Stroke Prediction Project

Pravalika Javvadi

Business Understanding

```
#For my machine learning project, I have chosen the Stroke Prediction Dataset. This dataset contains information
on over 5,000 patients and includes features such as age, hypertension, smoking status, and whether or not the pa
tient had a stroke. The dataset can be found on Kaggle at the following URL: https://www.kaggle.com/fedesoriano/s
troke-prediction-dataset.

#Business Understanding for Stroke Prediction Dataset:

#The healthcare dataset on stroke prediction contains information about patients' demographic, lifestyle, and hea
lth-related characteristics, as well as whether or not they have had a stroke. The goal of this project is to bui
ld a model that can predict the likelihood of a patient having a stroke based on their characteristics.

#The stakeholders for this project include healthcare providers and insurers, as well as policy makers and public
health officials. By identifying patients at high risk for stroke, healthcare providers can target preventive int
erventions to those who need them most, potentially reducing the incidence and severity of strokes. Insurers can
also use this model to more accurately price policies and manage risk. Finally, policy makers and public health o
fficials can use the insights gained from this analysis to develop targeted public health campaigns to educate th
e public about stroke risk factors and promote healthy behaviors.

#To achieve these goals, we will need to perform exploratory data analysis to understand the distribution of the
variables and identify any missing data or outliers. We will also need to evaluate the predictive power of each v
ariable and identify any potential confounding factors that could influence the results. Finally, we will build a
nd test several machine learning models to determine which one performs best on this dataset.
```

Installations needed:

```
#remotes::install_version("DMwR", version="0.4.1")
## The package DMwR is not available for my current R version so I downloaded it like this. Just sharing it here
just in case if it doesn't work on the other's R studio as well.
```

Libraries:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
library(fastDummies)
```

```
## Thank you for using fastDummies!
```

```
## To acknowledge our work, please cite the package:
```

```
## Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categoric
al Variables. Version 1.7.1. URL: https://github.com/jacobkap/fastDummies, https://jacobkap.github.io/fastDummie
s/.
```

```
library(psych)
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(ggmosaic)
library(reshape2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(nnet)
library(DMwR)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'DMwR'
```

```
## The following object is masked from 'package:psych':
##
##     crossValidation
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
library(RColorBrewer)
library(C50)
library(ipred)
```

Data Understanding

```
## Data Collection and exploration

## The URL of the stroke prediction dataset
# "https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset?select=healthcare-dataset-stroke-data.cs
v"


url <- "https://drive.google.com/uc?id=1F_TDiOvhghFSWda-bpHO3ml5PwfpWP1Q&export=download"

#Read the CSV file
stroke <- read.csv(url, header = T)


# check the structure of the dataset
str(stroke)
```

```
## '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 ...
```

```
paste0("We have a 5110 observations and 12 variables which a couple of numerical, char and int variables. Some ne
cessary conversions should be made for the variables.")
```

```
## [1] "We have a 5110 observations and 12 variables which a couple of numerical, char and int variables. Some ne
cessary conversions should be made for the variables."
```

```
# Check the statistics of the dataset
summary(stroke)
```

```
##        id           gender               age         hypertension    
##  Min.   :   67   Length:5110        Min.   : 0.08   Min.   :0.00000  
##  1st Qu.:17741   Class :character   1st Qu.:25.00   1st Qu.:0.00000  
##  Median :36932   Mode  :character   Median :45.00   Median :0.00000  
##  Mean   :36518                      Mean   :43.23   Mean   :0.09746  
##  3rd Qu.:54682                      3rd Qu.:61.00   3rd Qu.:0.00000  
##  Max.   :72940                      Max.   :82.00   Max.   :1.00000  
##  heart_disease     ever_married        work_type         Residence_type    
##  Min.   :0.00000   Length:5110        Length:5110        Length:5110       
##  1st Qu.:0.00000   Class :character   Class :character   Class :character  
##  Median :0.00000   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :0.05401                                                           
##  3rd Qu.:0.00000                                                           
##  Max.   :1.00000                                                           
##  avg_glucose_level     bmi            smoking_status         stroke       
##  Min.   : 55.12    Length:5110        Length:5110        Min.   :0.00000  
##  1st Qu.: 77.25    Class :character   Class :character   1st Qu.:0.00000  
##  Median : 91.89    Mode  :character   Mode  :character   Median :0.00000  
##  Mean   :106.15                                          Mean   :0.04873  
##  3rd Qu.:114.09                                          3rd Qu.:0.00000  
##  Max.   :271.74                                          Max.   :1.00000  
```

```
paste0("We have found that the age & avg_glucose_level features have some un-usual statistics like in the age col
umn the data is positively skewed, as the mean is smaller than the median. Additionally, there are no missing val
ues in the age column, and the range of values is between 0.08 and 82, which indicates that there is a wide varia
tion in age. Also, the average glucose level variable appears to have a right-skewed distribution, with a median
of 91.89 and a mean of 106.15. Additionally, there appears to be some potential outliers as the maximum value is
significantly higher than the third quartile.")
```

```
## [1] "We have found that the age & avg_glucose_level features have some un-usual statistics like in the age col
umn the data is positively skewed, as the mean is smaller than the median. Additionally, there are no missing val
ues in the age column, and the range of values is between 0.08 and 82, which indicates that there is a wide varia
tion in age. Also, the average glucose level variable appears to have a right-skewed distribution, with a median
of 91.89 and a mean of 106.15. Additionally, there appears to be some potential outliers as the maximum value is
significantly higher than the third quartile."
```

```
summary(stroke$age) #check the summary of age
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.08   25.00   45.00   43.23   61.00   82.00
```

```
paste0("The summary seemed off since the min value is 0.08 which is ridiculous to be a person's age.")
```

```
## [1] "The summary seemed off since the min value is 0.08 which is ridiculous to be a person's age."
```

```
stroke$age <- ceiling(stroke$age) # round up the values to the nearest integer

# now re-check the summary of the age to check if the conversion happened
summary(stroke$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00   25.00   45.00   43.24   61.00   82.00
```

Data cleaning, preparation and shaping

```
# Data preparation

stroke <- subset(stroke, select = -c(id)) # the id column doesn't serve any valuable purpose in this data set so
I removed it.

# Convert bmi to numeric
stroke$bmi <- as.numeric(stroke$bmi) # Also, I observed that the BMI column is in char so I converted it to numer
ic
```

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

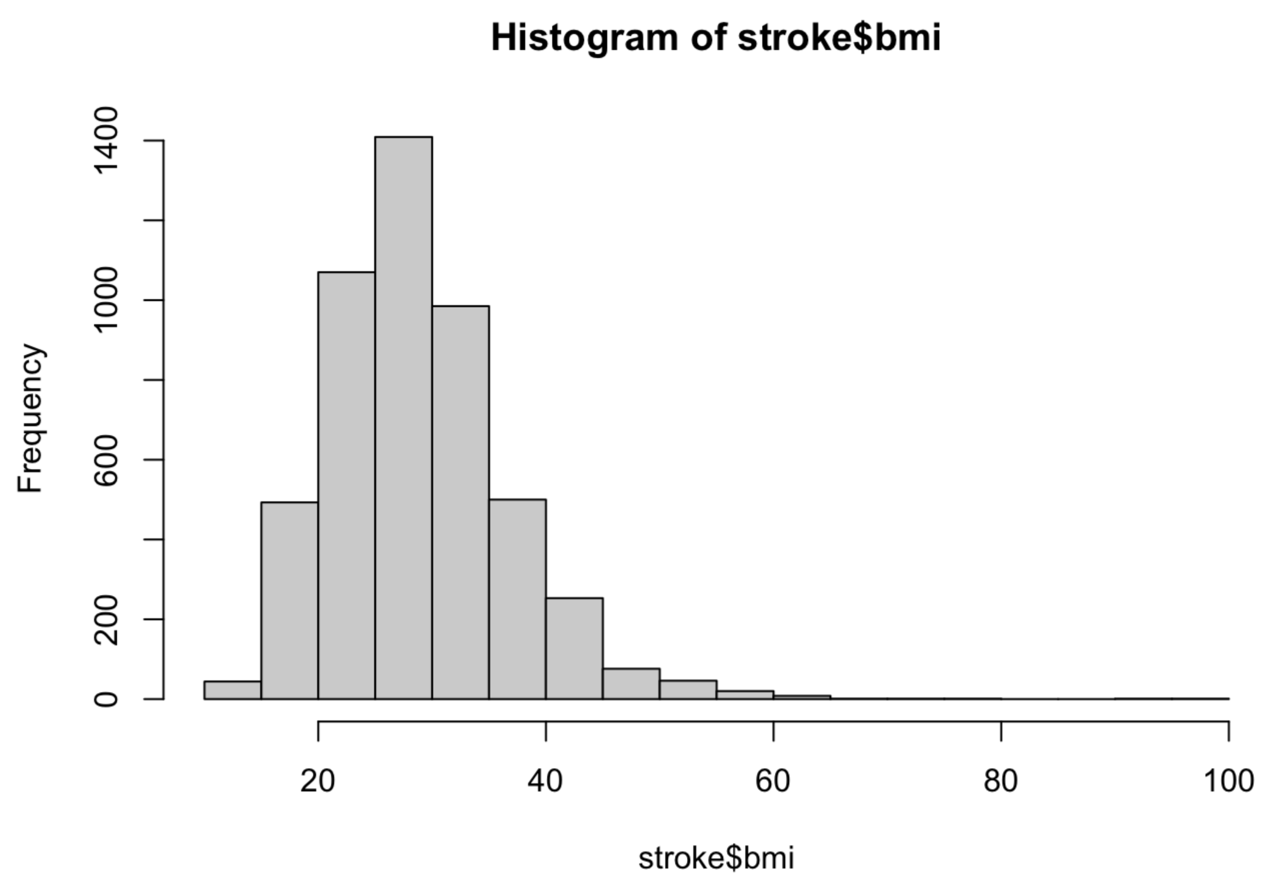
```
summary(stroke$bmi) # We have checked the summary to see the statistics of the BMI after its conversion and found
that there are presence of some missing values
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   10.30   23.50   28.10   28.89   33.10   97.60     201
```

```
# Check for missing values
colSums(is.na(stroke)) # to display the number of missing values found in the BMI column
```

```
##            gender               age      hypertension     heart_disease 
##                 0                 0                 0                 0 
##      ever_married         work_type    Residence_type avg_glucose_level 
##                 0                 0                 0                 0 
##               bmi    smoking_status            stroke 
##               201                 0                 0
```

```
# Check the distribution of the BMI column to select the best method to impute it
hist(stroke$bmi)
```



```
paste0("According to the distribution, it is positively skewed and it would be better to impute the missing value
s with the median rather than the mean. This is because the median is less sensitive to outliers and extreme valu
es than the mean, and therefore more appropriate for skewed distributions.")
```

```
## [1] "According to the distribution, it is positively skewed and it would be better to impute the missing value
s with the median rather than the mean. This is because the median is less sensitive to outliers and extreme valu
es than the mean, and therefore more appropriate for skewed distributions."
```

```
# impute the missing values in BMI
median <- median(stroke$bmi, na.rm = TRUE)
stroke$bmi[is.na(stroke$bmi)] <- median
head(stroke$bmi, 20)
```

```
##  [1] 36.6 28.1 32.5 34.4 24.0 29.0 27.4 22.8 28.1 24.2 29.7 36.8 27.3 28.1 28.2
## [16] 30.9 37.5 25.8 37.8 28.1
```

```
## Now, lets examine the categorical variables

#First, lets start with the gender column
table(stroke$gender)
```

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

```
paste0("Here, we can see there is only a single observation for the 'other' so I decided to exclude the single ob
servation as it doesnt make any huge impact in the gender column.")
```

```
## [1] "Here, we can see there is only a single observation for the 'other' so I decided to exclude the single ob
servation as it doesnt make any huge impact in the gender column."
```

```
# Filter the data frame to exclude rows where "gender" is equal to other
stroke<- stroke %>%
  filter(gender != "Other")

# check if the conversion happened
table(stroke$gender)
```

```
##
## Female   Male
##   2994   2115
```

```
# Now,lets examine the remaining discrete columns
table(stroke$hypertension)
```

```
##
##    0    1
## 4611  498
```

```
table(stroke$heart_disease)
```

```
##
##    0    1
## 4833  276
```

```
table(stroke$ever_married)
```

```
##
##   No  Yes
## 1756 3353
```

```
table(stroke$Residence_type)
```

```
##
## Rural Urban
##  2513  2596
```

```
table(stroke$smoking_status)
```

```
##
## formerly smoked    never smoked          smokes         Unknown
##             884            1892             789            1544
```

```
table(stroke$work_type)
```

```
##
##      children      Govt_job  Never_worked       Private Self-employed
##           687           657            22          2924           819
```

```
table(stroke$stroke)
```

```
##
##    0    1
## 4860  249
```

```
# Convert the Unknown to never smoked because there are huge number of unknowns which technically means we don't
know the information so rather i decided to change it to the most frequent category i.e., never smoked.
stroke$smoking_status <- replace(stroke$smoking_status, stroke$smoking_status == "Unknown", "never smoked")

# check if the conversion happened
table(stroke$smoking_status)
```

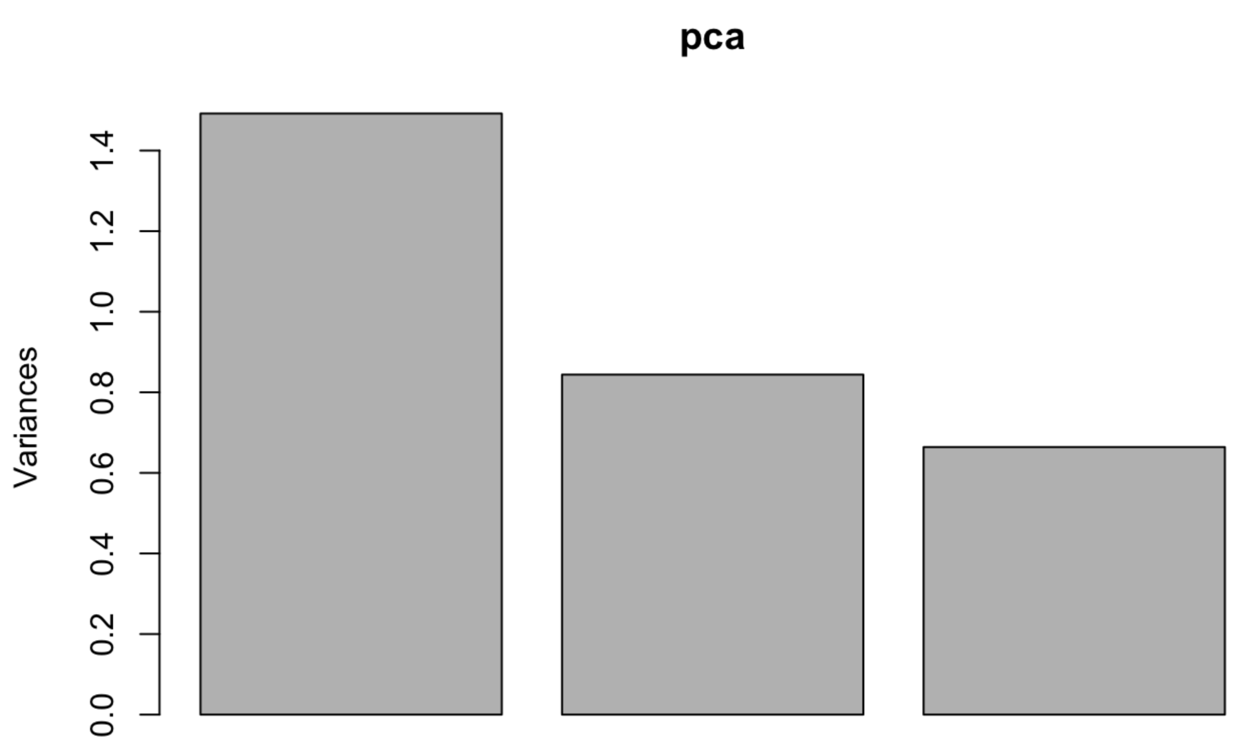
```
##
## formerly smoked    never smoked          smokes
##             884            3436             789
```

```
# Conversion of data types

# Convert the characters into factor columns
stroke$hypertension <- as.factor(stroke$hypertension)
stroke$heart_disease <- as.factor(stroke$heart_disease)
stroke$Residence_type <- as.factor(stroke$Residence_type)
stroke$ever_married <- as.factor(stroke$ever_married)
stroke$stroke <- as.factor(stroke$stroke)

# Check the structure of the dataset to see if the conversion happened
str(stroke)
```

```
## 'data.frame':    5109 obs. of  11 variables:
##  $ gender           : chr  "Male" "Female" "Male" "Female" ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 1 1 1 ...
##  $ heart_disease    : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 1 1 ...
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : chr  "Private" "Self-employed" "Private" "Private" ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ smoking_status   : chr  "formerly smoked" "never smoked" "never smoked" "smokes" ...
##  $ stroke           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Identification of PCA

```
# Perform PCA
pca <- prcomp(stroke[, c("age", "avg_glucose_level", "bmi")], scale = TRUE)

# Plot proportion of variance explained by each principal component
plot(pca)
```



```
# Check the loadings of the first principal component
loadings <- pca$rotation

## Check all loadings
loadings[,1]
```

```
##               age avg_glucose_level               bmi
##        -0.6317760        -0.5056257        -0.5875387
```

```
loadings[,2]
```

```
##               age avg_glucose_level               bmi
##        -0.1742942         0.8312197        -0.5279160
```

```
loadings[,3]
```

```
##               age avg_glucose_level               bmi
##        -0.7553017         0.2311201         0.6132723
```

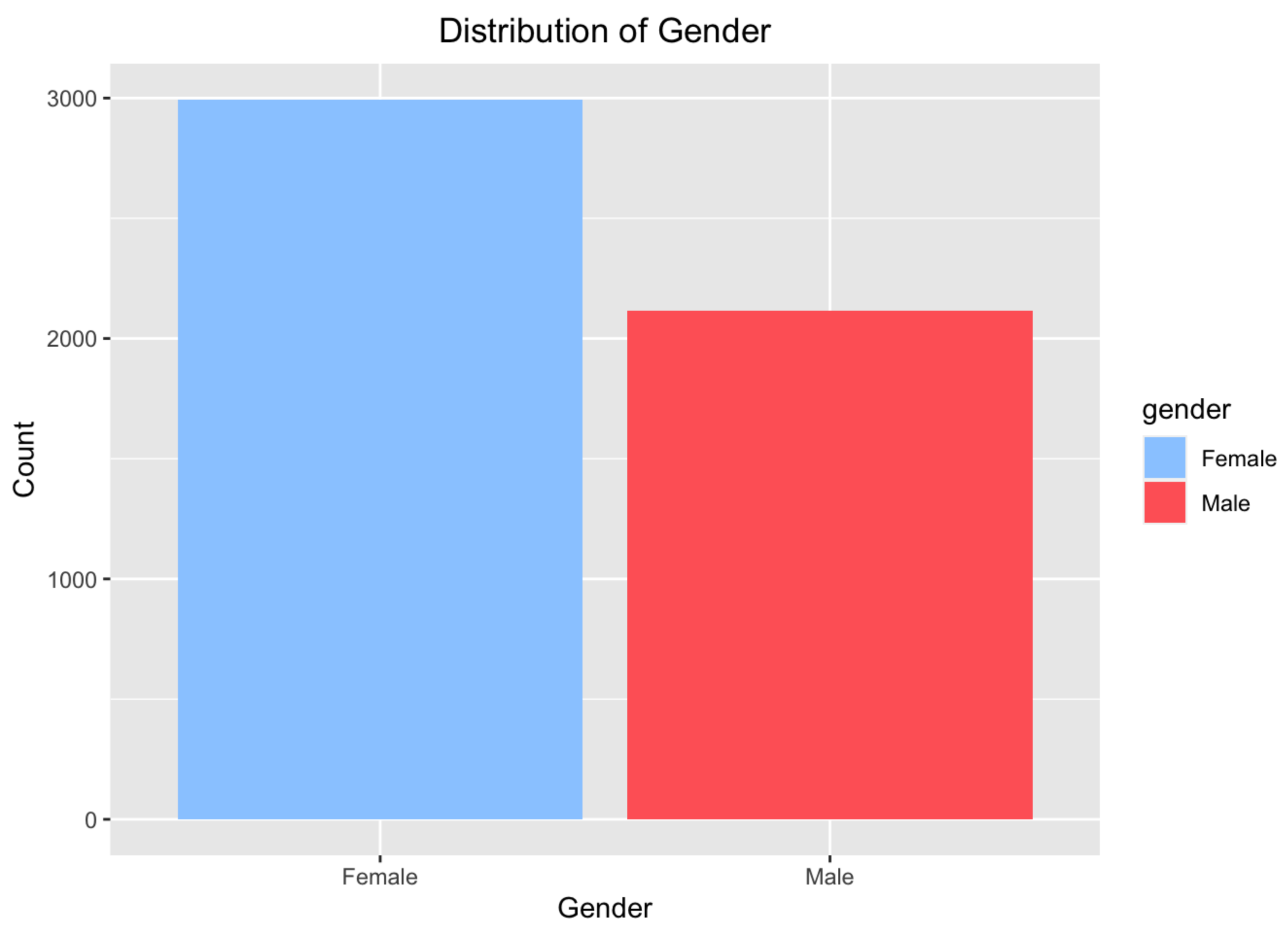
```
# Display the first pca values
first_pca_loadings <- loadings[,1]

# Check which features are strongly associated with the first principal component
sorted_loadings <- sort(abs(first_pca_loadings), decreasing = TRUE)
print(sorted_loadings)
```

```
##               age               bmi avg_glucose_level
##         0.6317760         0.5875387         0.5056257
```

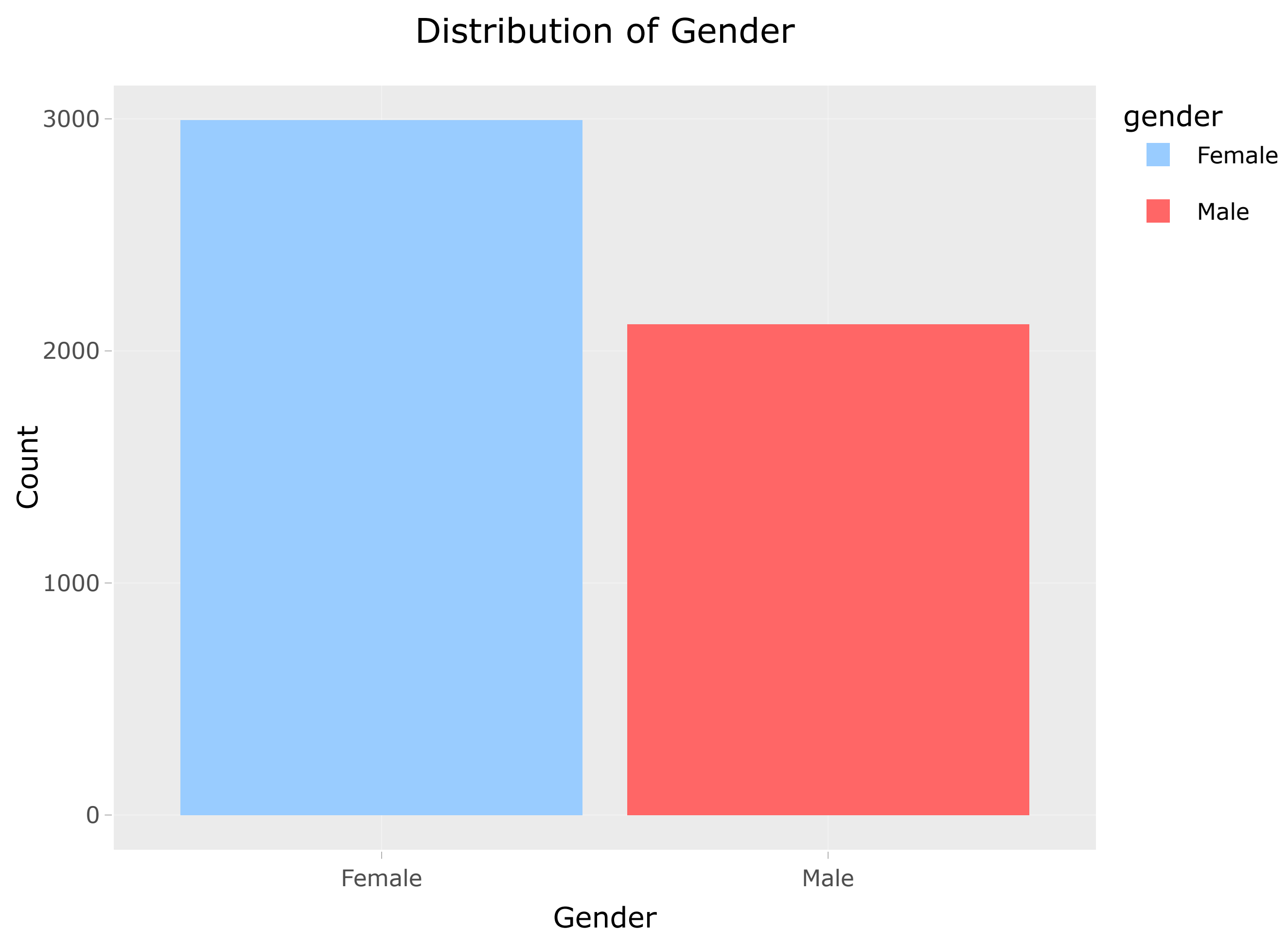
This is the PCA distribution of the continuous variables. There are no features necessary to be removed and I believe for this dataset, since there are only 3 continuous features and 11 categorical features, the performance of PCA wouldn't be necessary given that the dimensionality of the dataset is relatively low. However, I want to try and see the distribution so I performed PCA but in other case I don't think PCA is necessary here.

Data exploratory analysis

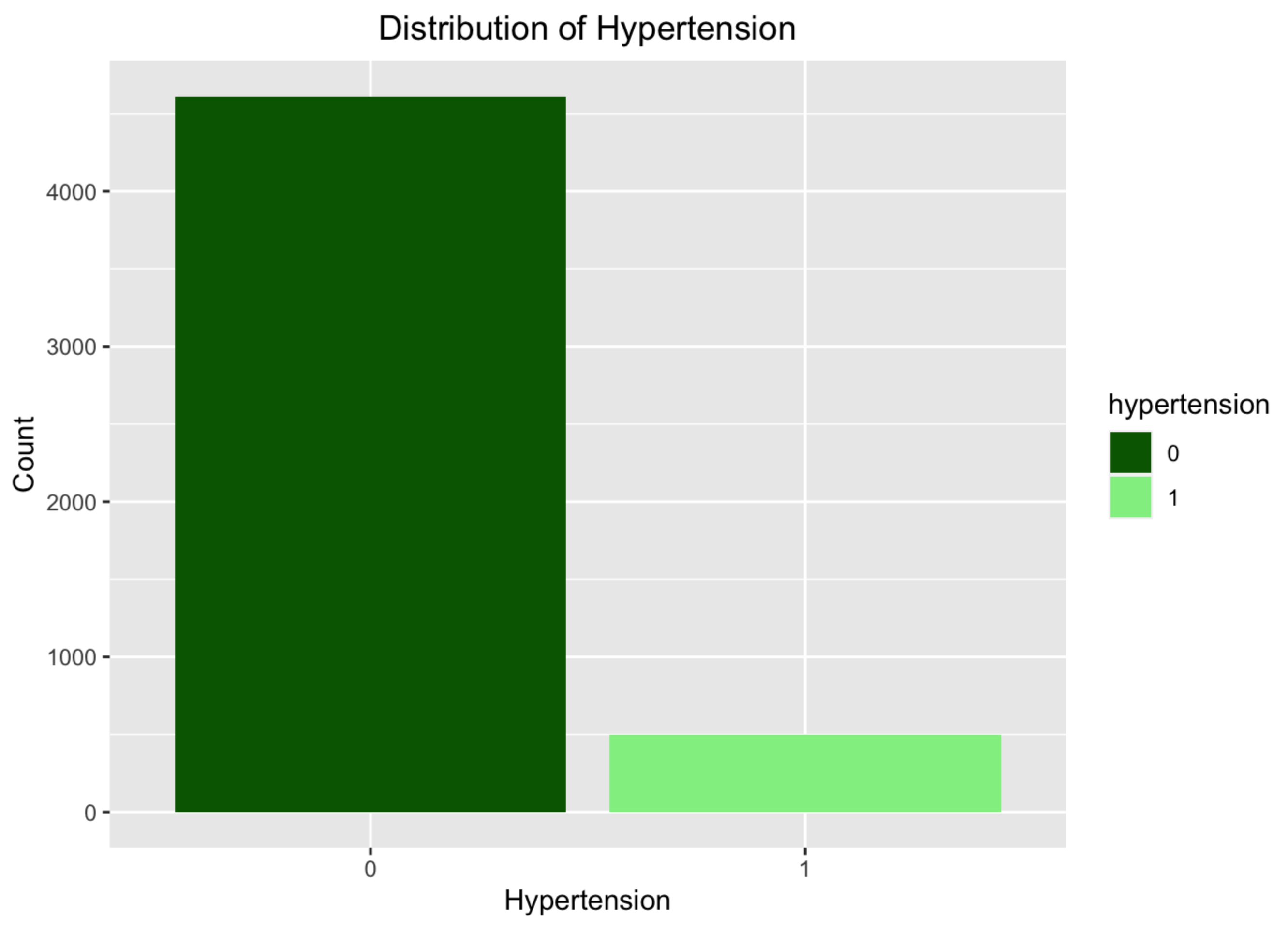
```
# Data exploratory analysis

# Discrete variables distribution
#Gender
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = gender, fill = gender)) +
  labs(title = "Distribution of Gender", x = "Gender", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("#99CCFF", "#FF6666"))
```



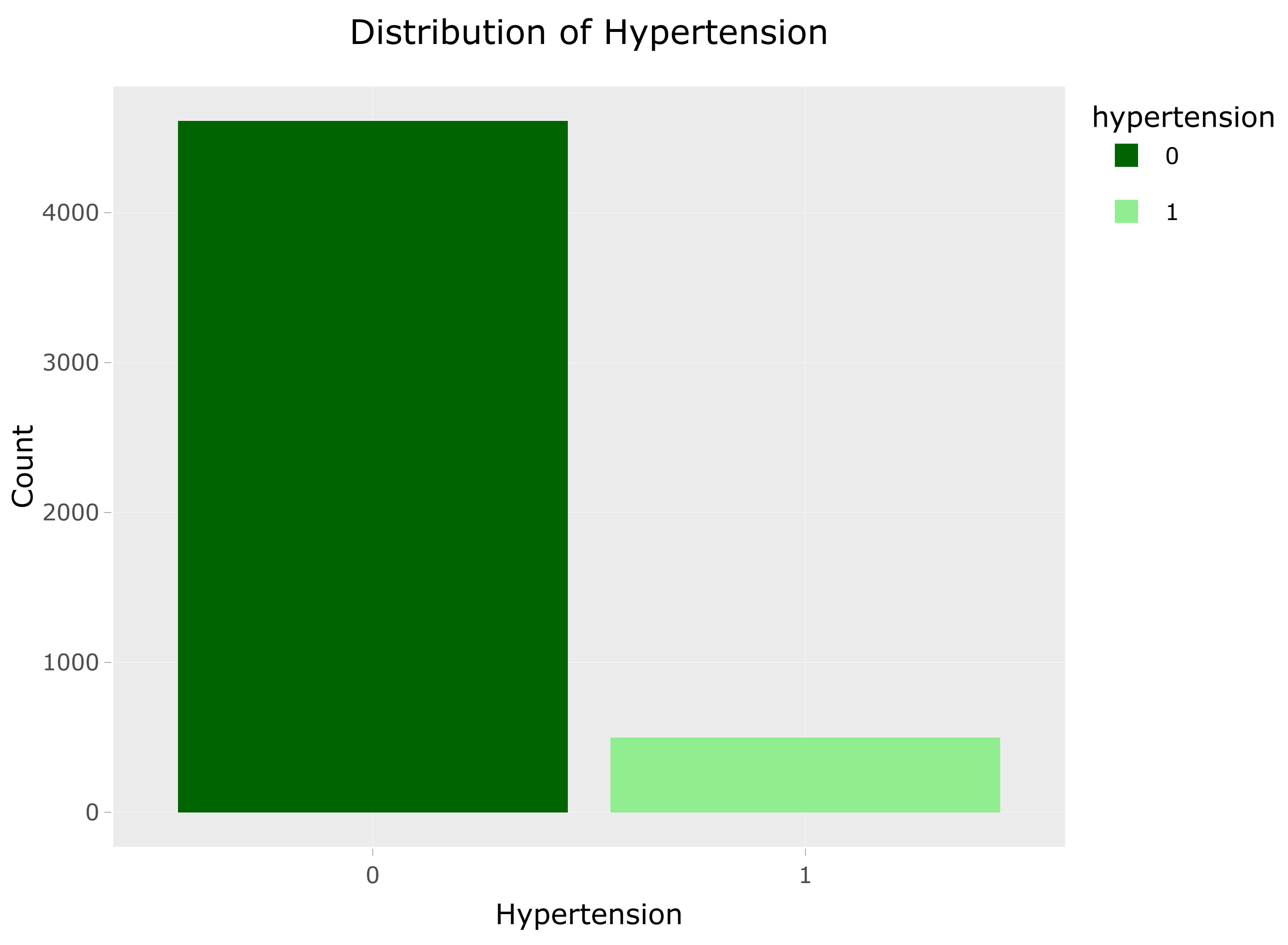
```
ggplotly()
```



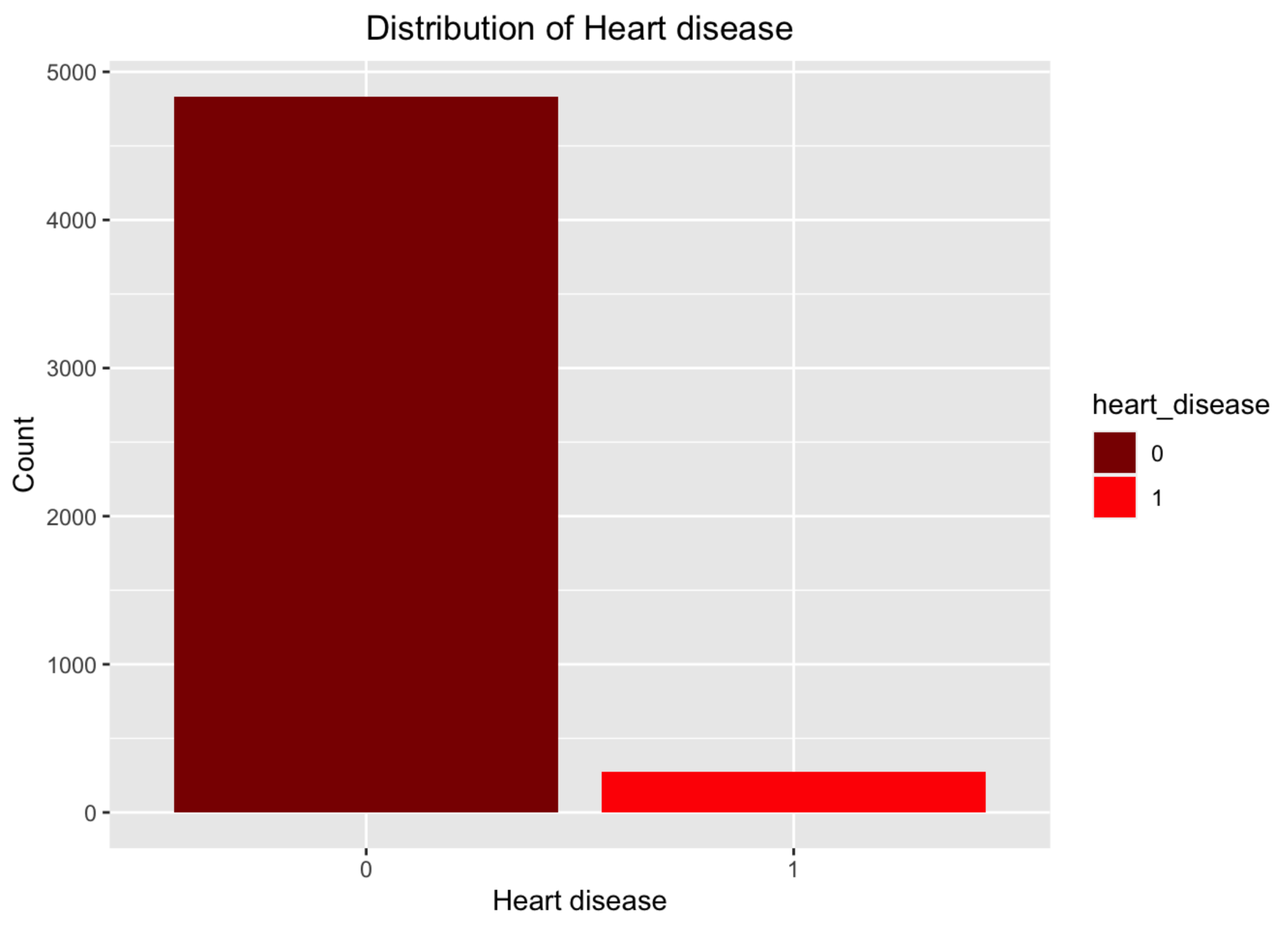
```
#Hypertension
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = hypertension, fill = hypertension)) +
  labs(title = "Distribution of Hypertension", x = "Hypertension", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("darkgreen", "lightgreen"))
```



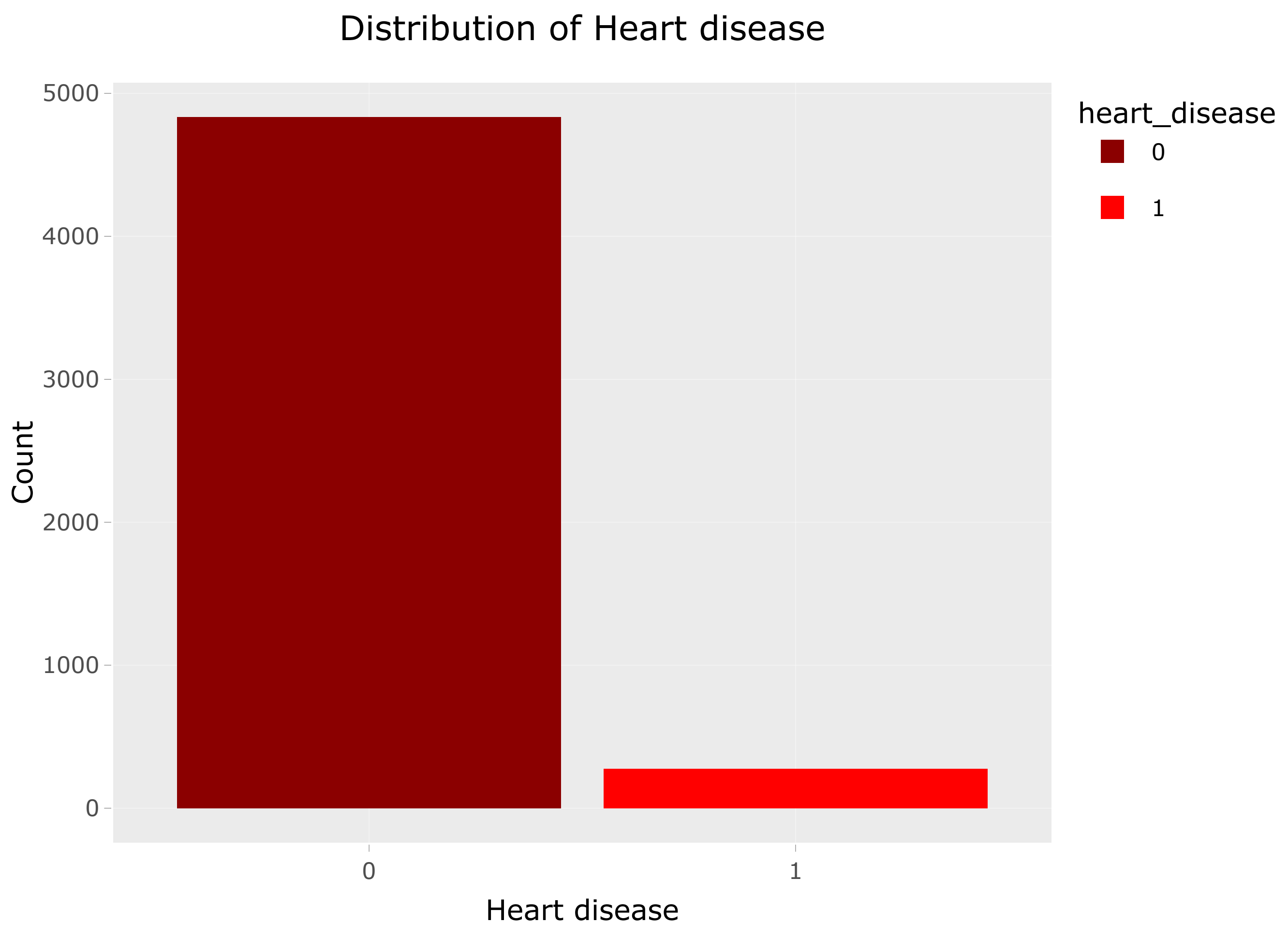
```
ggplotly()
```



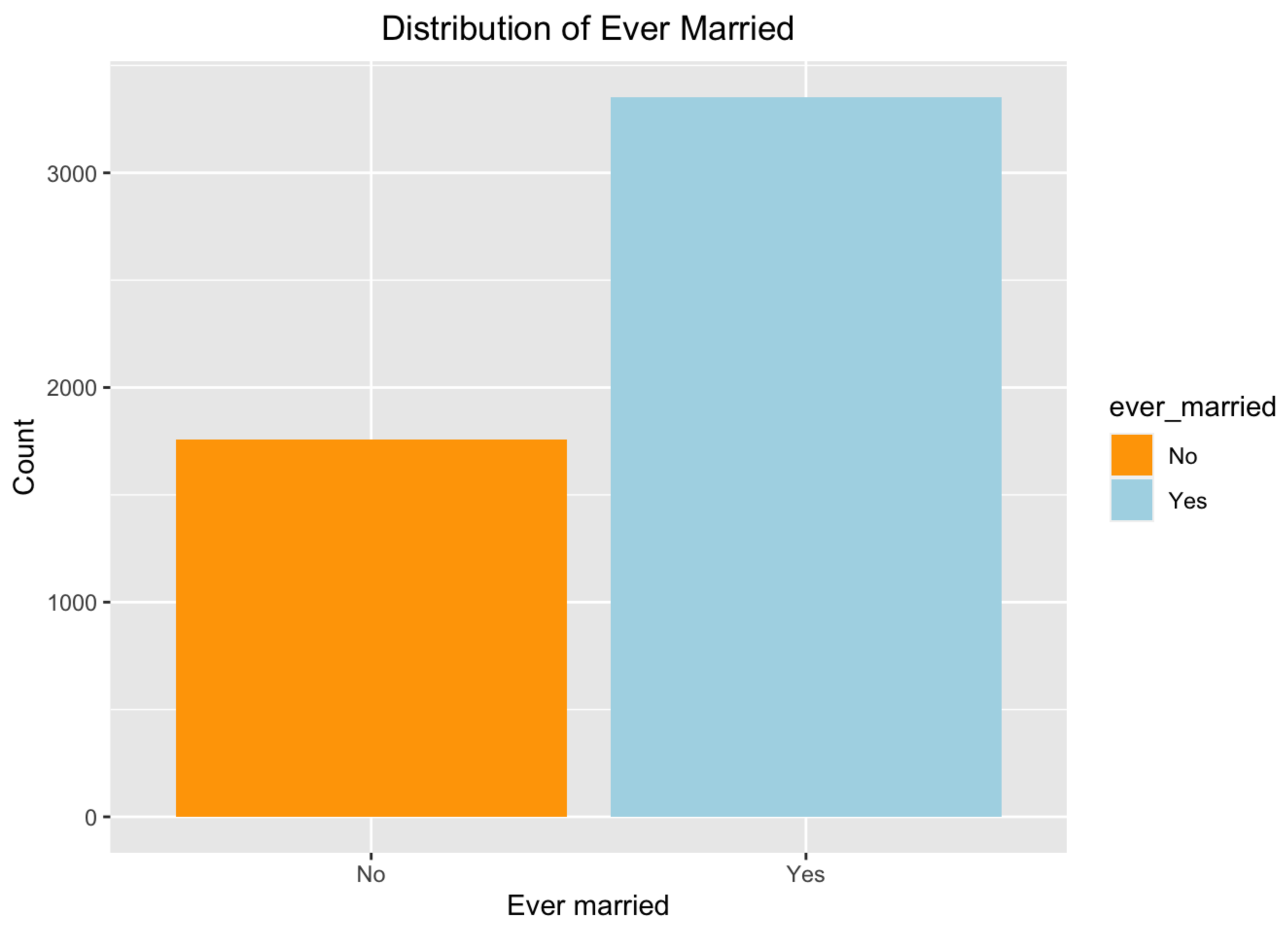
```
#Heart disease
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = heart_disease, fill = heart_disease)) +
  labs(title = "Distribution of Heart disease", x = "Heart disease", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("darkred", "red"))
```



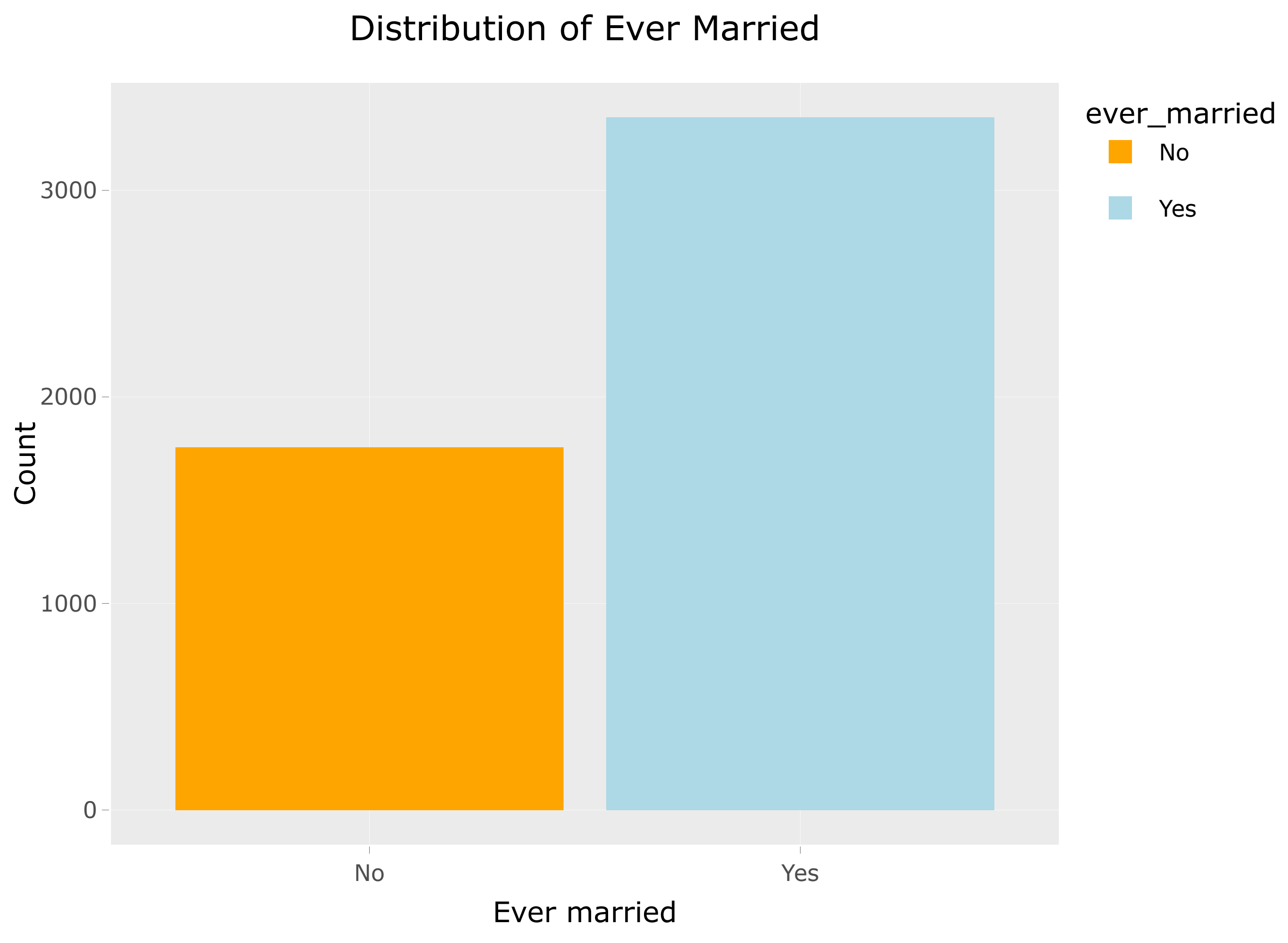
```
ggplotly()
```



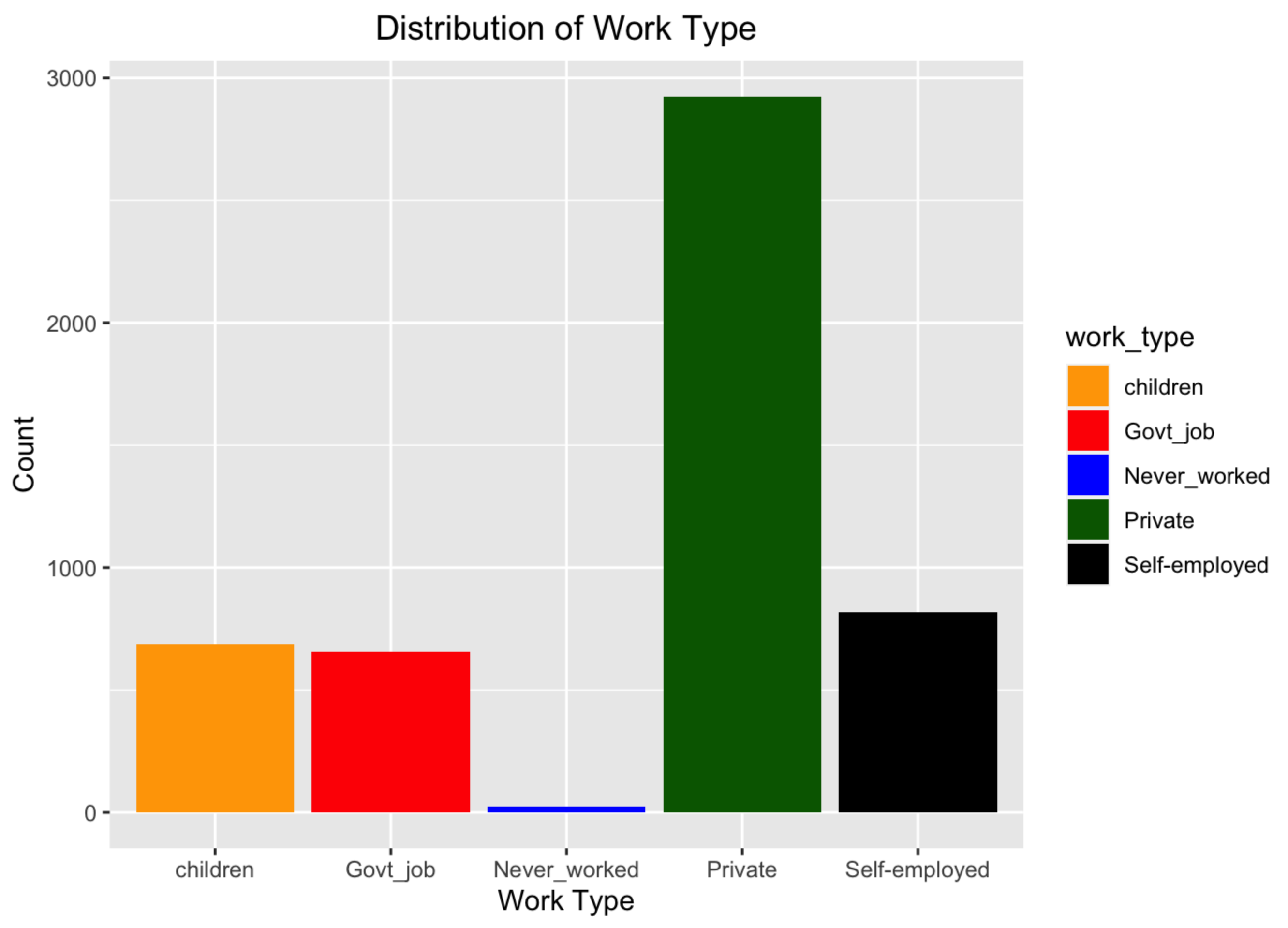
```
# Ever married
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = ever_married, fill = ever_married)) +
  labs(title = "Distribution of Ever Married", x = "Ever married", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("Orange", "lightblue"))
```



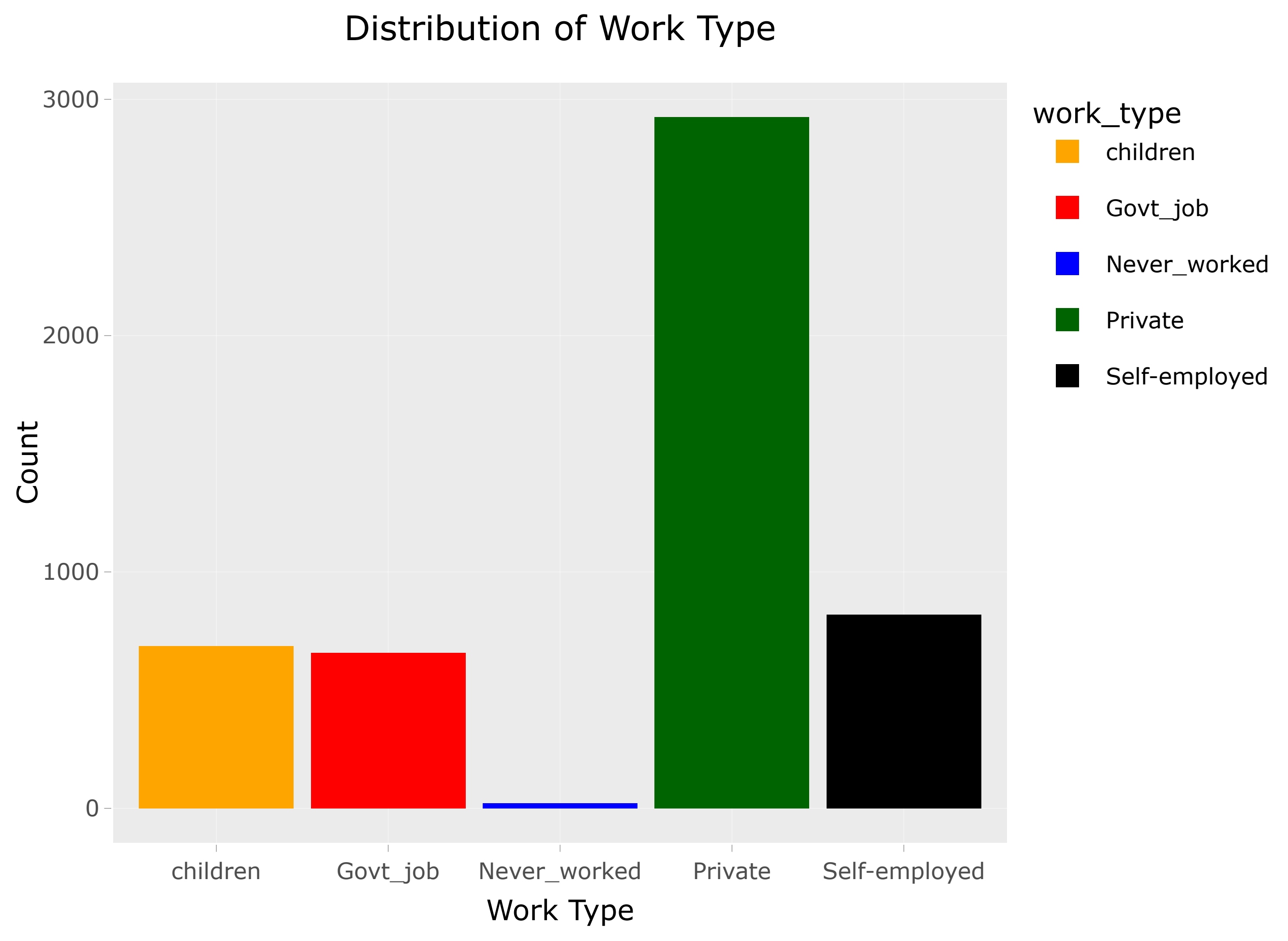
```
ggplotly()
```



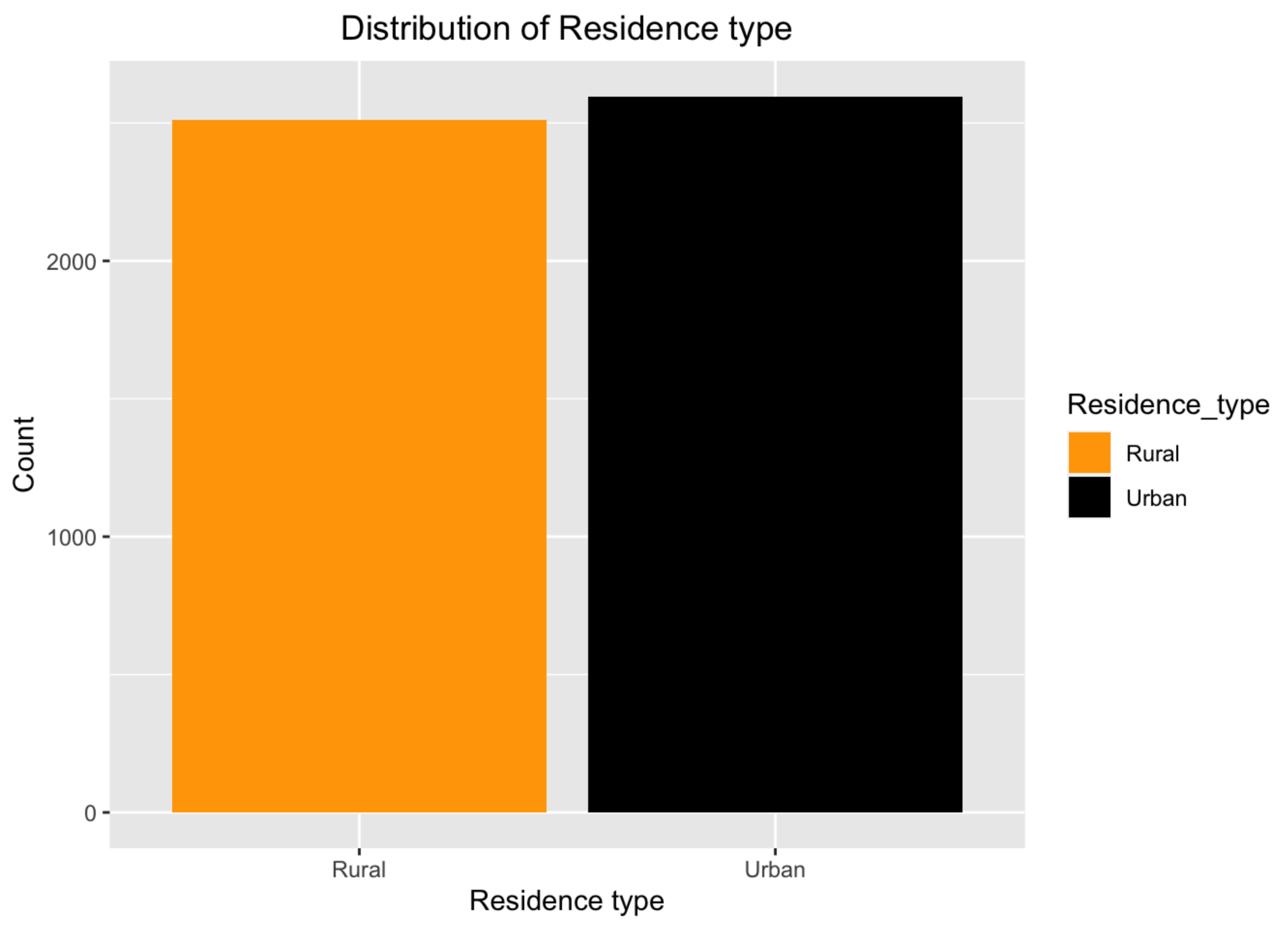
```
# Work type
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = work_type, fill = work_type)) +
  labs(title = "Distribution of Work Type", x = "Work Type", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("Orange", "red", "blue", "darkgreen", "black"))
```



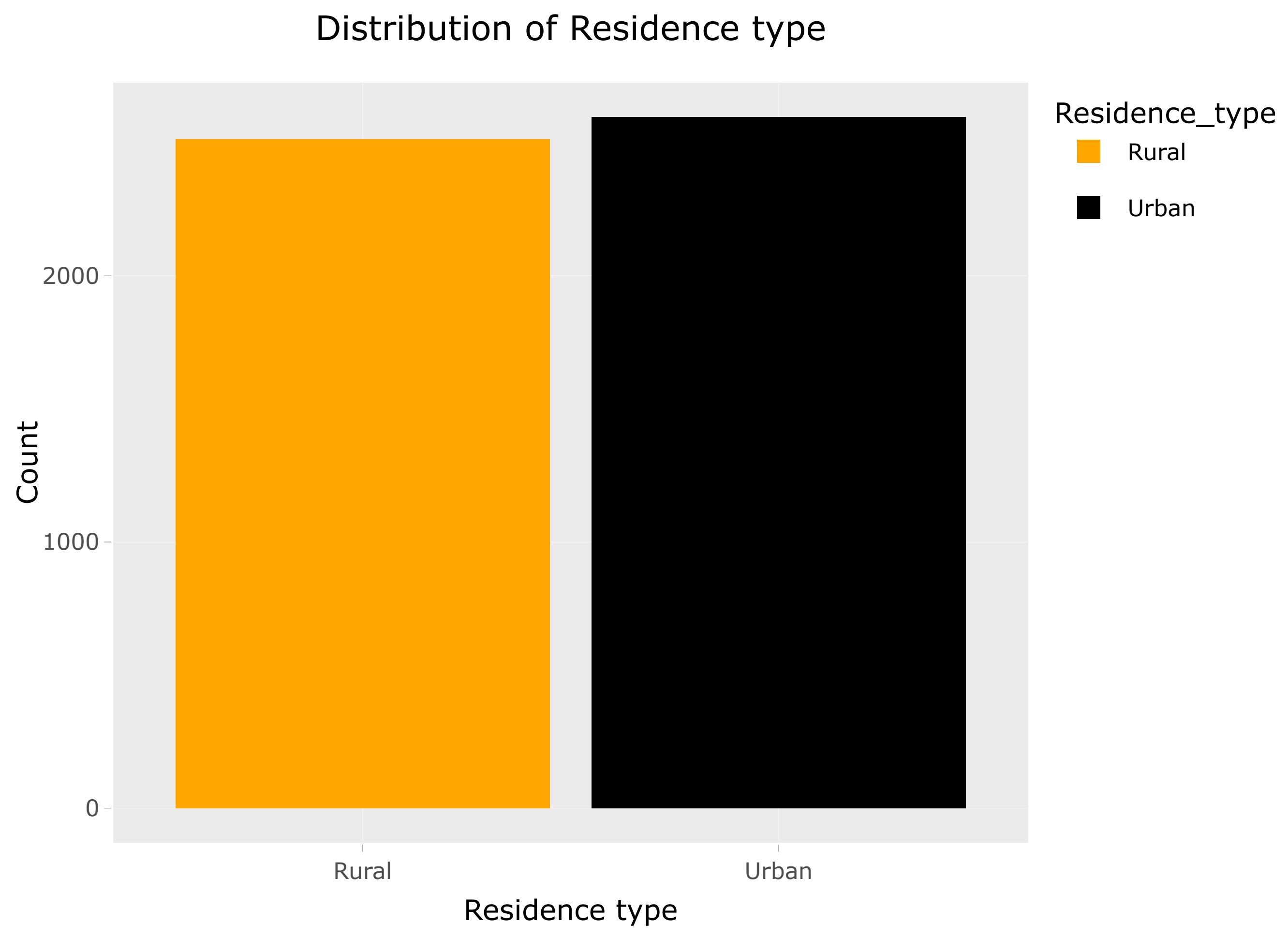
```
ggplotly()
```



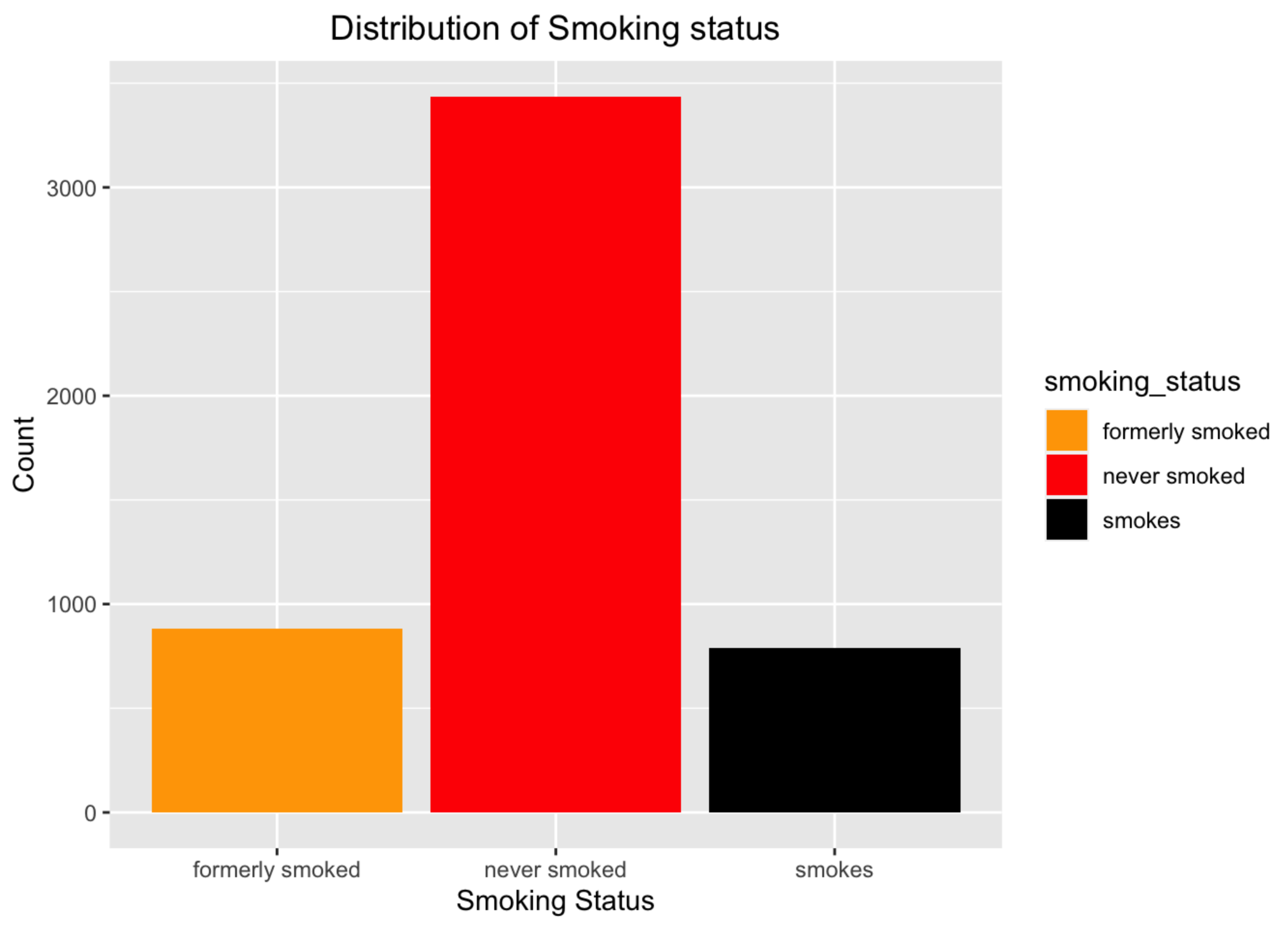
```
# Residence type
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = Residence_type, fill = Residence_type)) +
  labs(title = "Distribution of Residence type", x = "Residence type", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("Orange", "black"))
```



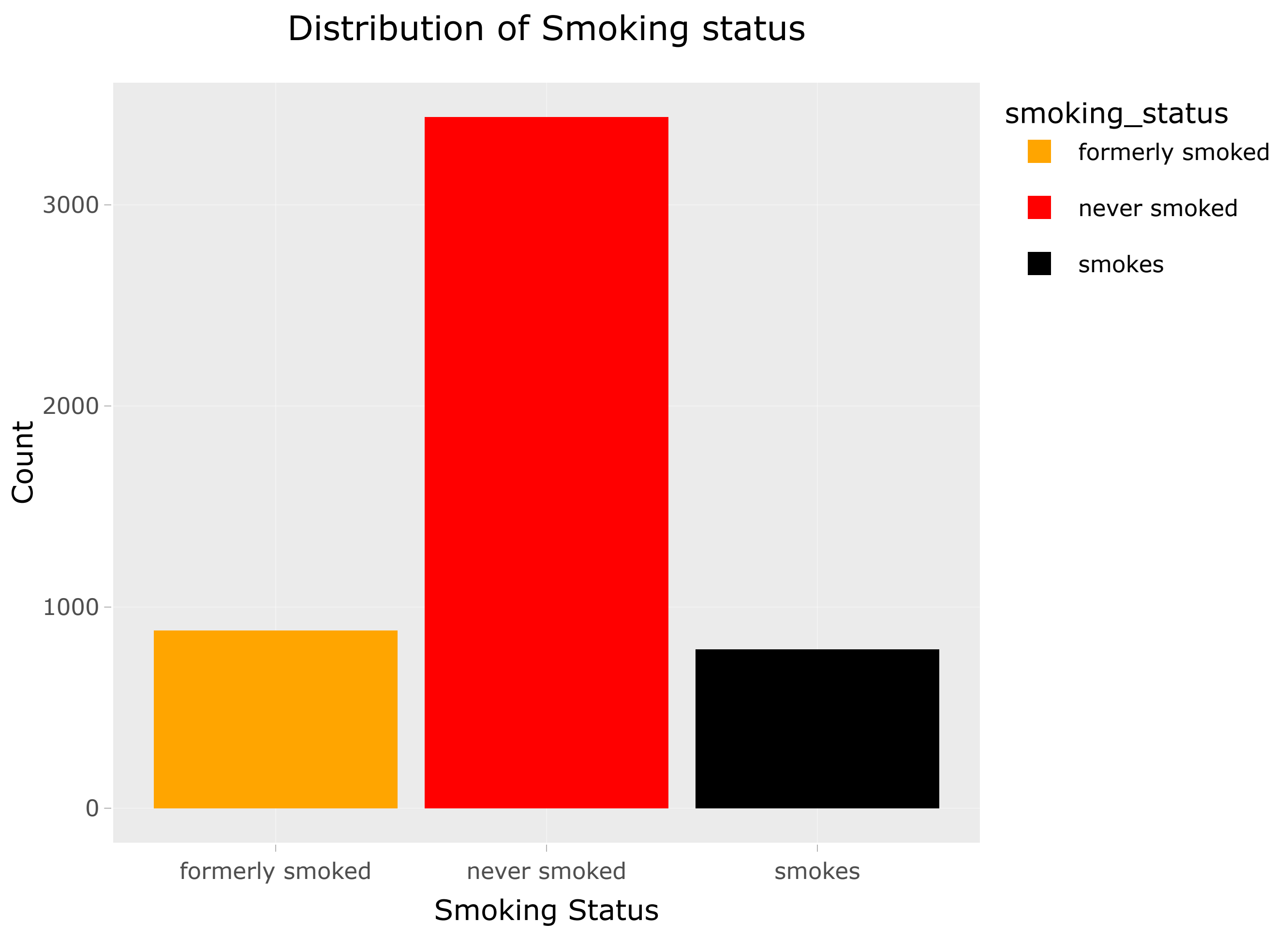
```
ggplotly()
```



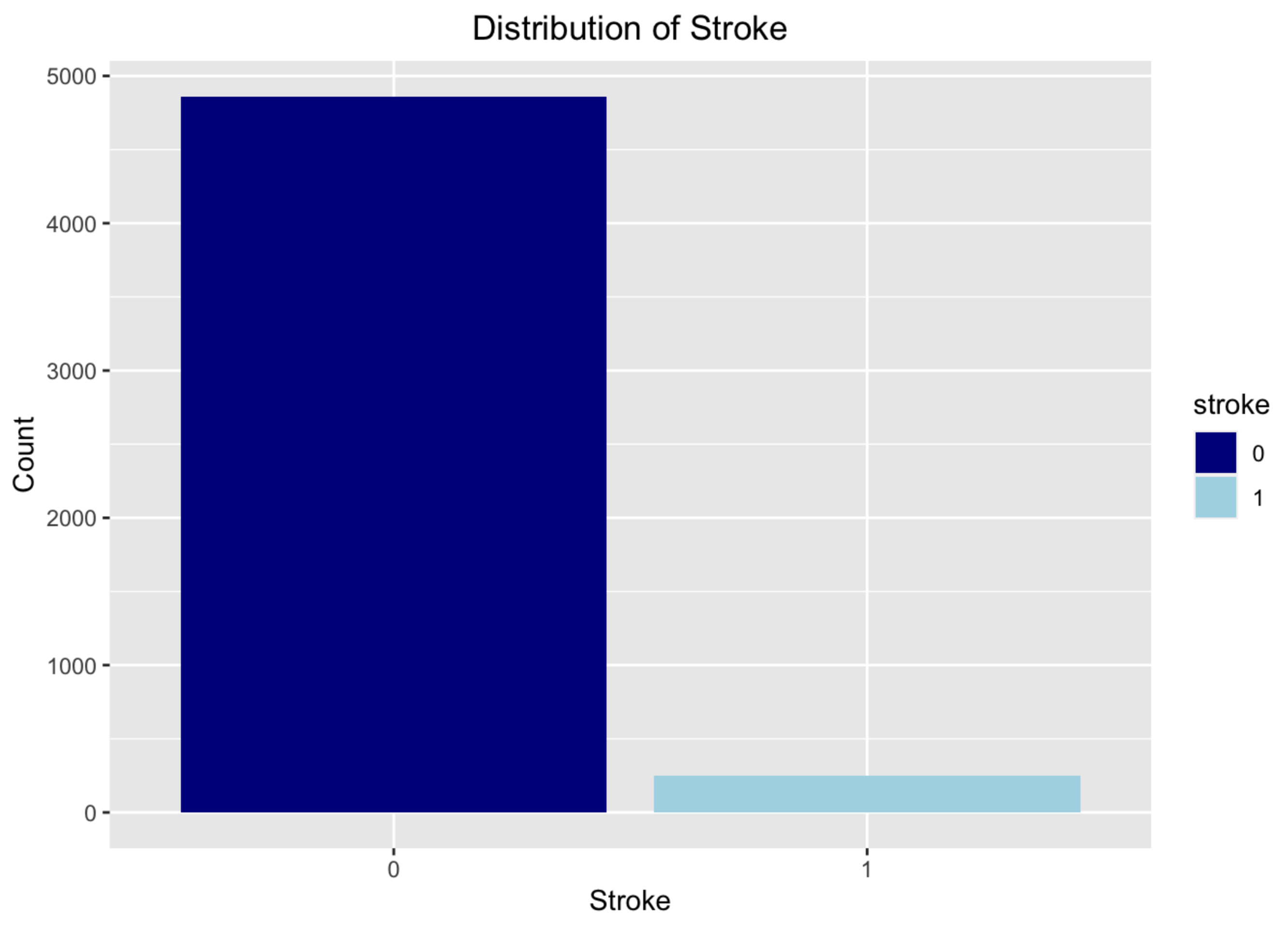
```
# smoking status
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = smoking_status, fill = smoking_status)) +
  labs(title = "Distribution of Smoking status", x = "Smoking Status", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("Orange", "red", "black"))
```



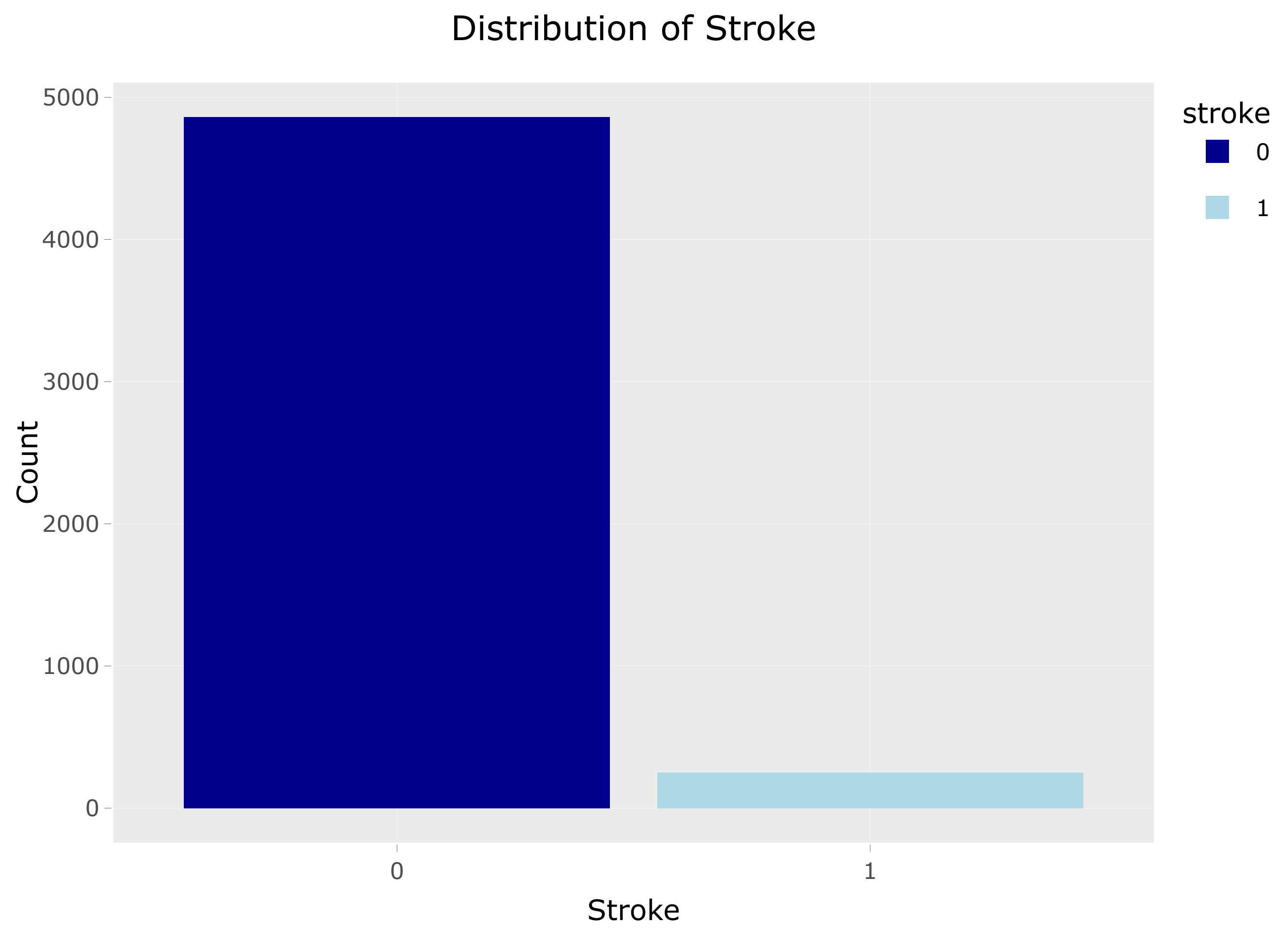
```
ggplotly()
```



```
#stroke
ggplot(data = stroke) +
  geom_bar(mapping = aes(x = stroke, fill = stroke)) +
  labs(title = "Distribution of Stroke", x = "Stroke", y = "Count") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("darkblue", "lightblue"))
```

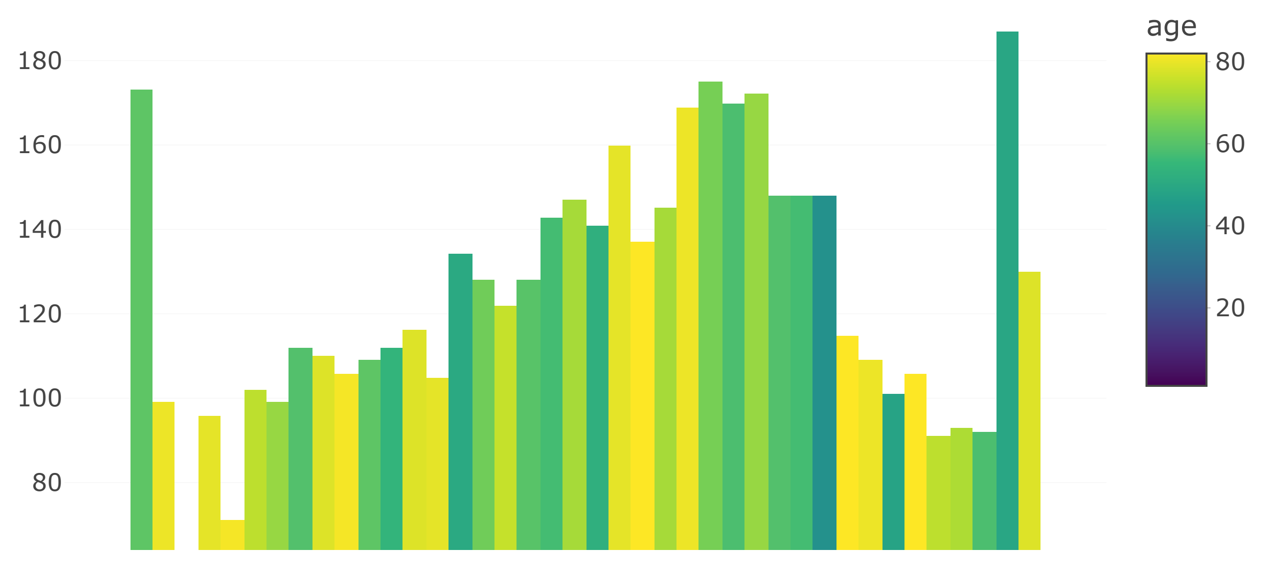


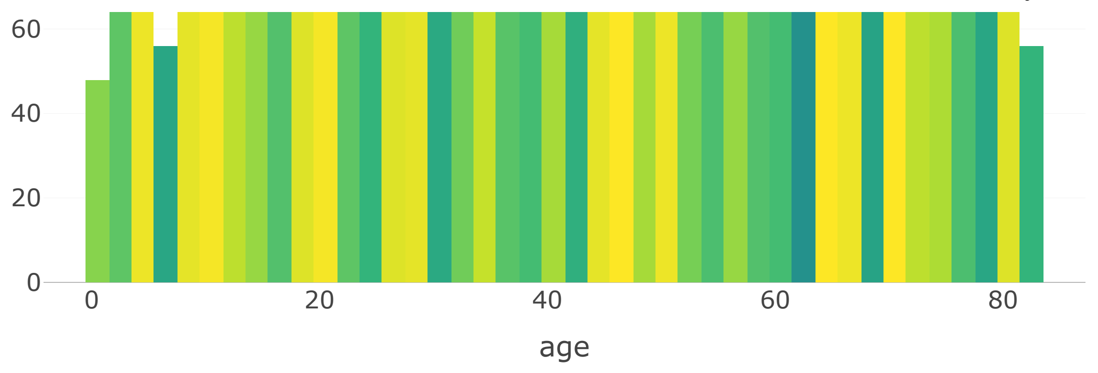
```
ggplotly()
```



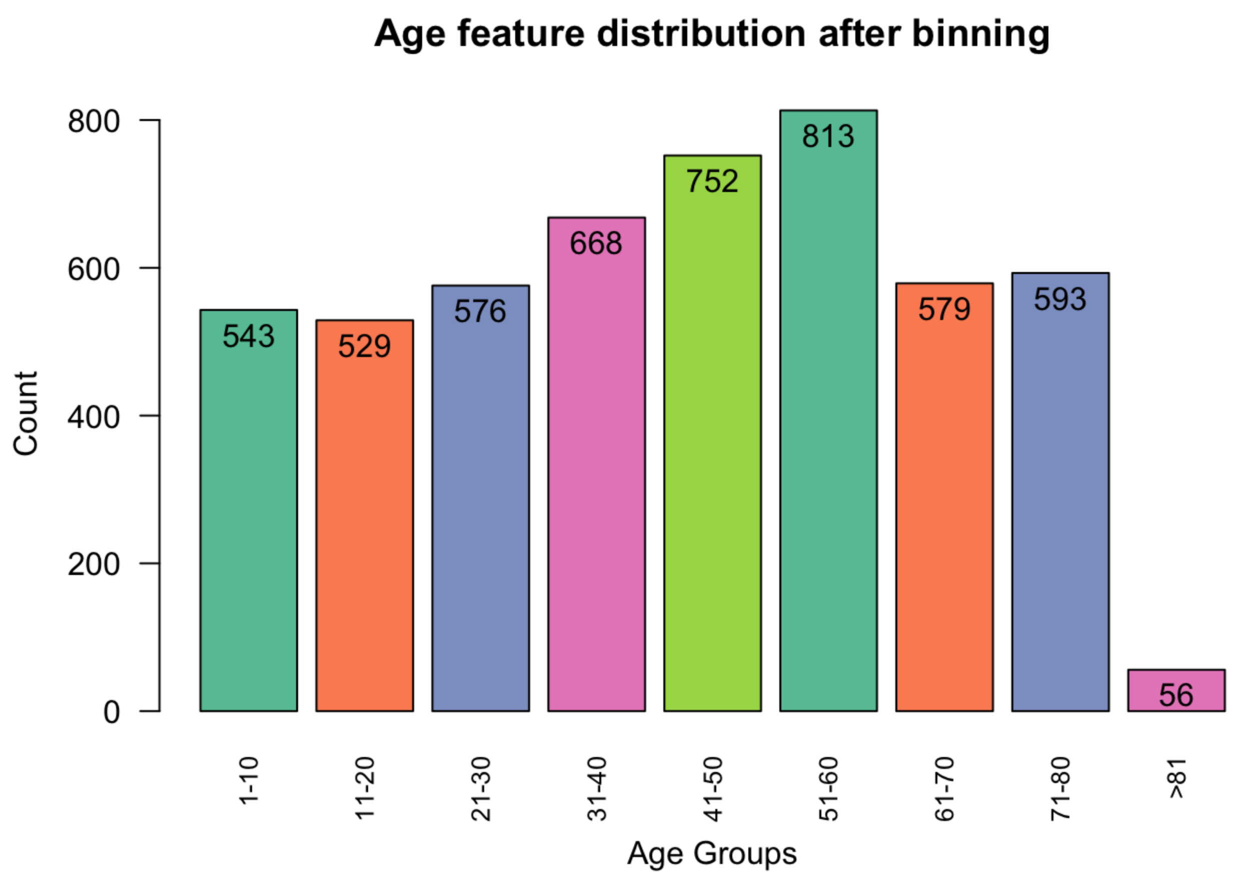
From the above distributions, these are what I observed: 1. The percentage of individuals not having a heart disease and a hypertension is majority. 2. According to the data, there are more number of females than males and the percentage of individuals being married is the majority. 3. Also, the percentage of not having a stroke is the majority as per the data. 4. The private workers category and never smoked category are the majority in work type and smoking status.

```
# Continuous variables

#Age
plot_ly(stroke, x = ~age, color = ~age)%>%
  add_histogram()
```

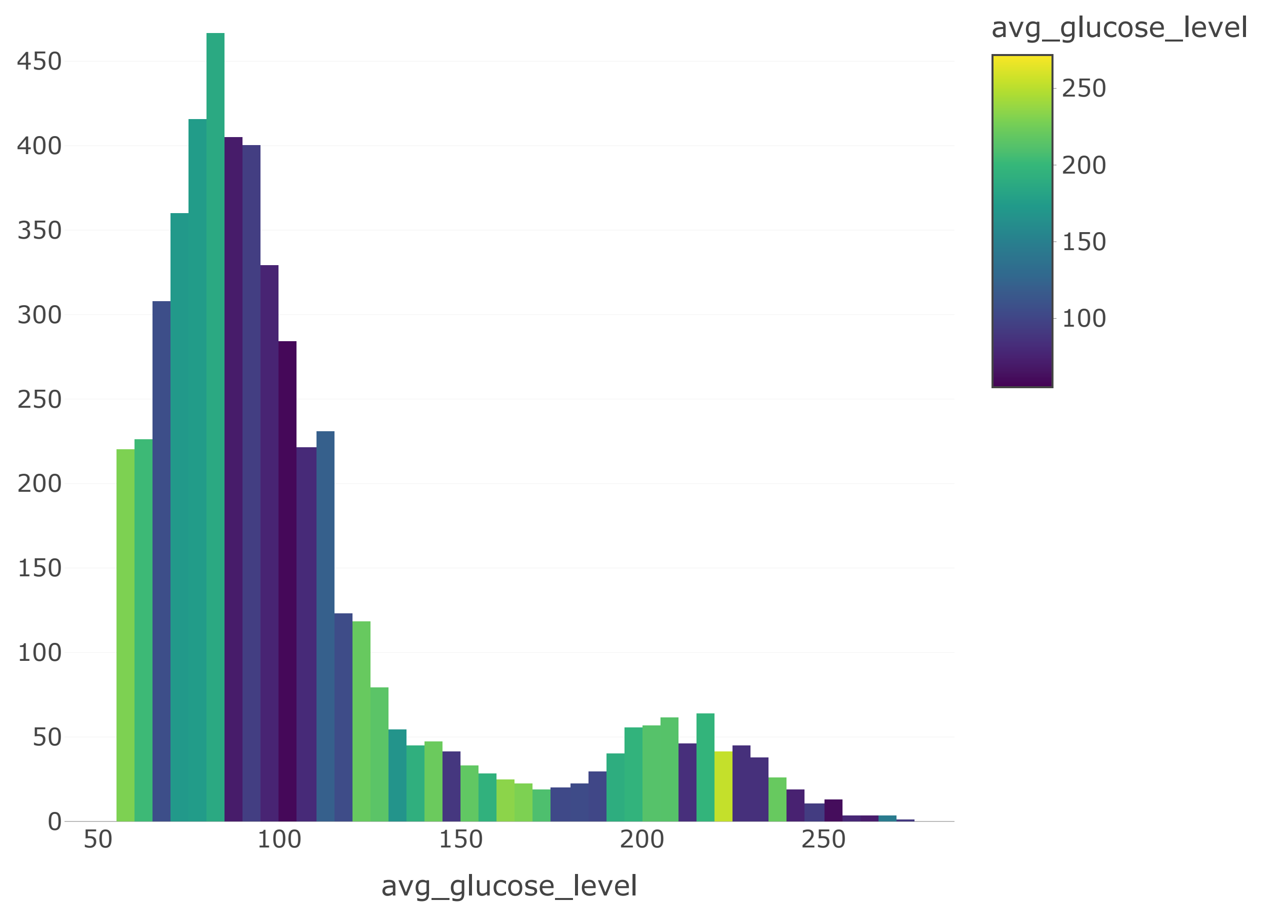




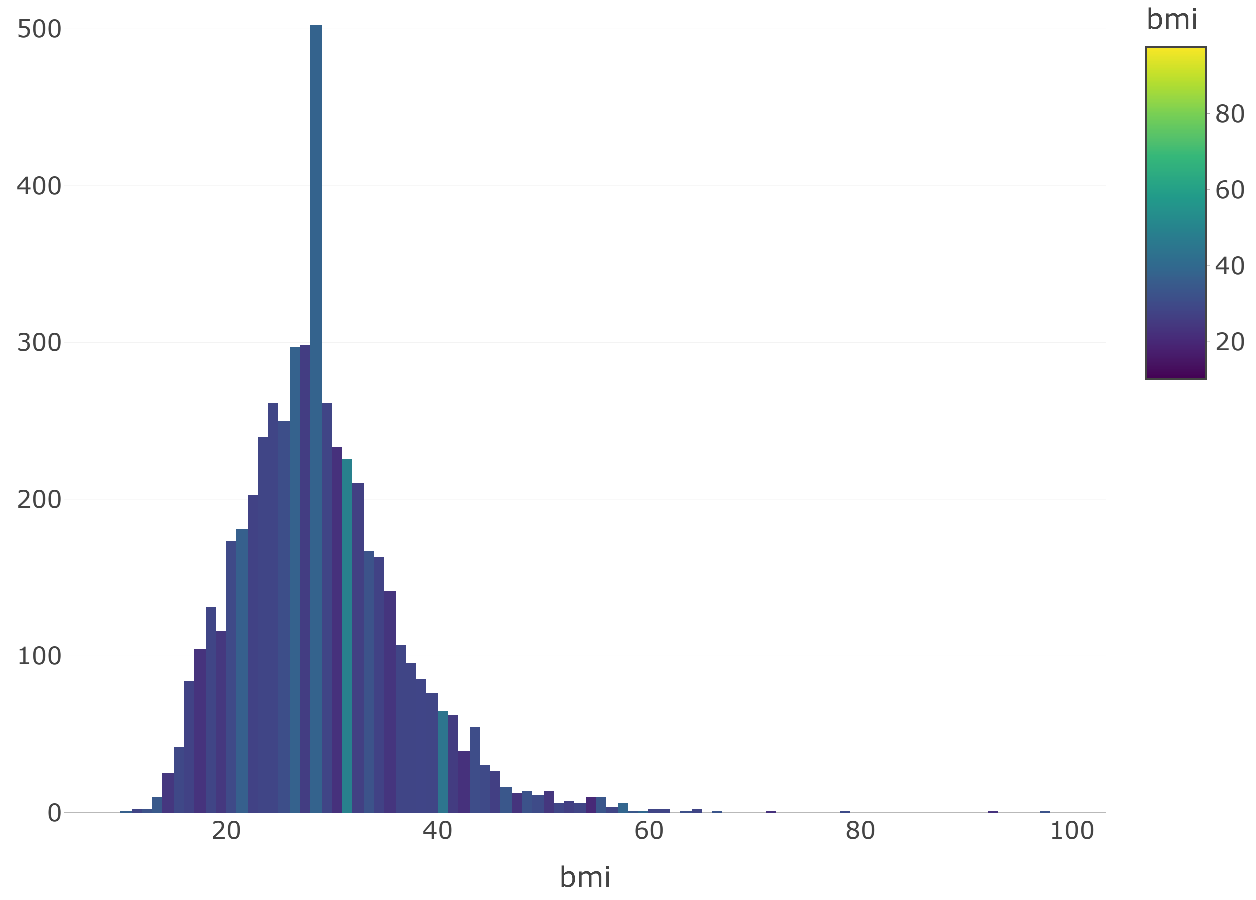
```
# To visualize the Age feature distribution in a better way, we can bin the age column and then display it
age <- stroke$age
age.bins <- cut(age, c(1, 11, 21, 31, 41, 51, 61,  71, 81, Inf), c("1-10", "11-20", "21-30", "31-40", "41-50", "5
1-60", "61-70", "71-80",  ">81"), include.lowest=TRUE, ordered_result = TRUE)
age_table <- table(age.bins)
age.plot <- barplot(table(age.bins), main = "Age feature distribution after binning", col = brewer.pal(5, "Set
2"), xlab = "Age Groups", ylab = "Count", names.arg = levels(age.bins), las = 2, cex.names = 0.8)
text(x = age.plot, y = age_table + 3, labels = as.character(age_table), pos = 1)
```



```
#Average glucose level
plot_ly(stroke, x = ~avg_glucose_level, color = ~avg_glucose_level)%>%
  add_histogram()
```

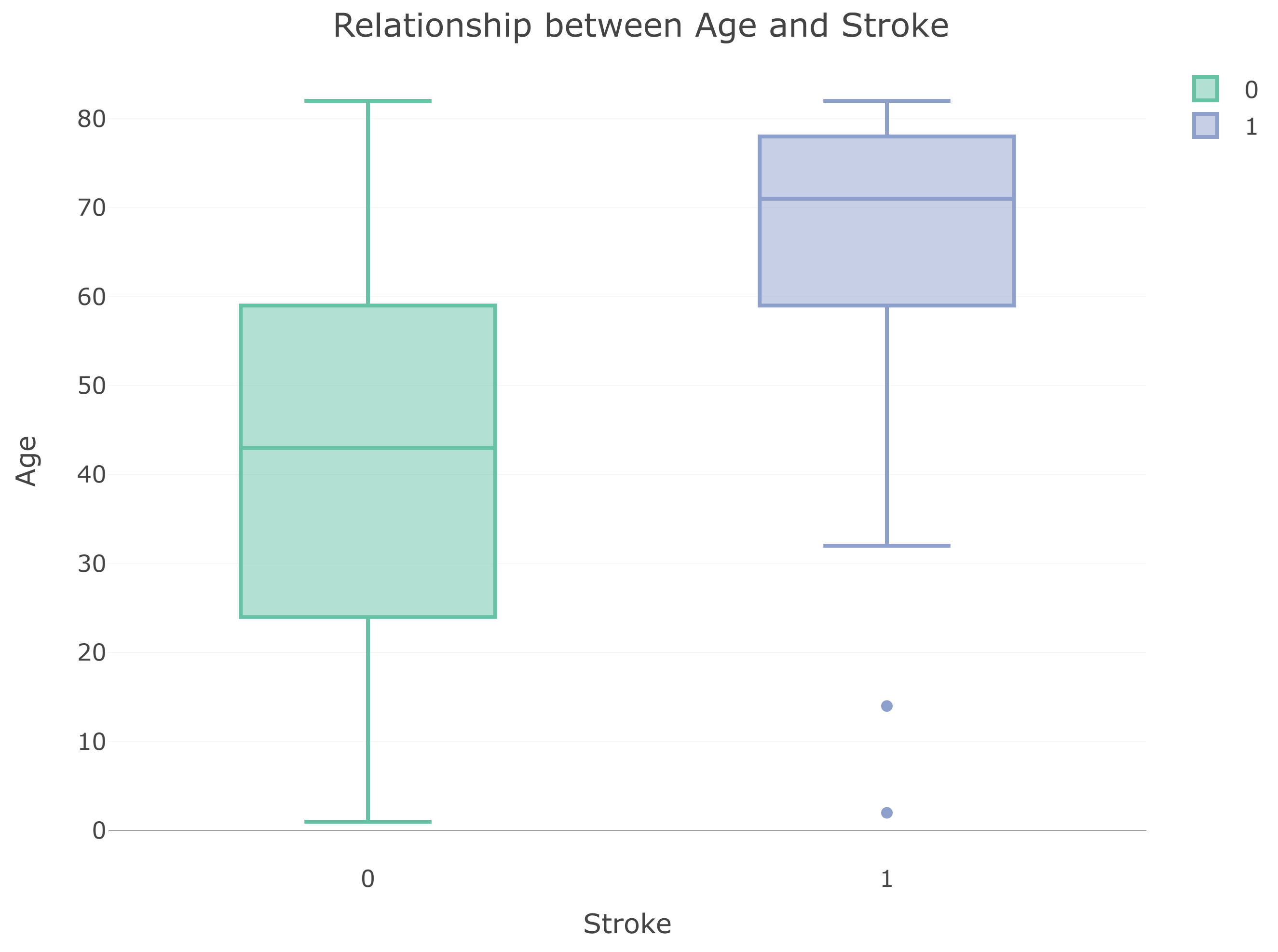


```
#BMI
plot_ly(stroke, x = ~bmi, color = ~bmi)%>%
  add_histogram()
```

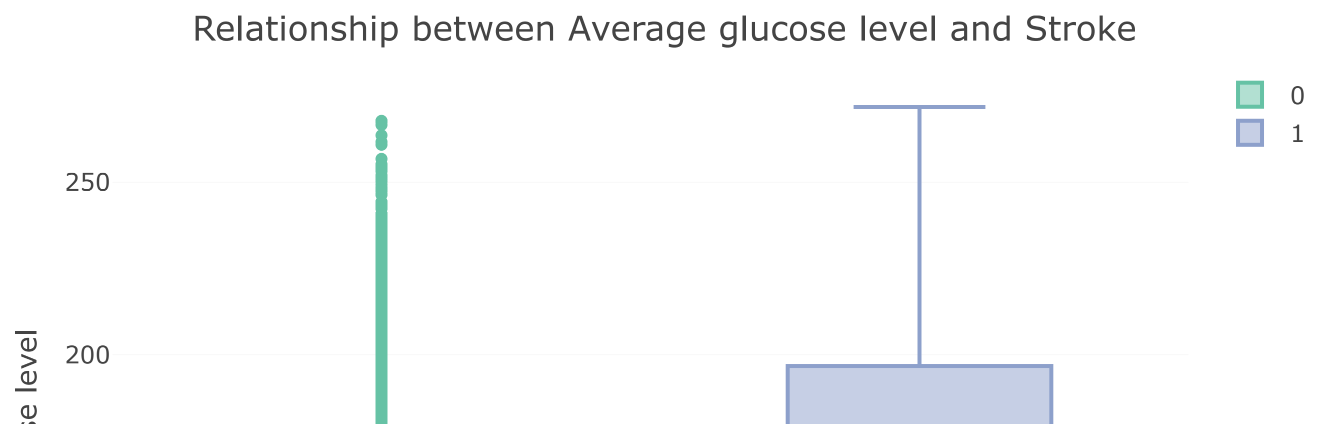


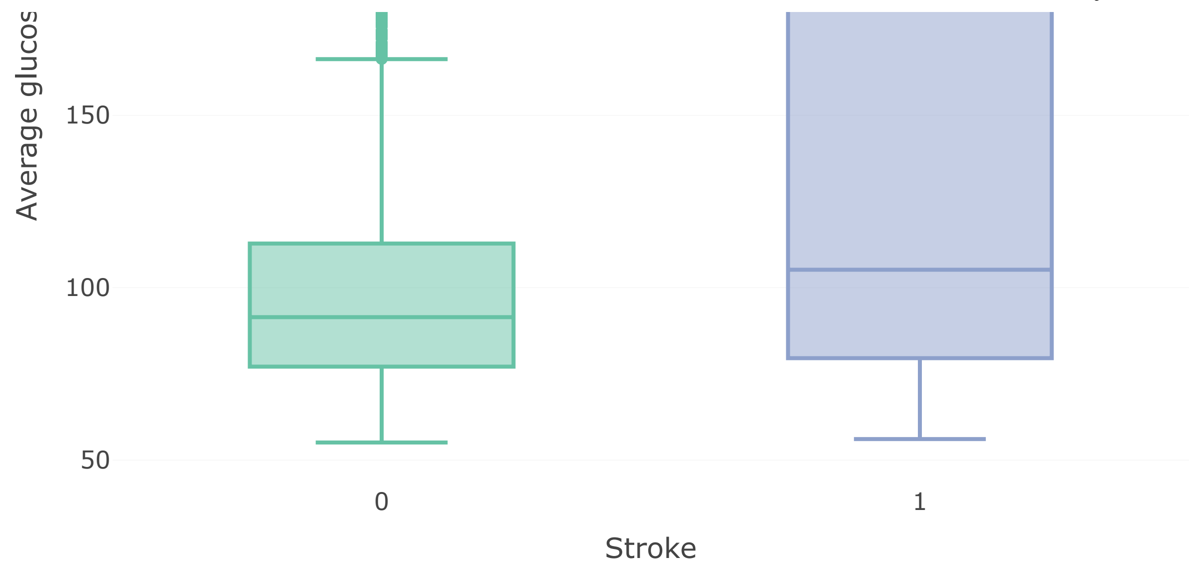
From the above distributions, these are what I observed: 1. The age distribution is slightly negative-skewed 2. The average glucose level and BMI distributions are slightly positive-skewed. 3. The pike in the BMI is due to replacement of missing values with mean value of the column. 4. From the binned age barplot, we can see that the highest number of people are in the age group of 51-60.

```
# Now, we check the relationship between stroke (the outcome variable) with different input variables

# Using boxplots is best for checking the relationship between the continuous variables and categorical variable
s.

# Age & Stroke
plot_ly(stroke, x = ~stroke, y = ~age, type = "box", color = ~stroke) %>%
  layout(xaxis = list(title = "Stroke"), yaxis = list(title = "Age")) %>%
  layout(title = "Relationship between Age and Stroke")
```

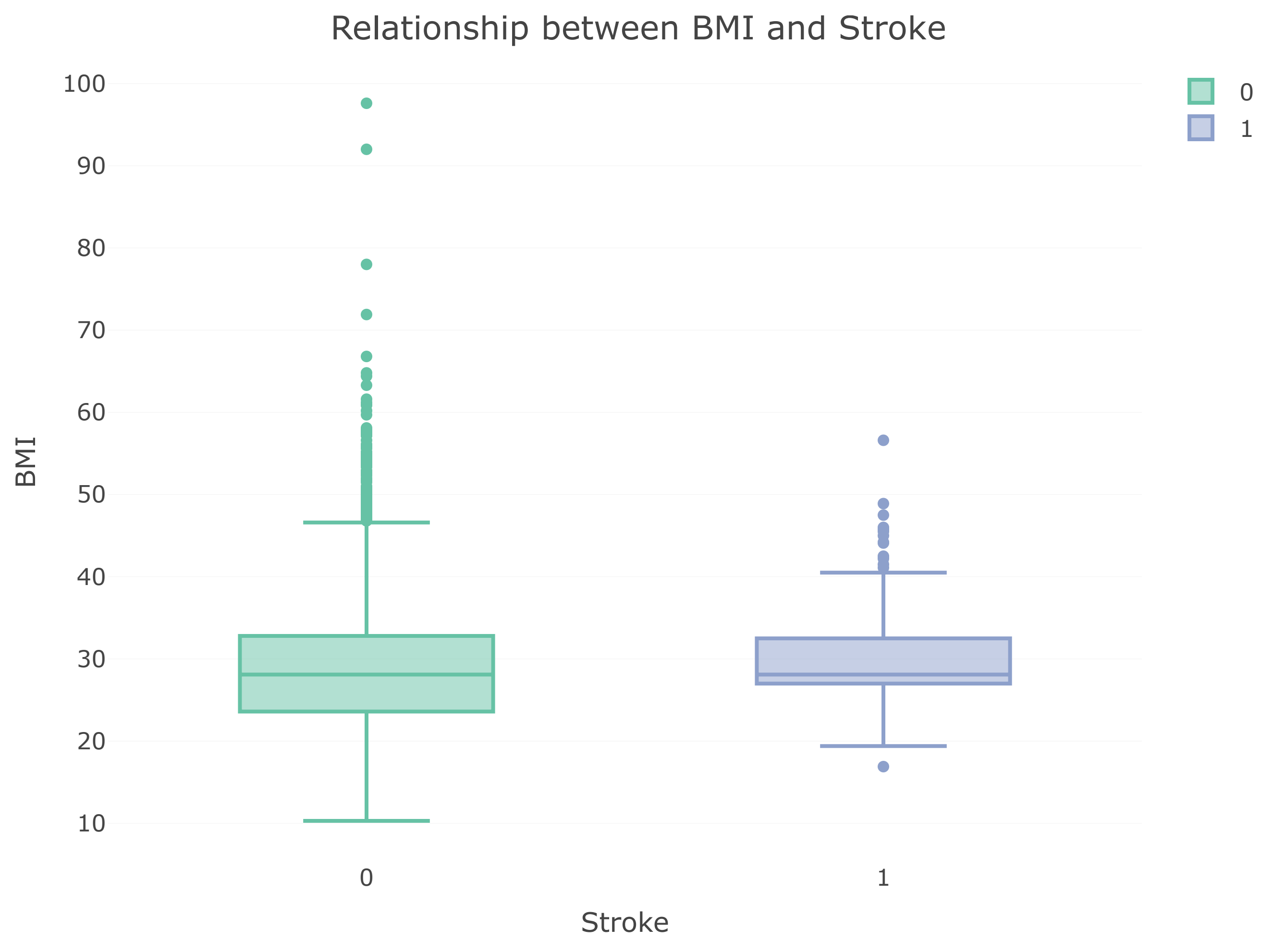


```
# Average glucose level & Stroke
plot_ly(stroke, x = ~stroke, y = ~avg_glucose_level, type = "box", color = ~stroke) %>%
  layout(xaxis = list(title = "Stroke"), yaxis = list(title = "Average glucose level")) %>%
  layout(title = "Relationship between Average glucose level and Stroke")
```



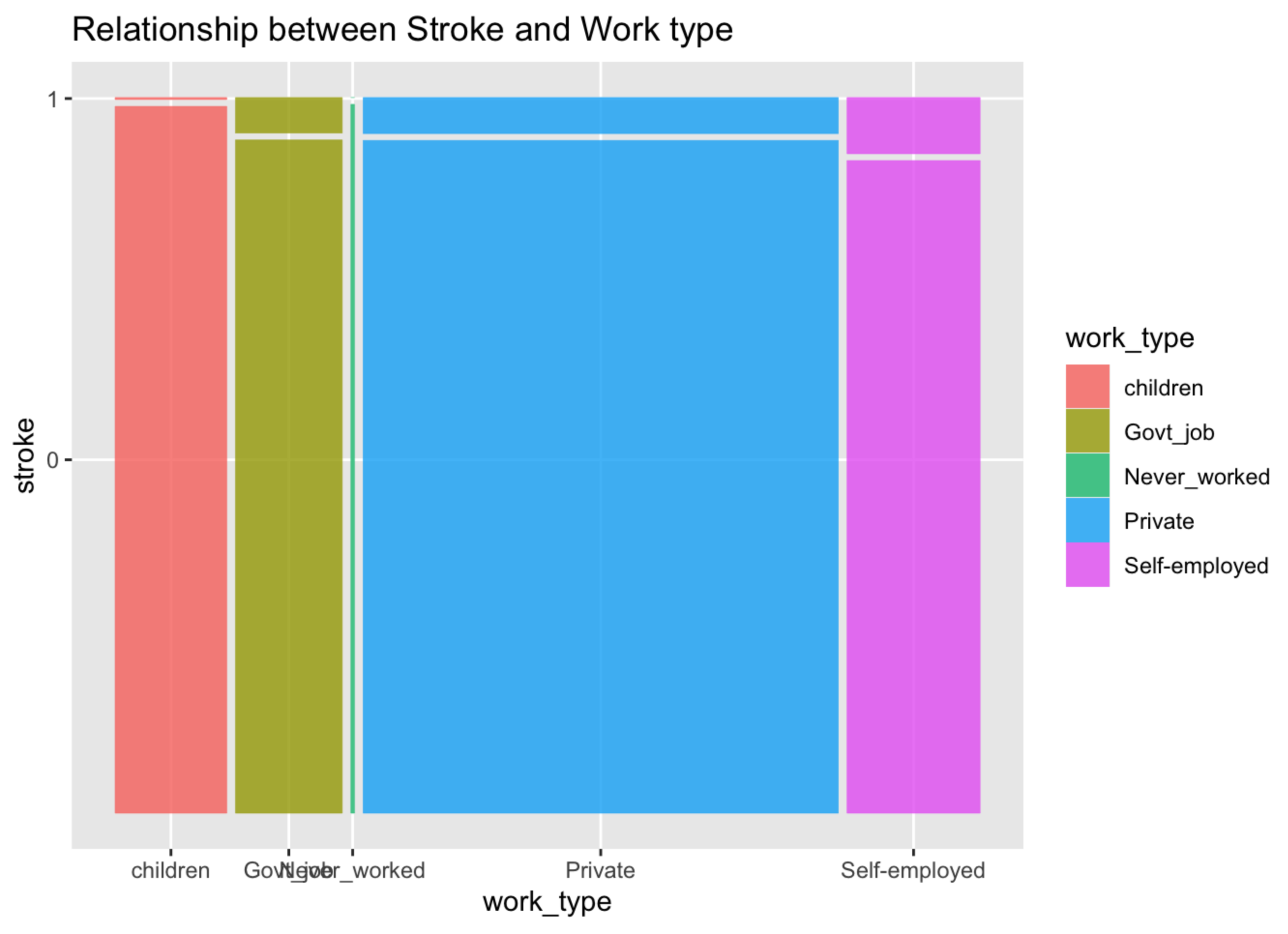


```
# BMI & Stroke
plot_ly(stroke, x = ~stroke, y = ~bmi, type = "box", color = ~stroke) %>%
  layout(xaxis = list(title = "Stroke"), yaxis = list(title = "BMI")) %>%
  layout(title = "Relationship between BMI and Stroke")
```

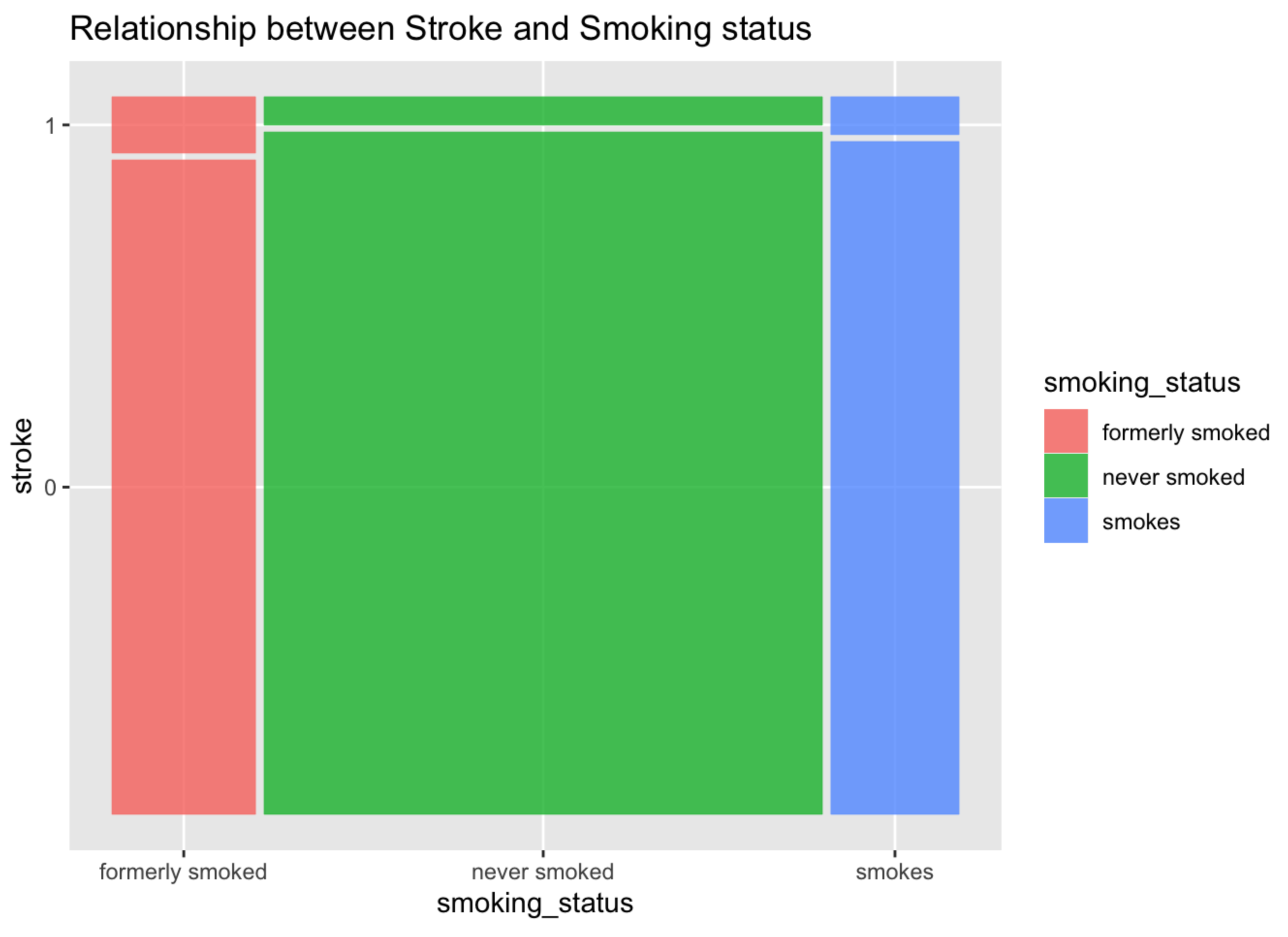


From the above distributions, this is what I observed: 1. The elder individuals are most prone for a stroke prediction. 2. The individuals with high average glucose level are more inclined for a stroke. 3. The individuals with high BMI levels are the ones without stroke.

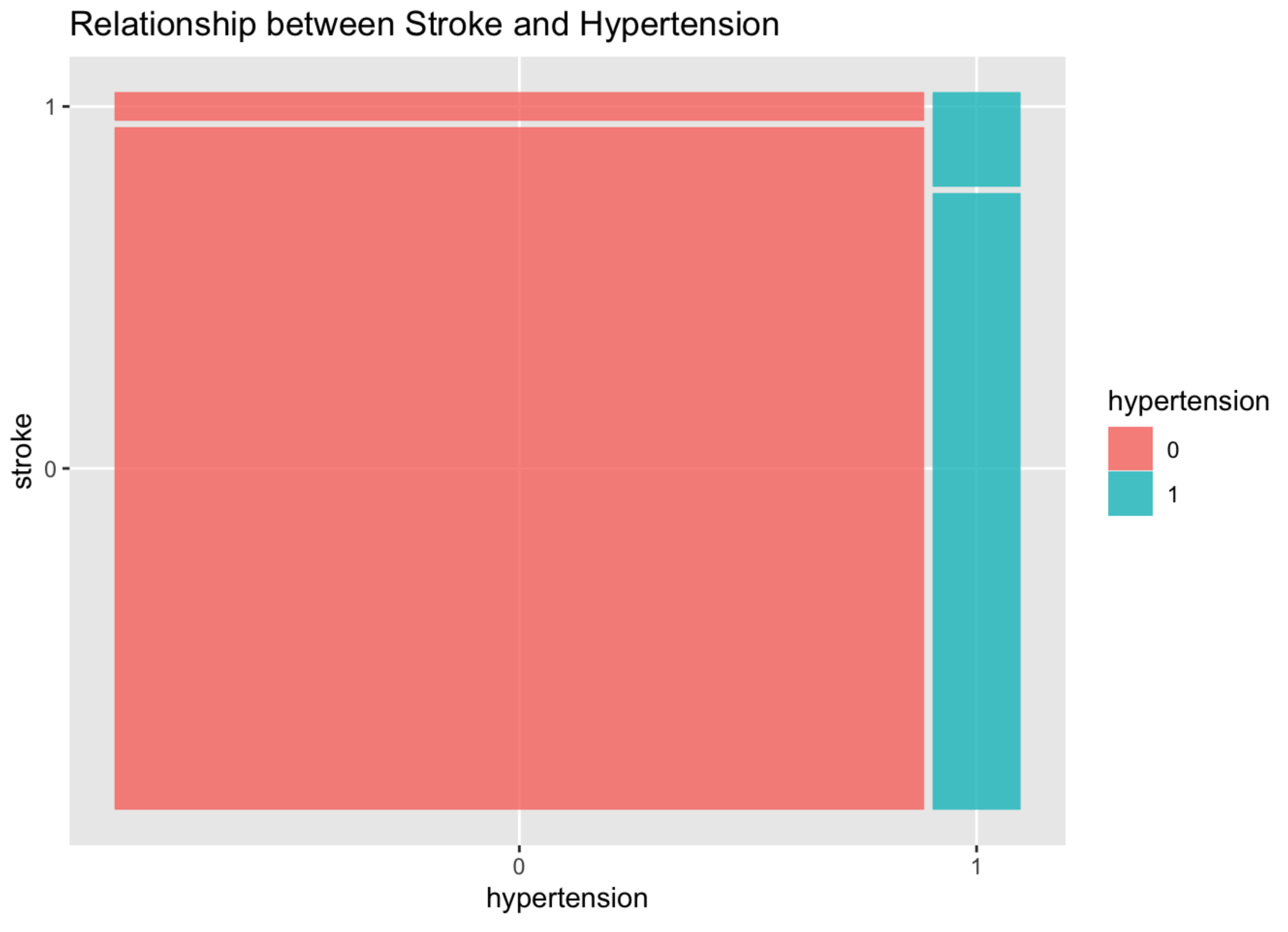
```
# Check the relationship between the discrete variables and the stroke variable using mosaic plot

# Work type & Stroke
ggplot(data = stroke) +geom_mosaic(aes(x = product(stroke,work_type), fill=work_type)) + labs(title='Relationship
between Stroke and Work type')
```



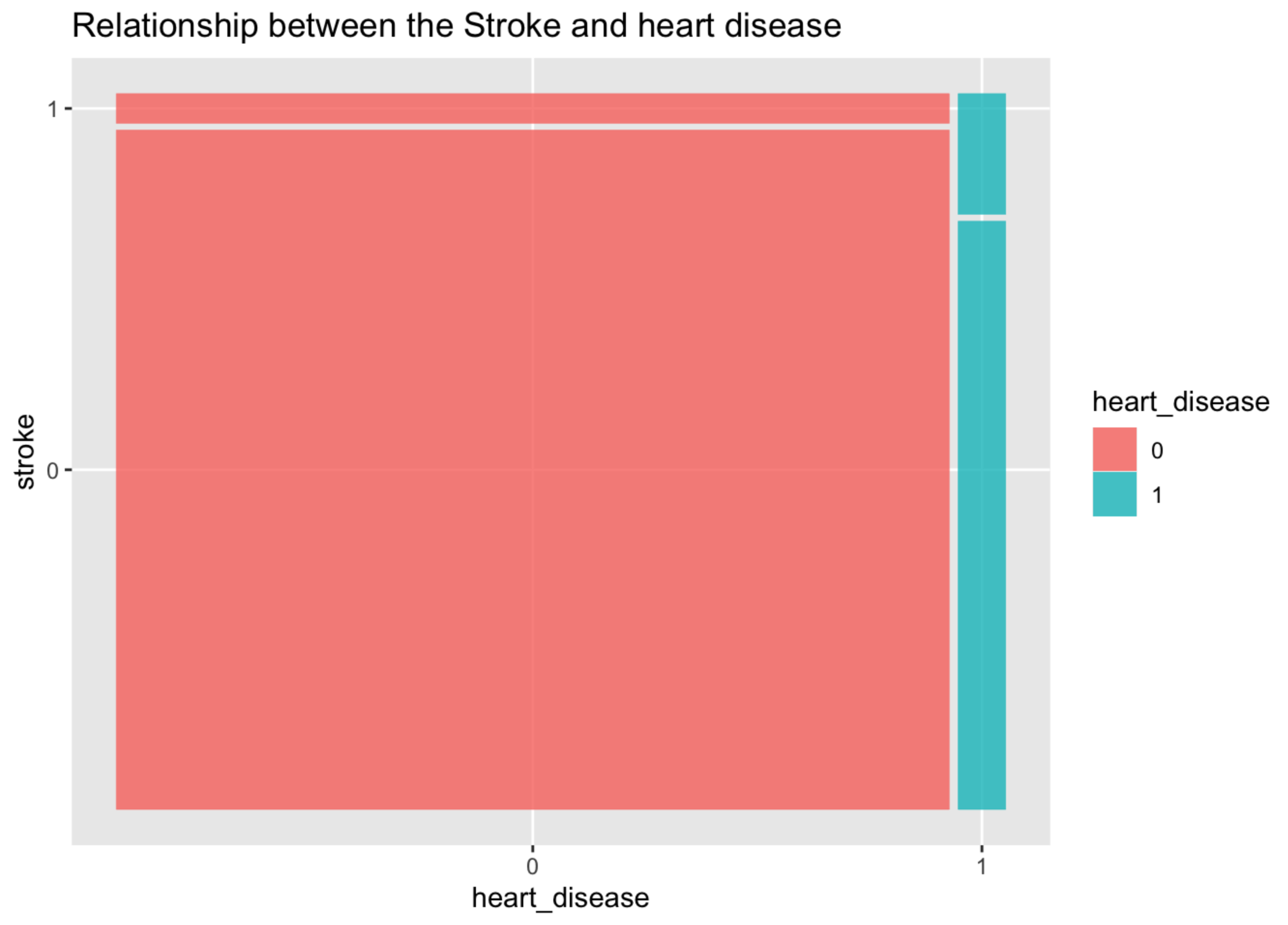
```
# Smoking status & Stroke
ggplot(data = stroke) +geom_mosaic(aes(x = product(stroke,smoking_status), fill=smoking_status)) + labs(title='Re
lationship between Stroke and Smoking status')
```



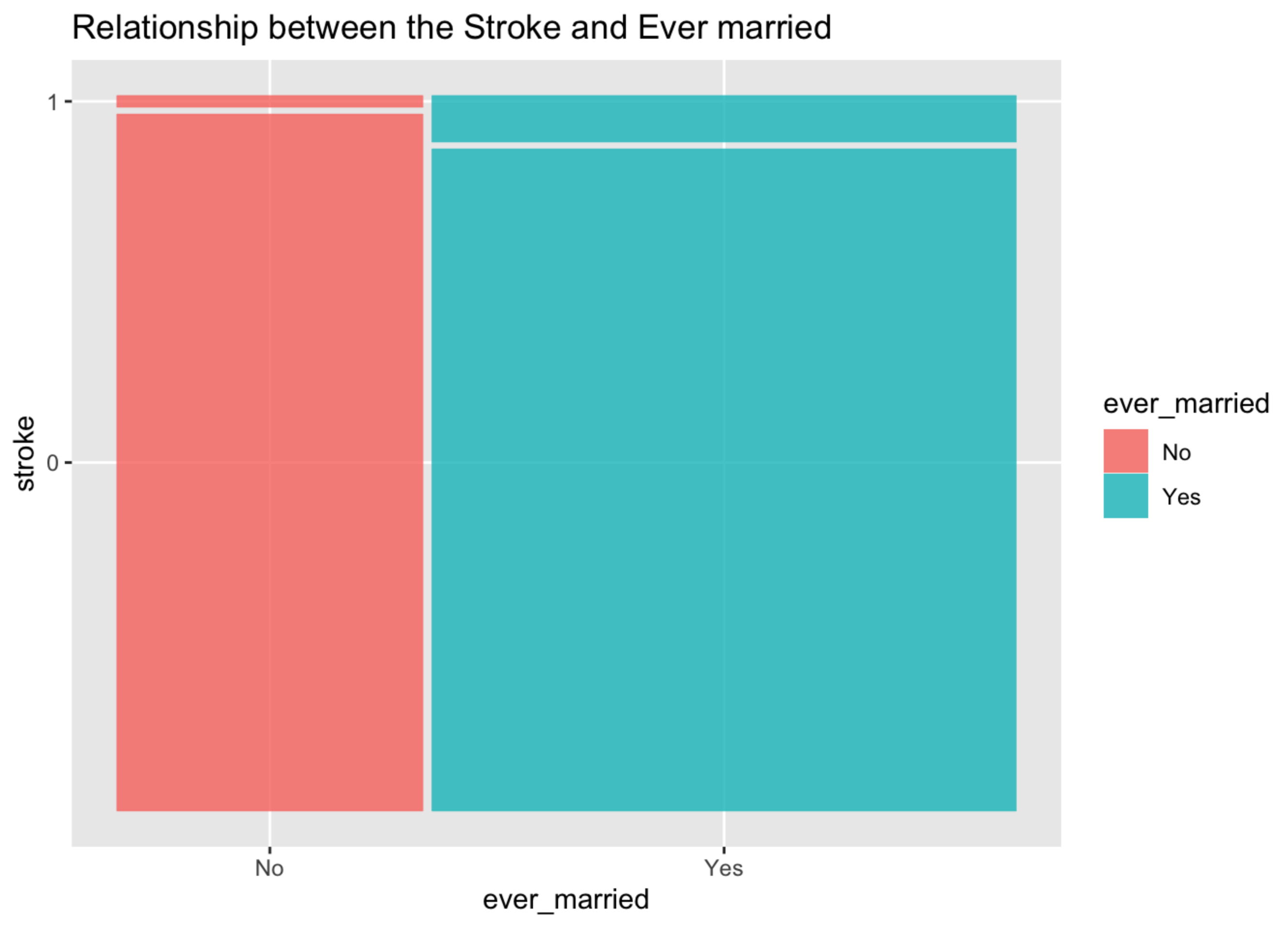
```
# Hypertension & Stroke
ggplot(data = stroke) +geom_mosaic(aes(x = product(stroke,hypertension), fill=hypertension)) + labs(title='Relati
onship between Stroke and Hypertension')
```



```
# Heart disease & Stroke
ggplot(data = stroke) +geom_mosaic(aes(x = product(stroke,heart_disease), fill=heart_disease)) + labs(title='Rela
tionship between the Stroke and heart disease')
```



```
# Ever married & Stroke
ggplot(data = stroke) +geom_mosaic(aes(x = product(stroke,ever_married), fill= ever_married)) + labs(title='Relat
ionship between the Stroke and Ever married')
```



From the above distributions, this is what I observed: 1. The individuals whose work type is self employed are most likely to get a stroke. 2. Individuals with high hypertension or heart disease are most likely to get a stroke. 3. Smoking has a little effect on the stroke prediction. 4. The individuals who are married are most likely to get a stroke.

Correlation between the dependent and the independent variables

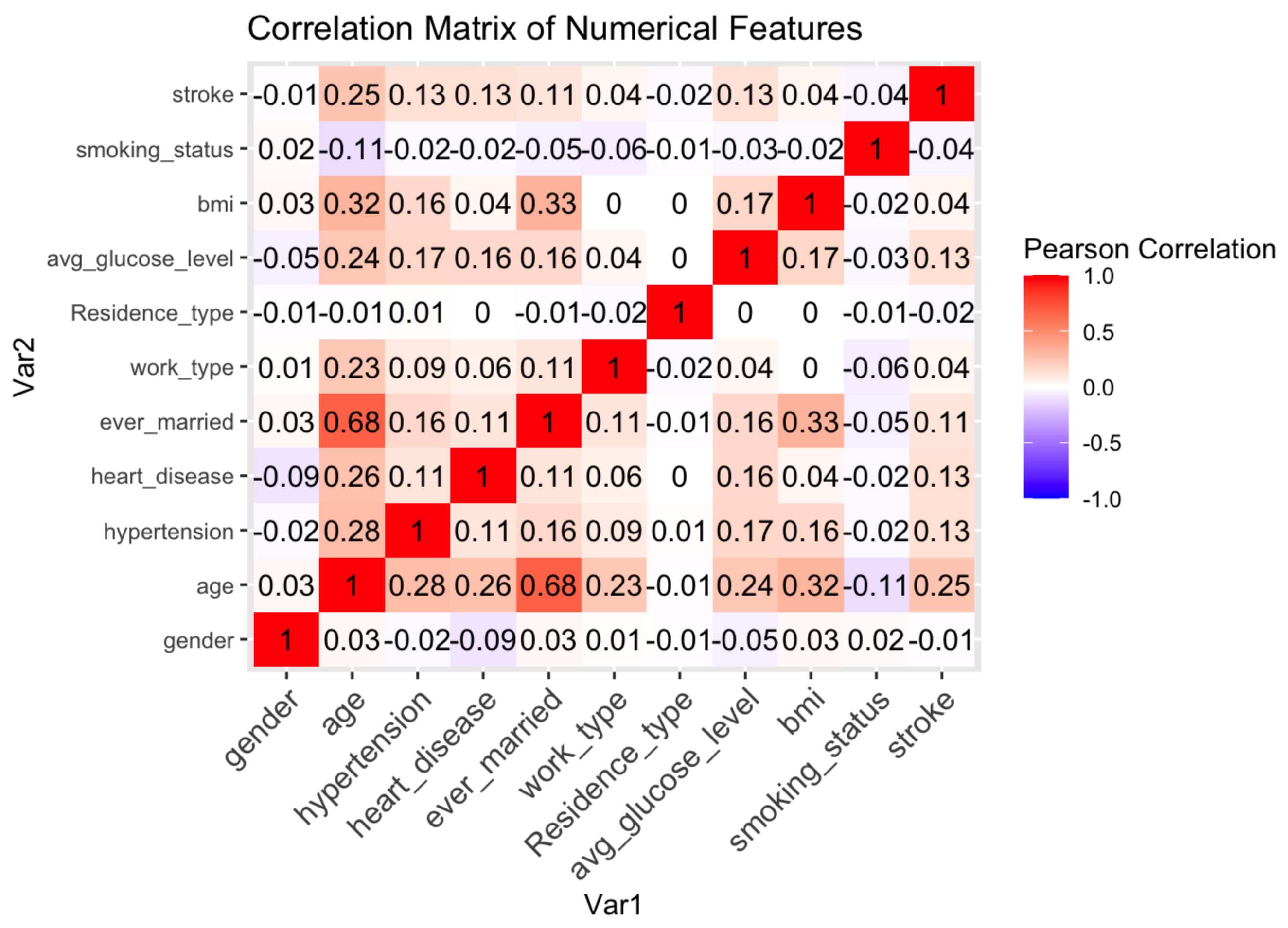
```
# Create another copy of the stroke dataset to work on the correlation to check the relationship between the feat
ures and the target variable
stroke_data  <- stroke

# Convert the categorical variables into binary values which are easier to analyse
#ever married
stroke_data$ever_married <- ifelse(stroke_data$ever_married == "Yes", 1, 0)

#gender
stroke_data$gender <- ifelse(stroke$gender == "Female", 1, 0)

# residence type
stroke_data$Residence_type <- ifelse(stroke_data$Residence_type == "Rural", 1, 0)

#smoking status
stroke_data$smoking_status[stroke_data$smoking_status == "formerly smoked"] <- 0
stroke_data$smoking_status[stroke_data$smoking_status == "never smoked"] <- 1
stroke_data$smoking_status[stroke_data$smoking_status == "smokes"] <- 2

# work type
stroke_data$work_type[stroke_data$work_type == "Private"] <- 0
stroke_data$work_type[stroke_data$work_type == "children"] <- 1
stroke_data$work_type[stroke_data$work_type == "Govt_job"] <- 2
stroke_data$work_type[stroke_data$work_type == "Never_worked"] <- 3
stroke_data$work_type[stroke_data$work_type == "Self-employed"] <- 4

# Check if the converted values are ready for the analysis
str(stroke_data)
```

```
## 'data.frame':    5109 obs. of  11 variables:
##  $ gender           : num  0 1 0 1 1 0 0 1 1 1 ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 1 1 1 ...
##  $ heart_disease    : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 1 1 ...
##  $ ever_married     : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ work_type        : chr  "0" "4" "0" "0" ...
##  $ Residence_type   : num  0 1 1 0 1 0 1 0 1 0 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ smoking_status   : chr  "0" "1" "1" "2" ...
##  $ stroke           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
# Convert the factor variables into numeric variables
stroke_data$heart_disease <- as.numeric(stroke_data$heart_disease)
stroke_data$hypertension <- as.numeric(stroke_data$hypertension)
stroke_data$stroke <- as.numeric(stroke_data$stroke)
stroke_data$work_type <- as.numeric(as.character(stroke_data$work_type))
stroke_data$smoking_status <- as.numeric(as.character(stroke_data$smoking_status))

# Check if the conversion happened successfully or not
str(stroke_data)
```

```
## 'data.frame':    5109 obs. of  11 variables:
##  $ gender           : num  0 1 0 1 1 0 0 1 1 1 ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : num  1 1 1 1 2 1 2 1 1 1 ...
##  $ heart_disease    : num  2 1 2 1 1 1 2 1 1 1 ...
##  $ ever_married     : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ work_type        : num  0 4 0 0 4 0 0 0 0 0 ...
##  $ Residence_type   : num  0 1 1 0 1 0 1 0 1 0 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ smoking_status   : num  0 1 1 2 1 0 1 1 1 1 ...
##  $ stroke           : num  2 2 2 2 2 2 2 2 2 2 ...
```

```
# check the correlation between the quantitative variables
cor <- round(cor(stroke_data),2)
corr_melted <- melt(cor)

# A heatmap that shows the correlation between the numerical features and the target feature
ggplot(corr_melted, aes(x=Var1, y=Var2, fill=value)) + geom_tile() +  scale_fill_gradient2(low = "blue", high =
"red", mid = "white", midpoint = 0, limit = c(-1,1), space = "Lab", name="Pearson Correlation") + geom_text(aes(V
ar2, Var1, label = value), color = "black", size = 4) + theme(axis.text.x = element_text(angle = 45, vjust = 1, s
ize = 12, hjust = 1))+ labs(title = "Correlation Matrix of Numerical Features")
```

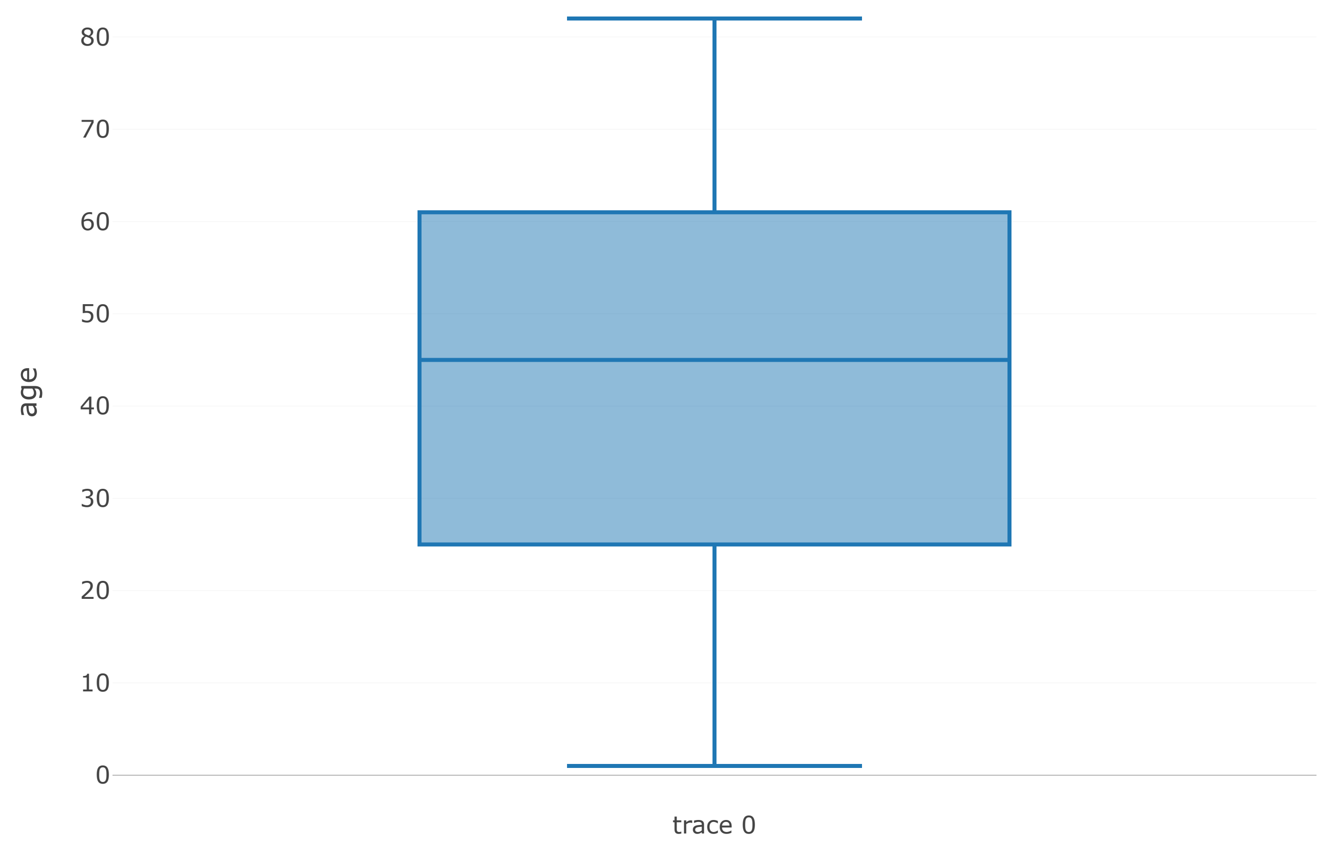


Var2	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
stroke	-0.01	0.25	0.13	0.13	0.11	0.04	-0.02	0.13	0.04	-0.04	1
smoking_status	0.02	-0.11	-0.02	-0.02	-0.05	-0.06	-0.01	-0.03	-0.02	1	-0.04
bmi	0.03	0.32	0.16	0.04	0.33	0	0	0.17	1	-0.02	0.04
avg_glucose_level	-0.05	0.24	0.17	0.16	0.16	0.04	0	1	0.17	-0.03	0.13
Residence_type	-0.01	-0.01	0.01	0	-0.01	-0.02	1	0	0	-0.01	-0.02
work_type	0.01	0.23	0.09	0.06	0.11	1	-0.02	0.04	0	-0.06	0.04
ever_married	0.03	0.68	0.16	0.11	1	0.11	-0.01	0.16	0.33	-0.05	0.11
heart_disease	-0.09	0.26	0.11	1	0.11	0.06	0	0.16	0.04	-0.02	0.13
hypertension	-0.02	0.28	1	0.11	0.16	0.09	0.01	0.17	0.16	-0.02	0.13
age	0.03	1	0.28	0.26	0.68	0.23	-0.01	0.24	0.32	-0.11	0.25
gender	1	0.03	-0.02	-0.09	0.03	0.01	-0.01	-0.05	0.03	0.02	-0.01

These are the couple of observations I had regarding the above output: 1. Most of the features are not highly correlated with each other. This is desirable in a regression analysis because highly correlated features can cause multicollinearity issues, which can affect the accuracy and interpret ability of the regression model. Therefore, having features with low correlation can be beneficial for building a more reliable and accurate regression model. 2. After checking for the correlation between different variables in the data set, it was found that the only correlation worth noticing is the one between the age and ever_married variables. However, this correlation is not surprising or interesting because it is expected that older individuals are more likely to be married than younger individuals. Therefore, the correlation between age and ever_married is not a significant finding and does not provide any new insights or information about the data set. 3. Most important finding is that out of all the features, the age variable has the highest correlation coefficient with the stroke. 4. Additionally, the avg_glucose_level, heart_disease and the hypertension features have the second highest correlation coefficient with the target variable. 5. Marriage also has a positive correlation with the stroke variable.(Well, now we know that marriage == stroke!)

Outliers detection and removal

```
# Now, lets check the individual distributions of the continuous variables

#Age

# check the boxplot distributions of the age
plot_ly(stroke, y = ~age, type = "box")
```



```
paste0("Well, the boxplot works on the Interquartile range and it showed that there are no outliers in the age co
lumn.")
```

```
## [1] "Well, the boxplot works on the Interquartile range and it showed that there are no outliers in the age co
lumn."
```

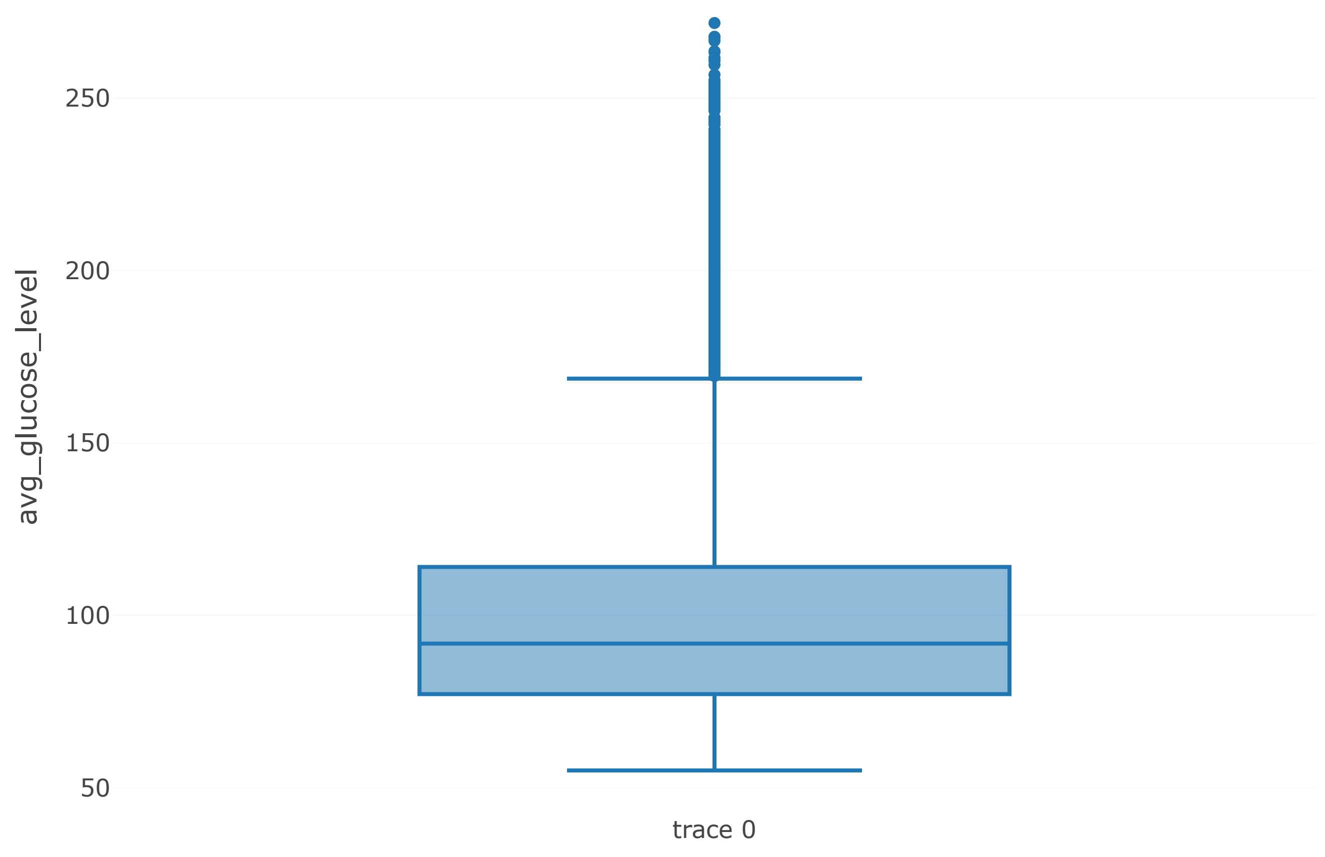
```
# Even though the boxplot showed there are no outliers in the age column, lets also perform the z-score standardi
sation to double check about the outliers
age.scale <- scale(stroke$age)
age.out <- apply(age.scale, 1, function(x) any(x>3))
sum(age.out)
```

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

```
paste0("Even using the z-score standardization method has proved that there are no outliers in the age column.")
```

```
## [1] "Even using the z-score standardization method has proved that there are no outliers in the age column."
```

```
## Average glucose level

# Check the boxplot distributions of the avg_glucose_level
plot_ly(stroke, y = ~avg_glucose_level, type = "box")
```



```
paste0("we observed that there are some outliers present in the avg_glucose_level column.")
```

```
## [1] "we observed that there are some outliers present in the avg_glucose_level column."
```

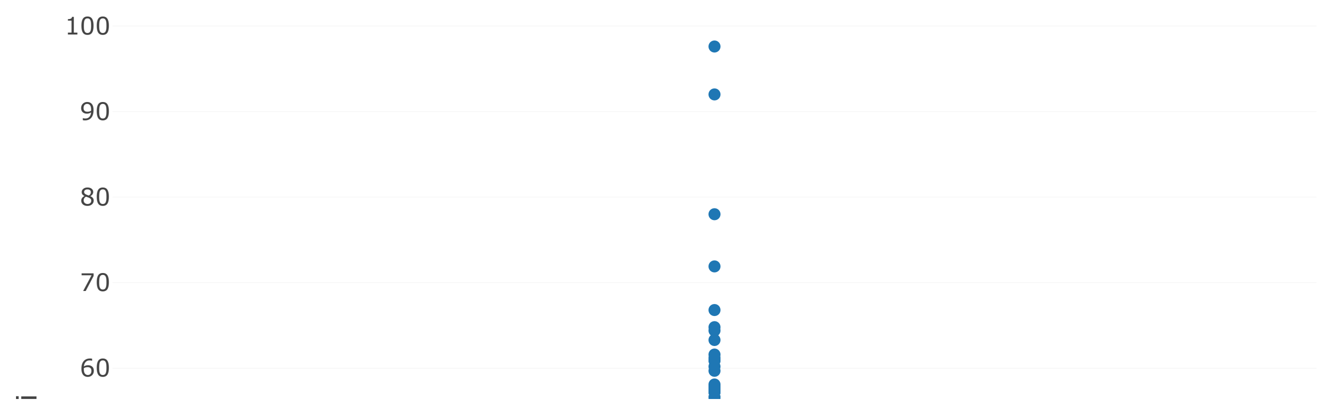
```
# Perform the z-score standardisation and find the outliers
avg.scale <- scale(stroke$avg_glucose_level) #scale the data
avg.out <- apply(avg.scale, 1, function(x) any(x>3)) # find outliers
sum(avg.out) # total number of outliers present in the column
```

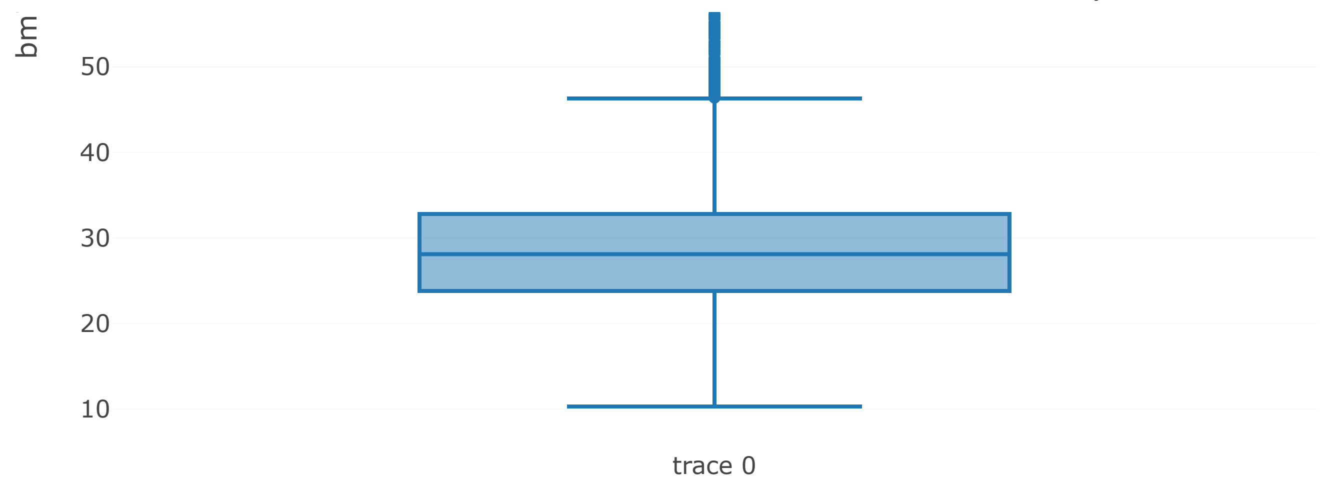
```
## [1] 49
```

```
head(stroke[avg.out, 8], 10) # heads first 10 values of the outlier data points found in the column.
```

```
##  [1] 252.72 243.58 259.63 249.31 263.32 271.74 242.52 250.89 247.51 243.53
```

```
## BMI - Body Mass Index

# Check the boxplot distributions of the BMI
plot_ly(stroke, y = ~bmi, type = "box")
```





```
paste0("we observed that there are some outliers present in the bmi column.")
```

```
## [1] "we observed that there are some outliers present in the bmi column."
```

```
# Perform the z-score standardisation and find the outliers
bmi.scale <- scale(stroke$bmi) #scale the data
bmi.out <- apply(bmi.scale, 1, function(x) any(x>3)) # find outliers
sum(bmi.out) # total number of outliers present in the column
```

```
## [1] 59
```

```
head(stroke[bmi.out, 9], 10) # heads first 10 values of the outlier data points found in the column.
```

```
##  [1] 56.6 54.6 60.9 54.7 64.8 54.7 60.2 71.9 54.6 55.7
```

```
## Conclusions
sum(avg.out, bmi.out)
```

```
## [1] 108
```

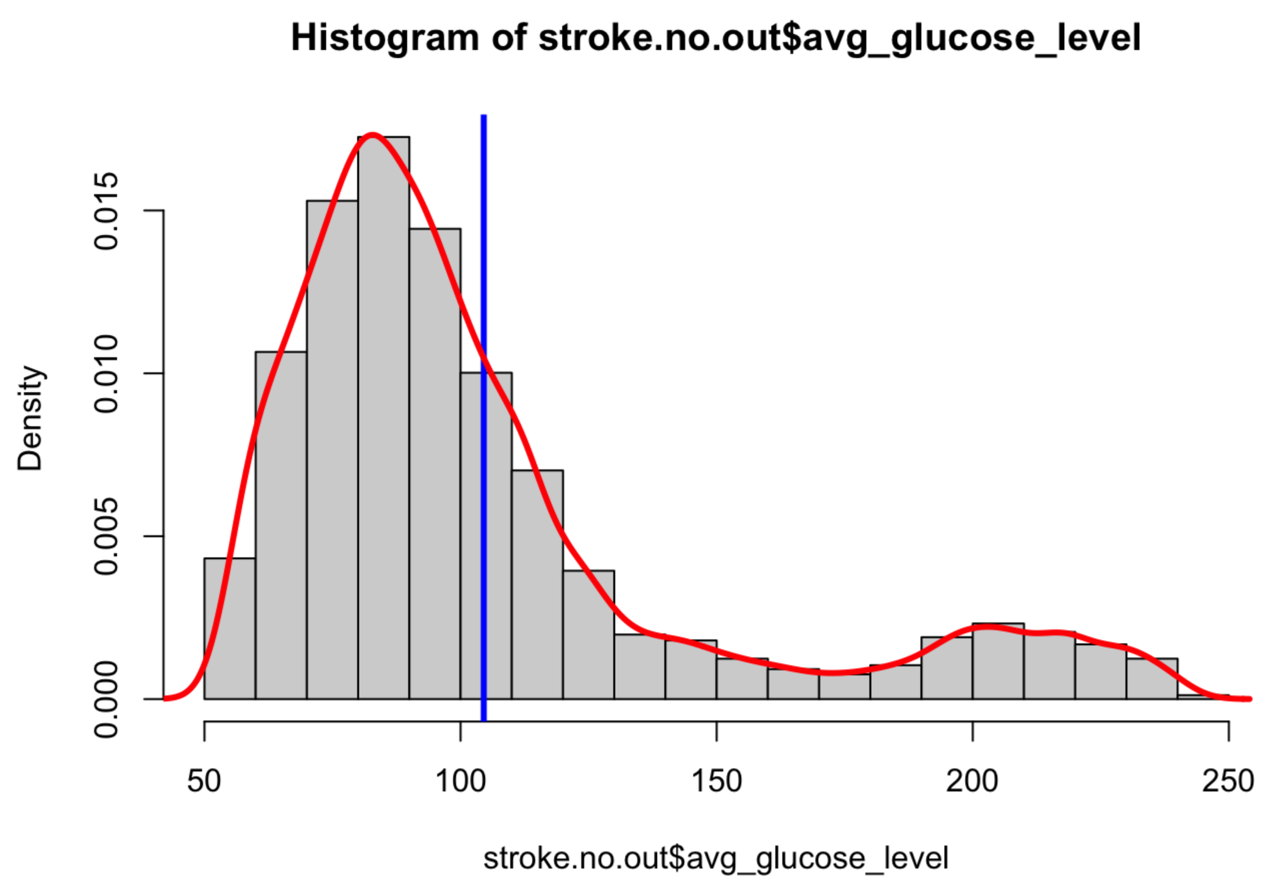
```
# Out of the three continuous variables, the outliers were found only in the avg_glucose_level and bmi columns wi
th a total of 108 observations.

# Assemble the outliers together to remove them
outliers <- which(avg.out | bmi.out)
stroke.no.out <- stroke[-outliers,]
```

Feature transformation

```
## Check the distributions of the continuous variables after the removal of the outliers
summary(stroke.no.out$avg_glucose_level)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   55.12   77.07   91.57  104.53  112.96  240.86
```

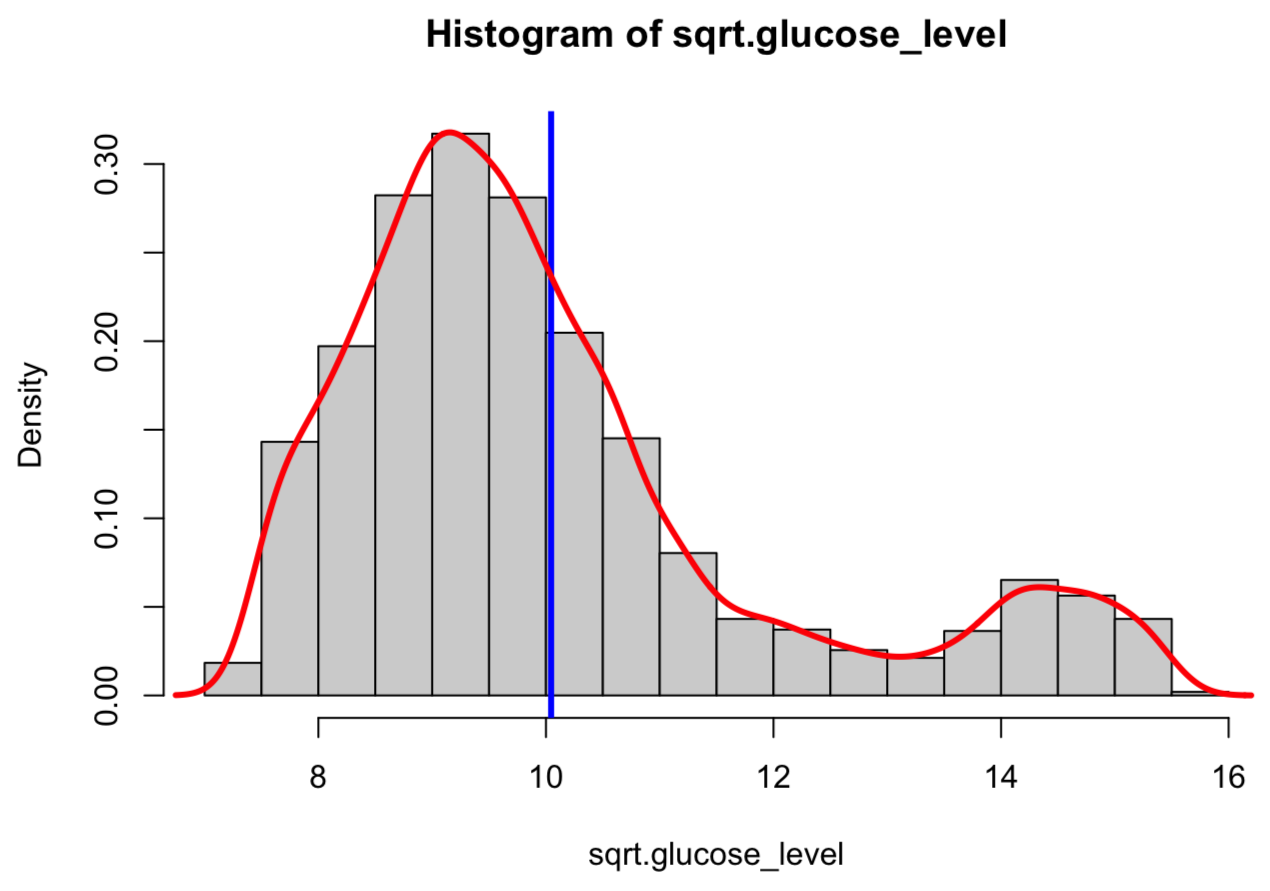
```
# Histogram of the avg_glucose_level after the outliers are removed
hist(stroke.no.out$avg_glucose_level, probability = T)
abline(v= mean(stroke.no.out$avg_glucose_level), col = "blue", lwd =3)
lines(density(stroke.no.out$avg_glucose_level), col = "red", lwd = 3)
```



```
# Perform square root transformation on the avg_glucose_level column
sqrt.glucose_level <- sqrt(stroke.no.out$avg_glucose_level)

# check the summary of the transformed values
summary(sqrt.glucose_level)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   7.424   8.779   9.569  10.045  10.628  15.520
```

```
# visualize the distribution of the transformed avg_glucose_level values
hist(sqrt.glucose_level, probability = T)
abline(v= mean(sqrt.glucose_level), col = "blue", lwd =3)
lines(density(sqrt.glucose_level), col = "red", lwd = 3)
```



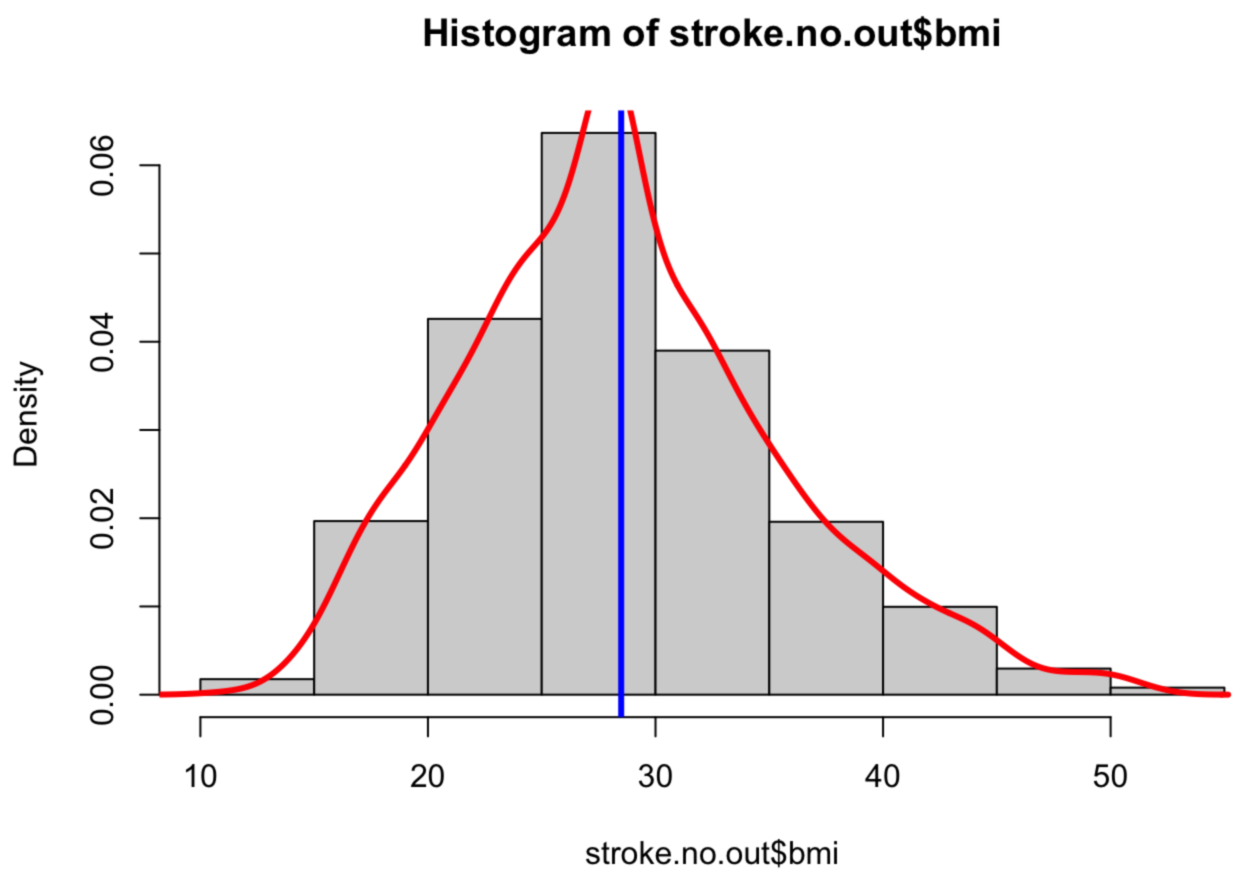
```
paste0("The intial histogram of the avg_glucose_level column has a bimodal normal distribution but the scale is a
bit big so i have performed the data transformation using the square root method because there is a positive corr
elation between the feature and the target variable and after the transformation, the bimodal normal distribution
is more clear and the values are more clear. Therefore, I will proceed with the square root transformation of the
glucose level values for the rest of the project.")
```

```
## [1] "The intial histogram of the avg_glucose_level column has a bimodal normal distribution but the scale is a
bit big so i have performed the data transformation using the square root method because there is a positive corr
elation between the feature and the target variable and after the transformation, the bimodal normal distribution
is more clear and the values are more clear. Therefore, I will proceed with the square root transformation of the
glucose level values for the rest of the project."
```

```
# Data transformation using sqrt method
stroke.no.out$avg_glucose_level <- sqrt(stroke.no.out$avg_glucose_level)

# Now, lets check the summary and histogram of the BMI column
summary(stroke.no.out$bmi)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   10.30   23.60   28.10   28.49   32.50   51.90
```

```
hist(stroke.no.out$bmi, probability = T)
abline(v= mean(stroke.no.out$bmi), col = "blue", lwd =3)
lines(density(stroke.no.out$bmi), col = "red", lwd = 3)
```



```
paste0("From the above histogram, the BMI has a normal distribution hence no transformation is necessary.")
```

```
## [1] "From the above histogram, the BMI has a normal distribution hence no transformation is necessary."
```

Dummy Coding

```
# Transform categorical variables into quantitative ones
stroke.no.out$gender <- ifelse(stroke.no.out$gender == "Female", 1, 0)
stroke.no.out$ever_married <- ifelse(stroke.no.out$ever_married == "Yes", 1, 0)
stroke.no.out$Residence_type <- ifelse(stroke.no.out$Residence_type == "Urban", 1, 0)

# Create dummy variables for a categorical variables to convert them into binary
stroke_dummy <- dummy_columns(stroke.no.out, select_columns = c("work_type", "smoking_status"), remove_first_dumm
y = TRUE, remove_selected_columns = TRUE)

#check the structure of the new stroke data
str(stroke_dummy)
```

```
## 'data.frame':    5001 obs. of  15 variables:
##  $ gender                     : num  0 1 0 1 1 0 0 1 1 1 ...
##  $ age                        : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension               : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 2 1 1 1 ...
##  $ heart_disease              : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 1 1 ...
##  $ ever_married               : num  1 1 1 1 1 1 1 0 1 1 ...
##  $ Residence_type             : num  1 0 0 1 0 1 0 1 0 1 ...
##  $ avg_glucose_level          : num  15.1 14.2 10.3 13.1 13.2 ...
##  $ bmi                        : num  36.6 28.1 32.5 34.4 24 29 27.4 22.8 28.1 24.2 ...
##  $ stroke                     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ work_type_Govt_job         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Never_worked     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type_Private          : int  1 0 1 1 0 1 1 1 1 1 ...
##  $ work_type_Self-employed    : int  0 1 0 0 1 0 0 0 0 0 ...
##  $ smoking_status_never smoked: int  0 1 1 0 1 0 1 1 1 1 ...
##  $ smoking_status_smokes      : int  0 0 0 1 0 0 0 0 0 0 ...
```

```
# Rename these columns so that R wont get confused while predicting some models
names(stroke_dummy)[names(stroke_dummy) == "work_type_Self-employed"] <- "work_type_self_employed"
names(stroke_dummy)[names(stroke_dummy) == "smoking_status_never smoked"] <- "smoking_status_never_smoked"

# Factorize the binary variables
stroke_dummy$gender <- as.factor(stroke_dummy$gender)
stroke_dummy$ever_married <- as.factor(stroke_dummy$ever_married)
stroke_dummy$Residence_type <- as.factor(stroke_dummy$Residence_type)
stroke_dummy$work_type_Govt_job <- as.factor(stroke_dummy$work_type_Govt_job)
stroke_dummy$work_type_Never_worked <- as.factor(stroke_dummy$work_type_Never_worked)
stroke_dummy$work_type_Private <- as.factor(stroke_dummy$work_type_Private)
stroke_dummy$work_type_self_employed <- as.factor(stroke_dummy$work_type_self_employed)
stroke_dummy$smoking_status_never_smoked <- as.factor(stroke_dummy$smoking_status_never_smoked)
stroke_dummy$smoking_status_smokes <- as.factor(stroke_dummy$smoking_status_smokes)

# check first few data of the new stroke data
head(stroke_dummy)
```

	gender <fct>	a... <dbl>	hypertension <fct>	heart_disease <fct>	ever_married <fct>	Residence_type <fct>	avg_glucose_level <dbl>	bmi <dbl>	stroke <fct>
1	0	67	0	1	1	1	15.12250	36.6	1
2	1	61	0	0	1	0	14.22006	28.1	1
3	0	80	0	1	1	0	10.29174	32.5	1
4	1	49	0	0	1	1	13.08549	34.4	1
5	1	79	1	0	1	0	13.19545	24.0	1
6	0	81	0	0	1	1	13.64588	29.0	1

6 rows | 1-10 of 16 columns

```
# Also, check the summary of the stroke dummy data
summary(stroke_dummy)
```

```
##  gender        age     hypertension heart_disease ever_married Residence_type
##  0:2076   Min.   : 1   0:4532       0:4735        0:1741       0:2459
##  1:2925   1st Qu.:25   1: 469       1: 266        1:3260       1:2542
##           Median :44
##           Mean   :43
##           3rd Qu.:61
##           Max.   :82
##  avg_glucose_level      bmi        stroke   work_type_Govt_job
##  Min.   : 7.424    Min.   :10.30   0:4762   0:4360
##  1st Qu.: 8.779    1st Qu.:23.60   1: 239   1: 641
##  Median : 9.569    Median :28.10
##  Mean   :10.045    Mean   :28.49
##  3rd Qu.:10.628    3rd Qu.:32.50
##  Max.   :15.520    Max.   :51.90
##  work_type_Never_worked work_type_Private work_type_self_employed
##  0:4979                 0:2147            0:4204
##  1:  22                 1:2854            1: 797
##
##
##
##
##  smoking_status_never_smoked smoking_status_smokes
##  0:1628                      0:4234
##  1:3373                      1: 767
##
##
##
##
```

From the summary, we can see that for certain features, the distribution is imbalanced with the majority class being 0 and minority class being 1.

Modeling & Evaluation

Model Training

```
# Data split into 70-30
set.seed(123) # set seed for reproducibility

# create an index vector for the split using randomization
split_index <- sample(1:nrow(stroke_dummy), size = round(0.7*nrow(stroke_dummy)), replace = F)

# create training and validation datasets using the index
train <- stroke_dummy[split_index, ]
valid <- stroke_dummy[-split_index, ]

# Check the dimensions of the train and validation datasets
dim(train)
```

```
## [1] 3501   15
```

```
dim(valid)
```

```
## [1] 1500   15
```

```
# Check the structure of the train and validation datasets
str(train)
```

```
## 'data.frame':    3501 obs. of  15 variables:
##  $ gender                     : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ age                        : num  81 27 64 9 21 62 63 23 55 58 ...
##  $ hypertension               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ heart_disease              : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 1 1 1 ...
##  $ ever_married               : Factor w/ 2 levels "0","1": 2 1 2 1 1 2 2 1 2 2 ...
##  $ Residence_type             : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 2 2 2 2 ...
##  $ avg_glucose_level          : num  12.86 8.52 9.1 11.06 8.62 ...
##  $ bmi                        : num  28.1 38.5 29.5 17.7 32.7 26.2 37.8 22.3 36.6 25.9 ...
##  $ stroke                     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ work_type_Govt_job         : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ work_type_Never_worked     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ work_type_Private          : Factor w/ 2 levels "0","1": 2 2 2 1 2 1 1 2 1 2 ...
##  $ work_type_self_employed    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ smoking_status_never_smoked: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 2 2 2 ...
##  $ smoking_status_smokes      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
str(valid)
```

```
## 'data.frame':    1500 obs. of  15 variables:
##  $ gender                     : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 1 1 1 1 ...
##  $ age                        : num  49 81 69 59 50 79 82 58 69 48 ...
##  $ hypertension               : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ heart_disease              : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 2 1 ...
##  $ ever_married               : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 2 2 2 1 ...
##  $ Residence_type             : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 1 1 2 2 ...
##  $ avg_glucose_level          : num  13.09 13.65 9.72 8.73 12.94 ...
##  $ bmi                        : num  34.4 29 22.8 28.1 30.9 26.6 32.5 28.1 28.3 29.7 ...
##  $ stroke                     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ work_type_Govt_job         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ work_type_Never_worked     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ work_type_Private          : Factor w/ 2 levels "0","1": 2 2 2 2 1 1 2 2 1 1 ...
##  $ work_type_self_employed    : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 2 1 ...
##  $ smoking_status_never_smoked: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 2 1 2 ...
##  $ smoking_status_smokes      : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
```

```
# Check the summary of the train and validation datasets
summary(train)
```

```
##  gender        age       hypertension heart_disease ever_married Residence_type
##  0:1456   Min.   : 1.0   0:3178       0:3313        0:1216       0:1732
##  1:2045   1st Qu.:25.0   1: 323       1: 188        1:2285       1:1769
##           Median :45.0
##           Mean   :43.3
##           3rd Qu.:61.0
##           Max.   :82.0
##  avg_glucose_level      bmi        stroke   work_type_Govt_job
##  Min.   : 7.424    Min.   :10.30   0:3338   0:3051
##  1st Qu.: 8.782    1st Qu.:23.80   1: 163   1: 450
##  Median : 9.569    Median :28.10
##  Mean   :10.055    Mean   :28.47
##  3rd Qu.:10.642    3rd Qu.:32.50
##  Max.   :15.518    Max.   :51.90
##  work_type_Never_worked work_type_Private work_type_self_employed
##  0:3489                 0:1514            0:2924
##  1:  12                 1:1987            1: 577
##
##
##
##
##  smoking_status_never_smoked smoking_status_smokes
##  0:1147                      0:2962
##  1:2354                      1: 539
##
##
##
##
```

```
summary(valid)
```

```
##  gender       age        hypertension heart_disease ever_married Residence_type
##  0:620   Min.   : 1.00   0:1354       0:1422        0:525        0:727
##  1:880   1st Qu.:25.00   1: 146       1:  78        1:975        1:773
##          Median :44.00
##          Mean   :42.31
##          3rd Qu.:59.25
##          Max.   :82.00
##  avg_glucose_level      bmi        stroke   work_type_Govt_job
##  Min.   : 7.435    Min.   :13.00   0:1424   0:1309
##  1st Qu.: 8.762    1st Qu.:23.57   1:  76   1: 191
##  Median : 9.570    Median :28.10
##  Mean   :10.023    Mean   :28.55
##  3rd Qu.:10.600    3rd Qu.:32.80
##  Max.   :15.520    Max.   :51.90
##  work_type_Never_worked work_type_Private work_type_self_employed
##  0:1490                 0:633             0:1280
##  1:  10                 1:867             1: 220
##
##
##
##
##  smoking_status_never_smoked smoking_status_smokes
##  0: 481                      0:1272
##  1:1019                      1: 228
##
##
##
##
```

```
# Check the number of rows for train data
nrow(train)
```

```
## [1] 3501
```

```
# check the target variable distribution in the train data
table(train$stroke)
```

```
##
##    0    1
## 3338  163
```

```
# check the number of rows for validation data
nrow(valid)
```

```
## [1] 1500
```

```
# check the target variable distribution in the validation data
table(valid$stroke)
```

```
##
##    0    1
## 1424   76
```

From the table outputs, we can see that the majority class is 0 and minority class is 1 for the target feature values in both the train and validation dataset. This calls for an imbalanced classification. However, lets classify some models and check the results.

Model Construction - Imbalanced dataset

#Model 1 : Logistic regression

```
# Logistic Regression
# We are first creating a model with logistic regression because it is well suited for binary classification task
s.
set.seed(123)
model.reg.imb <- glm(stroke ~.,family=binomial(link='logit'), data=train)
summary(model.reg.imb)
```

```
##
## Call:
## glm(formula = stroke ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -2.073e+01  4.892e+02  -0.042   0.9662
## gender1                      6.241e-02  1.766e-01   0.353   0.7238
## age                          7.667e-02  7.371e-03  10.400   <2e-16 ***
## hypertension1                3.846e-01  2.057e-01   1.870   0.0615 .
## heart_disease1               4.154e-01  2.344e-01   1.772   0.0763 .
## ever_married1                1.036e-02  2.888e-01   0.036   0.9714
## Residence_type1              2.738e-01  1.726e-01   1.586   0.1127
## avg_glucose_level            8.716e-02  3.645e-02   2.391   0.0168 *
## bmi                         -9.027e-03  1.559e-02  -0.579   0.5626
## work_type_Govt_job1          1.270e+01  4.892e+02   0.026   0.9793
## work_type_Never_worked1     -7.419e-01  3.120e+03   0.000   0.9998
## work_type_Private1           1.251e+01  4.892e+02   0.026   0.9796
## work_type_self_employed1     1.224e+01  4.892e+02   0.025   0.9800
## smoking_status_never_smoked1 -2.660e-01  1.958e-01  -1.359   0.1743
## smoking_status_smokes1       4.053e-04  2.674e-01   0.002   0.9988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1318.1  on 3500  degrees of freedom
## Residual deviance: 1023.9  on 3486  degrees of freedom
## AIC: 1053.9
##
## Number of Fisher Scoring iterations: 18
```

```
# Make predictions on validation data
pred.reg.imb <- predict(model.reg.imb, newdata = valid[,-9], type = "response")

# Convert the predicted values into binary for better outcome understanding
pred.bin.imb <- ifelse(pred.reg.imb > 0.5, 1, 0)

# Create a confusionmatrix to check the prediction result
cm.reg.imb <- confusionMatrix(as.factor(pred.bin.imb), valid$stroke, positive = "1")
```

```
## Warning in confusionMatrix.default(as.factor(pred.bin.imb), valid$stroke, :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
dimnames(cm.reg.imb$table) <- list(Prediction = c("Non-stroke", "Stroke"), Actual = c("Non-stroke", "Stroke"))
cm.reg.imb
```

```
## Confusion Matrix and Statistics
##
##             Actual
## Prediction   Non-stroke Stroke
##   Non-stroke       1424     76
##   Stroke              0      0
##
##                Accuracy : 0.9493
##                  95% CI : (0.937, 0.9599)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 0.5305
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.94933
##              Prevalence : 0.05067
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
precision.reg <- (0 / (0 + 0))
recall.reg <- 0 / (0 + 76 )

F1_score.reg <- 2 * (precision.reg * recall.reg) / (precision.reg + recall.reg)
F1_score.reg
```

```
## [1] NaN
```

According to the confusion matrix, the model only predicted the true negatives (1424) and even misclassified the true positives (76) as false negatives. Overall, model is not suitable for predicting the stroke cases. Also, the F1 score is Nan which typically means that there are no true positives in the prediction. I am not considering accuracy here because the model is trained with the majority class being 0 so it is calculating accuracy according to the majority class 0.

According to the summary, the age and avg_glucose_level have a significant positive coefficient.

Model 2: Decision Tree

```
# Build the decision tree model
set.seed(123)
model.tree.imb <- C5.0(stroke ~ ., data = train) # I have used the C5.0 algorithm because it is better in handlin
g imbalanced datasets
model.tree.imb
```

```
##
## Call:
## C5.0.formula(formula = stroke ~ ., data = train)
##
## Classification Tree
## Number of samples: 3501
## Number of predictors: 14
##
## Tree size: 1
##
## Non-standard options: attempt to group attributes
```

```
# Check the summary of the model's outcome
summary(model.tree.imb)
```

```
##
## Call:
## C5.0.formula(formula = stroke ~ ., data = train)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Jan 27 15:11:51 2024
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 3501 cases (15 attributes) from undefined.data
##
## Decision tree:
##  0 (3501/163)
##
##
## Evaluation on training data (3501 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       1  163( 4.7%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    3338          (a): class 0
##     163          (b): class 1
##
##
## Time: 0.0 secs
```

```
# Predict the validation data
pred.tree.imb <- predict(model.tree.imb, newdata = valid[,-9], type = "class")

# Create a confusionmatrix
cm.tree.imb <- confusionMatrix(pred.tree.imb, valid$stroke, positive = "1")
dimnames(cm.tree.imb$table) <- list(Prediction = c("Non-stroke", "Stroke"), Actual = c("Non-stroke", "Stroke"))
cm.tree.imb
```

```
## Confusion Matrix and Statistics
##
##             Actual
## Prediction   Non-stroke Stroke
##   Non-stroke       1424     76
##   Stroke              0      0
##
##                Accuracy : 0.9493
##                  95% CI : (0.937, 0.9599)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 0.5305
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.94933
##              Prevalence : 0.05067
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
# Calculate the F1 scores
precision.tree <- (0 / (0 + 0))
recall.tree <- 0 / (0 + 76 )

F1_score.tree <- 2 * (precision.tree * recall.tree) / (precision.tree + recall.tree)
F1_score.tree
```

```
## [1] NaN
```

According to the confusion matrix, the model only predicted the true negatives and even misclassified the true positives as false negatives. Overall, model is not suitable for predicting the stroke in an individual. Also, the F1 score is Nan which typically means that there are no true positives in the prediction. I am not considering accuracy here because the model is trained with the majority class being 0 so it is calculating accuracy according to the majority class 0.

Model 3: Random Forest

```
# Set seed for reproducibility
set.seed(123)


# Fit random forest model
model.rf.imb <- randomForest(stroke ~ ., data = train, importance = TRUE, proximity = TRUE)

# Print model summary
print(model.rf.imb)
```

```
##
## Call:
##  randomForest(formula = stroke ~ ., data = train, importance = TRUE,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4.66%
## Confusion matrix:
##      0 1 class.error
## 0 3338 0           0
## 1  163 0           1
```

```
# Make predictions on the test data
pred.forest.imb <- predict(model.rf.imb, newdata = valid[,-9])

# Create a confusion matrix
cm.rf.imb <- confusionMatrix(pred.forest.imb, valid$stroke, positive = "1")
dimnames(cm.rf.imb$table) <- list(Prediction = c("Non-stroke", "Stroke"), Actual = c("Non-stroke", "Stroke"))
cm.rf.imb
```

```
## Confusion Matrix and Statistics
##
##             Actual
## Prediction   Non-stroke Stroke
##   Non-stroke       1423     76
##   Stroke              1      0
##
##                Accuracy : 0.9487
##                  95% CI : (0.9363, 0.9593)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 0.5768
##
##                   Kappa : -0.0013
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9992978
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.9492995
##              Prevalence : 0.0506667
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0006667
##       Balanced Accuracy : 0.4996489
##
##        'Positive' Class : 1
##
```

```
# Calculate the F1 scores
precision.rf <- (0 / (0 + 1))
recall.rf <- 0 / (0 + 76 )

F1_score.rf <- 2 * (precision.rf * recall.rf) / (precision.rf + recall.rf)
F1_score.rf
```

```
## [1] NaN
```

According to the confusion matrix, the model predicted 1423 true negatives and even misclassified the true positives as false negatives but also it predicted 1 non-stroke case as a stroke case. Overall, model is not suitable for predicting the stroke in an individual. I am not considering accuracy here because the model is trained with the majority class being 0 so it is calculating accuracy according to the majority class 0. Also, the F1 score is Nan which typically means that there are no true positives in the prediction.

Model 4: Artificial Neural Networks

```
# Train the model using the nnet package
set.seed(123)
model.ann.imb <- nnet(stroke~., data = train, size = 13, decay = 0.1, maxit = 100)
```

```
## # weights:  209
## initial  value 1869.973916
## iter  10 value 578.216439
## iter  20 value 549.613206
## iter  30 value 538.551679
## iter  40 value 520.121861
## iter  50 value 505.005565
## iter  60 value 500.293532
## iter  70 value 499.285014
## iter  80 value 495.693172
## iter  90 value 492.009770
## iter 100 value 486.799835
## final  value 486.799835
## stopped after 100 iterations
```

```
# Make predictions on the test data
pred.ann.imb <- predict(model.ann.imb, newdata = valid[,-9], type = "class")

# create a confusion matrix
cm.ann.imb <- confusionMatrix(as.factor(pred.ann.imb), valid$stroke, positive = "1")
dimnames(cm.ann.imb$table) <- list(Prediction = c("Non-stroke", "Stroke"), Actual = c("Non-stroke", "Stroke"))
cm.ann.imb
```

```
## Confusion Matrix and Statistics
##
##             Actual
## Prediction   Non-stroke Stroke
##   Non-stroke       1421     74
##   Stroke              3      2
##
##                Accuracy : 0.9487
##                  95% CI : (0.9363, 0.9593)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 0.5768
##
##                   Kappa : 0.0434
##
##  Mcnemar's Test P-Value : 1.496e-15
##
##             Sensitivity : 0.026316
##             Specificity : 0.997893
##          Pos Pred Value : 0.400000
##          Neg Pred Value : 0.950502
##              Prevalence : 0.050667
##          Detection Rate : 0.001333
##    Detection Prevalence : 0.003333
##       Balanced Accuracy : 0.512105
##
##        'Positive' Class : 1
##
```

```
# Calculate the f1 scores
precision.ann <- (2 / (2 + 3))
recall.ann <- 2 / (2 + 74 )

F1_score.ann <- 2 * (precision.ann * recall.ann) / (precision.ann + recall.ann)
F1_score.ann
```

```
## [1] 0.04938272
```

According to the confusion matrix, the model predicts 2 stroke cases properly and misclassifies 74 stroke cases as non-stroke cases. On the other hand, it properly predicts the 1421 non-stroke cases and misclassifies 3 non-stroke cases as stroke cases. This model is comparatively better than the previous three models because it at least predicts 2 cases of getting stroke correctly. Additionally, The kappa value is 0.04 which indicates a poor agreement between the actual and the predicted values. The F1 score is 0.04 which is a low score means the precision and recall of the model are both low. It indicates that the model is not performing well in predicting the positive class. In other words, the model is not correctly identifying the minority class in the imbalanced dataset.

In conclusion, out of the 4 models, only the ANN model performed somewhat better in predicting the true positive class. The major issue here is that the dataset is imbalanced with majority of values in the true negative class and minority of values in the true positive class. Due to this, when the model is being trained, it is considering the majority class and even it is predicting the minority class values into the majority class because of the extremely high number of values in the majority class. This is the reason why we got Nan values for the F1 score of those 3 models. Hence, to solve this problem, we'll peform SMOTE to balance the imbalanced data and train the model using balanced data to properly predict the test dataset.

#SMOTE (Synthetic Minority Over-sampling Technique) - To Handle imbalanced data

```
set.seed(123)
train.smote <- SMOTE(train$stroke~., train, perc.over = 200, perc.under = 100) ## perc.over is a parameter for th
e oversampling of minority cases and perc.under is a parameter for the undersampling of majority cases

#summary of the train.smote data
summary(train.smote)
```

```
##  gender       age        hypertension heart_disease ever_married Residence_type
##  0:292   Min.   : 1.00   0:698        0:696         0:163        0:351
##  1:523   1st Qu.:48.00   1:117        1:119         1:652        1:464
##          Median :59.39
##          Mean   :55.95
##           3rd Qu.:71.33
##          Max.   :82.00
##  avg_glucose_level      bmi        stroke  work_type_Govt_job
##  Min.   : 7.491    Min.   :14.20   0:326   0:719
##  1st Qu.: 9.095    1st Qu.:26.25   1:489   1: 96
##  Median :10.128    Median :29.24
##  Mean   :10.812    Mean   :29.84
##  3rd Qu.:12.497    3rd Qu.:32.56
##  Max.   :15.511    Max.   :51.00
##  work_type_Never_worked work_type_Private work_type_self_employed
##  0:813                  0:303             0:643
##  1:  2                  1:512             1:172
##
##
##
##
##  smoking_status_never_smoked smoking_status_smokes
##  0:321                       0:687
##  1:494                       1:128
##
##
##
##
```

```
paste0("Check the distribution of the factored features!")
```

```
## [1] "Check the distribution of the factored features!"
```

```
# Compare your results with the intial train data
summary(train)
```

```
##  gender        age       hypertension heart_disease ever_married Residence_type
##  0:1456   Min.   : 1.0   0:3178       0:3313        0:1216       0:1732
##  1:2045   1st Qu.:25.0   1: 323       1: 188        1:2285       1:1769
##           Median :45.0
##           Mean   :43.3
##           3rd Qu.:61.0
##           Max.   :82.0
##  avg_glucose_level      bmi        stroke   work_type_Govt_job
##  Min.   : 7.424    Min.   :10.30   0:3338   0:3051
##  1st Qu.: 8.782    1st Qu.:23.80   1: 163   1: 450
##  Median : 9.569    Median :28.10
##  Mean   :10.055    Mean   :28.47
##  3rd Qu.:10.642    3rd Qu.:32.50
##  Max.   :15.518    Max.   :51.90
##  work_type_Never_worked work_type_Private work_type_self_employed
##  0:3489                 0:1514            0:2924
##  1:  12                 1:1987            1: 577
##
##
##
##
##  smoking_status_never_smoked smoking_status_smokes
##  0:1147                      0:2962
##  1:2354                      1: 539
##
##
##
##
```

```
paste0("The results are having a drastic change for factored features in comparison to the continuous features.")
```

```
## [1] "The results are having a drastic change for factored features in comparison to the continuous features."
```

```
# after smote, this is the distribution of the target variable values
table(train.smote$stroke)
```

```
##
##   0   1
## 326 489
```

```
paste0("Now, this looks like a balanced data that I can work with.")
```

```
## [1] "Now, this looks like a balanced data that I can work with."
```

Model Construction - After SMOTE

#Model 1 : Logistic regression

```
# Logistic Regression
# We are first creating a model with logistic regression because it is well suited for binary classification task
s.
set.seed(123)
model.reg <- glm(stroke ~.,family=binomial(link='logit'), data=train.smote)

# check the summary of the regression model
summary(model.reg)
```

```
##
## Call:
## glm(formula = stroke ~ ., family = binomial(link = "logit"),
##     data = train.smote)
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -7.686820   0.972019  -7.908 2.61e-15 ***
## gender1                       0.157071   0.201241   0.781 0.435091
## age                           0.090231   0.008707  10.363  < 2e-16 ***
## hypertension1                 0.090083   0.273969   0.329 0.742301
## heart_disease1                1.350544   0.359436   3.757 0.000172 ***
## ever_married1                 0.902761   0.321396   2.809 0.004972 **
## Residence_type1               0.559811   0.195835   2.859 0.004255 **
## avg_glucose_level             0.032286   0.048980   0.659 0.509785
## bmi                           0.034238   0.018083   1.893 0.058304 .
## work_type_Govt_job1          -0.003796   0.370514  -0.010 0.991825
## work_type_Never_worked1      -9.738926 619.160585  -0.016 0.987450
## work_type_Private1            0.418170   0.319962   1.307 0.191235
## work_type_self_employed1      0.193720   0.343137   0.565 0.572376
## smoking_status_never_smoked1 -0.256368   0.220022  -1.165 0.243940
## smoking_status_smokes1        0.316923   0.275871   1.149 0.250635
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1097.01  on 814  degrees of freedom
## Residual deviance:  683.07  on 800  degrees of freedom
## AIC: 713.07
##
## Number of Fisher Scoring iterations: 13
```

From the above summary, I have observed the following:

1.The intercept term has a large negative coefficient value (-7.686820), which indicates that in the absence of any other factors, the model predicts a low probability of stroke.

2. Age is a significant predictor of stroke with a positive coefficient value (0.090231) and a very small p-value (< 0.001).
3. Heart disease is also a significant predictor with a large positive coefficient value (1.350544) and a very small p-value (< 0.001).
4. Ever married and residence type are significant predictors with positive coefficient values (0.902761 and 0.559811 respectively) and small p-values (< 0.01).
5. BMI has a slightly positive coefficient value (0.034238) and a p-value just above the significance level (0.058304).
6. Work type, smoking status, and gender are not significant predictors, as their p-values are above the significance level (0.05).
7. The model's goodness-of-fit is decent, as indicated by the residual deviance being much smaller than the null deviance.
8. The model's AIC is relatively low (713.07), indicating that it is a good fit for the data.

Overall, this logistic regression model provides insights into the predictors that have the greatest impact on the likelihood of a stroke occurring.

model prediction & evaluation

```
# Make predictions on validation data
pred.reg <- predict(model.reg, newdata = valid[,-9], type = "response")

# Convert the predicted values into binary for better outcome understanding
pred.bin <- ifelse(pred.reg > 0.5, 1, 0)

# Create a confusionmatrix to check the prediction result
confusionMatrix(as.factor(pred.bin), valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 953  16
##          1 471  60
##
##                Accuracy : 0.6753
##                  95% CI : (0.651, 0.699)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1197
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.78947
##             Specificity : 0.66924
##          Pos Pred Value : 0.11299
##          Neg Pred Value : 0.98349
##              Prevalence : 0.05067
##          Detection Rate : 0.04000
##    Detection Prevalence : 0.35400
##       Balanced Accuracy : 0.72936
##
##        'Positive' Class : 1
##
```

From the confusion matrix output, I have observed the following:

1. The model has predicted 60 stroke cases correctly out of 76 and misclassified 11 stroke cases as non-stroke cases. However, it also misclassified 471 non-stroke cases as stroke cases which is again a big deal. Apart from this, it correctly classified 953 non-stroke cases.
2. The accuracy of the model is 0.6753, which means the model is correctly predicting 67.53% of the cases.
3. The sensitivity of the model is 0.78947, which means the model correctly identifies 78.95% of the positive cases.
4. The specificity of the model is 0.66924, which means the model correctly identifies 66.92% of the negative cases.
5. The positive predictive value (PPV) of the model is 0.11299, which means that only 11.3% of the cases predicted as positive are actually positive.
6. The negative predictive value (NPV) of the model is 0.98349, which means that 98.3% of the cases predicted as negative are actually negative.
7. The Kappa statistic measures the agreement between the predicted and actual labels, and its value is 0.1197, which indicates poor agreement.

Model 2: Decision Tree

```
# Build the decision tree model
set.seed(123)
model.tree <- C5.0(stroke ~ ., data = train.smote)
model.tree
```

```
##
## Call:
## C5.0.formula(formula = stroke ~ ., data = train.smote)
##
## Classification Tree
## Number of samples: 815
## Number of predictors: 14
##
## Tree size: 24
##
## Non-standard options: attempt to group attributes
```

```
# Check the summary of the model's outcome
summary(model.tree)
```

```
##
## Call:
## C5.0.formula(formula = stroke ~ ., data = train.smote)
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Jan 27 15:11:58 2024
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 815 cases (15 attributes) from undefined.data
##
## Decision tree:
##
## age <= 49:
## :...age <= 37: 0 (143)
## :   age > 37:
## :   :...work_type_self_employed = 0: 0 (58/11)
## :       work_type_self_employed = 1:
## :       :...avg_glucose_level <= 8.984431: 0 (4)
## :           avg_glucose_level > 8.984431: 1 (7/1)
## age > 49:
## :...heart_disease = 1: 1 (114/8)
##     heart_disease = 0:
##     :...age > 57:
##         :...work_type_Govt_job = 0: 1 (325/59)
##         :   work_type_Govt_job = 1:
##         :   :...work_type_Private = 1: 1 (8)
##         :       work_type_Private = 0:
##         :       :...age > 67.49702: 1 (20/2)
##         :           age <= 67.49702:
##         :           :...avg_glucose_level <= 10.61662: 0 (13/2)
##         :               avg_glucose_level > 10.61662: 1 (6/1)
##         age <= 57:
##         :...bmi <= 27: 0 (13/1)
##             bmi > 27:
##             :...Residence_type = 0:
##                 :...avg_glucose_level > 13.39795: 0 (6)
##                 :   avg_glucose_level <= 13.39795:
##                 :   :...smoking_status_never_smoked = 0:
##                 :       :...bmi <= 32.2: 1 (18/1)
##                 :       :   bmi > 32.2: 0 (4/1)
##                 :       smoking_status_never_smoked = 1:
##                 :       :...smoking_status_smokes = 0: 0 (16/5)
##                 :           smoking_status_smokes = 1: 1 (2)
##                 Residence_type = 1:
##                 :...gender = 1:
##                     :...avg_glucose_level <= 13.61294: 1 (25/3)
##                     :   avg_glucose_level > 13.61294:
##                     :   :...avg_glucose_level <= 14.80439: 0 (3)
##                     :       avg_glucose_level > 14.80439: 1 (2)
##                     gender = 0:
##                     :...smoking_status_smokes = 1: 0 (5/1)
##                         smoking_status_smokes = 0:
##                         :...bmi <= 28.91209: 1 (9)
##                             bmi > 28.91209:
##                             :...work_type_Private = 0: 0 (3/1)
##                                 work_type_Private = 1:
##                                 :...age <= 55.94515: 1 (6)
##                                     age > 55.94515: 0 (5)
##
##
## Evaluation on training data (815 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      24   97(11.9%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     251    75    (a): class 0
##      22   467    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% age
##   73.99% heart_disease
##   45.64% work_type_Govt_job
##   14.36% bmi
##   13.01% avg_glucose_level
##   12.76% Residence_type
##    8.47% work_type_self_employed
##    7.48% work_type_Private
##    7.12% gender
##    5.64% smoking_status_smokes
##    4.91% smoking_status_never_smoked
##
##
## Time: 0.0 secs
```

From the above summary, I have observed the following:

1. The decision tree model has a size of 24 nodes and an error rate of 11.9% on the training data.
2. The attribute “age” is used 100% of the time in the tree, making it the most important predictor for stroke. The attribute “heart_disease” is used 73.99% of the time, followed by “work_type_Govt_job” at 45.64%, indicating their importance in predicting stroke.
3. The attribute “bmi” is only used 14.36% of the time, suggesting that it may not be as important as other predictors in this model.

#model prediction & evaluation

```r
# Predict the validation data
pred.tree <- predict(model.tree, newdata = valid[,-9], type = "class")

# Create a confusionmatrix
confusionMatrix(pred.tree, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1002   18
##          1  422   58
##
##                Accuracy : 0.7067
##                  95% CI : (0.6829, 0.7296)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1328
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.76316
##             Specificity : 0.70365
##          Pos Pred Value : 0.12083
##          Neg Pred Value : 0.98235
##              Prevalence : 0.05067
##          Detection Rate : 0.03867
##    Detection Prevalence : 0.32000
##       Balanced Accuracy : 0.73340
##
##        'Positive' Class : 1
##
```

From the above confusion matrix, I have observed the following:

1. The model has correctly classified 58 true positives but it misclassified 18 stroke cases as non-stroke cases. However, it also predicted 422 non-stroke cases as stroke cases and correctly predicted 1002 non stroke cases.
2. The overall accuracy of the model is 0.7067 or 70.67%, which means that the model correctly classified 70.67% of the instances.
3. The sensitivity of the model is 0.76316, which means that the model correctly classified 76.32% of the positive instances.
4. The specificity of the model is 0.70365, which means that the model correctly classified 70.37% of the negative instances.
5. The positive predictive value (PPV) of the model is 0.12083, which means that out of all the instances that the model classified as positive (1), only 12.08% of them were actually positive (1).
6. The negative predictive value (NPV) of the model is 0.98235, which means that out of all the instances that the model classified as negative (0), 98.24% of them were actually negative (0).
7. The Kappa statistic measures the agreement between the predicted and actual class labels, and its value is 0.1328 for this model. A Kappa value of 1 indicates perfect agreement, while this value is low.

Model 3: Random Forest

```
# Set seed for reproducibility
set.seed(123)


# Fit random forest model
model.rf <- randomForest(stroke ~ ., data = train.smote, importance = TRUE, proximity = TRUE)

# Print model summary
print(model.rf)
```

```
##
## Call:
##  randomForest(formula = stroke ~ ., data = train.smote, importance = TRUE,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 15.95%
## Confusion matrix:
##     0   1 class.error
## 0 225 101   0.3098160
## 1  29 460   0.0593047
```

From the above summary output, I have observed the following:

1. The number of trees grown in the forest is 500, which can provide high accuracy in the predictions.
2. At each split in the tree, only 3 variables were tried, which can make the model run faster.
3. The out-of-bag (OOB) estimate of the error rate is 15.95%, which is a measure of how well the model will perform on new, unseen data.
4. The confusion matrix shows the number of true positives (225), true negatives (460), false positives (101), and false negatives (29).
5. The class error for predicting 0 is 30.98% and for predicting 1 it is 5.93%, which means the model is better at predicting 1 than 0.

model prediction & evaluation

```
# Make predictions on the test data
pred.forest <- predict(model.rf, newdata = valid[,-9])

# Create a confusion matrix
confusionMatrix(pred.forest, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 963  17
##          1 461  59
##
##                Accuracy : 0.6813
##                  95% CI : (0.6571, 0.7049)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1202
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77632
##             Specificity : 0.67626
##          Pos Pred Value : 0.11346
##          Neg Pred Value : 0.98265
##              Prevalence : 0.05067
##          Detection Rate : 0.03933
##    Detection Prevalence : 0.34667
##       Balanced Accuracy : 0.72629
##
##        'Positive' Class : 1
##
```

From the above confusion matrix, I have observed the following:

1. The model correctly classified 59 stroke cases and incorrectly classified 17 stroke cases as non-stroke cases. Additionally, it correctly classified 963 non-stroke cases and incorrectly classified 461 non-stroke cases as stroke cases.
2. The overall accuracy of the model is 0.6813 or 68.13%.
3. The sensitivity of the model is 0.77632 or 77.63%, which means that the model correctly identified 77.63% of all positive cases.
4. The specificity of the model is 0.67626 or 67.63%, which means that the model correctly identified 67.63% of all negative cases.
5. The positive predictive value (PPV) of the model is 0.11346 or 11.35%, which means that when the model predicts a positive case, it is correct only 11.35% of the time.

6. The negative predictive value (NPV) of the model is 0.98265 or 98.27%, which means that when the model predicts a negative case, it is correct 98.27% of the time.
7. The Kappa statistic is 0.1202 or 12.02%, which indicates slight agreement between the predicted and actual values beyond chance.

Model 4: Artificial Neural Networks

```
# Train the model using the nnet package
set.seed(123)
model.ann <- nnet(stroke~., data = train.smote, size = 2, decay = 0.1, maxit = 100) # since size = 1 is the defau
lt value, I have taken the next value.
```

```
## # weights:  33
## initial  value 639.499703
## iter  10 value 428.719157
## iter  20 value 379.295575
## iter  30 value 360.523951
## iter  40 value 348.142805
## iter  50 value 332.709938
## iter  60 value 328.837612
## iter  70 value 327.611535
## iter  80 value 326.656614
## iter  90 value 325.143356
## iter 100 value 325.100020
## final  value 325.100020
## stopped after 100 iterations
```

```
# Make predictions on the test data
pred.ann <- predict(model.ann, newdata = valid[,-9], type = "class")

# create a confusion matrix
confusionMatrix(as.factor(pred.ann), valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 890  12
##          1 534  64
##
##                Accuracy : 0.636
##                  95% CI : (0.6111, 0.6604)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1099
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.84211
##             Specificity : 0.62500
##          Pos Pred Value : 0.10702
##          Neg Pred Value : 0.98670
##              Prevalence : 0.05067
##          Detection Rate : 0.04267
##    Detection Prevalence : 0.39867
##       Balanced Accuracy : 0.73355
##
##        'Positive' Class : 1
##
```

From the above confusion matrix, I have observed the following:

1. The model has 64 true positives, 12 false negatives, 534 false positives and 890 true negatives.
2. The overall accuracy of the model is 0.636, which means that 63.6% of cases were correctly classified. The sensitivity of the model, which is the proportion of true positives out of all actual positives, is 0.842, and the specificity, which is the proportion of true negatives out of all actual negatives, is 0.625.
3. The Kappa statistic is 0.1099, which indicates a slight agreement between the model's predictions and the actual values.

Model Tuning - To improve the model's performance

```
# Logistic Regression

# Stepwise backward elimination
step(model.reg, direction = "backward")
```

```
## Start:  AIC=713.07
## stroke ~ gender + age + hypertension + heart_disease + ever_married +
##     Residence_type + avg_glucose_level + bmi + work_type_Govt_job +
##     work_type_Never_worked + work_type_Private + work_type_self_employed +
##     smoking_status_never_smoked + smoking_status_smokes
##
##                               Df Deviance    AIC
## - work_type_Govt_job           1   683.07 711.07
## - work_type_Never_worked       1   683.10 711.10
## - hypertension                 1   683.18 711.18
## - work_type_self_employed      1   683.39 711.39
## - avg_glucose_level            1   683.51 711.51
## - gender                       1   683.68 711.68
## - smoking_status_smokes        1   684.41 712.41
## - smoking_status_never_smoked  1   684.43 712.43
## - work_type_Private            1   684.79 712.79
## <none>                             683.07 713.07
## - bmi                          1   686.69 714.69
## - ever_married                 1   691.01 719.01
## - Residence_type               1   691.30 719.30
## - heart_disease                1   700.41 728.41
## - age                          1   837.45 865.45
##
## Step:  AIC=711.07
## stroke ~ gender + age + hypertension + heart_disease + ever_married +
##     Residence_type + avg_glucose_level + bmi + work_type_Never_worked +
##     work_type_Private + work_type_self_employed + smoking_status_never_smoked +
##     smoking_status_smokes
##
##                               Df Deviance    AIC
## - work_type_Never_worked       1   683.10 709.10
## - hypertension                 1   683.18 709.18
## - avg_glucose_level            1   683.51 709.51
## - work_type_self_employed      1   683.56 709.56
## - gender                       1   683.68 709.68
## - smoking_status_smokes        1   684.41 710.41
## - smoking_status_never_smoked  1   684.44 710.44
## <none>                             683.07 711.07
## - work_type_Private            1   686.18 712.18
## - bmi                          1   686.71 712.71
## - ever_married                 1   691.01 717.01
## - Residence_type               1   691.31 717.31
## - heart_disease                1   700.42 726.42
## - age                          1   838.59 864.59
##
## Step:  AIC=709.1
## stroke ~ gender + age + hypertension + heart_disease + ever_married +
##     Residence_type + avg_glucose_level + bmi + work_type_Private +
##     work_type_self_employed + smoking_status_never_smoked + smoking_status_smokes
##
##                               Df Deviance    AIC
## - hypertension                 1   683.21 707.21
## - avg_glucose_level            1   683.54 707.54
## - work_type_self_employed      1   683.60 707.60
## - gender                       1   683.71 707.71
## - smoking_status_smokes        1   684.44 708.44
## - smoking_status_never_smoked  1   684.47 708.47
## <none>                             683.10 709.10
## - work_type_Private            1   686.22 710.22
## - bmi                          1   686.76 710.76
## - ever_married                 1   691.06 715.06
## - Residence_type               1   691.34 715.34
## - heart_disease                1   700.45 724.45
## - age                          1   838.97 862.97
##
## Step:  AIC=707.21
## stroke ~ gender + age + heart_disease + ever_married + Residence_type +
##     avg_glucose_level + bmi + work_type_Private + work_type_self_employed +
##     smoking_status_never_smoked + smoking_status_smokes
##
##                               Df Deviance    AIC
## - avg_glucose_level            1   683.63 705.63
## - work_type_self_employed      1   683.73 705.73
## - gender                       1   683.80 705.80
## - smoking_status_never_smoked  1   684.55 706.55
## - smoking_status_smokes        1   684.57 706.57
## <none>                             683.21 707.21
## - work_type_Private            1   686.23 708.23
## - bmi                          1   686.99 708.99
## - ever_married                 1   691.25 713.25
## - Residence_type               1   691.45 713.45
## - heart_disease                1   700.49 722.49
## - age                          1   843.89 865.89
```

```
##
## Step:  AIC=705.63
## stroke ~ gender + age + heart_disease + ever_married + Residence_type +
##     bmi + work_type_Private + work_type_self_employed + smoking_status_never_smoked +
##     smoking_status_smokes
##
##                               Df Deviance    AIC
## - work_type_self_employed      1   684.11 704.11
## - gender                       1   684.24 704.24
## - smoking_status_smokes        1   685.00 705.00
## - smoking_status_never_smoked  1   685.04 705.04
## <none>                             683.63 705.63
## - work_type_Private            1   686.76 706.76
## - bmi                          1   688.53 708.53
## - ever_married                 1   691.81 711.81
## - Residence_type               1   692.03 712.03
## - heart_disease                1   703.82 723.82
## - age                          1   850.09 870.09
##
## Step:  AIC=704.11
## stroke ~ gender + age + heart_disease + ever_married + Residence_type +
##     bmi + work_type_Private + smoking_status_never_smoked + smoking_status_smokes
##
##                               Df Deviance    AIC
## - gender                       1   684.69 702.69
## - smoking_status_never_smoked  1   685.45 703.45
## - smoking_status_smokes        1   685.50 703.50
## <none>                             684.11 704.11
## - work_type_Private            1   686.79 704.79
## - bmi                          1   689.18 707.18
## - Residence_type               1   692.20 710.20
## - ever_married                 1   692.26 710.26
## - heart_disease                1   704.20 722.20
## - age                          1   865.98 883.98
##
## Step:  AIC=702.69
## stroke ~ age + heart_disease + ever_married + Residence_type +
##     bmi + work_type_Private + smoking_status_never_smoked + smoking_status_smokes
##
##                               Df Deviance    AIC
## - smoking_status_never_smoked  1   685.92 701.92
## - smoking_status_smokes        1   686.13 702.13
## <none>                             684.69 702.69
## - work_type_Private            1   687.37 703.37
## - bmi                          1   689.82 705.82
## - Residence_type               1   692.45 708.45
## - ever_married                 1   692.65 708.65
## - heart_disease                1   704.64 720.64
## - age                          1   867.95 883.95
##
## Step:  AIC=701.92
## stroke ~ age + heart_disease + ever_married + Residence_type +
##     bmi + work_type_Private + smoking_status_smokes
##
##                         Df Deviance    AIC
## <none>                       685.92 701.92
## - work_type_Private      1   689.17 703.17
## - smoking_status_smokes  1   689.59 703.59
## - bmi                    1   691.79 705.79
## - Residence_type         1   693.60 707.60
## - ever_married           1   693.96 707.96
## - heart_disease          1   705.21 719.21
## - age                    1   875.58 889.58
```

```
##
## Call:  glm(formula = stroke ~ age + heart_disease + ever_married + Residence_type +
##     bmi + work_type_Private + smoking_status_smokes, family = binomial(link = "logit"),
##     data = train.smote)
##
## Coefficients:
##           (Intercept)                    age          heart_disease1
##              -7.72863                0.09368                 1.36918
##         ever_married1        Residence_type1                     bmi
##               0.90924                0.53405                 0.04095
##    work_type_Private1  smoking_status_smokes1
##               0.36873                0.46480
##
## Degrees of Freedom: 814 Total (i.e. Null);  807 Residual
## Null Deviance:      1097
## Residual Deviance: 685.9     AIC: 701.9
```

```
# New model after elimination of non-significant variables
set.seed(123)
model.reg.1 <- glm(stroke ~ age + heart_disease + ever_married + Residence_type +
    bmi + work_type_Private + smoking_status_smokes, family = binomial(link = "logit"),
    data = train.smote)

# summarize the model
summary(model.reg.1)
```

```
##
## Call:
## glm(formula = stroke ~ age + heart_disease + ever_married + Residence_type +
##     bmi + work_type_Private + smoking_status_smokes, family = binomial(link = "logit"),
##     data = train.smote)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -7.728635   0.843119  -9.167  < 2e-16 ***
## age                     0.093685   0.008429  11.114  < 2e-16 ***
## heart_disease1          1.369175   0.351365   3.897 9.75e-05 ***
## ever_married1           0.909237   0.321308   2.830  0.00466 **
## Residence_type1         0.534055   0.193266   2.763  0.00572 **
## bmi                     0.040955   0.017104   2.395  0.01664 *
## work_type_Private1      0.368726   0.204864   1.800  0.07188 .
## smoking_status_smokes1  0.464802   0.246150   1.888  0.05899 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1097.01  on 814  degrees of freedom
## Residual deviance:  685.92  on 807  degrees of freedom
## AIC: 701.92
##
## Number of Fisher Scoring iterations: 6
```

```
# check whether the initial model without backward elimination is better or not
anova(model.reg.1, model.reg, test = "Chisq")
```

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	807	685.9243	NA	NA	NA
2	800	683.0701	7	2.854275	0.8981477

2 rows

```
# Evaluation of the new regression model after backward stepwise elimination
pred.reg.1 <- predict(model.reg.1, newdata = valid[,-9], type = "response")

# Convert the predicted values into binary for better outcome understanding
pred.bin.1 <- ifelse(pred.reg.1 > 0.5, 1, 0)

# Create a confusionmatrix to check the prediction result
confusionMatrix(as.factor(pred.bin.1), valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 946  15
##          1 478  61
##
##                Accuracy : 0.6713
##                  95% CI : (0.6469, 0.6951)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1202
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.80263
##             Specificity : 0.66433
##          Pos Pred Value : 0.11317
##          Neg Pred Value : 0.98439
##              Prevalence : 0.05067
##          Detection Rate : 0.04067
##    Detection Prevalence : 0.35933
##       Balanced Accuracy : 0.73348
##
##        'Positive' Class : 1
##
```

From the summary output of the new regression model, I have observed the following:

1. The model has an intercept of -7.73, which is the estimated log-odds of stroke occurrence when all predictor variables are zero.
2. Age is a significant predictor of stroke occurrence, with an estimated coefficient of 0.094 and a very small p-value (< 0.001). This suggests that the odds of stroke increase by a factor of exp(0.094) = 1.099 for each one-year increase in age, holding other variables constant.
3. Heart disease is also a significant predictor of stroke occurrence, with an estimated coefficient of 1.369 and a very small p-value (< 0.001). This suggests that individuals with heart disease have exp(1.369) = 3.930 times higher odds of stroke occurrence than those without heart disease, holding other variables constant.
4. Ever married and residence type are also significant predictors of stroke occurrence, with positive coefficients indicating that being married and living in an urban area are associated with higher odds of stroke occurrence.
5. BMI, work type (private), and smoking status (smokes) are not statistically significant predictors of stroke occurrence, based on their p-values (> 0.05).
6. The residual deviance is 685.92 on 807 degrees of freedom, which indicates that the model fits the data reasonably well. The AIC value of 701.92 suggests that this model is a good fit for the data, as it has a low AIC value.

Comparison of the regression model’s before and after the stepwise backward elimination: In this case, Model 2 has a slightly lower residual deviance than Model 1, but the change in deviance is not significant based on the p-value (0.8981). This suggests that the additional predictor variables in Model 2 do not significantly improve the model fit compared to Model 1. Therefore, the model which has been created using the backward elimination elimination has a better model fit.

However, according to the confusion matrix results of the new regression model:

1. The model has 61 true positives, 946 true negatives, 478 false positives, 15 false negatives.
2. Compared to the model’s performance without the backward step elimination, there is a slight improvement.
3. The accuracy of the model is 0.6713, which means that the model correctly predicted the outcome for 67.13% of the test set.
4. The kappa statistic is 0.1202, indicating slight agreement between the predicted and true classes.
5. The sensitivity of the model is 0.80263, which means that the model correctly predicted the positive class for 80.26% of the test set.
6. The specificity of the model is 0.66433, which means that the model correctly predicted the negative class for 66.43% of the test set

In conclusion, the ANN model is better suited for predicting the stroke cases in comparison to other three models because it correctly predicted 64 stroke cases out of the 76. The second model which is a better suited is the logistic regression with backward elimination step performed because it correctly predicted 61 stroke cases out of the 76. The remaining models random forest and decision tree performance could be improved by hyperparameter tuning and we can check the outputs again.

Evaluation with K-fold cross validation

```
# Logistic regression
set.seed(123)
reg.fit <- train(stroke~age + heart_disease + ever_married + Residence_type +
    bmi + work_type_Private + smoking_status_smokes, data = train.smote, method = "glmnet", trControl = trainCont
rol(method = "cv", number = 5), family = "binomial")

reg.fit
```

```
## glmnet
##
## 815 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 652, 652, 651, 653, 652
## Resampling results across tuning parameters:
##
##   alpha  lambda        Accuracy   Kappa
##   0.10   0.0005955545  0.8086463  0.5856473
##   0.10   0.0059555453  0.8209163  0.6103465
##   0.10   0.0595554531  0.8172278  0.5949669
##   0.55   0.0005955545  0.8086463  0.5856473
##   0.55   0.0059555453  0.8196894  0.6074346
##   0.55   0.0595554531  0.8135543  0.5864601
##   1.00   0.0005955545  0.8098733  0.5884853
##   1.00   0.0059555453  0.8147889  0.5973101
##   1.00   0.0595554531  0.8172278  0.5949138
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.005955545.
```

```
# Evaluation of the new regression model after backward stepwise elimination
pred.reg.fit <- predict(reg.fit, newdata = valid[,-9], type = "raw")

# Create a confusionmatrix to check the prediction result
confusionMatrix(pred.reg.fit, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 935  15
##          1 489  61
##
##                Accuracy : 0.664
##                  95% CI : (0.6395, 0.6879)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1162
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.80263
##             Specificity : 0.65660
##          Pos Pred Value : 0.11091
##          Neg Pred Value : 0.98421
##              Prevalence : 0.05067
##          Detection Rate : 0.04067
##    Detection Prevalence : 0.36667
##       Balanced Accuracy : 0.72962
##
##        'Positive' Class : 1
##
```

From the summary above, I have observed the following:

1. The output shows the results of a cross-validated glmnet model with 5 folds, which was used to classify two classes ('0' and '1') based on 7 predictor variables. The model was trained on 815 samples without any pre-processing.
2. The accuracy and kappa coefficient (a measure of agreement between predicted and actual classes) for each combination of alpha (the elastic net mixing parameter) and lambda (the regularization parameter). The optimal model was selected based on the highest accuracy, which was achieved with an alpha of 0.1 and a lambda of 0.005955545.

Overall, the model achieved relatively high accuracy (around 81%) and moderate agreement (kappa coefficients around 0.6) for the optimal model.

From the confusion matrix, I have observed the following:

1.The overall accuracy of the model is 0.664, which means that 66.4% of the predictions were correct. The kappa value of 0.1162 indicates only slight agreement between the model's predictions and the actual classes. The sensitivity of the model, which measures the proportion of actual positive cases that were correctly identified as positive, is 0.80263. The specificity, which measures the proportion of actual negative cases that were correctly identified as negative, is 0.65660.

2. The model has 61 true positives, 935 true negatives, 489 false positives, 15 false negatives.

```
# C5.0 Decision trees
set.seed(123)
tree.fit <- train(stroke ~., data = train.smote, tuneGrid = expand.grid( .winnow = c(TRUE,FALSE), .trials= 1:6, .
model="tree"), trControl = trainControl(method = "cv", number = 5), method = "C5.0", metric = "Kappa", verbose =
F)

tree.fit
```

```
## C5.0
##
## 815 samples
##  14 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 652, 652, 651, 653, 652
## Resampling results across tuning parameters:
##
##   winnow  trials  Accuracy   Kappa
##   FALSE   1       0.8074643  0.5873789
##   FALSE   2       0.8099033  0.5912190
##   FALSE   3       0.7877790  0.5437764
##   FALSE   4       0.8245973  0.6244936
##   FALSE   5       0.8196516  0.6160327
##   FALSE   6       0.8295056  0.6349392
##    TRUE   1       0.8123348  0.5919200
##    TRUE   2       0.8098732  0.5866464
##    TRUE   3       0.8000648  0.5694729
##    TRUE   4       0.8294828  0.6285050
##    TRUE   5       0.8184847  0.6103546
##    TRUE   6       0.8184548  0.6047335
##
## Tuning parameter 'model' was held constant at a value of tree
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were trials = 6, model = tree and winnow
##  = FALSE.
```

```
# Predict the validation data
pred.tree.fit <- predict(tree.fit, newdata = valid[,-9], type = "raw")

# Create a confusionmatrix
confusionMatrix(pred.tree.fit, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 970  16
##          1 454  60
##
##                Accuracy : 0.6867
##                  95% CI : (0.6625, 0.7101)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1263
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.78947
##             Specificity : 0.68118
##          Pos Pred Value : 0.11673
##          Neg Pred Value : 0.98377
##              Prevalence : 0.05067
##          Detection Rate : 0.04000
##    Detection Prevalence : 0.34267
##       Balanced Accuracy : 0.73533
##
##        'Positive' Class : 1
##
```

From the summary above, I have observed that:

This output shows the results of a cross-validated C5.0 model on a dataset with 815 samples and 14 predictors. The goal is to classify the samples into two classes, '0' and '1'.

The model was trained with different values of the tuning parameters 'winnow' and 'trials', and the performance was evaluated using accuracy and kappa as metrics. The 'winnow' parameter determines whether to use the winnow algorithm for attribute weighting, and the 'trials' parameter specifies the number of boosting trials to perform.

The best performing model had ‘trials’ set to 6, ‘model’ set to ‘tree’, and ‘winnow’ set to FALSE. It achieved an accuracy of 82.95% and a kappa of 0.6349.

From the confusion matrix, I have observed that: 1. The model has 60 true positives, 970 true negatives, 454 false positives, 16 false negatives.

2. The accuracy of the model is 0.6867, which means that it correctly classified 68.67% of the instances.
3. The kappa statistic is 0.1263, indicating fair agreement between the observed and predicted classifications.
4. The sensitivity of the model is 0.78947, which means that it correctly identified 78.95% of the positive instances. The specificity of the model is 0.68118, which means that it correctly identified 68.12% of the negative instances.

```
# Random forest
set.seed(123)
rf.fit <- train(stroke ~., data = train.smote, method = "rf", trControl = trainControl(method = "cv", number = 5,
search = "grid"), metric= "Kappa", tuneGrid = expand.grid( mtry = c(1:10)), importance = T)

rf.fit
```

```
## Random Forest
##
## 815 samples
##  14 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 652, 652, 651, 653, 652
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.7681924  0.4671402
##    2    0.8380495  0.6443604
##    3    0.8491151  0.6739631
##    4    0.8405260  0.6570530
##    5    0.8442371  0.6660767
##    6    0.8442370  0.6661402
##    7    0.8430025  0.6642802
##    8    0.8491375  0.6772005
##    9    0.8442595  0.6674197
##   10    0.8442520  0.6680442
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```
# Make predictions on the test data
pred.rf.fit <- predict(rf.fit, newdata = valid[,-9])

# Create a confusion matrix
confusionMatrix(pred.rf.fit, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1013   17
##          1  411   59
##
##                Accuracy : 0.7147
##                  95% CI : (0.6911, 0.7374)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1412
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.77632
##             Specificity : 0.71138
##          Pos Pred Value : 0.12553
##          Neg Pred Value : 0.98350
##              Prevalence : 0.05067
##          Detection Rate : 0.03933
##    Detection Prevalence : 0.31333
##       Balanced Accuracy : 0.74385
##
##        'Positive' Class : 1
##
```

From the summary, I have observed that:

The resampling method used was cross-validation with 5 folds, and the summary of sample sizes shows that each fold had a similar number of samples.

The tuning parameter used was “mtry”, which determines the number of variables randomly sampled as candidates at each split. The model was evaluated for mtry values ranging from 1 to 10.

The optimal Random Forest model was selected based on the highest Kappa value, which is a measure of agreement between predicted and actual class labels, and the model with mtry = 8 was chosen.

The accuracy of the Random Forest model for the optimal mtry value was 0.8491, and the Kappa value was 0.6772, which indicates a substantial agreement between predicted and actual class labels.

Overall, the Random Forest model with mtry = 8 seems to have performed better than the previous models we have looked at, based on its higher accuracy and Kappa values.

From the confusion matrix, I have observed that:

1. The accuracy of the model is 0.7147 which indicates that it correctly classified 71.47% of the instances.

2.The Kappa coefficient is 0.1412 which indicates a fair agreement.

3. The sensitivity of the model is 0.77632 which indicates the proportion of actual positive instances that were correctly identified by the model. The specificity is 0.71138 which indicates the proportion of actual negative instances that were correctly identified by the model.
4. The model has 1013 true negatives, 17 false negatives, 411 false positives, 59 true positives.

```
# Artificial Neural Networks
set.seed(123)
ann.fit <- train(stroke ~ ., data = train.smote, method = "nnet", trControl = trainControl(method = "cv", number
= 5), tunelength = 5, metric = "Kappa")
```

```
## # weights:  17
## initial  value 447.388830
## iter  10 value 298.730700
## iter  20 value 279.423959
## iter  30 value 277.211883
## iter  40 value 273.443755
## iter  50 value 272.409924
## iter  60 value 268.406210
## iter  70 value 267.972805
## iter  80 value 267.971580
## iter  90 value 267.970133
## final  value 267.969217
## converged
## # weights:  49
## initial  value 441.705555
## iter  10 value 438.884560
## iter  20 value 361.350372
## iter  30 value 288.458436
## iter  40 value 287.227295
## iter  50 value 284.247813
## iter  60 value 282.280792
## iter  70 value 282.183939
## iter  80 value 282.180766
## iter  80 value 282.180765
## iter  80 value 282.180765
## final  value 282.180765
## converged
## # weights:  81
## initial  value 543.421214
## iter  10 value 327.570808
## iter  20 value 297.734067
## iter  30 value 281.057799
## iter  40 value 269.219882
## iter  50 value 268.270143
## iter  60 value 262.192473
## iter  70 value 261.142940
## iter  80 value 261.018403
## iter  90 value 260.972588
## iter 100 value 260.914918
## final  value 260.914918
## stopped after 100 iterations
## # weights:  17
## initial  value 453.977444
## iter  10 value 316.662269
## iter  20 value 289.576609
## iter  30 value 283.881740
## iter  40 value 277.254458
## iter  50 value 276.878675
## final  value 276.855072
## converged
## # weights:  49
## initial  value 554.954279
## iter  10 value 392.647747
## iter  20 value 308.128015
## iter  30 value 293.476612
## iter  40 value 281.898840
## iter  50 value 280.205504
## iter  60 value 272.027065
## iter  70 value 258.905391
## iter  80 value 254.994281
## iter  90 value 248.002753
## iter 100 value 244.670849
## final  value 244.670849
## stopped after 100 iterations
## # weights:  81
## initial  value 450.697073
## iter  10 value 317.467854
## iter  20 value 300.474394
## iter  30 value 286.218063
## iter  40 value 278.819325
## iter  50 value 264.494803
## iter  60 value 260.604633
## iter  70 value 260.162697
## iter  80 value 260.002188
## iter  90 value 258.644430
## iter 100 value 252.384910
## final  value 252.384910
## stopped after 100 iterations
## # weights:  17
## initial  value 446.051825
## final  value 438.884940
## converged
## # weights:  49
## initial  value 443.505456
## iter  10 value 288.640191
## iter  20 value 276.972072
## iter  30 value 273.713103
## iter  40 value 273.448771
## iter  50 value 272.854231
## iter  60 value 272.588923
## iter  70 value 271.549227
## iter  80 value 271.216454
## iter  90 value 270.960546
## iter 100 value 270.842860
## final  value 270.842860
## stopped after 100 iterations
## # weights:  81
## initial  value 509.416403
## iter  10 value 319.329408
## iter  20 value 275.085310
## iter  30 value 258.094754
## iter  40 value 250.052224
## iter  50 value 247.226484
## iter  60 value 246.172824
## iter  70 value 245.884737
## iter  80 value 245.725297
## iter  90 value 245.423300
## iter 100 value 245.073218
## final  value 245.073218
## stopped after 100 iterations
## # weights:  17
## initial  value 526.119335
## final  value 438.884550
## converged
## # weights:  49
## initial  value 439.237941
## iter  10 value 307.531896
## iter  20 value 288.858111
## iter  30 value 283.638558
## iter  40 value 279.985531
## iter  50 value 279.797160
## final  value 279.796906
## converged
## # weights:  81
## initial  value 479.834229
## iter  10 value 317.264643
## iter  20 value 306.393906
## iter  30 value 303.926921
## iter  40 value 302.139588
## iter  50 value 300.054947
## iter  60 value 296.078765
## iter  70 value 292.845851
## iter  80 value 277.028597
## iter  90 value 275.004841
## iter 100 value 270.396144
## final  value 270.396144
## stopped after 100 iterations
## # weights:  17
## initial  value 473.249928
## iter  10 value 313.039823
## iter  20 value 304.105043
## iter  30 value 297.776389
## iter  40 value 293.221287
## iter  50 value 283.805425
## iter  60 value 276.003667
## final  value 275.942486
## converged
## # weights:  49
## initial  value 422.290957
## iter  10 value 305.021769
## iter  20 value 294.335629
## iter  30 value 270.108626
## iter  40 value 259.434717
## iter  50 value 257.869375
## iter  60 value 256.275059
## iter  70 value 256.233301
## iter  80 value 256.231229
## iter  90 value 256.206894
## iter 100 value 255.968192
## final  value 255.968192
## stopped after 100 iterations
## # weights:  81
## initial  value 473.587506
## iter  10 value 349.505651
## iter  20 value 314.927238
## iter  30 value 288.773076
## iter  40 value 261.498624
## iter  50 value 248.547992
## iter  60 value 235.683396
## iter  70 value 230.326824
## iter  80 value 229.269016
## iter  90 value 228.408176
## iter 100 value 225.066864
## final  value 225.066864
## stopped after 100 iterations
## # weights:  17
## initial  value 439.391865
## iter  10 value 410.807716
## iter  20 value 309.359412
## iter  30 value 277.947665
## iter  40 value 276.575812
## iter  50 value 273.254766
## iter  60 value 270.706584
## iter  70 value 270.094805
## iter  80 value 269.934192
## iter  90 value 269.927939
## iter 100 value 269.854145
## final  value 269.854145
## stopped after 100 iterations
## # weights:  49
## initial  value 574.594664
## iter  10 value 362.321490
## iter  20 value 295.085131
## iter  30 value 290.603809
## iter  40 value 276.943619
## iter  50 value 271.012734
## iter  60 value 270.068345
## iter  70 value 269.916891
## iter  80 value 269.821148
## iter  90 value 269.573271
## iter 100 value 268.900339
## final  value 268.900339
## stopped after 100 iterations
## # weights:  81
## initial  value 435.777349
## iter  10 value 327.179527
## iter  20 value 275.778146
## iter  30 value 266.627363
## iter  40 value 265.363651
## iter  50 value 264.966140
## iter  60 value 264.928703
## iter  70 value 264.909477
## iter  80 value 264.888829
## iter  90 value 264.866541
## iter 100 value 264.859599
## final  value 264.859599
## stopped after 100 iterations
## # weights:  17
## initial  value 533.859627
## final  value 437.967930
## converged
## # weights:  49
## initial  value 437.786443
## iter  10 value 304.012901
## iter  20 value 280.249182
## iter  30 value 260.386998
## iter  40 value 254.532434
## iter  50 value 254.152349
## iter  60 value 254.060133
## iter  70 value 253.972881
## iter  80 value 253.951731
## iter  90 value 253.948013
## iter 100 value 253.947223
## final  value 253.947223
## stopped after 100 iterations
## # weights:  81
## initial  value 403.033616
## iter  10 value 293.177275
## iter  20 value 270.772383
## iter  30 value 259.638768
## iter  40 value 235.204996
## iter  50 value 224.062466
## iter  60 value 215.673530
## iter  70 value 213.220838
## iter  80 value 212.978129
## final  value 212.977252
## converged
## # weights:  17
## initial  value 461.145801
## iter  10 value 305.687373
## iter  20 value 281.580015
## iter  30 value 277.638330
## iter  40 value 272.925146
## final  value 272.845909
## converged
## # weights:  49
## initial  value 435.406442
## iter  10 value 309.818622
## iter  20 value 304.300943
## iter  30 value 292.696084
## iter  40 value 277.850323
## iter  50 value 269.260054
## iter  60 value 263.417832
## iter  70 value 256.055475
## iter  80 value 254.798210
## iter  90 value 254.271071
## iter 100 value 253.496942
## final  value 253.496942
## stopped after 100 iterations
## # weights:  81
## initial  value 455.422433
## iter  10 value 320.884051
## iter  20 value 294.085388
## iter  30 value 268.742944
## iter  40 value 254.578322
## iter  50 value 242.567362
## iter  60 value 237.821460
## iter  70 value 233.935026
## iter  80 value 232.493370
## iter  90 value 232.230628
## iter 100 value 232.173894
## final  value 232.173894
## stopped after 100 iterations
## # weights:  17
## initial  value 443.925001
## final  value 437.969551
## converged
## # weights:  49
## initial  value 634.599687
## iter  10 value 300.113643
## iter  20 value 272.994428
## iter  30 value 264.827913
## iter  40 value 249.182529
## iter  50 value 245.350071
## iter  60 value 244.703369
## iter  70 value 244.484063
## iter  80 value 244.458129
## iter  90 value 244.436706
## iter 100 value 244.397147
## final  value 244.397147
## stopped after 100 iterations
## # weights:  81
## initial  value 437.145339
## iter  10 value 305.692002
## iter  20 value 290.909414
## iter  30 value 254.620942
## iter  40 value 240.533132
## iter  50 value 237.875934
## iter  60 value 237.241787
## iter  70 value 236.667601
## iter  80 value 236.187971
## iter  90 value 235.815687
## iter 100 value 235.543018
## final  value 235.543018
## stopped after 100 iterations
## # weights:  17
## initial  value 435.684238
## iter  10 value 338.795505
## iter  20 value 318.249136
## iter  30 value 306.076127
## iter  40 value 298.675123
## iter  50 value 296.420726
## iter  60 value 295.124908
## iter  70 value 294.184506
## iter  80 value 293.953839
## iter  90 value 293.900179
## iter 100 value 293.845592
## final  value 293.845592
## stopped after 100 iterations
## # weights:  49
## initial  value 475.102435
## iter  10 value 312.180529
## iter  20 value 292.301994
## iter  30 value 286.288620
## iter  40 value 281.085503
## iter  50 value 273.943676
## iter  60 value 273.364471
## iter  70 value 273.334719
## iter  80 value 273.316862
## iter  90 value 273.236884
## iter 100 value 272.772399
## final  value 272.772399
## stopped after 100 iterations
## # weights:  81
## initial  value 503.345268
## iter  10 value 325.164075
## iter  20 value 293.961521
## iter  30 value 283.223141
## iter  40 value 280.459611
## iter  50 value 278.425993
## iter  60 value 273.557072
## iter  70 value 272.278187
## iter  80 value 271.799833
## iter  90 value 271.565937
## iter 100 value 269.452459
## final  value 269.452459
## stopped after 100 iterations
## # weights:  17
## initial  value 436.469994
## iter  10 value 336.457543
## iter  20 value 313.473112
## iter  30 value 292.830781
## iter  40 value 280.406063
## iter  50 value 274.839697
## iter  60 value 274.740711
## iter  70 value 274.732588
## iter  70 value 274.732587
## iter  70 value 274.732587
## final  value 274.732587
## converged
## # weights:  49
## initial  value 488.958639
## iter  10 value 323.827337
## iter  20 value 308.681635
## iter  30 value 287.171597
## iter  40 value 270.600714
## iter  50 value 260.718609
## iter  60 value 246.354325
## iter  70 value 241.169216
## iter  80 value 240.274922
## iter  90 value 240.262313
## final  value 240.262299
## converged
## # weights:  81
## initial  value 451.168094
## iter  10 value 320.234637
## iter  20 value 297.771928
## iter  30 value 270.529073
## iter  40 value 262.792659
## iter  50 value 260.304641
## iter  60 value 259.172189
## iter  70 value 258.524930
## iter  80 value 254.207573
## iter  90 value 237.347242
## iter 100 value 234.580770
## final  value 234.580770
## stopped after 100 iterations
## # weights:  17
## initial  value 507.944567
## iter  10 value 398.280981
## iter  20 value 366.029513
## iter  30 value 358.155373
## iter  40 value 345.942435
## iter  50 value 303.235627
## iter  60 value 299.103505
## iter  70 value 294.452246
## iter  80 value 290.084368
## iter  90 value 288.861797
## iter 100 value 287.354392
## final  value 287.354392
## stopped after 100 iterations
## # weights:  49
## initial  value 433.379693
## iter  10 value 320.051024
## iter  20 value 292.227008
## iter  30 value 286.526470
## iter  40 value 272.184570
## iter  50 value 264.997050
## iter  60 value 263.079693
## iter  70 value 262.993052
## iter  80 value 262.787699
## iter  90 value 262.704199
## iter 100 value 262.305948
## final  value 262.305948
## stopped after 100 iterations
## # weights:  81
## initial  value 434.777851
## iter  10 value 322.208770
## iter  20 value 287.114902
## iter  30 value 261.275954
## iter  40 value 253.101632
## iter  50 value 247.948221
## iter  60 value 244.501256
## iter  70 value 239.885275
## iter  80 value 236.246487
## iter  90 value 235.113200
## iter 100 value 234.815198
## final  value 234.815198
## stopped after 100 iterations
## # weights:  17
## initial  value 438.014114
## iter  10 value 330.765667
## iter  20 value 284.127087
## iter  30 value 281.059833
## iter  40 value 280.851476
## iter  50 value 280.216274
## iter  60 value 279.276662
## iter  70 value 277.982506
## iter  80 value 277.442295
## iter  90 value 276.586528
## iter 100 value 276.254143
## final  value 276.254143
## stopped after 100 iterations
## # weights:  49
## initial  value 557.196335
## iter  10 value 347.659241
## iter  20 value 313.809489
## iter  30 value 285.960771
## iter  40 value 278.605367
## iter  50 value 269.469231
## iter  60 value 264.147102
## iter  70 value 263.036086
## iter  80 value 262.468245
## final  value 262.465452
## converged
## # weights:  81
## initial  value 506.031550
## iter  10 value 314.966321
## iter  20 value 312.576965
## iter  30 value 301.001142
## iter  40 value 291.102443
## iter  50 value 289.268466
## iter  60 value 287.667413
## iter  70 value 287.605915
## iter  80 value 287.574445
## iter  90 value 287.340163
## iter 100 value 283.970269
## final  value 283.970269
## stopped after 100 iterations
## # weights:  17
## initial  value 491.114968
## iter  10 value 419.887129
## iter  20 value 308.340646
## iter  30 value 287.443733
## iter  40 value 277.950018
## iter  50 value 277.167041
## final  value 277.156521
## converged
## # weights:  49
## initial  value 457.195490
## iter  10 value 339.976180
## iter  20 value 285.177795
## iter  30 value 269.443418
## iter  40 value 265.853496
## iter  50 value 263.022849
## iter  60 value 262.089355
## iter  70 value 261.466856
## iter  80 value 256.431551
## iter  90 value 250.646417
## iter 100 value 248.526311
## final  value 248.526311
## stopped after 100 iterations
## # weights:  81
## initial  value 439.157712
## iter  10 value 410.396774
## iter  20 value 331.484172
## iter  30 value 319.354152
## iter  40 value 294.761761
## iter  50 value 284.816630
## iter  60 value 279.507018
## iter  70 value 264.169448
## iter  80 value 252.461112
## iter  90 value 250.577990
## iter 100 value 248.171112
## final  value 248.171112
## stopped after 100 iterations
## # weights:  17
## initial  value 519.338798
## iter  10 value 438.898911
## iter  20 value 438.889864
## final  value 438.885083
## converged
## # weights:  49
## initial  value 436.911359
## iter  10 value 300.537807
## iter  20 value 282.866298
## iter  30 value 279.101693
## iter  40 value 273.543145
## iter  50 value 272.462858
## iter  60 value 272.295259
## iter  70 value 271.615773
## iter  80 value 268.988652
## iter  90 value 267.633542
## iter 100 value 267.507216
## final  value 267.507216
## stopped after 100 iterations
## # weights:  81
## initial  value 445.305358
## iter  10 value 366.691282
## iter  20 value 308.428133
## iter  30 value 305.539219
## iter  40 value 302.848654
## iter  50 value 293.889958
## iter  60 value 281.257410
## iter  70 value 278.034058
## iter  80 value 276.519739
## iter  90 value 275.292473
## iter 100 value 275.219446
## final  value 275.219446
## stopped after 100 iterations
## # weights:  49
## initial  value 518.023201
## iter  10 value 387.512195
## iter  20 value 355.593374
## iter  30 value 336.174362
## iter  40 value 329.553093
## iter  50 value 322.947500
## iter  60 value 314.900491
## iter  70 value 310.574827
## iter  80 value 309.724885
## iter  90 value 308.825353
## iter 100 value 308.816296
## final  value 308.816296
## stopped after 100 iterations
```

```
ann.fit
```

```
## Neural Network
##
## 815 samples
##  14 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 652, 652, 651, 653, 652
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##   1     0e+00  0.7254351  0.3451995
##   1     1e-04  0.6799833  0.2226425
##   1     1e-01  0.8086235  0.5825008
##   3     0e+00  0.8025110  0.5592374
##   3     1e-04  0.8000495  0.5571504
##   3     1e-01  0.8160157  0.6005999
##   5     0e+00  0.7975880  0.5469025
##   5     1e-04  0.8061696  0.5786849
##   5     1e-01  0.8062221  0.5824882
##
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.1.
```

```
# To determine which neurons in the hidden layer performed best
ann.fit$results[order(-ann.fit$results$Kappa), ] # 3 neurons model has better Kappa statistic compared to other n
eurons which indicates a good agreement between the actual and the predicted values.
```

	size	decay	Accuracy	Kappa	AccuracySD	KappaSD
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
6	3	1e-01	0.8160157	0.6005999	0.02570661	0.05676857
3	1	1e-01	0.8086235	0.5825008	0.01643347	0.03524268
9	5	1e-01	0.8062221	0.5824882	0.03383460	0.06519875
8	5	1e-04	0.8061696	0.5786849	0.01345359	0.02233109
4	3	0e+00	0.8025110	0.5592374	0.02279026	0.04763843
5	3	1e-04	0.8000495	0.5571504	0.02257785	0.04419988
7	5	0e+00	0.7975880	0.5469025	0.01722085	0.03871427
1	1	0e+00	0.7254351	0.3451995	0.11608354	0.31726890
2	1	1e-04	0.6799833	0.2226425	0.11010784	0.30581900

9 rows

```
# Make predictions on the test data
pred.ann.fit <- predict(ann.fit, newdata = valid[,-9], type = "raw")

# create a confusion matrix
confusionMatrix(pred.ann.fit, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 934  17
##          1 490  59
##                                          
##                Accuracy : 0.662          
##                  95% CI : (0.6374, 0.6859)
##     No Information Rate : 0.9493         
##     P-Value [Acc > NIR] : 1              
##                                          
##                   Kappa : 0.1095         
##                                          
##  Mcnemar's Test P-Value : <2e-16         
##                                          
##             Sensitivity : 0.77632        
##             Specificity : 0.65590        
##          Pos Pred Value : 0.10747        
##          Neg Pred Value : 0.98212        
##              Prevalence : 0.05067        
##          Detection Rate : 0.03933        
##    Detection Prevalence : 0.36600        
##       Balanced Accuracy : 0.71611        
##                                          
##        'Positive' Class : 1              
## 
```

From the summary, I have observed that:

In the neural network model, 5-fold cross-validation was performed using 14 predictor variables to predict a binary outcome with 815 samples. The model was tuned with different values of size and decay. The selected values for the final model were size=3 and decay=0.1, which resulted in an accuracy of 0.816 and a Kappa value of 0.601.

From the confusion matrix, I have observed that:

1. The accuracy of the model is 0.662 which indicates that it correctly classified 66.2% of the instances.

2.The Kappa coefficient is 0.1095 which indicates a slight agreement.

3. The sensitivity of the model is 0.77632 which indicates the proportion of actual positive instances that were correctly identified by the model. The specificity is 0.65590 which indicates the proportion of actual negative instances that were correctly identified by the model.
4. The model has 934 true negatives, 17 false negatives, 490 false positives, 59 true positives.

In conclusion, I have used the Kappa metric in the train model because Kappa metric is better for imbalanced data and if we train the model on the kappa metric then it will be helpful when evaluating the test data which is imbalanced. Additionally, After hyperparamter tuning, I have observed that the Logistic regression model is better suited for the stroke prediction in comparison to the other models after hyperparameter tuning. The second model which is better suited for the stroke prediction would be Decision Tree.

On overall comparsion of the models before and after hyper parameter tuning, the decision tree model has showed a good improvement after hyperparameter tuning. However, other models like the ANN, random forest and logistic regression didn't show much improvement in terms of predicting the stroke cases better, on the other hand, the ANN model's true positive values prediction fell after the hyper parameter tuning indicating the previous model was better at its capacity.

Model Tuning with ensemble

```
# Bagging method
set.seed(123)

# Create model using the bagging method
model.bag <- bagging(stroke ~ . , data = train.smote, nbagg = 25 )

predict.bag <- predict(model.bag, valid[,-9])

confusionMatrix(predict.bag, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1020   22
##          1  404   54
##
##                Accuracy : 0.716
##                  95% CI : (0.6924, 0.7387)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1263
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71053
##             Specificity : 0.71629
##          Pos Pred Value : 0.11790
##          Neg Pred Value : 0.97889
##              Prevalence : 0.05067
##          Detection Rate : 0.03600
##    Detection Prevalence : 0.30533
##       Balanced Accuracy : 0.71341
##
##        'Positive' Class : 1
##
```

From the confusion matrix, I have observed that:

1. The accuracy of the model is 71.6%, which means that 71.6% of the predictions made by the model were correct.
2. The sensitivity of the model for class 1 is 71.05%, which means that 71.05% of the actual class 1 observations were correctly identified as class 1 by the model. The specificity of the model for class 0 is 71.63%, which means that 71.63% of the actual class 0 observations were correctly identified as class 0 by the model.
3. The kappa statistic is 0.1263, which indicates that the agreement between the predicted and actual values is only slightly better than chance.
4. The model has 54 true positives, 22 false negatives, 1020 true negatives, 404 false positives.

Overall, the model has a low PPV for class 1, indicating that it may be better at identifying class 0 observations than class 1 observations

```
# Performing the bagging method with cross validation

set.seed(123)

# create model
model.bag1 <- train(stroke ~ ., data = train.smote, method = "treebag", metric = "Kappa", trControl = trainContro
l(method = "cv", number = 9))

#predict the values
predict.bag1 <- predict(model.bag1, valid[,-9])

# create a confusion matrix
confusionMatrix(predict.bag1, valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 1022   20
##          1  402   56
##                                           
##                Accuracy : 0.7187          
##                  95% CI : (0.6952, 0.7413)
##     No Information Rate : 0.9493          
##     P-Value [Acc > NIR] : 1               
##                                           
##                   Kappa : 0.1345          
##                                           
##  Mcnemar's Test P-Value : <2e-16          
##                                           
##             Sensitivity : 0.73684         
##             Specificity : 0.71770         
##          Pos Pred Value : 0.12227         
##          Neg Pred Value : 0.98081         
##              Prevalence : 0.05067         
##          Detection Rate : 0.03733         
##    Detection Prevalence : 0.30533         
##       Balanced Accuracy : 0.72727         
##                                           
##        'Positive' Class : 1               
## 
```

From the confusion matrix, I have observed that:

After performing 9 k-fold cross validation, the model's prediction for true positives slightly improved. So, it is something we can experiment with.

The accuracy of the model is 71.87%, which means that 71.87% of the predictions made by the model were correct.

The sensitivity of the model for class 1 is 73.68%, which means that 73.68% of the actual class 1 observations were correctly identified as class 1 by the model. The specificity of the model for class 0 is 71.77%, which means that 71.77% of the actual class 0 observations were correctly identified as class 0 by the model.

The kappa statistic is 0.1345, which indicates that the agreement between the predicted and actual values is only slightly better than chance.

The model has predicted 56 true positives, 20 false negatives, 1022 true negatives, 402 false positives.

Ensemble Model

```
# create an ensemble function - An ensemble model generally means taking predictions of different models and comb
ining them in some fashion to get one set of predictions.

Ensemble.model <- function(testdata){

  # Use the four models to predict the stroke class for the test data
  pred.tree <- predict(model.tree, newdata = valid[,-9], type = "class")
  pred.reg <- predict(model.reg, newdata = valid[,-9])
  pred.forest <- predict(model.rf, newdata = valid[,-9])
  pred.ann <- predict(model.ann, newdata = valid[,-9], type = "class")

  # Combine the predictions into a single data frame
  predictions <- data.frame(pred.tree, pred.reg, pred.forest, pred.ann)

  # Calculate the mode for each row (i.e. the most common class)
  ensemble_pred <- apply(predictions, 1, function(x) {
    names(sort(table(x), decreasing = TRUE))[1]
  })

  return(ensemble_pred)
}
```

```
# prediction of the ensemble model
ensemble.pred <- Ensemble.model(valid[,-9])

confusionMatrix(as.factor(ensemble.pred), valid$stroke, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 952  14
##          1 472  62
##
##                Accuracy : 0.676
##                  95% CI : (0.6517, 0.6997)
##     No Information Rate : 0.9493
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1257
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.81579
##             Specificity : 0.66854
##          Pos Pred Value : 0.11610
##          Neg Pred Value : 0.98551
##              Prevalence : 0.05067
##          Detection Rate : 0.04133
##    Detection Prevalence : 0.35600
##       Balanced Accuracy : 0.74216
##
##        'Positive' Class : 1
##
```

From the confusion matrix, I have observed that:

The accuracy of the model is 67.6%, which means that 67.6% of the predictions made by the model were correct.

The sensitivity of the model for class 1 is 81.58%, which means that 81.58% of the actual class 1 observations were correctly identified as class 1 by the model. The specificity of the model for class 0 is 66.85%, which means that 66.85% of the actual class 0 observations were correctly identified as class 0 by the model.

The PPV for class 1 is 11.61%, which means that only 11.61% of the predicted class 1 observations were actually class 1. The NPV for class 0 is 98.55%, which means that 98.55% of the predicted class 0 observations were actually class 0.

The kappa statistic is 0.1257, which indicates that the agreement between the predicted and actual values is only slightly better than chance.

The model has predicted 62 true positives, 14 false negatives, 952 true negatives, 472 false positives.

```
# Comparison of Ensemble model's performance vs Individual models

## Ensemble model
accuracy.en = 0.674 * 100
accuracy.en
```

```
## [1] 67.4
```

```
precision.en <- 62 / (62 + 475)
precision.en
```

```
## [1] 0.1154562
```

```
recall.en <- 62/ (62+14)
recall.en
```

```
## [1] 0.8157895
```

```
F1_score.en <- 2 * (precision.en * recall.en) / (precision.en + recall.en)
F1_score.en
```

```
## [1] 0.2022838
```

```
# Individual Models

# Logistic Regression
accuracy.reg1 <- 0.6713 * 100
accuracy.reg1
```

```
## [1] 67.13
```

```
precision.reg1 <- 61/ (61+478)
precision.reg1
```

```
## [1] 0.1131725
```

```
recall.reg1 <- 61/(61+15)
recall.reg1
```

```
## [1] 0.8026316
```

```
F1.score.reg1 <- 2 * (precision.reg1 * recall.reg1) / (precision.reg1 + recall.reg1)
F1.score.reg1
```

```
## [1] 0.198374
```

```
# Decision Tree
accuracy.tree1 <- 0.7067 * 100
accuracy.tree1
```

```
## [1] 70.67
```

```
precision.tree1 <- 58/(58+422)
precision.tree1
```

```
## [1] 0.1208333
```

```
recall.tree1 <- 58/(58+18)
recall.tree1
```

```
## [1] 0.7631579
```

```
F1.score.tree1 <-  2 * (precision.tree1 * recall.tree1) / (precision.tree1 + recall.tree1)
F1.score.tree1
```

```
## [1] 0.2086331
```

```
#Random forest
accuracy.rf1 <- 0.6813 * 100
accuracy.rf1
```

```
## [1] 68.13
```

```
precision.rf1 <- 59/(59+461)
precision.rf1
```

```
## [1] 0.1134615
```

```
recall.rf1 <- 59/(59+17)
recall.rf1
```

```
## [1] 0.7763158
```

```
F1.score.rf1 <-  2 * (precision.rf1 * recall.rf1) / (precision.rf1 + recall.rf1)
F1.score.rf1
```

```
## [1] 0.1979866
```

```
# Artificial Neural Network
accuracy.ann1 <- 0.636 * 100
accuracy.ann1
```

```
## [1] 63.6
```

```
precision.ann1 <- 64/(64+534)
precision.ann1
```

```
## [1] 0.1070234
```

```
recall.ann1 <- 64/(64+12)
recall.ann1
```

```
## [1] 0.8421053
```

```
F1.score.ann1 <-  2 * (precision.ann1 * recall.ann1) / (precision.ann1 + recall.ann1)
F1.score.ann1
```

```
## [1] 0.189911
```

To give a better display of the statistics, I will provide a manual table here:

```
                         Accuracy        Precision         Recall        F1 score
```

Ensemble 67.4 0.1154562 0.8157895 0.2022838

Logistic regression 67.13 0.1131725 0.8026316 0.198374

Decision Tree 70.67 0.1208333 0.7631579 0.2086331

Random Forest 68.13. 0.1134615 0.7763158 0.1979866

Artificial Neural Net 63.6 0.1070234 0.8421053 0.189911

According to the above table,

1. Decision Tree has the highest accuracy, precision and F1 score.
2. ANN has the highest recall score indicating that it has the highest ability to correctly identify positive cases.
3. The Ensemble and logistic regression models also has a good recall score.
4. The Random Forest model performs relatively well across all metrics but does not achieve the highest score in any of them.
5. The Logistic Regression model performs comparably to the Ensemble model in terms of accuracy, precision, and F1 score.

Overall, the ANN model is better suited for predicting the stroke cases in comparison to the other models. This is followed by ensemble model and logistic regression model.

Deployement

To perform this phase, we have the ANN model ready to perform on real-world healthcare datasets. However, on this stroke prediction dataset, the ANN model was successful in predicting 64 stroke cases out of 76.

References

Huang, C., Zhang, Y., Hu, Y., & Yang, J. (n.d.). Visual analysis and prediction of stroke - stat.cmu.edu. Visual Analysis and Prediction of Stroke. Retrieved April 27, 2023, from https://www.stat.cmu.edu/capstoneresearch/spring2021/315files/team16.html (https://www.stat.cmu.edu/capstoneresearch/spring2021/315files/team16.html)

Yeap, Z. C. (2020, December 9). Exploratory Data Analysis on Stroke Dataset. Medium. Retrieved April 26, 2023, from https://towardsdatascience.com/step-by-step-exploratory-data-analysis-on-stroke-dataset-840aefea8739 (https://towardsdatascience.com/step-by-step-exploratory-data-analysis-on-stroke-dataset-840aefea8739)

Arnaldo, M. (2021, April 23). Smote and best subset selection for linear regression in R. Analytics Vidhya. Retrieved April 26, 2023, from https://www.analyticsvidhya.com/blog/2021/04/smote-and-best-subset-selection-for-linear-regression-in-r/ (https://www.analyticsvidhya.com/blog/2021/04/smote-and-best-subset-selection-for-linear-regression-in-r/)

Alberg, J. (2015, June 21). R, caret, and parameter tuning c5.0. Euclidean Technologies ®. Retrieved April 26, 2023, from https://www.euclidean.com/machine-learning-in-practice/2015/6/12/r-caret-and-parameter-tuning-c50 (https://www.euclidean.com/machine-learning-in-practice/2015/6/12/r-caret-and-parameter-tuning-c50)