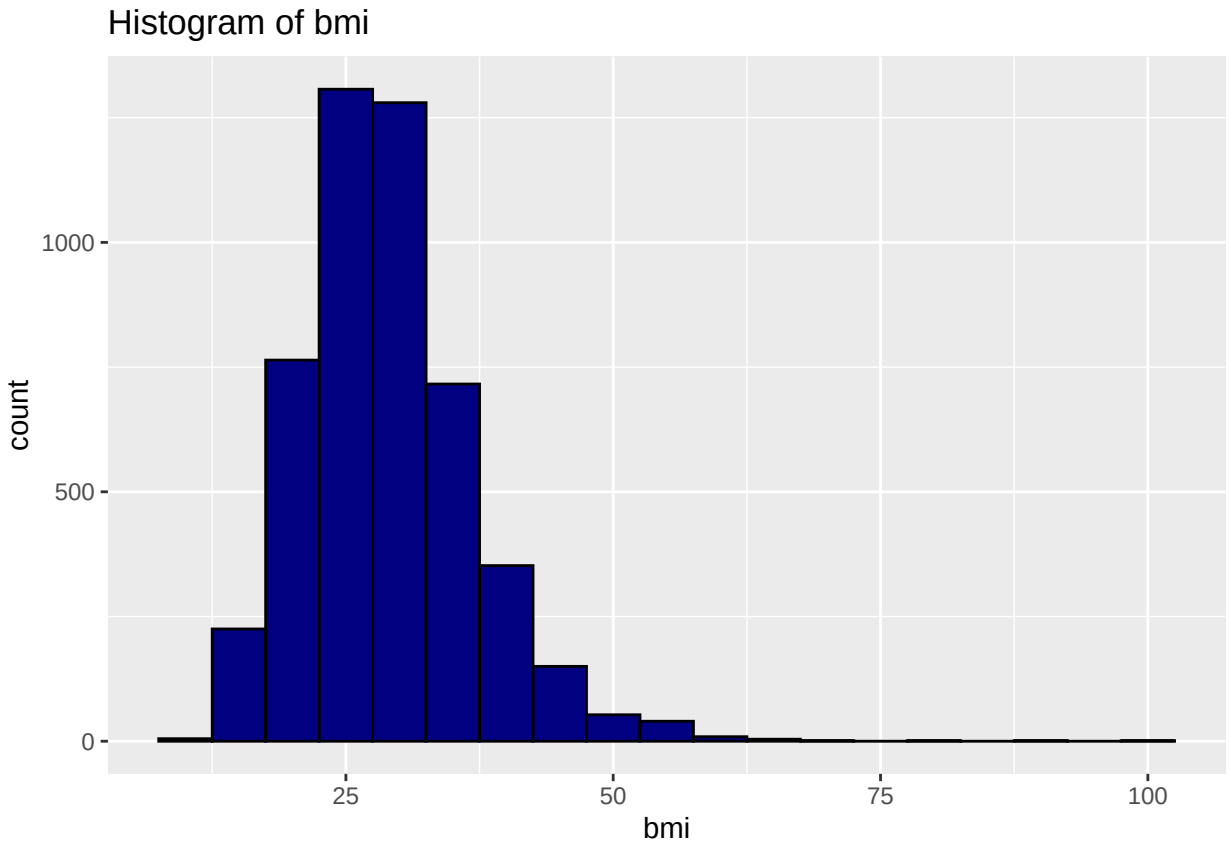
BMI variable The BMI stands for the Body mass index which is measured in (weight in kg/(height in m)^2). Therefore, this variable should be a numerical variable and not a categorical. We will convert it to numeric.

```
stroke_temp$bmi<-as.numeric(stroke_temp$bmi)
```

```
## Warning: NAs introduced by coercion
```

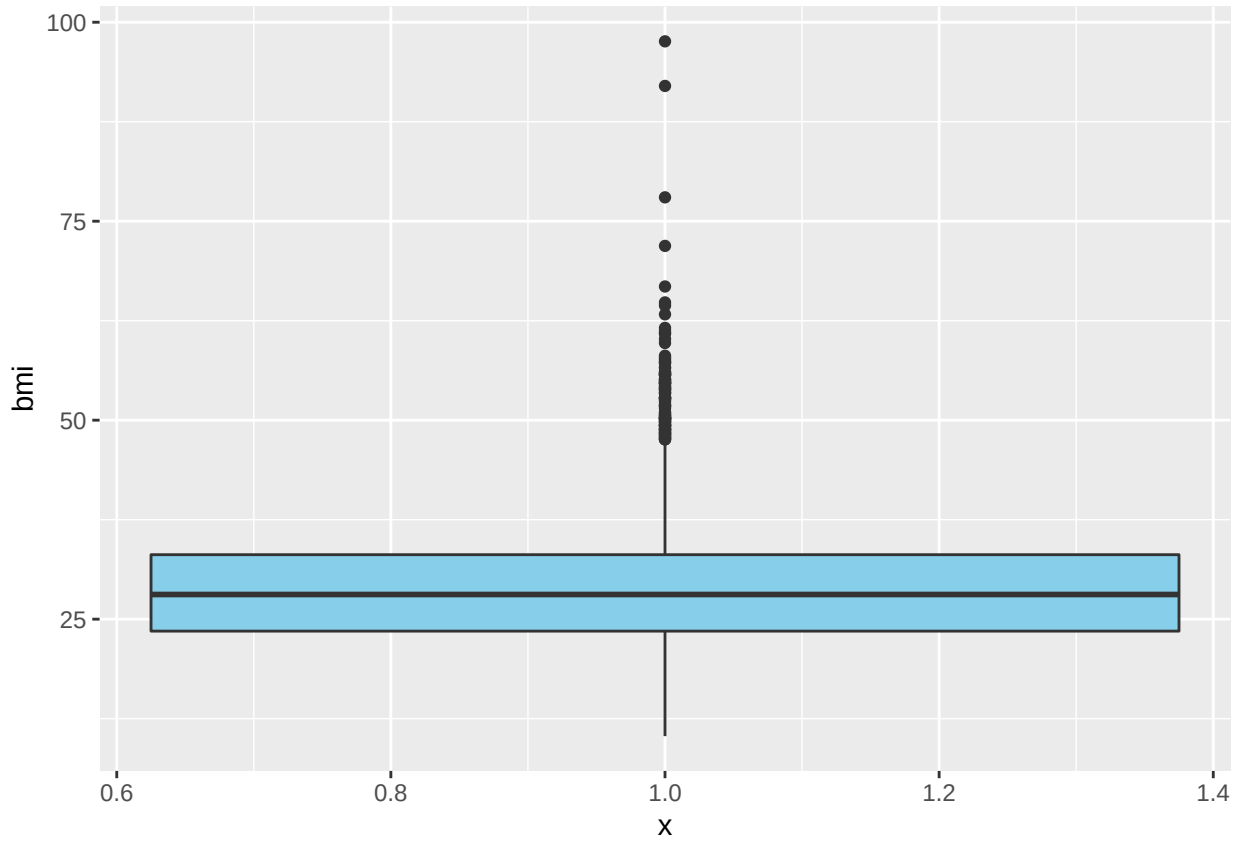
From the above, we get the message that "NAs introduced by coercion", hence we have missing values.

Let us now explore the bmi variable by creating a histogram.



We observe that we have a right skewed histogram. We know that the normal weight is between 18.5 – 24.9 kg/m^2, between 25 and 29.9 kg/m^2 are overweight and the Obese range is higher that 30.0 kg/m^2. The obese category is further divided to obese(Class I)(BMI 30 to 34.9), Obese (Class II) (BMI 35 to 39.9) and Morbidly Obese (Class III) (BMI 40 or more). That means that we have more people with normal and overweight body mass index in our dataset than people that are obese. We do not have to forget that we also have children in our dataset, hence their bosy mass index appear as underweight.

We continue with creating a boxplot and checking for outliers.

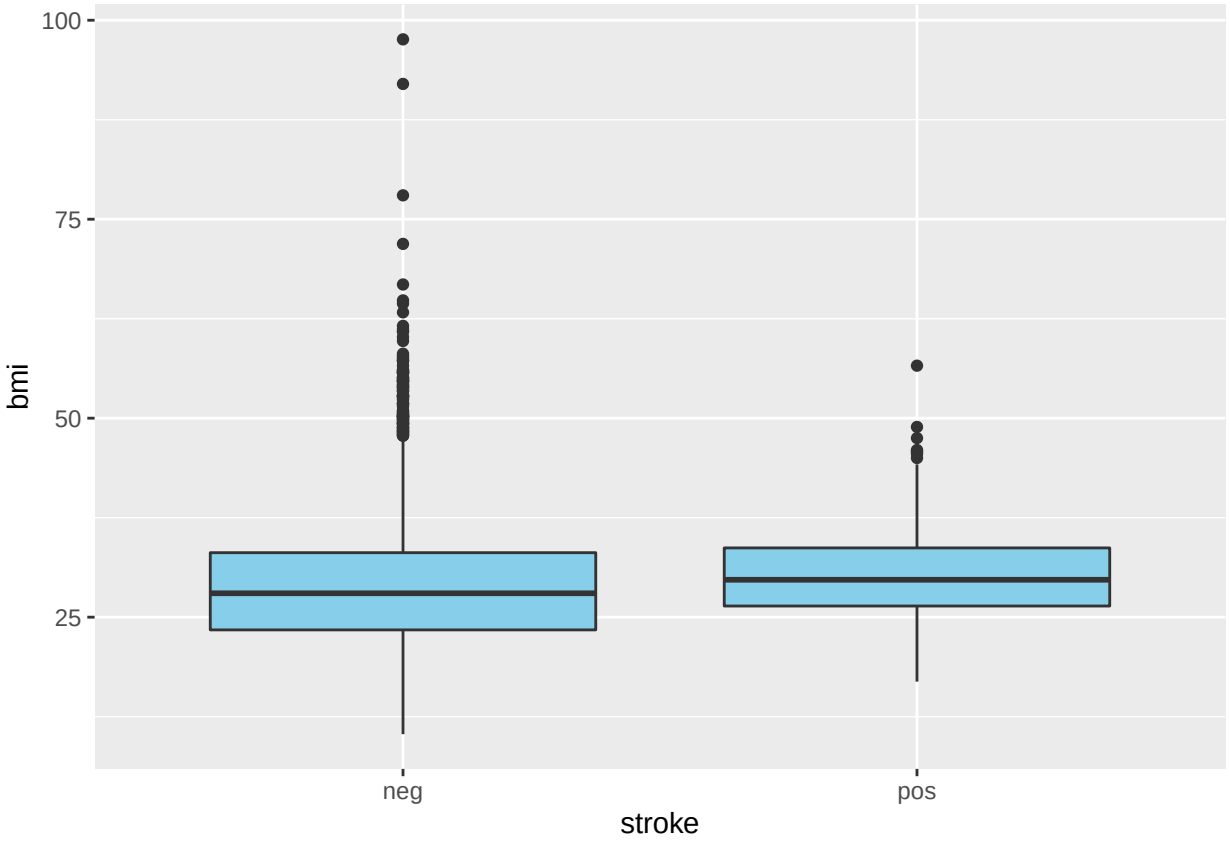


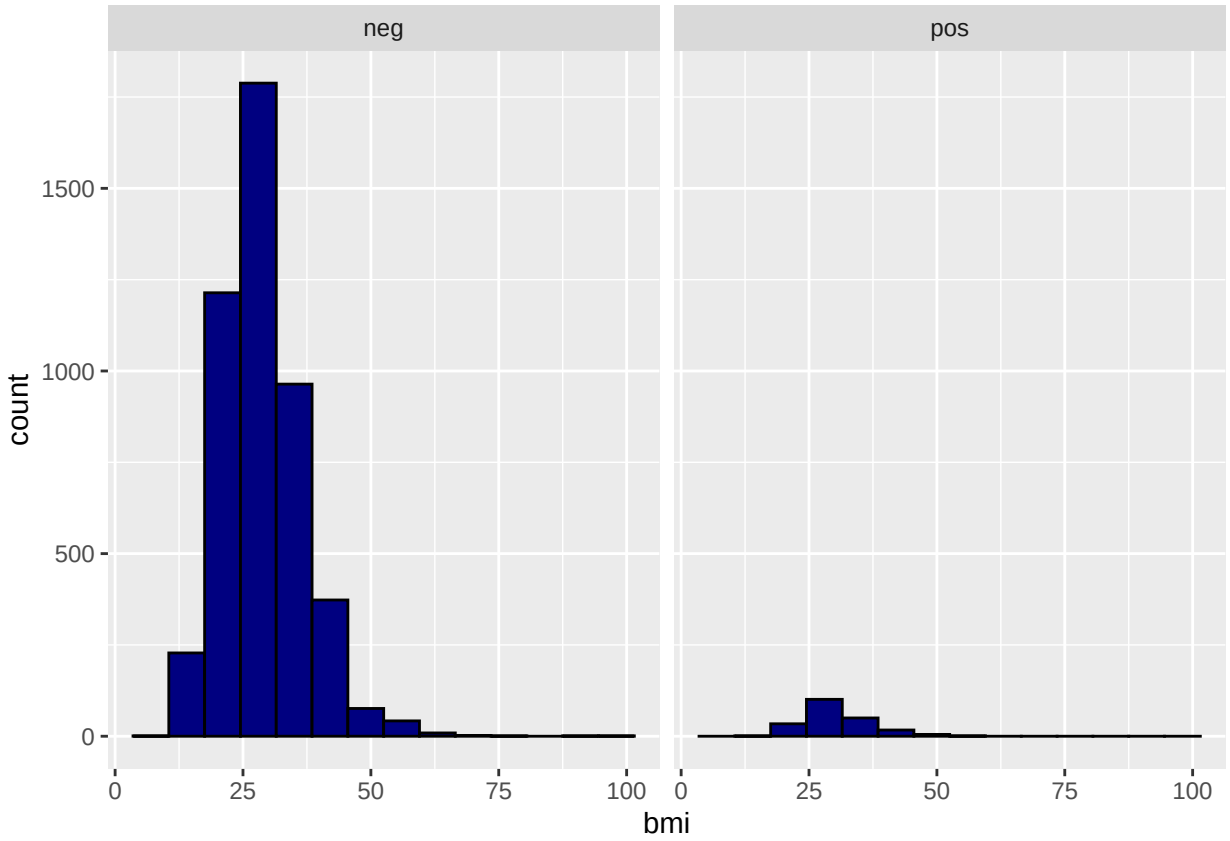

```
boxplot(stroke_temp$bmi, plot=FALSE)$out%>%sort(decreasing=TRUE)
```

```
##   [1] 97.6 92.0 78.0 71.9 66.8 64.8 64.4 63.3 61.6 61.2 60.9 60.9 60.2 59.7 58.1
##  [16] 57.9 57.7 57.5 57.3 57.2 57.2 56.6 56.6 56.1 56.0 55.9 55.9 55.7 55.7 55.7
##  [31] 55.7 55.2 55.1 55.0 55.0 54.8 54.7 54.7 54.7 54.6 54.6 54.3 54.2 54.1 54.0
##  [46] 53.9 53.8 53.8 53.5 53.4 53.4 52.9 52.8 52.8 52.8 52.7 52.7 52.5 52.3 51.9
##  [61] 51.9 51.8 51.7 51.5 51.0 50.9 50.8 50.6 50.6 50.5 50.4 50.3 50.3 50.2 50.2
##  [76] 50.2 50.2 50.1 50.1 49.9 49.8 49.8 49.8 49.5 49.5 49.4 49.3 49.3 49.3 49.2
##  [91] 48.9 48.9 48.9 48.8 48.8 48.7 48.5 48.5 48.4 48.3 48.3 48.2 48.1 48.0 47.9
## [106] 47.8 47.8 47.6 47.6 47.6
```

We have a median bmi of around 28 and many outliers. These outliers may be natural, some people may fall into the morbidly obesity class; there are records of people with BMI more than 100. However, they also may be data entry errors.

Following, we will create a boxplot and an histogram of stroke versus the bmi variable.


100
75
50
25
bmi
neg
pos
stroke



From the boxplot, we can observe that people that are having a stroke and those that do not, have almost the same median and around the same interquartile range. Moreover, from the histogram we see that for both a positive and a negative case, we have right skewed histograms.

We have seen above that the bmi distribution is right skewed, hence it does not follow the normal distribution. We can also run a Kolmogorov-Smirnov test to prove this

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  stroke_temp$bmi
## D = 0.058788, p-value < 2.2e-16
```

We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the the variable is not normally distributed.

As a result from the above, we can use the non parametric Mann Whitney U - Wilcoxon test to check the null hypothesis that people who get a stroke and those that do not get a stroke have on average the same body mass index, therefore the two populations are equal.

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  stroke_temp[stroke_temp$stroke == "pos", "bmi"] and stroke_temp[stroke_temp$stroke == "neg",
## W = 569022, p-value = 0.0001026
## alternative hypothesis: true location shift is not equal to 0
```

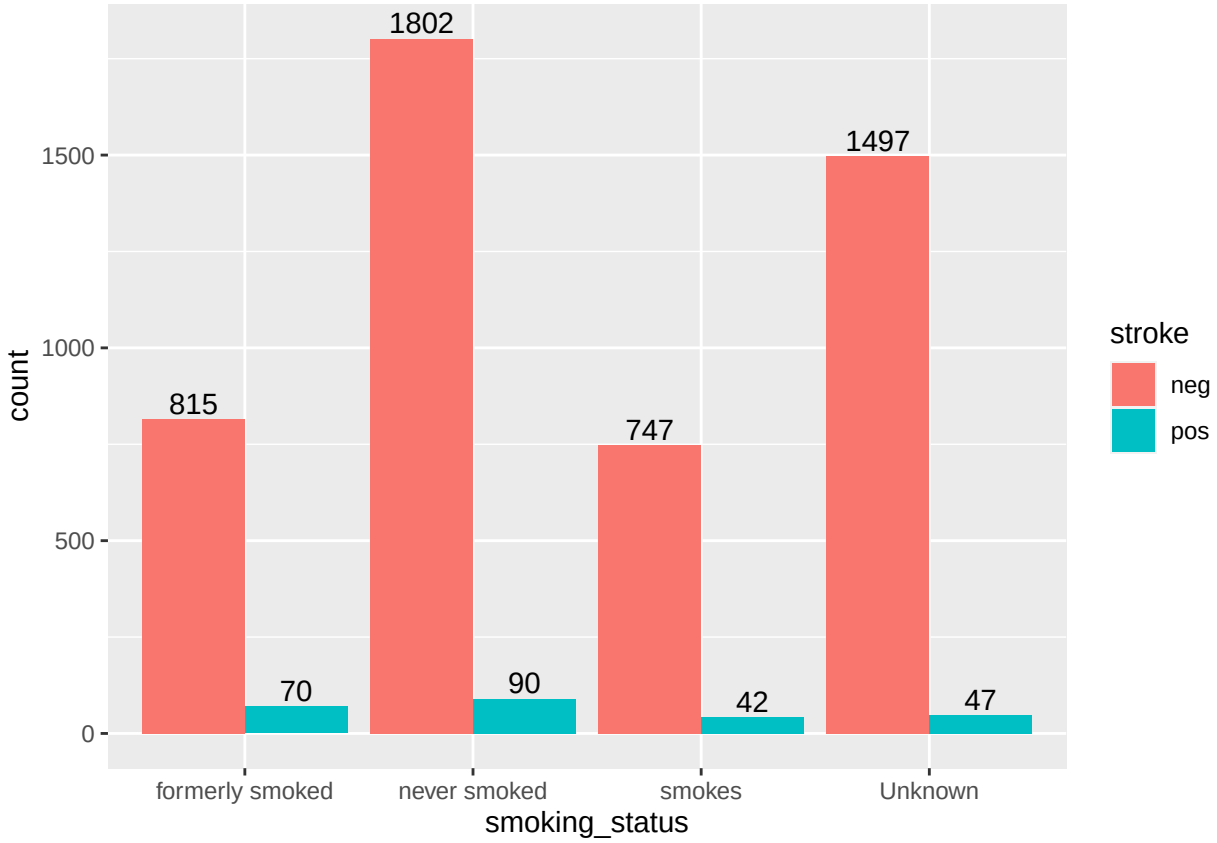
We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the the populations are not equal. Therefore, the bmi variable appear to be statistically significant.

Smoking status variable Finally, let us check the smoking status variable. smoking_status is a categorical variable. Let us create a frequency and a relative table of the work_type variable and check its categories.

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

```
##
## formerly smoked    never smoked          smokes         Unknown
##           0.173           0.370           0.154           0.302
```

We observe that in our dataset we have responders in four categories: "formerly smoked", "never smoked", "smokes", "unknown". "Unknown" is basically not available information. However, how is the stroke variable affected by the smoking_status variable? We continue with the creation of a barplot to check the relation of stroke versus the smoking_status variable.



Since both our variables are categorical, we will run a chi_square test in order to check if the variables are independent or not, with null hypothesis H_0: the two variables are independent.

```
##
##  Pearson's Chi-squared test
```

```
##
## data:  stroke_temp$smoking_status and stroke_temp$stroke
## X-squared = 29.147, df = 3, p-value = 2.085e-06
```

We get a p-Value less than the significance level of 0.05, hence, we reject the null hypothesis and conclude that the two variables are dependent.

Multicollinearity analysis In order to build a model in which the predictors will explain the response variable in the best and most reliable way, it is very important to check that these predictors are not correlated with each other. While correlation between a feature and the response variable is an indication that our model will have better predictability, correlation among the predictors/features is a problem. This has to be corrected, if we want to end up with a reliable model. We will use the ggpairs function of the GGally package in order to check the correlations between the features

We can see the scatterplots of each pair of the numeric variables (on the left part of the figure) and the Pearson correlation is displayed on the right. All the scatterplots associated with the variables hypertension and heart_disease are like two columns because as we mentioned above, they are basically categorical variables. Also, on the diagonal the variable distribution, the density plot, is available. From there, we get that all our variables do not follow the normal distribution. Furthermore, since we also have a response variable, at the bottom right diagonal, we get the barplots for the two classes of the stroke. At the last line, we get the histograms of each class of the stroke versus the features. Finally, at the last column we get the boxplots of the stroke variable versus the features.

From the plot, we observe that there is not correlation or there is a very low positive correlation for all the variables.

Final selected features From the feature selection procedure that we have seen above, we conclude that we will remove from our dataset the variables id, gender, Residence_type. Therefore, we get the below dataset.

```
stroke_final <- stroke_temp[,c(3:7, 9:12)]
```

Furthermore, we have also the issue that most machine learning algorithms only work with numeric values. Since these categorical features cannot be directly used in most machine learning algorithms, the categorical features need to be transformed into numerical features. Therefore, we will proceed with the following:

```
stroke_final$ever_married<-as.integer(factor(stroke_final$ever_married,
                                             levels=c("No","Yes")))
```

```
stroke_final$work_type<-as.integer(factor(stroke_final$work_type,
                                          levels=c("children","Govt_job",
                                                   "Never_worked","Private",
                                                   "Self-employed")))
```

```
stroke_final$smoking_status<-as.integer(factor(stroke_final$smoking_status,
                                               levels=c("formerly smoked", "never smoked",
                                                        "smokes","Unknown")))
```

However, we have seen that the class "Unknown" is basically not available information. Therefore, we can put the class value 4 for as NAs.

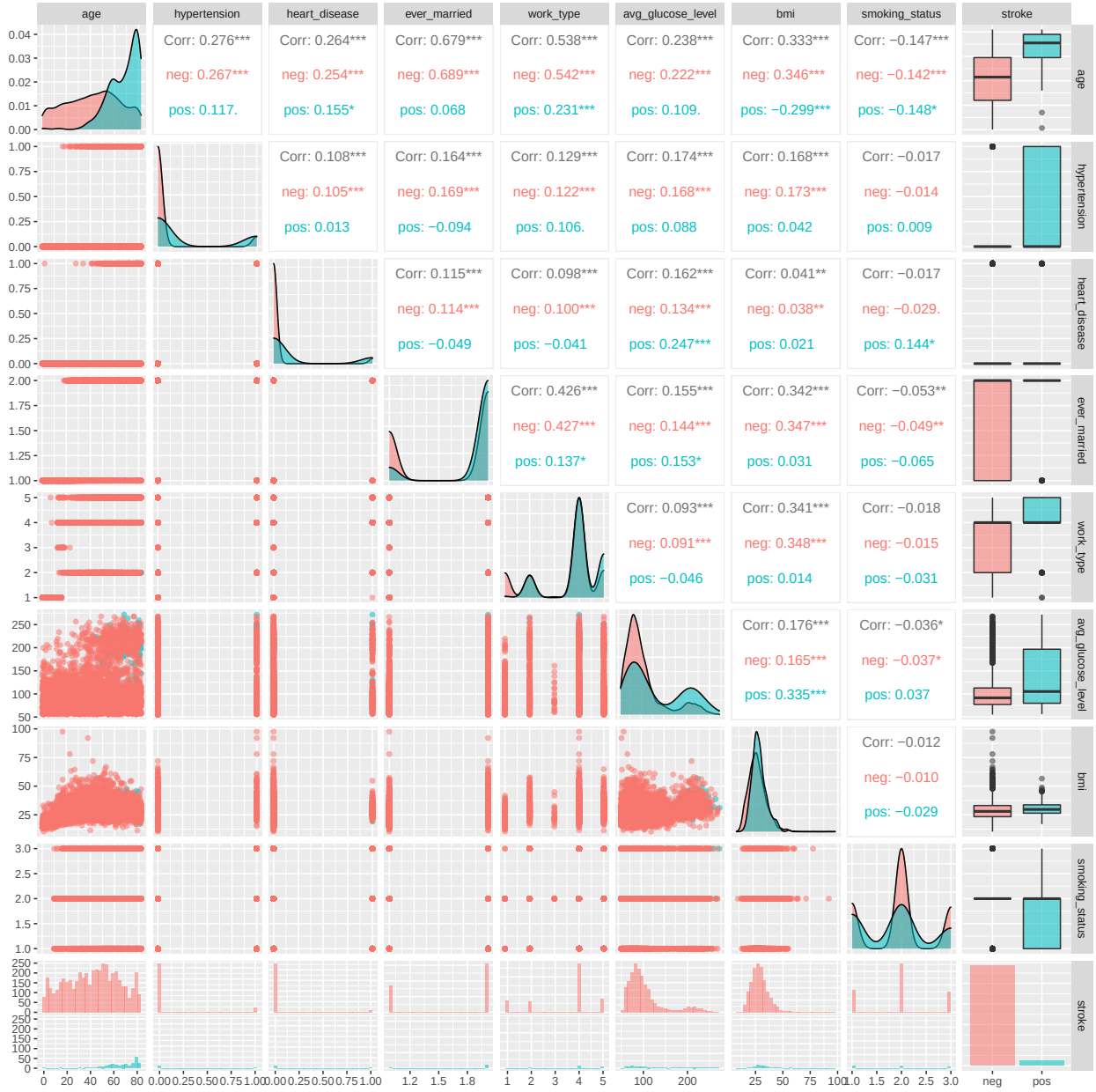
```
stroke_final$smoking_status[stroke_final$smoking_status==4] <-NA
```

Let us now have a look of the structure of our final dataset.

```
str(stroke_final)
```

```
## 'data.frame':    5110 obs. of  9 variables:
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : int  2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : int  4 5 4 4 5 4 4 4 4 4 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 NA 32.5 34.4 24 29 27.4 22.8 NA 24.2 ...
##  $ smoking_status   : int  1 2 2 3 2 1 2 2 NA NA ...
##  $ stroke           : Factor w/ 2 levels "neg","pos": 2 2 2 2 2 2 2 2 2 2 ...
```

Now that we have converted all our variables to numerical we can check again the correlations between the features.



From the plot, we observe again that there is a very low positive or negative correlation (in some cases not even a correlation) between almost all the variables. Exception are the wotk_type - age that have low positive correlation and the ever_married - age that have a positive correlation. In general, we accept that an absolute correlation of over >0.7 among two or more features indicates the presence of multicollinearity.

Normalization of the predictors

We have seen that our data has varying scales. If we do not normalize the data, the variables that use a larger scale will dominate over the machine learning algorithm, causing to negatively affect the model performance. Moreover, some algorithms used to build models (such as k-nearest neighbors) do not make assumptions regarding the distribution of the data.Therefore, we will normalize our data.

```
stroke_final[,c(1:8)] <- as.data.frame(scale(stroke_final[,c(1:8)]))
```

Data splitting

Now it is the time of splitting our data.

Train_temp and validation set partition Our dataset is small, we only have 5110 observations. Therefore, the splitting of the dataset should be done in a way that we will have enough observations for the train set, for the validation set and for the test set. If we do not have enough training data, then the machine learning model will have high variance while training. If we do not have enough testing data/validation data, our model evaluation/model performance statistic will have greater variance. Here, we firstly split our data in a ratio 80:20 for the temporary training set and the validation set. The validation set will only be used to test our final model after completing the training. Later on, we will also split the temporary training set, to a test set and a training set.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = stroke_final$stroke, times = 1, p = 0.2, list = FALSE)
train_temp <- stroke_final[-test_index,]
validation_set <- stroke_final[test_index,]
```

Train set and test set partition Let us continue with splitting our train_temp data to a train set and a test set. For the reasons mentioned previously, we will split our data to a 80:20 ratio for the train set and the test set respectively. We will use these two sets during the building of our machine learning algorithm.

```
set.seed(123, sample.kind="Rounding")
test_index1 <- createDataPartition(y = train_temp$stroke, times = 1, p = 0.2, list = FALSE)
train_set <- train_temp[-test_index1,]
test_set <- train_temp[test_index1,]
```

Data exploration

We will now use the train_set dataset in order to explore our data and gain some useful insights. Let us check the structure of our data and the summary of our data.

```
str(train_set)
```

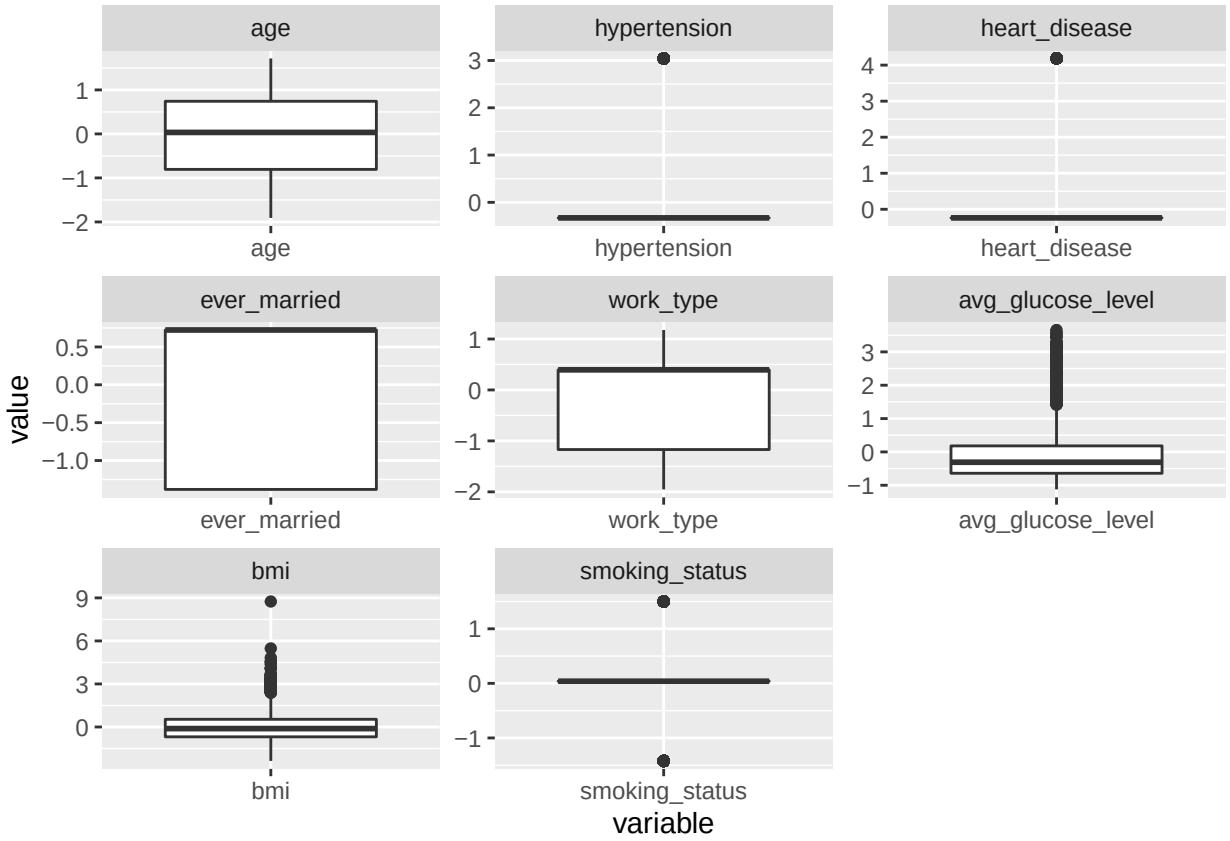
```
## 'data.frame':    3269 obs. of  9 variables:
##  $ age              : num  1.051 1.626 0.255 1.361 1.14 ...
##  $ hypertension     : num  -0.329 -0.329 -0.329 3.043 -0.329 ...
##  $ heart_disease    : num  4.185 4.185 -0.239 4.185 -0.239 ...
##  $ ever_married     : num  0.724 0.724 0.724 0.724 -1.381 ...
##  $ work_type        : num  0.395 0.395 0.395 0.395 0.395 ...
##  $ avg_glucose_level: num  2.70611 -0.00503 1.43722 -0.79626 -0.25965 ...
##  $ bmi              : num  0.981 0.459 0.701 -0.19 -0.776 ...
##  $ smoking_status   : num  -1.4211 0.0393 1.4998 0.0393 0.0393 ...
##  $ stroke           : Factor w/ 2 levels "neg","pos": 2 2 2 2 2 2 2 2 2 2 ...
```

```
summary(train_set)
```

```
##       age            hypertension      heart_disease        ever_married
##  Min.   :-1.90807   Min.   :-0.32857   Min.   :-0.238923   Min.   :-1.381301
##  1st Qu.:-0.80604   1st Qu.:-0.32857   1st Qu.:-0.238923   1st Qu.:-1.381301
##  Median : 0.03420   Median :-0.32857   Median :-0.238923   Median : 0.723813
##  Mean   :-0.01504   Mean   :-0.01195   Mean   :-0.002117   Mean   : 0.001287
##  3rd Qu.: 0.74177   3rd Qu.:-0.32857   3rd Qu.:-0.238923   3rd Qu.: 0.723813
##  Max.   : 1.71468   Max.   : 3.04290   Max.   : 4.184622   Max.   : 0.723813
##
##    work_type         avg_glucose_level        bmi           smoking_status
##  Min.   :-1.951847   Min.   :-1.124639   Min.   :-2.36734   Min.   :-1.4211
##  1st Qu.:-1.169700   1st Qu.:-0.641020   1st Qu.:-0.68668   1st Qu.: 0.0393
##  Median : 0.394594   Median :-0.309995   Median :-0.11373   Median : 0.0393
##  Mean   :-0.007605   Mean   :-0.000155   Mean   :-0.00268   Mean   : 0.0214
##  3rd Qu.: 0.394594   3rd Qu.: 0.180470   3rd Qu.: 0.53562   3rd Qu.: 0.0393
##  Max.   : 1.176741   Max.   : 3.656787   Max.   : 8.74792   Max.   : 1.4998
##                                          NA's   :145        NA's   :990
##  stroke
##  neg:3110
##  pos: 159
##
##
##
##
##
```

We get some statistics for our variables including the mean, the median and the minimum value after their normalization. We also observe that the bmi variable has 145 missing values and the smoking_status 990.

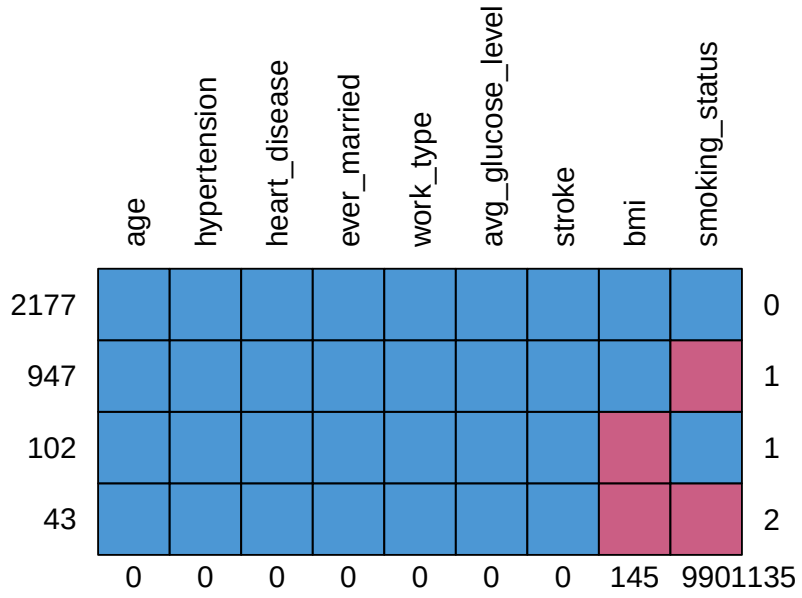
We will now create the boxplots of our variables and check for the outliers.



We observe outliers for four features, hypertension, heart disease, average glucose level and the body mass index. The hypertension and the heart_disease features are basically categorical variables. We have seen before normalization, that they were taking the value 1 in case you have hypertension and a heart disease respectively. As shown in the plots during feature selection procedure, the values of 1 for both features were not so many. Therefore, at the boxplots appear as outliers when in reality are not. Regarding the average glucose level, we have also shown in the feature selection part, that these high values of blood sugar can be observed to people with prediabetes or diabetes. Finally, regarding the bmi feature, we have seen that the outliers may be natural/true or measurement errors or input errors. Since we do not have access to the procedures that the data were collected, we will consider them as natural outliers.

Missing values We have seen that we have 145 missing values at the variable bmi. To the same feature we had also many outliers. The issue with the body mass index is that if someone is obese, it is less likely to disclose its weight. This could lead us that we have missing values not at random. However, since we do not have access at the procedures that the data were collected, we will assume that they are missing at random and we will handle these values as such. The same logic as with the bmi feature, we will also assume for the smoking status variable that we have 990 missing values and we are going to impute these values.

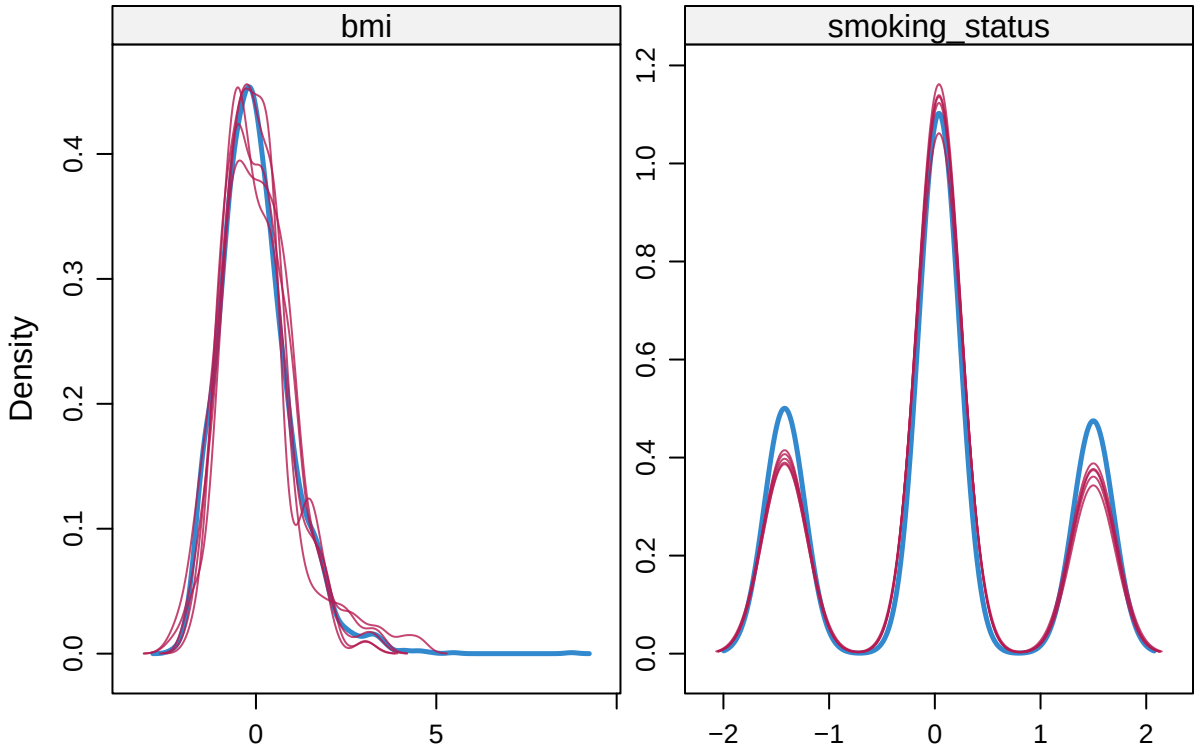
One solution would be to delete the observations with the missing values but this would further decrease our dataset. One other solution would be replace them with the median or the mean value or even better with the median or mean value of each class. However, we prefer to use the mice package to impute the missing values. This package creates multiple imputations (replacement values) for the missing data. Let us display the missing values pattern



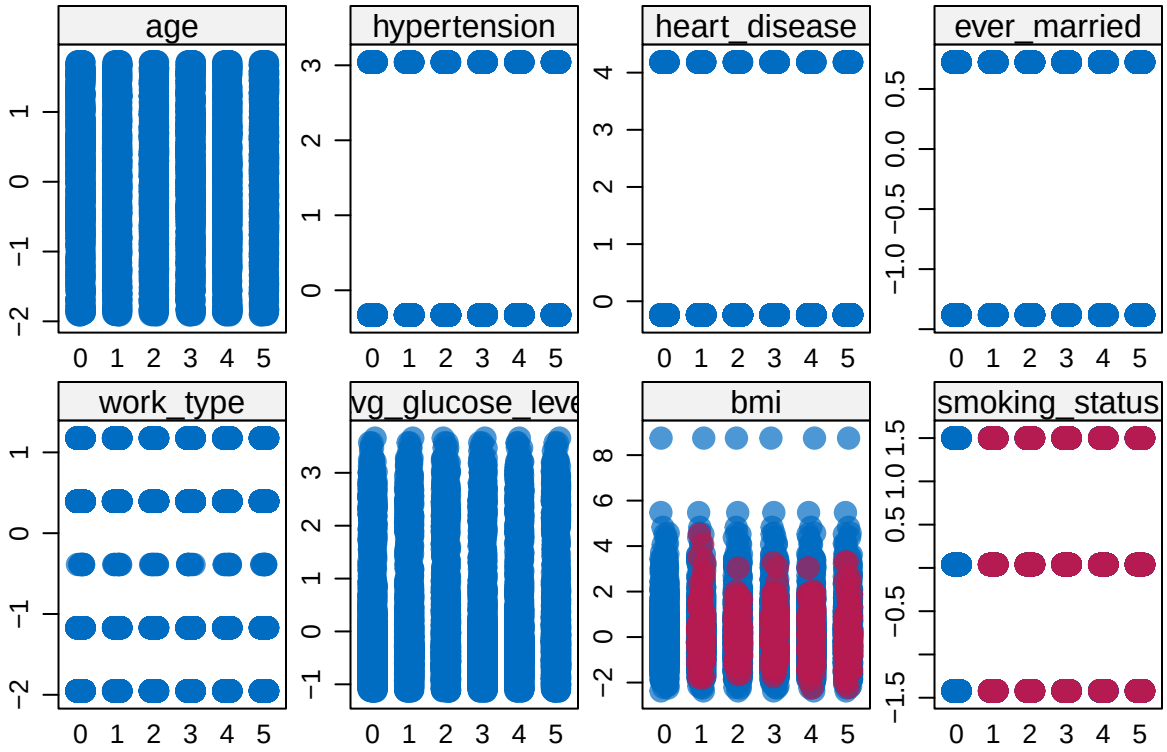
Let us now proceed with imputing our data.

This imputation computes as default 5 multiple imputations, 5 imputed data sets. We will have to check how good was the fitting of the missing values of these five imputations to our data. For this reason, we get the below plots

```
densityplot(imp1)
```




```
stripplot(imp1, pch=20, cex=2)
```

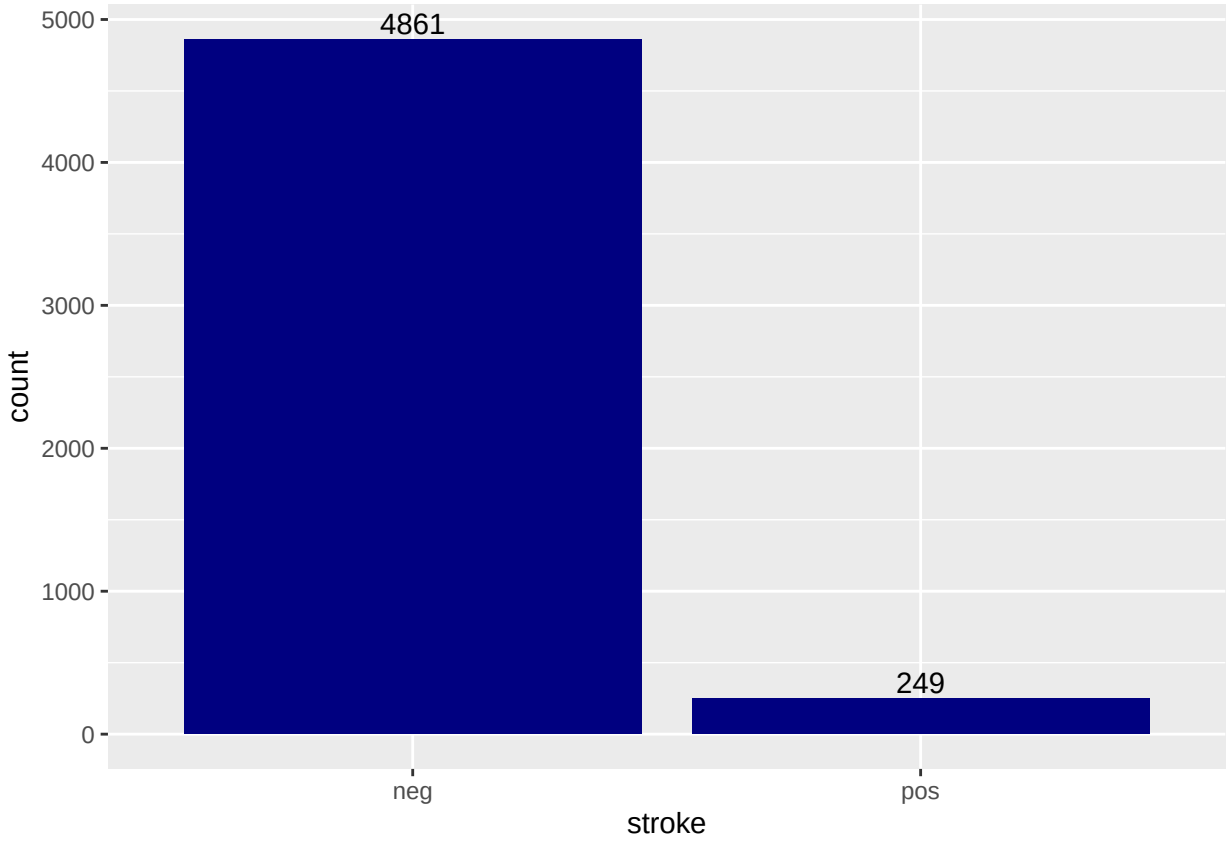


The blue dots and lines in the above plots are the observed data and the red ones are the imputed data. Ideally,the red points and lines should be similar with the blue ones. This would mean that the imputed values are similar to the observed, in other words they are plausible values.The features that have only blue dots is because they do not have missing values, hence they were not imputed.

It seems that we get good fit for all 5 imputations. Therefore, we will select one to complete our data set.

```
train_set1<- complete(imp1,2)
```

Response variable stroke Subsequently, we will create a barplot for the response variable stroke.



We will also create the table of relative frequencies for the stroke variable.

```
##
##   neg   pos
## 0.951 0.049
```

It is clear that the classes are highly imbalanced. This means that the number of observation for each class in the train set is not balanced. The issue is that most machine learning algorithms do not work very well with imbalanced datasets because they do not get the necessary information about the minority class. Therefore, it is difficult for the algorithms to make an accurate prediction. Hence, we should find a way to tackle this issue.

One first approach would be to not take into consideration the accuracy for the performance of our model but to look on the sensitivity, specificity and the F1 score. In this particular case, the specificity is more important, we would like to maximize the specificity over sensitivity. This happens because false positives can lead to healthy people begin a treatment. We therefore, want to be sure that negative tests are true negatives, we want to maximize the true negative error. Furthermore, we could make use of the F1-score which is the harmonic average of precision and recall and try to maximize this value.

However, the most efficient way to handle an imbalanced dataset is to balance it. This leads to an improved overall classification performance compared to that of an imbalanced dataset. A method that we could use in order to balance our dataset is the SMOTE (Synthetic Minority Oversampling Technique). As implied by the name, this technique will be oversampling the minority class and hence, will produce a dataset that has more balanced classes. For doing this, we will use the performanceEstimation pacakge.

```
smote_data <- smote(stroke~., train_set1,
                        k = 5, # indicates the number of nearest neighbours used to generate
```

```
                                # the new examples of the minority class.
                        perc.over = 19, # A number of how many extra cases from the minority
                                        # class are generated (known as over-sampling).
                        perc.under = 1.1) # A number of how many extra cases from the majority
                                          # classes are selected for each case generated from
#the minority class (known as under-sampling)

table(smote_data$stroke)
```

```
##
##  neg  pos
## 3323 3180
```

```
prop_stroke_smote <-prop.table(table(smote_data$stroke))
round(prop_stroke_smote,3)
```

```
##
##   neg   pos
## 0.511 0.489
```

As we observe, our classes are now balanced.

Modelling approach

In this part we will try different machine learning algorithms and we will test which have the best performance. Since we have a classification problem and our data is balanced, the measure that we will use to test the performance of our models is the Accuracy which results fro the confusion matrix. More details on this in Introduction to Data Science, Rafael A. Irizarry, Machine Learning part, Introduction to Machine Learning, Evaluation Metrics.

Logistic regression approach One way to build an algorithm would be with the use of the logistic regression approach. We could do this since our response variable is categorical with binary outcome. More details regarding logistic regression approach can be found in Introduction to Data Science, Rafael A. Irizarry, Machine Learning part, Examples of algorithms, Logistic Regression.

Let us now fit the logistic regression model with the function of generalized linear model (glm).

```
model_glm_stroke <- glm(stroke ~ ., family ="binomial", data = smote_data)
```

We will now obtain the prediction using the predict function.

```
predict_model_glm_stroke <- predict(model_glm_stroke, newdata = test_set, type = "response")
```

Now we will use the function ifelse to denote all the probabilities above 0.5 as positives and the remaining as negatives.

```
y_model_glm_stroke <- ifelse(predict_model_glm_stroke >= 0.5, "pos", "neg")
```

Finally, let us contsruct te confusion matrix and chech tha accuracy.

```r
tab_model_glm_stroke <- table(test_set$stroke, y_model_glm_stroke)

confusionMatrix_model_glm_stroke <- confusionMatrix(tab_model_glm_stroke,positive ="pos")

confusionMatrix_model_glm_stroke
```

```
## Confusion Matrix and Statistics
##
##      y_model_glm_stroke
##       neg pos
##   neg 350 171
##   pos   2  27
##
##                Accuracy : 0.6855
##                  95% CI : (0.6448, 0.7241)
##     No Information Rate : 0.64
##     P-Value [Acc > NIR] : 0.01413
##
##                   Kappa : 0.1607
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.13636
##             Specificity : 0.99432
##          Pos Pred Value : 0.93103
##          Neg Pred Value : 0.67179
##              Prevalence : 0.36000
##          Detection Rate : 0.04909
##    Detection Prevalence : 0.05273
##       Balanced Accuracy : 0.56534
##
##        'Positive' Class : pos
##
```

Therefore, the accuracy with the logistic regression model is:

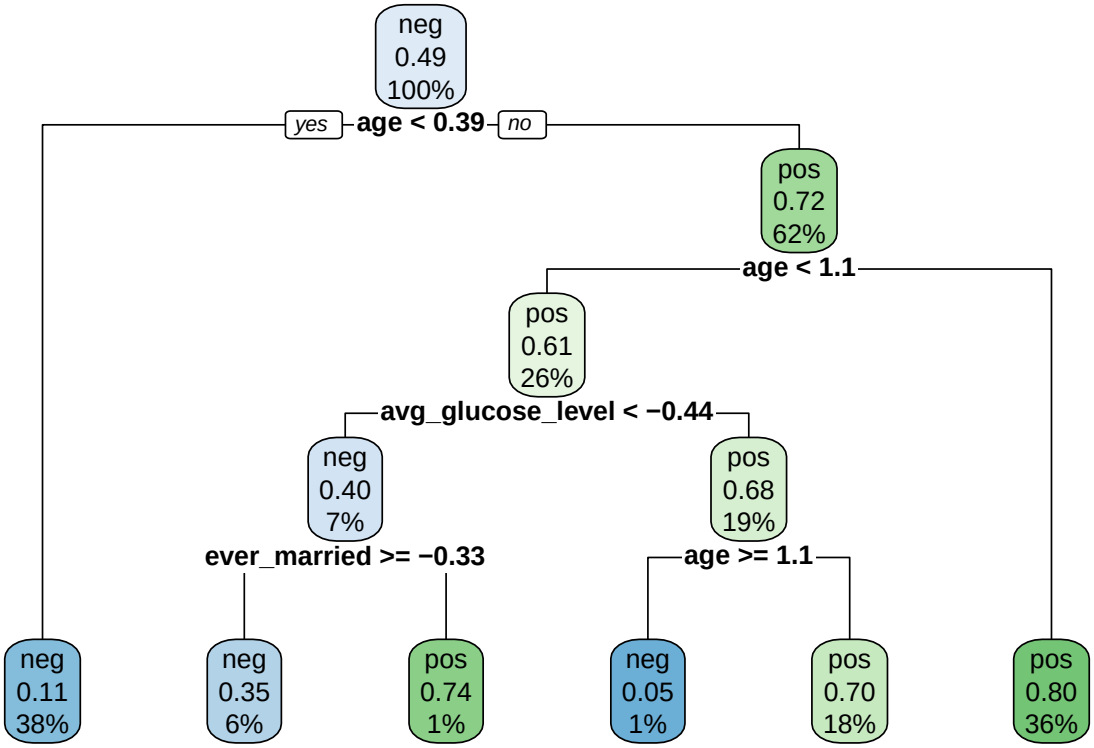
```
## [1] 0.6854545
```

Desicion Trees Another useful machine learning approach that we could use to build our model is the classification or decision tree since our response variable is categorical with a binary outcome. Basically, a classification tree is build through a process that is called recursive partitioning. That means that a repetitive process of splitting the data into partitions using some criteria is running and then, it continues to further splitting them up on each of the branches.

Let us now build our classification tree.

```r
set.seed(123, sample.kind="Rounding")
model_tree <- rpart(stroke ~ ., data = smote_data, method = "class",
                    control = rpart.control(xval=20, cp=0.009))
```

We will now plot our model



We do not have to forget that we have normalized our data that is why the number in the nodes (the criteria used to split the data) have this form. The advantage of using decision trees is that we would not have to normalize our data in order to use this machine learning algorithm.

We will continue with obtaining the prediction using the predict function.

```
predict_model_tree <- predict(model_tree, newdata = test_set, type = "class")
```

We will now check the metrics of our model.

```
confusionMatrix_model_tree <- confusionMatrix(predict_model_tree,test_set$stroke, positive ="pos",mode="
confusionMatrix_model_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 544   5
##        pos 234  35
##
##                Accuracy : 0.7078
##                  95% CI : (0.6753, 0.7388)
##     No Information Rate : 0.9511
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1546
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.87500
##             Specificity : 0.69923
##          Pos Pred Value : 0.13011
##          Neg Pred Value : 0.99089
##               Precision : 0.13011
##                  Recall : 0.87500
##                      F1 : 0.22654
##              Prevalence : 0.04890
##          Detection Rate : 0.04279
##    Detection Prevalence : 0.32885
##       Balanced Accuracy : 0.78711
##
##        'Positive' Class : pos
##
```

Therefore, the accuracy of our model using the decision tree algorithm is:

```
## [1] 0.707824
```

For the moment, we gwt a slightly better perfomance with this algorithm.

Random Forest The final machine learning algorithm that we will use is the random Forest. The idea of this algorithm is to average multiple classification trees and therefore, improve the performance of our prediction and reduce the instability.

Here, first of all, we will try to find the best value for the parameter mtry which indicates the number of variables that will be randomly sampled as candidates at each split.

```
##   mtry
## 4    4
```

Therefore, we get the best mtry parameter. We can now fit the random forest model with the best mtry parameter.

```
fit_rf <- randomForest(stroke~., smote_data, minNode = model_rf$bestTune$mtry)
```

We will continue with obtaining the prediction using the predict function.

```
predict_rf<- predict(fit_rf, test_set)
```

We will now check the confusion matrix of our model.

```
confusionMatrix_rf <- confusionMatrix(predict_rf,test_set$stroke, positive ="pos",
                                      mode="everything")
confusionMatrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 413  13
##        pos 108  16
##
##                Accuracy : 0.78
##                  95% CI : (0.743, 0.8139)
##     No Information Rate : 0.9473
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1352
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.55172
##             Specificity : 0.79271
##          Pos Pred Value : 0.12903
##          Neg Pred Value : 0.96948
##               Precision : 0.12903
##                  Recall : 0.55172
##                      F1 : 0.20915
##              Prevalence : 0.05273
##          Detection Rate : 0.02909
##    Detection Prevalence : 0.22545
##       Balanced Accuracy : 0.67222
##
##        'Positive' Class : pos
##
```

Therefore, the accuracy of our model using the decision tree algorithm is:

```
## [1] 0.78
```

We observe that this method gave us the greatest accuracy.

Results

In this section, the modeling results are going to be presented. We will create a results table

```
accuracy_results <- tibble(method = c("Logistic Regression Model",
                                      "Classification Trees", "Random Forest"),
                           Accuracy = c(glm, classification_tree, rf))

knitr::kable(accuracy_results,digits=5)
```

method	Accuracy
Logistic Regression Model	0.68545
Classification Trees	0.70782

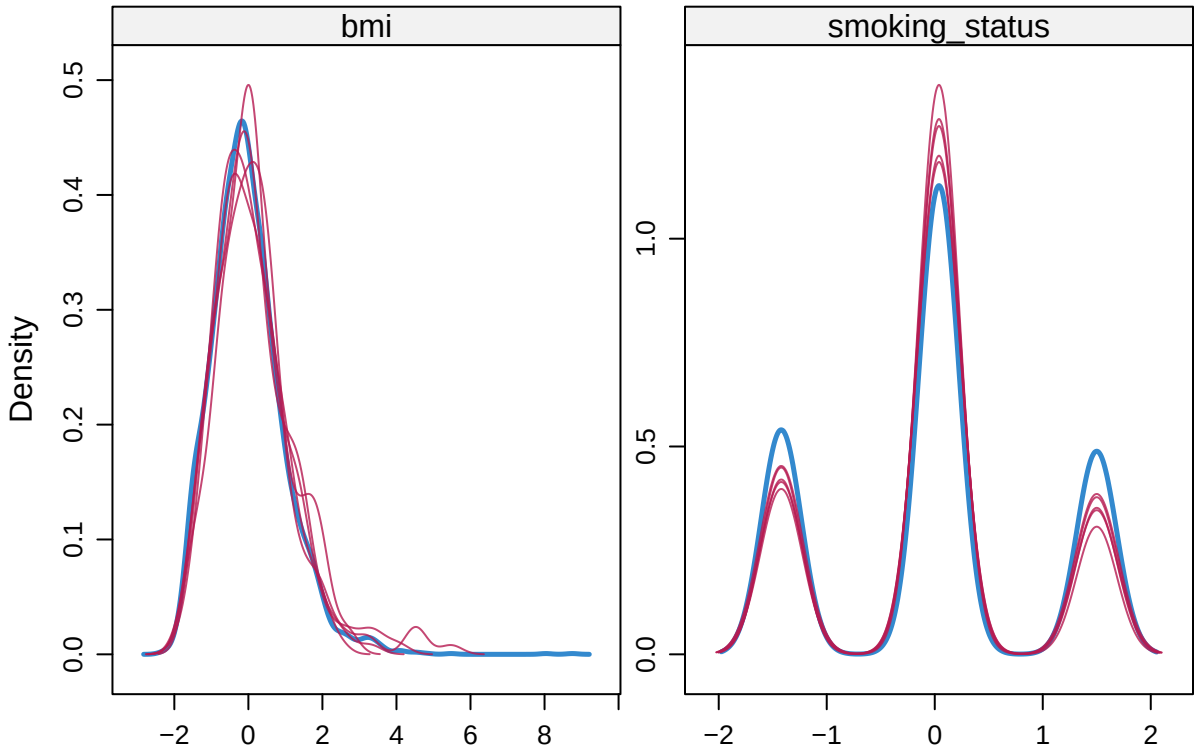
method	Accuracy
Random Forest	0.78000

As we notice, the best Accuracy value was achieved with the random forest algorithm, which maximized the most the accuracy. Therefore, we will choose this method and we will check our model performance taking into consideration the train_temp set and our final hold-out test set that means the validation set. Therefore, let us run the above procedure but now for the train_temp and the validation set.

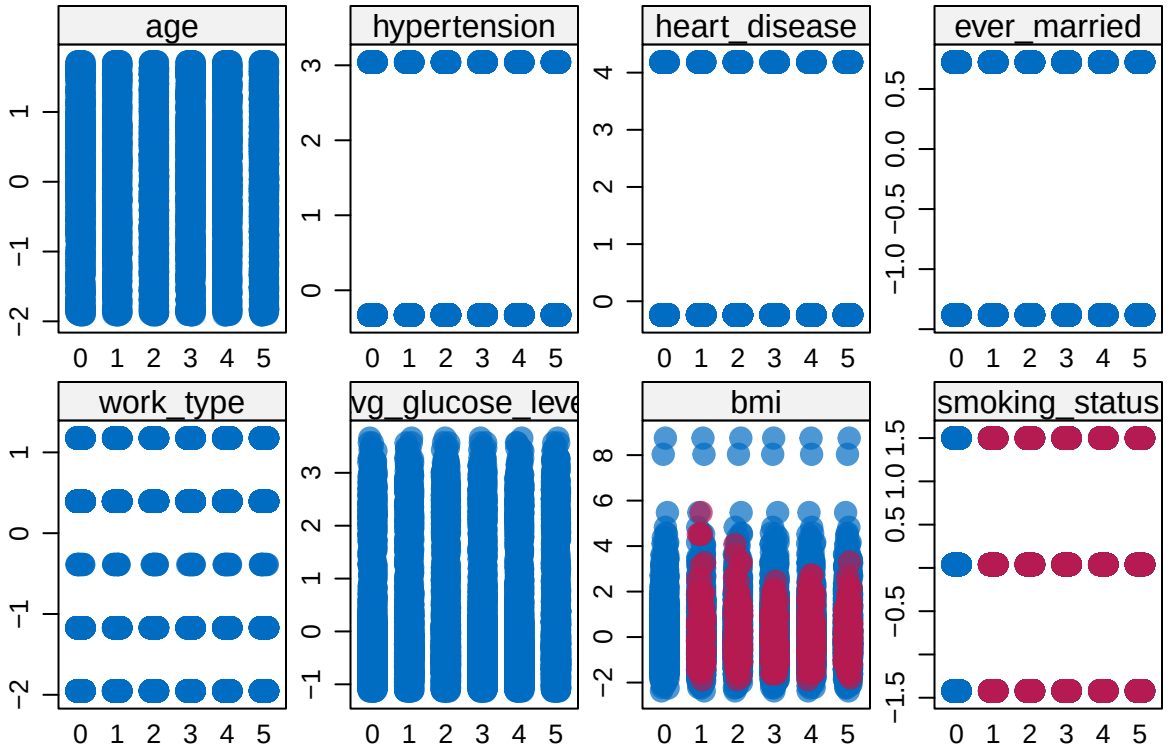
First of all we will impute the missing data to the train_temp set.

We will create the desnity plots and the stripplots.

```
densityplot(imp2)
```




```
stripplot(imp2, pch=20, cex=2)
```



It seems that we get good fit for all 5 imputations. Therefore, we will select one to complete our data set.

```
train_temp1<- complete(imp2,2)
```

We will continue with balancing the classes of our response variable but let us have a look at the table of relative frequencies for the stroke variable at the train_temp1 dataset.

```
##
##   neg   pos
## 0.951 0.049
```

It is clear that the classes are highly imbalanced. We will again use the SMOTE (Synthetic Minority Oversampling Technique). Hence, we balance our dataset as follows:

```
smote_data1 <- smote(stroke~.,
                     train_temp1,
                     k = 5,
                     perc.over = 19,
                     perc.under = 1.1)

table(smote_data1$stroke)
```

```
##
##  neg  pos
## 4159 3980
```

```
prop_stroke_smote_final <-prop.table(table(smote_data1$stroke))
round(prop_stroke_smote_final,3)
```

```
##
##   neg   pos
## 0.511 0.489
```

Now, we are ready to use the random forest algorithm in order to check the performance of the model.

We will find again the best value for the parameter mtry.

```
set.seed(321, sample.kind="Rounding")

control <- trainControl(method="cv", number = 10)

grid <- data.frame(mtry = c(1:8))

model_rf_final <- train(stroke~.,smote_data1,method = "rf",
                  trControl=control, tuneGrid = grid)

model_rf_final$bestTune
```

```
##   mtry
## 4    4
```

Therefore, we get the best mtry parameter. We can now fit the random forest model with the best mtry parameter and with the optimized minimum node size.

```
fit_rf_final <- randomForest(stroke~., smote_data1, minNode = model_rf_final$bestTune$mtry)
```

We will continue with obtaining the prediction using the predict function.

```
predict_rf_final<- predict(fit_rf_final, validation_set)
```

We will now check the confusion matrix of our model.

```
confusionMatrix_rf_final <- confusionMatrix(predict_rf_final,validation_set$stroke,
                                            positive ="pos", mode="everything")
confusionMatrix_rf_final
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 544  17
##        pos 118  20
##
##                Accuracy : 0.8069
##                  95% CI : (0.7756, 0.8355)
##     No Information Rate : 0.9471
##     P-Value [Acc > NIR] : 1
```

```
##
##                   Kappa : 0.1583
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.54054
##             Specificity : 0.82175
##          Pos Pred Value : 0.14493
##          Neg Pred Value : 0.96970
##               Precision : 0.14493
##                  Recall : 0.54054
##                      F1 : 0.22857
##              Prevalence : 0.05293
##          Detection Rate : 0.02861
##    Detection Prevalence : 0.19742
##       Balanced Accuracy : 0.68115
##
##        'Positive' Class : pos
##
```

Therefore, the accuracy of our model using the decision tree algorithm is:

```
## [1] 0.806867
```

Our model performance is quite good since we have maximized our accuracy, hence our predictions using this model can be more accurate and trustworthy.

Conclusion

In this project, we have explored our Stroke Prediction Dataset from Kaggle website. We got useful insights from this exploration and based on this, we have used different modeling approaches for creating an algorithm that could predict the potentiality of having a stroke. We concluded that the best modeling approach and therefore, the potential best evaluation system that could successfully predict the possibility of having a stroke is the Random Forest algorithm.

The scope of this work was to build an algorithm which will help us predict the possibility of having a stroke. This project gives us a hint of the relation of specific predictors with the stroke and can be the first step for a future better algorithm. Supervised machine learning was used in the construction of the algorithm, with data acquired from the Kaggle website. Restricted number of observations in our dataset limited the performance of our models. Future work that will be based in a larger dataset, will take into account supplementary factors/predictors that may increase the risk of a stroke and with the use of additional machine learning algorithms, could help us built a more efficient model with potential widespread use.

References

Stroke Prediction Dataset,

Introduction to Data Science, Rafael A. Irizarry,

mice: mice: Multivariate Imputation by Chained Equations,

Package 'performanceEstimation',

American Stroke Association,

National Health Service (NHS)