Homework 5

RakeshDR

2023-11-12

Importing all the required Libraries

```r
knitr::opts_chunk$set(warning = FALSE, message = FALSE)

library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(rpart)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ggplot2)
library(rattle)
```

```
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(dplyr)
library(e1071)
library(stats)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.3.2

##
## Attaching package: 'kknn'
##
## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(cluster)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 4.3.2

## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Loading required package: iterators
## Loading required package: parallel
```

a. Data Gathering and Integration

I am using the heart stroke data-set for this homework assignment 5 which I downloaded from the Kaggle data portal

This data is used to predict whether a patient is likely to get a stroke based on the input parameters like gender, age, heart conditions like hyper-tension, heart disease, glucose level in blood and smoking status

This data-set contains both categorical and numerical data. And attributes of the data-set are listed below -

1. id: unique identifier
2. gender: "Male", "Female" or "Other"
3. age: age of the patient
4. hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
5. heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
6. ever_married: "No" or "Yes"
7. work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
8. Residence_type: "Rural" or "Urban"
9. avg_glucose_level: average glucose level in blood
10. bmi: body mass index
11. smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
12. stroke: 1 if the patient had a stroke or 0 if not

```r
# Loading Stroke Data from CSV File
stroke_data_raw <- read.csv("C:/Users/harsh/Downloads/Fundamentals of DataScience/healthcare-dataset-st

# Creating a Data Frame for Stroke Analysis
stroke_analysis_df <- stroke_data_raw

# Previewing the Top Six Records of the Data Frame
head(stroke_analysis_df)
```

```
##      id gender age hypertension heart_disease ever_married     work_type
## 1  9046   Male  67            0             1          Yes       Private
## 2 51676 Female  61            0             0          Yes Self-employed
## 3 31112   Male  80            0             1          Yes       Private
## 4 60182 Female  49            0             0          Yes       Private
## 5  1665 Female  79            1             0          Yes Self-employed
## 6 56669   Male  81            0             0          Yes       Private
##   Residence_type avg_glucose_level  bmi  smoking_status stroke
## 1          Urban            228.69 36.6 formerly smoked      1
## 2          Rural            202.21  N/A    never smoked      1
## 3          Rural            105.92 32.5    never smoked      1
## 4          Urban            171.23 34.4          smokes      1
## 5          Rural            174.12   24    never smoked      1
## 6          Urban            186.21   29 formerly smoked      1
```

Removing the unique identifier - id

```r
# Excluding the ID Column for Data Anonymization
stroke_analysis_df <- stroke_analysis_df %>% dplyr::select(-c(id))
```

Missing values - Checking for NA/missing values using the summary function

```r
# Generating a Statistical Summary of the Stroke Data to fing missing values
summary(stroke_analysis_df)
```

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

Looking at the summary function we conclude that all the numerical variables are clean and ready for further step

Checking for missing values in categorical data -

```r
# Identifying Unique Genders in the Dataset
unique(stroke_analysis_df$gender)
```

```
## [1] "Male"   "Female" "Other"
```

```r
# Listing Distinct Marital Status Entries
unique(stroke_analysis_df$ever_married)
```

```
## [1] "Yes" "No"
```

```r
# Enumerating Various Types of Employment
unique(stroke_analysis_df$work_type)
```

```
## [1] "Private"       "Self-employed" "Govt_job"      "children"
## [5] "Never_worked"
```

```r
# Categorizing by Residence Type
unique(stroke_analysis_df$Residence_type)
```

```
## [1] "Urban" "Rural"
```

```r
# Exploring Different Smoking Statuses Recorded
unique(stroke_analysis_df$smoking_status)
```

```
## [1] "formerly smoked" "never smoked"    "smokes"          "Unknown"
```

We notice that there are no missing values in categorical data

Checking for missing values of BMI

```r
unique(stroke_analysis_df$bmi)
```

```
##   [1] "36.6" "N/A"  "32.5" "34.4" "24"   "29"   "27.4" "22.8" "24.2" "29.7"
##  [11] "36.8" "27.3" "28.2" "30.9" "37.5" "25.8" "37.8" "22.4" "48.9" "26.6"
##  [21] "27.2" "23.5" "28.3" "44.2" "25.4" "22.2" "30.5" "26.5" "33.7" "23.1"
##  [31] "32"   "29.9" "23.9" "28.5" "26.4" "20.2" "33.6" "38.6" "39.2" "27.7"
##  [41] "31.4" "36.5" "33.2" "32.8" "40.4" "25.3" "30.2" "47.5" "20.3" "30"
##  [51] "28.9" "28.1" "31.1" "21.7" "27"   "24.1" "45.9" "44.1" "22.9" "29.1"
##  [61] "32.3" "41.1" "25.6" "29.8" "26.3" "26.2" "29.4" "24.4" "28"   "28.8"
##  [71] "34.6" "19.4" "30.3" "41.5" "22.6" "56.6" "27.1" "31.3" "31"   "31.7"
##  [81] "35.8" "28.4" "20.1" "26.7" "38.7" "34.9" "25"   "23.8" "21.8" "27.5"
##  [91] "24.6" "32.9" "26.1" "31.9" "34.1" "36.9" "37.3" "45.7" "34.2" "23.6"
## [101] "22.3" "37.1" "45"   "25.5" "30.8" "37.4" "34.5" "27.9" "29.5" "46"
## [111] "42.5" "35.5" "26.9" "45.5" "31.5" "33"   "23.4" "30.7" "20.5" "21.5"
## [121] "40"   "28.6" "42.2" "29.6" "35.4" "16.9" "26.8" "39.3" "32.6" "35.9"
## [131] "21.2" "42.4" "40.5" "36.7" "29.3" "19.6" "18"   "17.6" "19.1" "50.1"
## [141] "17.7" "54.6" "35"   "22"   "39.4" "19.7" "22.5" "25.2" "41.8" "60.9"
## [151] "23.7" "24.5" "31.2" "16"   "31.6" "25.1" "24.8" "18.3" "20"   "19.5"
## [161] "36"   "35.3" "40.1" "43.1" "21.4" "34.3" "27.6" "16.5" "24.3" "25.7"
## [171] "21.9" "38.4" "25.9" "54.7" "18.6" "24.9" "48.2" "20.7" "39.5" "23.3"
## [181] "64.8" "35.1" "43.6" "21"   "47.3" "16.6" "21.6" "15.5" "35.6" "16.7"
## [191] "41.9" "16.4" "17.1" "29.2" "37.9" "44.6" "39.6" "40.3" "41.6" "39"
## [201] "23.2" "18.9" "36.1" "36.3" "46.5" "16.8" "46.6" "35.2" "20.9" "13.8"
## [211] "31.8" "15.3" "38.2" "45.2" "17"   "49.8" "27.8" "60.2" "23"   "22.1"
## [221] "26"   "44.3" "51"   "39.7" "34.7" "21.3" "41.2" "34.8" "19.2" "35.7"
## [231] "40.8" "24.7" "19"   "32.4" "34"   "28.7" "32.1" "51.5" "20.4" "30.6"
## [241] "71.9" "19.3" "40.9" "17.2" "16.1" "16.2" "40.6" "18.4" "21.1" "42.3"
## [251] "32.2" "50.2" "17.5" "18.7" "42.1" "47.8" "20.8" "30.1" "17.3" "36.4"
## [261] "12"   "36.2" "55.7" "14.4" "43"   "41.7" "33.8" "43.9" "22.7" "57.5"
## [271] "37"   "38.5" "16.3" "44"   "32.7" "54.2" "40.2" "33.3" "17.4" "41.3"
## [281] "52.3" "14.6" "17.8" "46.1" "33.1" "18.1" "43.8" "50.3" "38.9" "43.7"
## [291] "39.9" "15.9" "19.8" "12.3" "78"   "38.3" "41"   "42.6" "43.4" "15.1"
## [301] "20.6" "33.5" "43.2" "30.4" "38"   "33.4" "44.9" "44.7" "37.6" "39.8"
## [311] "53.4" "55.2" "42"   "37.2" "42.8" "18.8" "42.9" "14.3" "37.7" "48.4"
## [321] "50.6" "46.2" "49.5" "43.3" "33.9" "18.5" "44.5" "45.4" "55"   "54.8"
## [331] "19.9" "17.9" "15.6" "52.8" "15.2" "66.8" "55.1" "18.2" "48.5" "55.9"
## [341] "57.3" "10.3" "14.1" "15.7" "56"   "44.8" "13.4" "51.8" "38.1" "57.7"
## [351] "44.4" "38.8" "49.3" "39.1" "54"   "56.1" "97.6" "53.9" "13.7" "11.5"
## [361] "41.4" "14.2" "49.4" "15.4" "45.1" "49.2" "48.7" "53.8" "42.7" "48.8"
## [371] "52.7" "53.5" "50.5" "15.8" "45.3" "14.8" "51.9" "63.3" "40.7" "61.2"
## [381] "48"   "46.8" "48.3" "58.1" "50.4" "11.3" "12.8" "13.5" "14.5" "15"
## [391] "59.7" "47.4" "52.5" "13.2" "52.9" "61.6" "49.9" "54.3" "47.9" "13"
## [401] "13.9" "50.9" "57.2" "64.4" "92"   "50.8" "57.9" "45.8" "47.6" "14"
## [411] "46.4" "46.9" "47.1" "13.3" "48.1" "51.7" "46.3" "54.1" "14.9"
```

We notice some missing values for BMI

We will deal with the missing values by replacing the N/A values with 0

```r
# Transforming BMI to Numeric Format for Analysis
stroke_analysis_df$bmi <- as.numeric(stroke_analysis_df$bmi)

# Substituting Missing BMI Values with Zero
stroke_analysis_df <- stroke_analysis_df %>% mutate(bmi = ifelse(is.na(bmi), 0, bmi))

# Verifying the Replacement of NAs in BMI by Reviewing the Data Summary
summary(stroke_analysis_df)
```

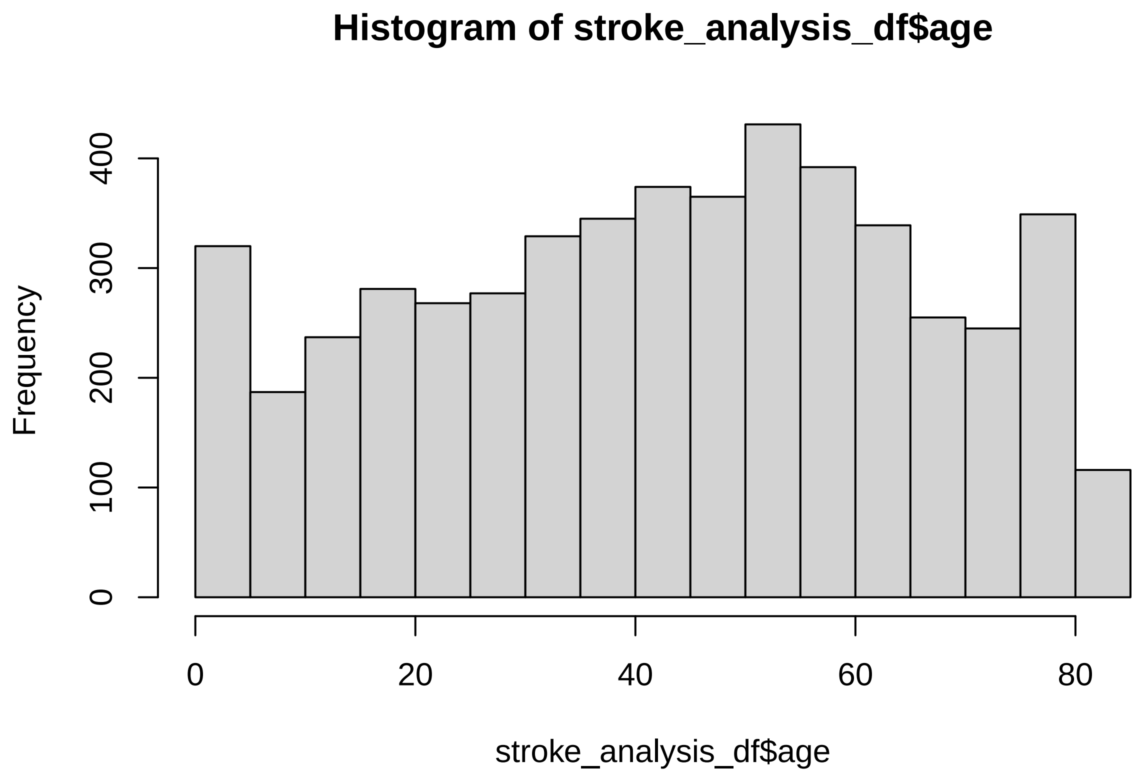
```
##     gender               age         hypertension     heart_disease
##  Length:5110        Min.   : 0.08   Min.   :0.00000   Min.   :0.00000
##  Class :character   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000
##  Mode  :character   Median :45.00   Median :0.00000   Median :0.00000
##                     Mean   :43.23   Mean   :0.09746   Mean   :0.05401
##                     3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000
##                     Max.   :82.00   Max.   :1.00000   Max.   :1.00000
##  ever_married        work_type         Residence_type     avg_glucose_level
##  Length:5110        Length:5110        Length:5110        Min.   : 55.12
##  Class :character   Class :character   Class :character   1st Qu.: 77.25
##  Mode  :character   Mode  :character   Mode  :character   Median : 91.89
##                                                           Mean   :106.15
##                                                           3rd Qu.:114.09
##                                                           Max.   :271.74
##       bmi        smoking_status         stroke
##  Min.   : 0.00   Length:5110        Min.   :0.00000
##  1st Qu.:22.90   Class :character   1st Qu.:0.00000
##  Median :27.70   Mode  :character   Median :0.00000
##  Mean   :27.76                      Mean   :0.04873
##  3rd Qu.:32.80                      3rd Qu.:0.00000
##  Max.   :97.60                      Max.   :1.00000
```

Outliers-

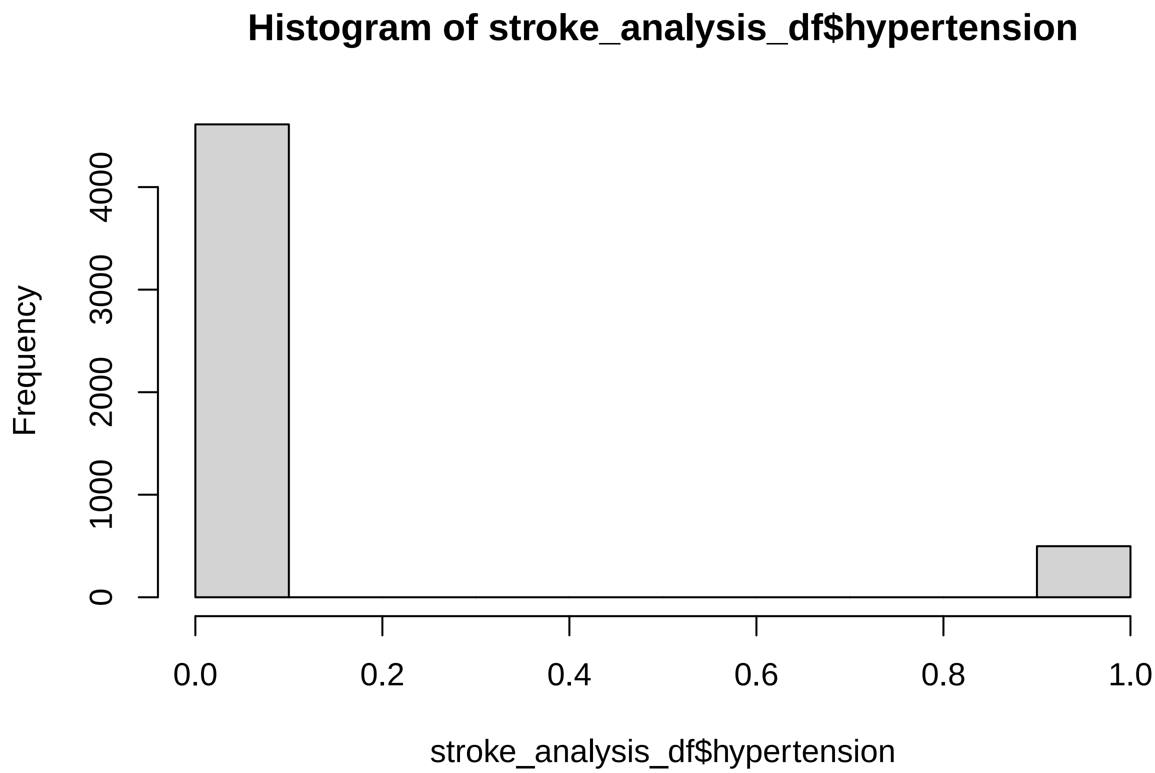
Checking for outliers in numerical variables -

No outliers were found

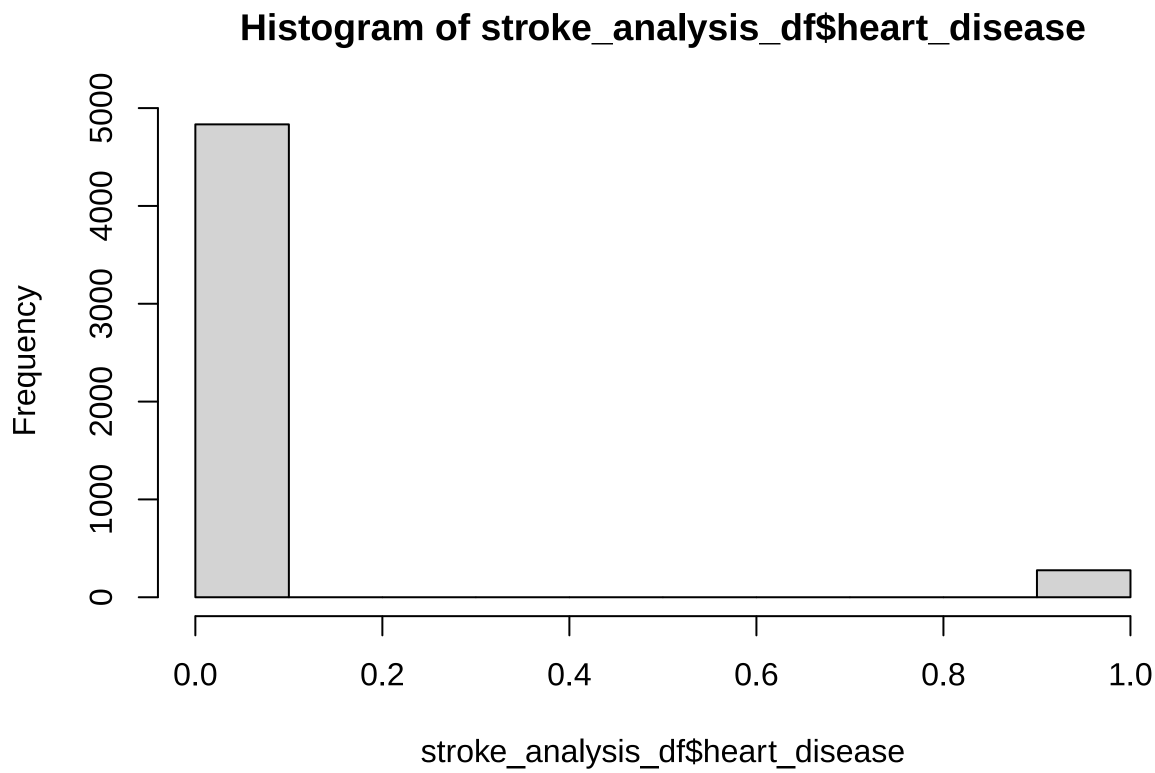
```r
# Creating a Histogram to Visualize Age Distribution
hist(stroke_analysis_df$age)
```



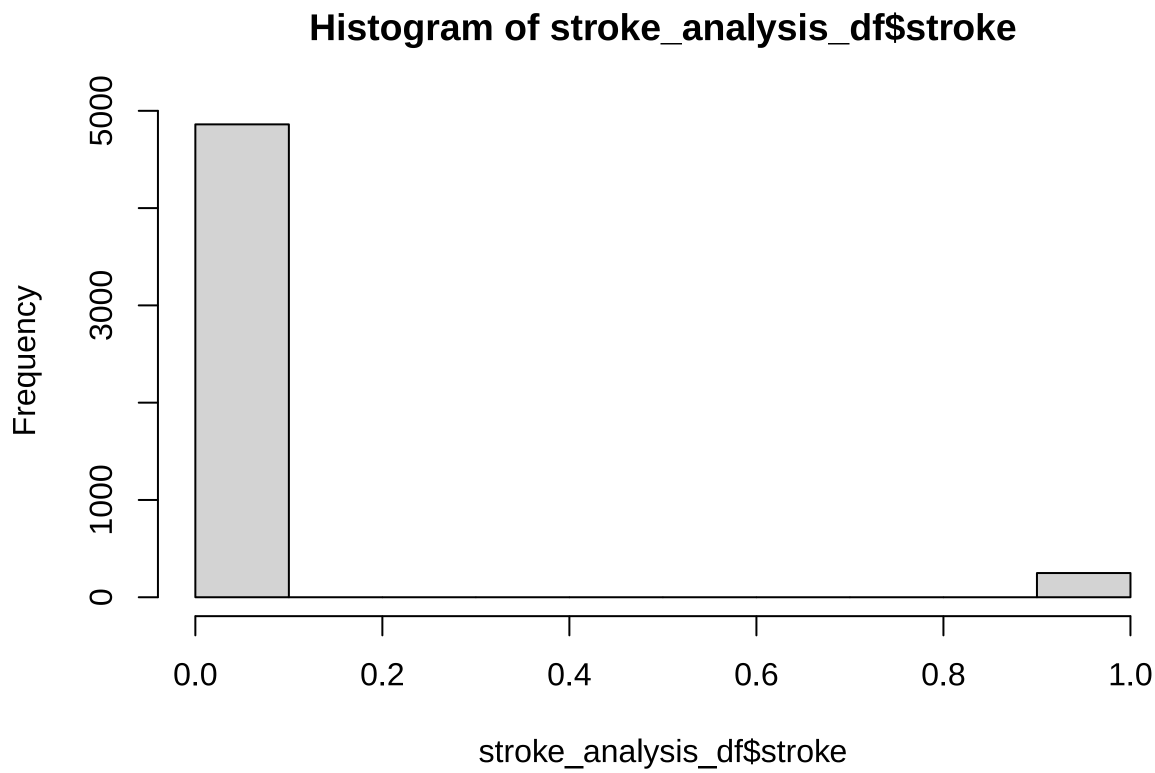

```r
# Plotting a Histogram for Hypertension Prevalence
hist(stroke_analysis_df$hypertension)
```



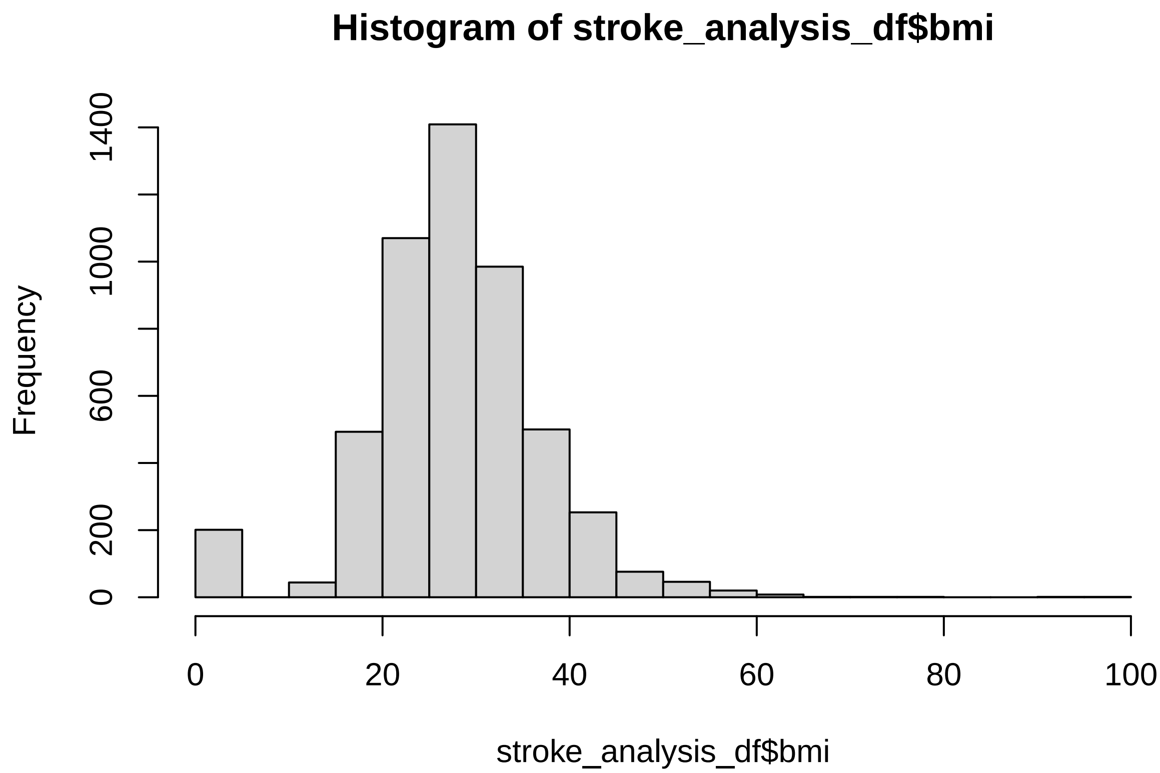

```r
# Histogram of Heart Disease Incidence in the Dataset
hist(stroke_analysis_df$heart_disease)
```



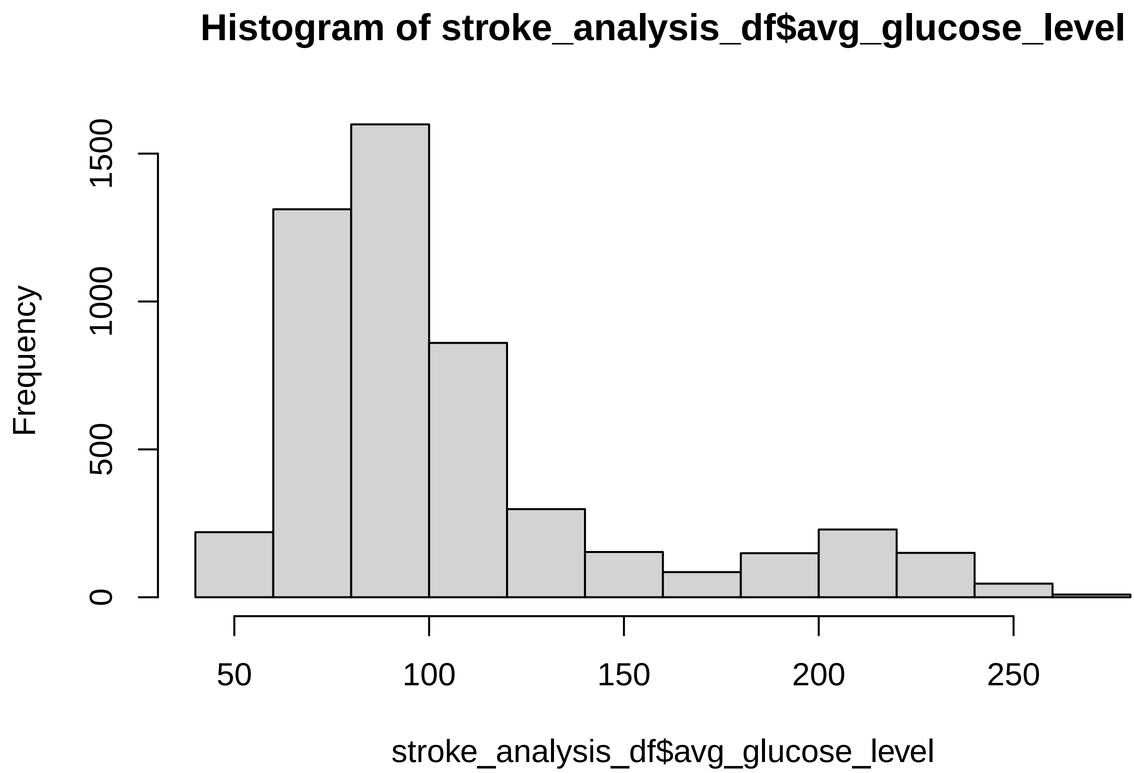

```r
# Visualizing Stroke Occurrences with a Histogram
hist(stroke_analysis_df$stroke)
```




```r
# BMI Distribution Among Participants: A Histogram Analysis
hist(stroke_analysis_df$bmi)
```




```r
# Histogram Depicting Average Glucose Level Variations
hist(stroke_analysis_df$avg_glucose_level)
```



Checking for outliers in categorical variables -

```
# Assessing Potential Outliers in Categorical Variables
unique(stroke_analysis_df$gender)
```

```
## [1] "Male"   "Female" "Other"
```

```
unique(stroke_analysis_df$ever_married)
```

```
## [1] "Yes" "No"
```

```
unique(stroke_analysis_df$work_type)
```

```
## [1] "Private"       "Self-employed" "Govt_job"      "children"
## [5] "Never_worked"
```

```
unique(stroke_analysis_df$Residence_type)
```

```
## [1] "Urban" "Rural"
```

```
unique(stroke_analysis_df$smoking_status)
```

```
## [1] "formerly smoked" "never smoked"    "smokes"          "Unknown"
```

We notice outliers - "other" and "unknown" in gender and smoking_status

Dealing with the outliers -

```
# Analyzing Frequency Counts to Identify Outliers in Gender and Smoking Status
table(stroke_analysis_df$gender)
```

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

```
table(stroke_analysis_df$smoking_status)
```

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

We see that there are 1544 unknown values for smoking_status, which I am deciding to keep as removing this would mean losing a lot of data, which could lead to heavy imbalance in the dataset We see that there is only 1 outlier - other in gender so we will remove it

```
# Filtering Out 'Other' Category from Gender to Remove Outliers
stroke_analysis_df <- stroke_analysis_df %>% filter(gender != "Other")

# Confirming Removal by Displaying Unique Gender Values
unique(stroke_analysis_df$gender)
```

```
## [1] "Male"   "Female"
```

b. Data Exploration

Before looking at the visualizations, I will be converting the categorical variables into factors

```
# Transforming Gender Variable to a Factor for Analysis
stroke_analysis_df$gender <- as.factor(stroke_analysis_df$gender)

# Converting Marital Status to Factor Type
stroke_analysis_df$ever_married <- as.factor(stroke_analysis_df$ever_married)

# Changing Work Type to Factor for Categorical Analysis
stroke_analysis_df$work_type <- as.factor(stroke_analysis_df$work_type)

# Updating Residence Type to Factor Format
stroke_analysis_df$Residence_type <- as.factor(stroke_analysis_df$Residence_type)

# Converting Smoking Status to Factor for Detailed Examination
stroke_analysis_df$smoking_status <- as.factor(stroke_analysis_df$smoking_status)
```

Let's look at the number of different factors of all the categorical variables -

```r
# Counting the Number of Entries by Gender
stroke_analysis_df %>%
  group_by(gender) %>%
  summarise("count" = n())
```

```
## # A tibble: 2 x 2
##   gender count
##   <fct>  <int>
## 1 Female  2994
## 2 Male    2115
```

```r
# Calculating the Total Count for Each Marital Status Category
stroke_analysis_df %>%
  group_by(ever_married) %>%
  summarise("count" = n())
```

```
## # A tibble: 2 x 2
##   ever_married count
##   <fct>        <int>
## 1 No            1756
## 2 Yes           3353
```

```r
# Summarizing the Dataset by Different Work Types
stroke_analysis_df %>%
  group_by(work_type) %>%
  summarise("count" = n())
```

```
## # A tibble: 5 x 2
##   work_type     count
##   <fct>         <int>
## 1 children        687
## 2 Govt_job        657
## 3 Never_worked     22
## 4 Private        2924
## 5 Self-employed   819
```

```r
# Grouping and Counting Entries Based on Residence Type
stroke_analysis_df %>%
  group_by(Residence_type) %>%
  summarise("count" = n())
```

```
## # A tibble: 2 x 2
##   Residence_type count
##   <fct>          <int>
## 1 Rural           2513
## 2 Urban           2596
```

```r
# Analyzing the Distribution of Smoking Status Among Participants
stroke_analysis_df %>%
  group_by(smoking_status) %>%
  summarise("count" = n())
```

```
## # A tibble: 4 x 2
##   smoking_status  count
##   <fct>           <int>
## 1 formerly smoked   884
## 2 never smoked     1892
## 3 smokes            789
## 4 Unknown          1544
```

We can do the same by using the summary function -

```r
# Generating Summary Statistics for Gender Distribution
summary(stroke_analysis_df$gender)
```

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

```r
# Overview of Marital Status Data
summary(stroke_analysis_df$ever_married)
```

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

```r
# Summarizing Work Type Categories in the Dataset
summary(stroke_analysis_df$work_type)
```

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

```r
# Analyzing Residence Type Distribution
summary(stroke_analysis_df$Residence_type)
```

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

```r
# Summary of Smoking Status Among Participants
summary(stroke_analysis_df$smoking_status)
```

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

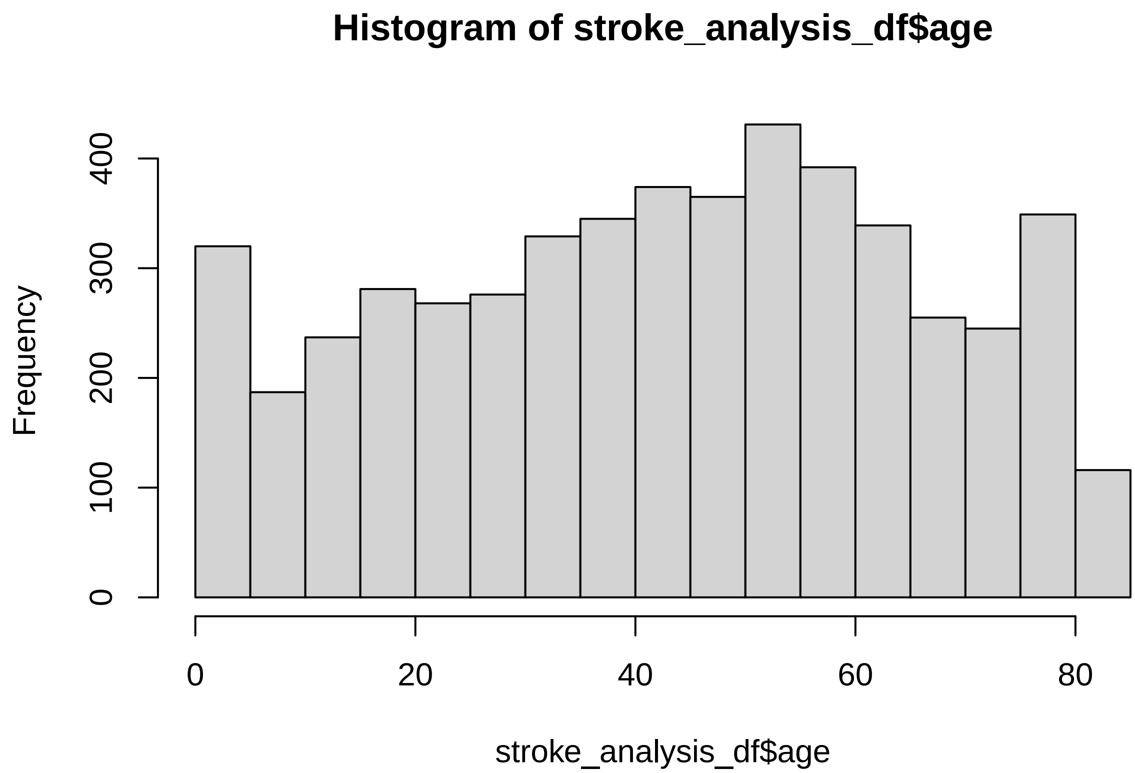
We get some important information about our dataset -

We learned about the diversity of genders present in the dataset, noting a balanced representation, which is favorable. The data revealed a predominance of married individuals compared to those who are not married. In terms of employment, a significant portion of the sample consists of individuals working in the private sector. The dataset also shows an equal distribution of urban and rural residents. Regarding smoking status, it was observed that many participants have never smoked, a positive aspect. There are also a notable number of unknown values in this category; however, we decided to retain them to avoid a substantial imbalance in the dataset.

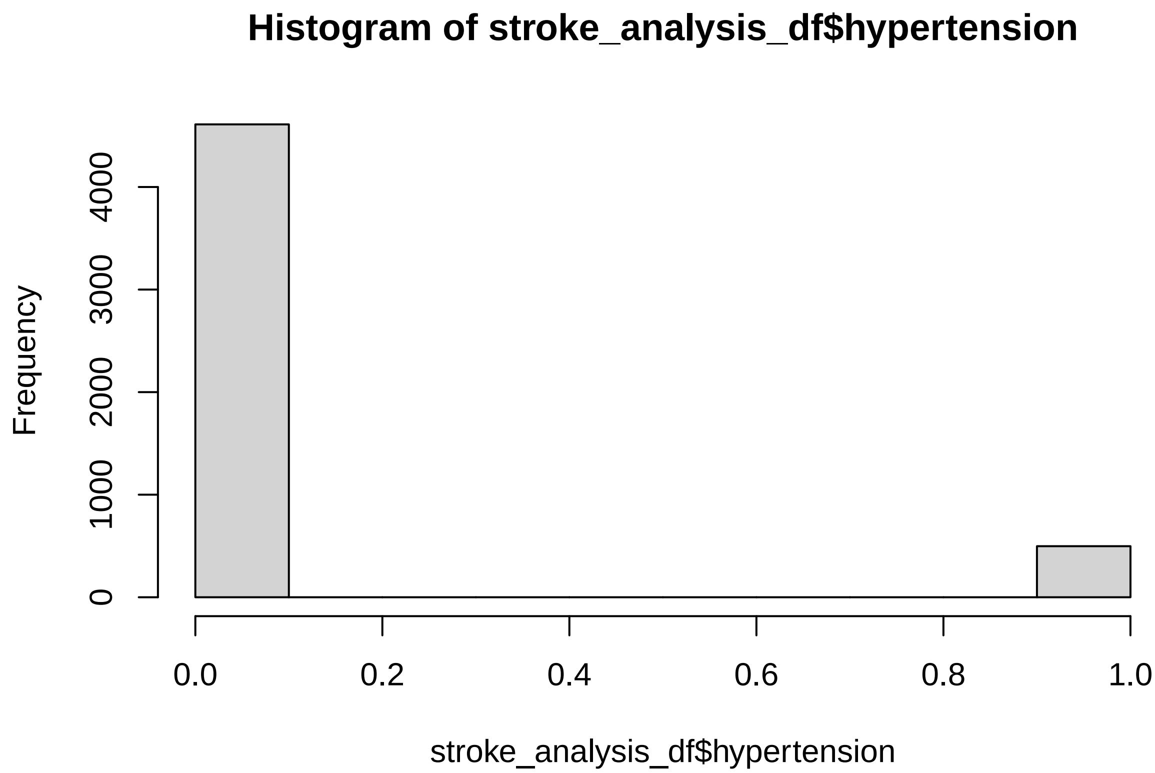
Visualization -

Let's look at visualizations for the numerical variables -

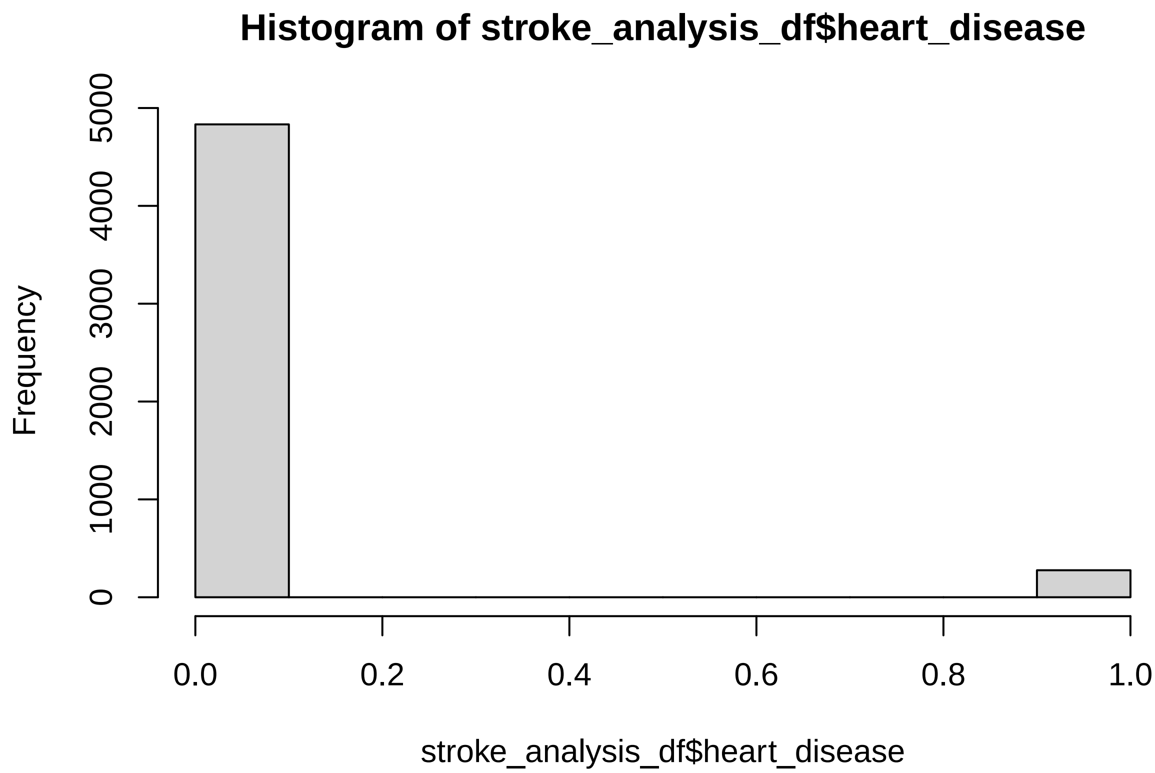
```r
# Plotting a Histogram to Visualize the Age Distribution
hist(stroke_analysis_df$age)
```



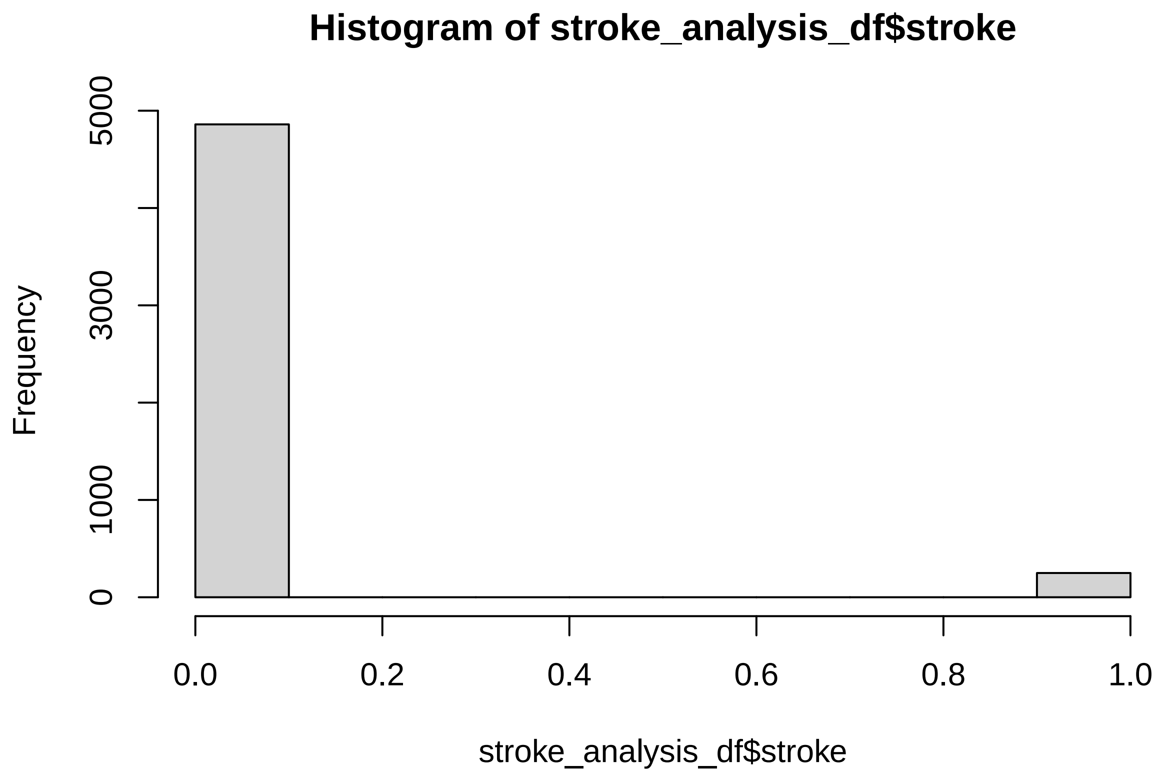

```r
# Histogram Showing Distribution of Hypertension Cases
hist(stroke_analysis_df$hypertension)
```



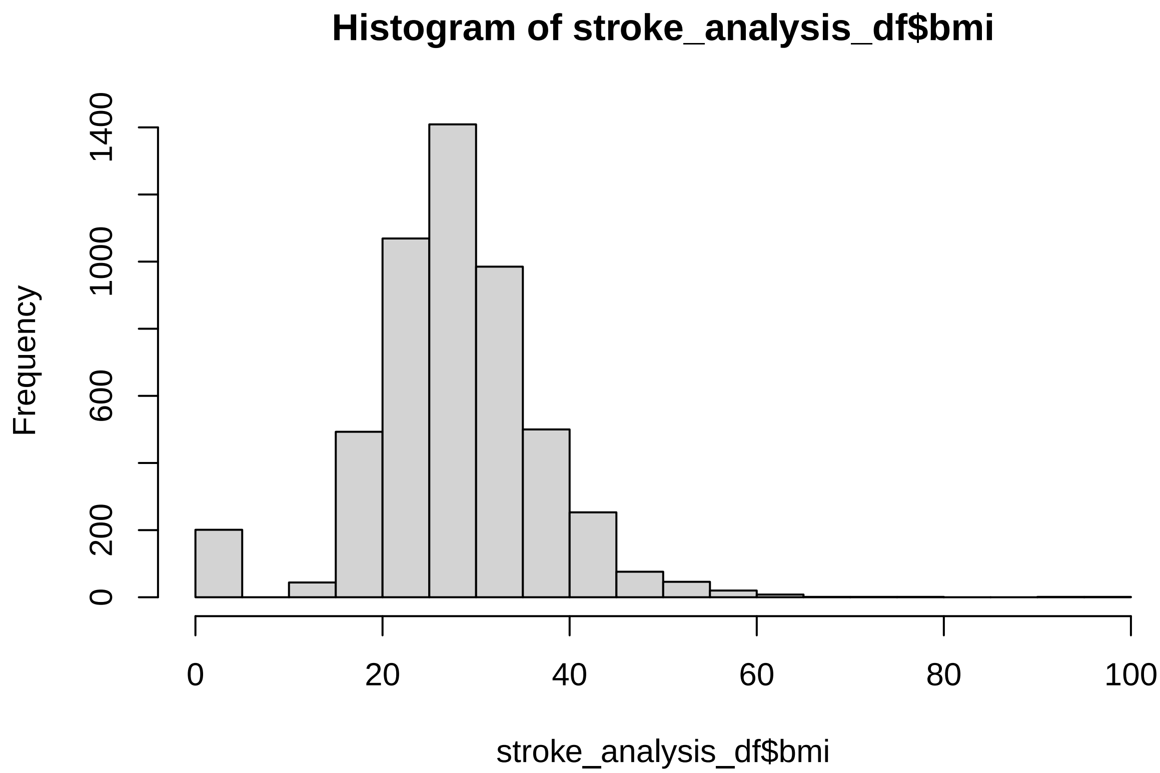

```
# Visualizing Heart Disease Incidence with a Histogram
hist(stroke_analysis_df$heart_disease)
```



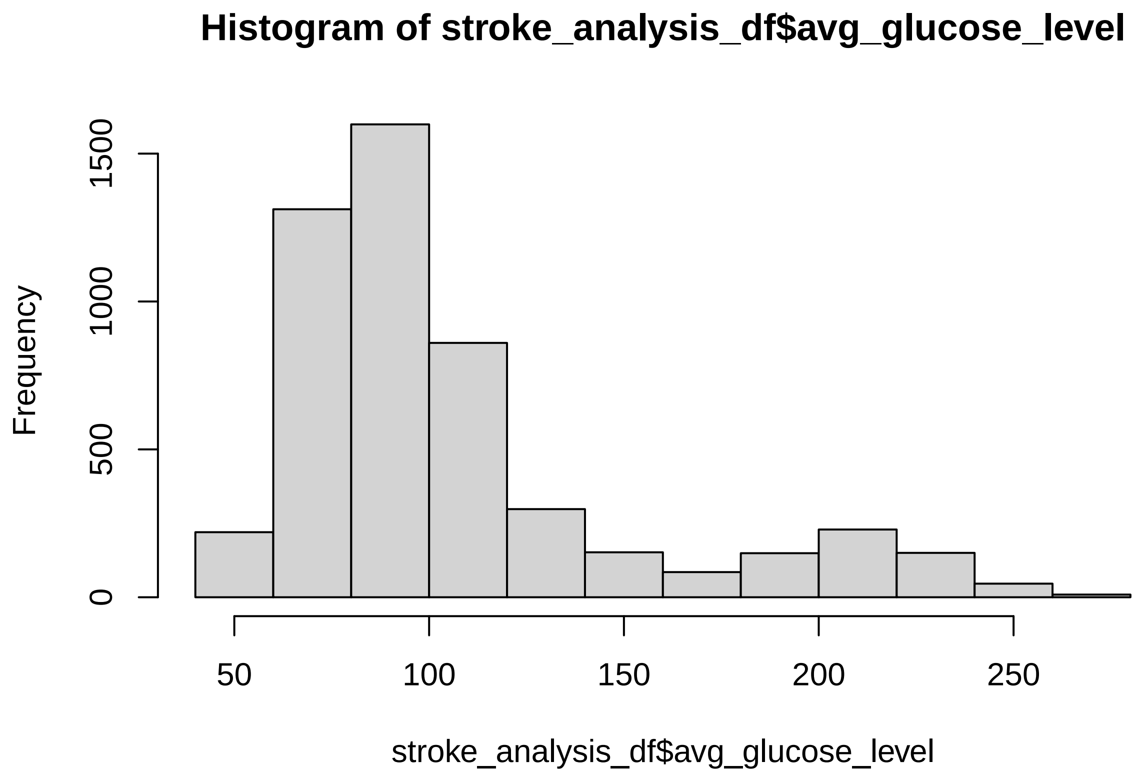

```r
# Histogram Analysis of Stroke Incidence
hist(stroke_analysis_df$stroke)
```




```r
# BMI Distribution Visualized Through a Histogram
hist(stroke_analysis_df$bmi)
```




```r
# Analyzing Average Glucose Level Distribution via Histogram
hist(stroke_analysis_df$avg_glucose_level)
```

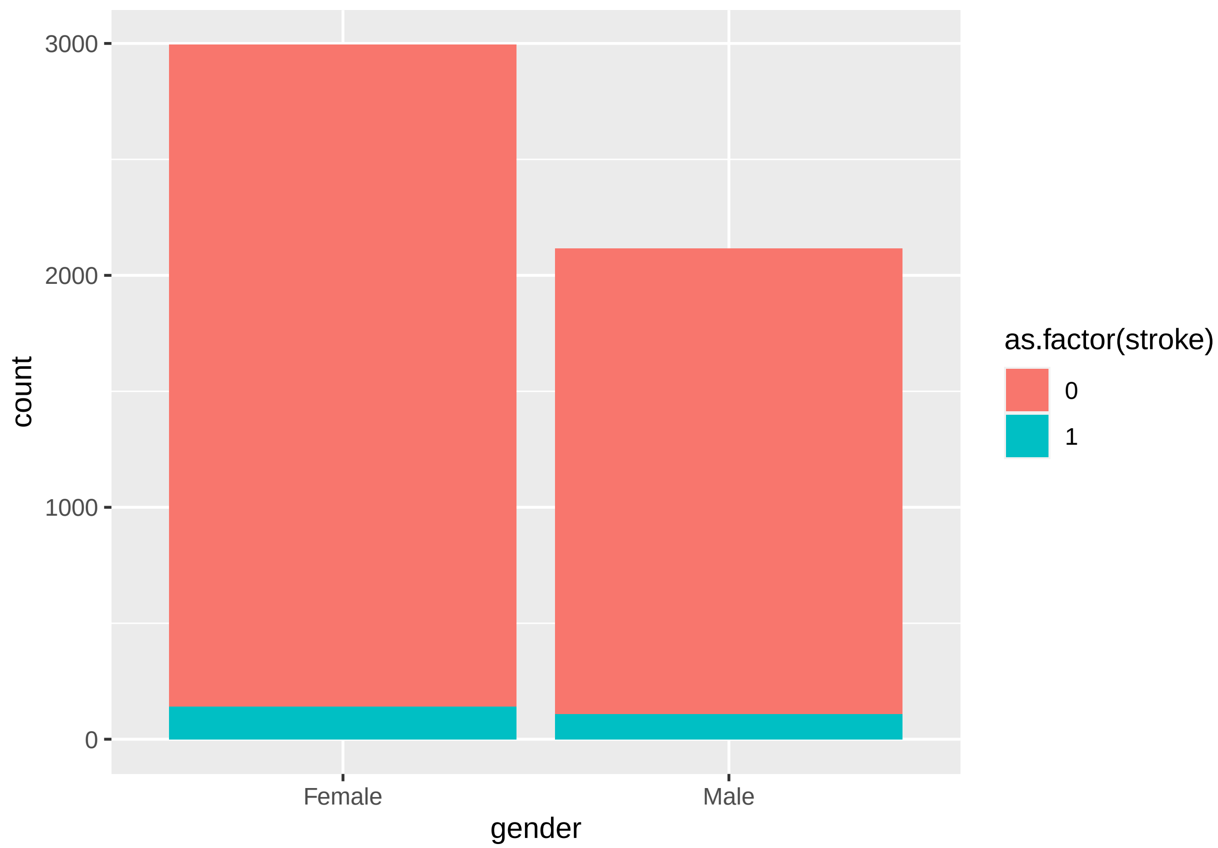


We have looked at these visualizations before,

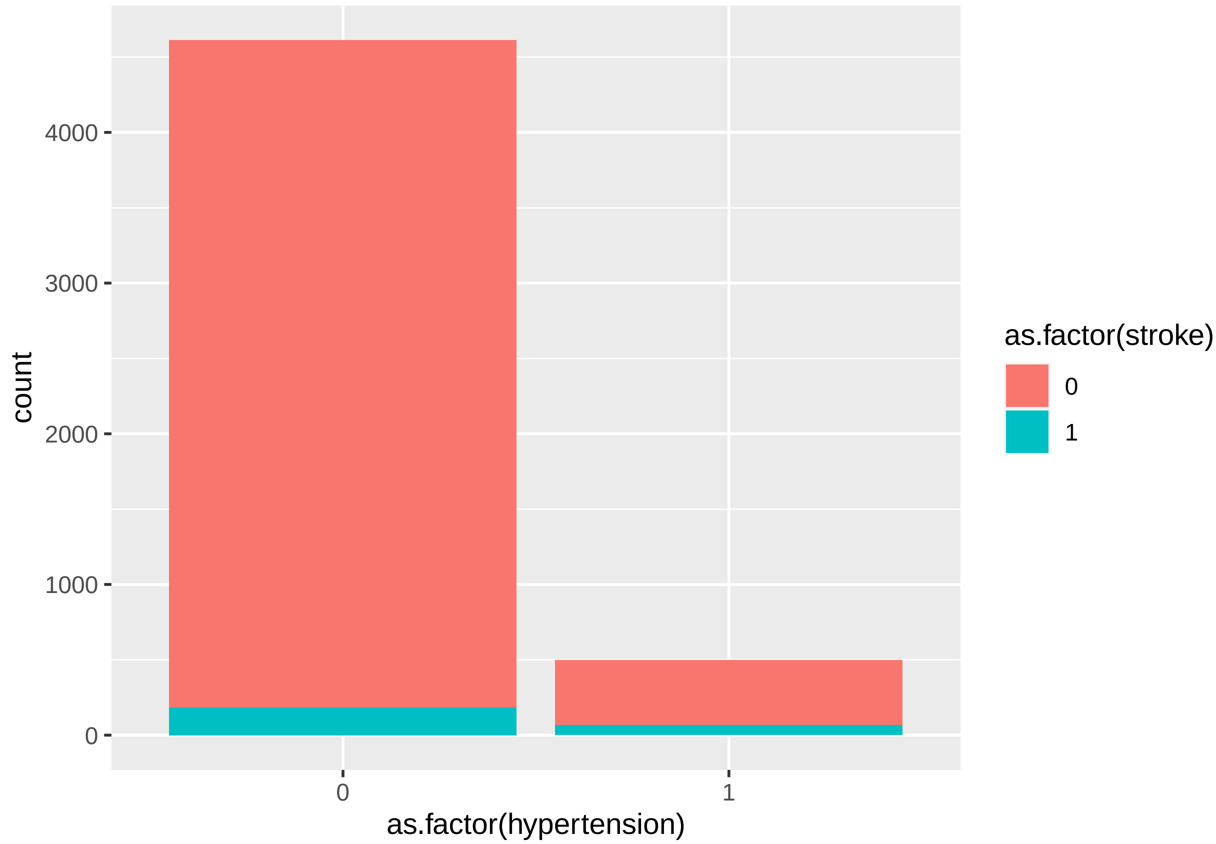
Examining the age histogram reveals a varied age distribution within the dataset. The hypertension histogram indicates that hypertension is relatively uncommon among the participants. Similarly, the histogram for heart disease suggests that few individuals in the sample have a history of heart disease. A significant imbalance is observed in the target variable, stroke, suggesting that sensitivity/recall should be a primary focus for assessing the model's reliability. The BMI histogram displays a broad range of values, indicating good diversity. Lastly, the histogram for glucose levels also shows a broad spectrum of values.

Let's look at some visualizations between variables -

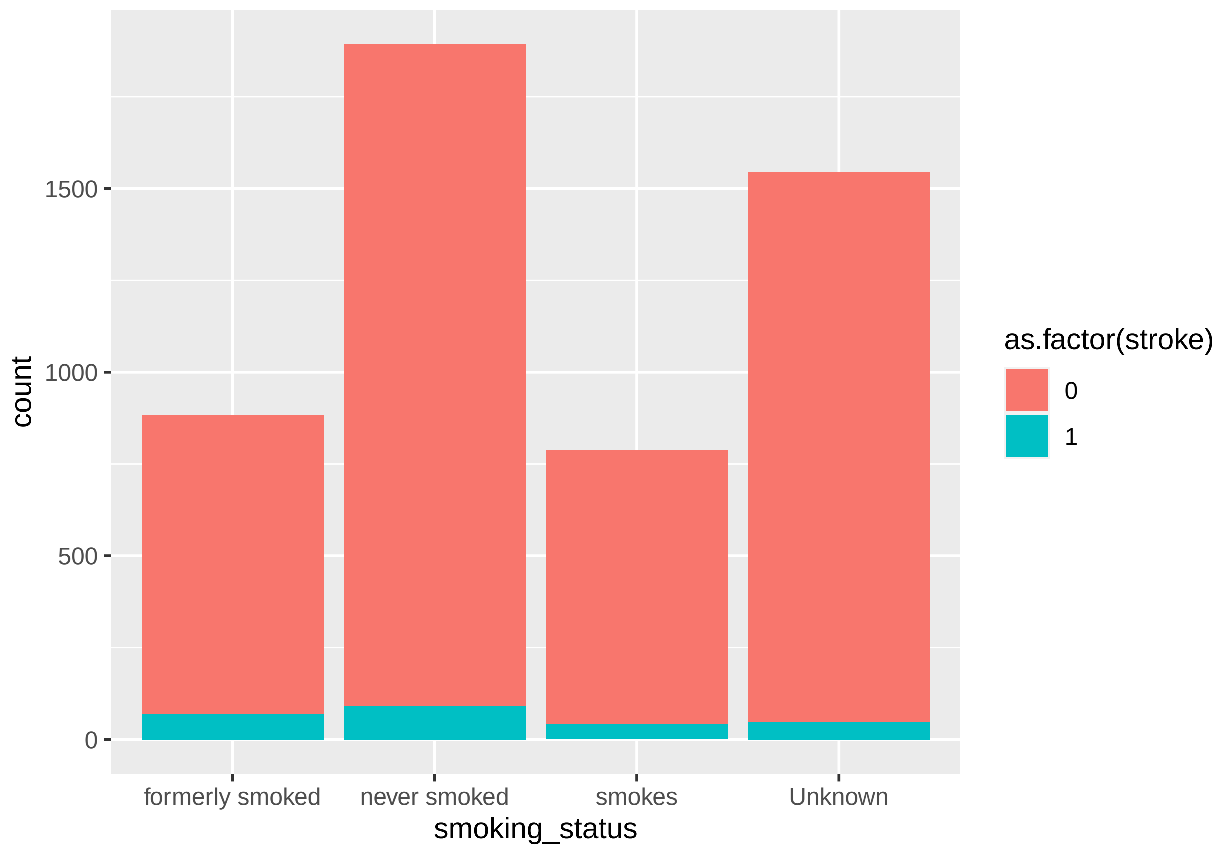
```
# Bar Chart Comparing Gender Distribution with Stroke Incidence
ggplot(stroke_analysis_df, aes(x=gender, fill=as.factor(stroke))) +
  geom_bar(aes(y=after_stat(count)))
```



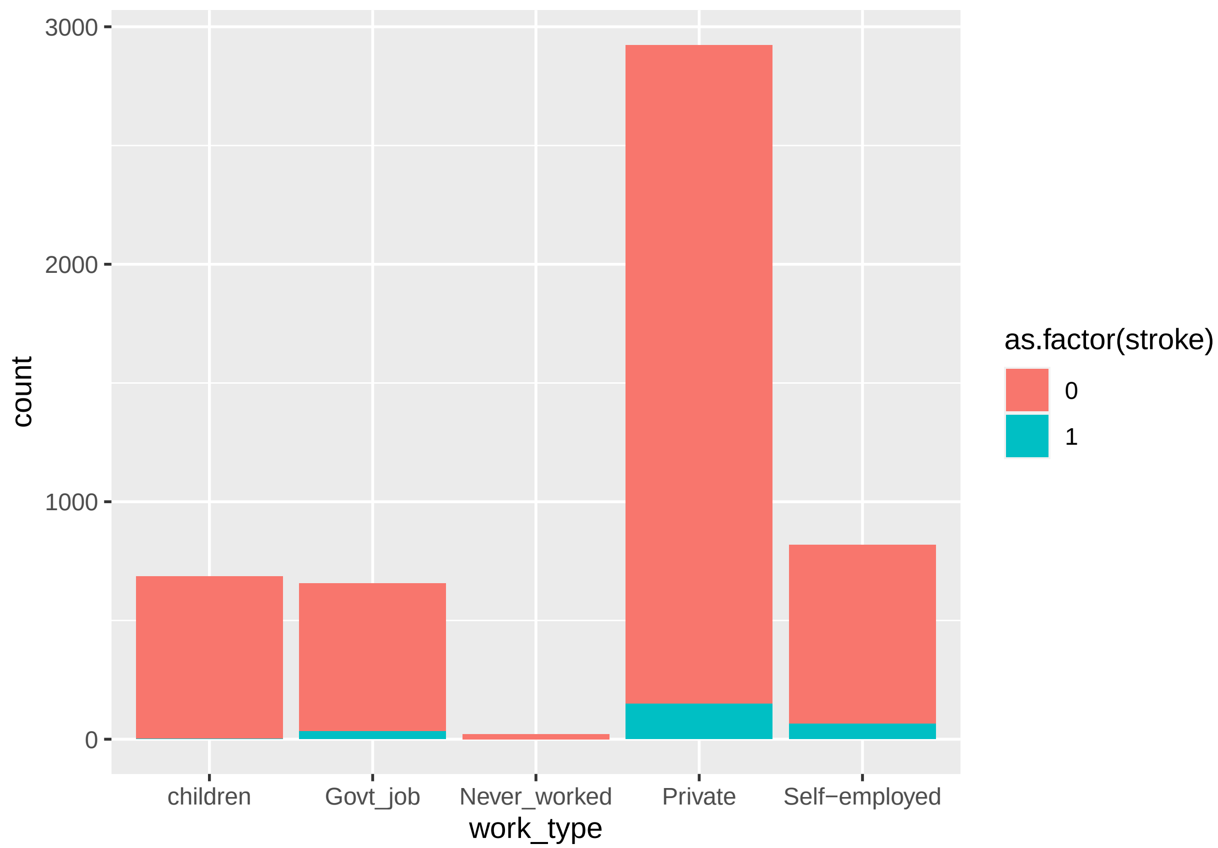

```r
# Bar Chart Showing Relationship Between Hypertension and Stroke
ggplot(stroke_analysis_df, aes(x=as.factor(hypertension), fill=as.factor(stroke))) +
  geom_bar(aes(y=after_stat(count)))
```




```
# Analyzing the Link Between Smoking Status and Stroke Through a Bar Chart
ggplot(stroke_analysis_df, aes(x=smoking_status, fill=as.factor(stroke))) +
  geom_bar(aes(y=after_stat(count)))
```




```r
# Exploring the Impact of Work Type on Stroke Occurrence with a Bar Chart
ggplot(stroke_analysis_df, aes(x=work_type, fill=as.factor(stroke))) +
  geom_bar(aes(y=after_stat(count)))
```



Despite some challenges in interpretation due to significant imbalances, the following insights were gleaned from the visualizations:

- Analyzing the bar graph correlating gender with stroke incidence reveals that gender does not significantly influence stroke prediction.
- The relationship between hypertension and stroke appears more pronounced, suggesting a potential link between the two.
- The analysis of smoking status indicates a surprisingly higher likelihood of stroke among individuals who have never smoked.
- Observations from the work type graph show that individuals employed in the private sector seem to have a higher risk of experiencing a stroke.

c. Data Cleaning

Missing Values -

Let's look at the summary statistics of our data

```
# Generating a Comprehensive Summary of the Stroke Dataset
summary(stroke_analysis_df)
```

```
##     gender          age         hypertension     heart_disease     ever_married
##  Female:2994   Min.   : 0.08   Min.   :0.00000   Min.   :0.00000   No :1756
##  Male  :2115   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000   Yes:3353
##                Median :45.00   Median :0.00000   Median :0.00000
```

```
##                  Mean   :43.23   Mean   :0.09748   Mean   :0.05402
##                  3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000
##                  Max.   :82.00   Max.   :1.00000   Max.   :1.00000
##          work_type    Residence_type avg_glucose_level      bmi
##  children     : 687   Rural:2513     Min.   : 55.12    Min.   : 0.00
##  Govt_job     : 657   Urban:2596     1st Qu.: 77.24    1st Qu.:22.90
##  Never_worked :  22                  Median : 91.88    Median :27.70
##  Private      :2924                  Mean   :106.14    Mean   :27.76
##  Self-employed: 819                  3rd Qu.:114.09    3rd Qu.:32.80
##                                      Max.   :271.74    Max.   :97.60
##          smoking_status     stroke
##  formerly smoked: 884   Min.   :0.00000
##  never smoked   :1892   1st Qu.:0.00000
##  smokes         : 789   Median :0.00000
##  Unknown        :1544   Mean   :0.04874
##                         3rd Qu.:0.00000
##                         Max.   :1.00000
```

The dataset's comprehensive review through the summary function confirms that it is currently free of missing values. Earlier, there were some missing values in the BMI data, which have been addressed by substituting them with 0.

Transforming Variables -

Let's look at the datatypes of our dataset

```r
# Displaying Data Structure and Types of Each Column in the Dataset
str(stroke_analysis_df)
```

```
## 'data.frame':    5109 obs. of  11 variables:
##  $ gender           : Factor w/ 2 levels "Female","Male": 2 1 2 1 1 2 2 1 1 1 ...
##  $ 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     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4 4 ...
##  $ 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 0 32.5 34.4 24 29 27.4 22.8 0 24.2 ...
##  $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...
##  $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

The majority of the categorical variables in the dataset have already been transformed for visualization purposes. Next, we plan to also convert the 'hypertension' and 'heart_disease' variables into factors. However, at this stage, we will not be converting the 'stroke' variable into a factor.

```r
# Creating a Cleaned Version of the Stroke Data Frame
stroke_data_cleaned <- stroke_analysis_df

# Converting 'Hypertension' Column to Factor for Categorical Analysis
stroke_data_cleaned$hypertension <- as.factor(stroke_data_cleaned$hypertension)

# Changing 'Heart Disease' Column to Factor Type
stroke_data_cleaned$heart_disease <- as.factor(stroke_data_cleaned$heart_disease)
```

Outliers -

We previously conducted an outlier analysis and identified outliers in the 'gender' and 'smoking_status' variables. To address this, we removed the outliers in the 'gender' category from the dataset. However, we decided to retain the outliers in 'smoking_status', as eliminating them would lead to a significant imbalance in the dataset. Apart from those in 'smoking_status', the dataset was found to be free of outliers.

```r
# Evaluating Unique Values in Categorical Columns to Identify Outliers
unique(stroke_data_cleaned$gender)
```

```
## [1] Male   Female
## Levels: Female Male
```

```r
unique(stroke_data_cleaned$hypertension)
```

```
## [1] 0 1
## Levels: 0 1
```

```r
unique(stroke_data_cleaned$heart_disease)
```

```
## [1] 1 0
## Levels: 0 1
```

```r
unique(stroke_data_cleaned$ever_married)
```

```
## [1] Yes No
## Levels: No Yes
```

```r
unique(stroke_data_cleaned$work_type)
```

```
## [1] Private       Self-employed Govt_job      children      Never_worked
## Levels: children Govt_job Never_worked Private Self-employed
```

```r
unique(stroke_data_cleaned$Residence_type)
```

```
## [1] Urban Rural
## Levels: Rural Urban
```

```r
unique(stroke_data_cleaned$smoking_status)
```

```
## [1] formerly smoked never smoked    smokes          Unknown
## Levels: formerly smoked never smoked smokes Unknown
```

We have confirmed that are data is clean enough to move forward

d. Data Preprocessing

Dummy Variables -

Converting the categorical variables into dummy variables -

```r
# Creating Dummy Variables for All Categorical Columns
dummy_model <- dummyVars(~., data = stroke_data_cleaned)

# Applying the Dummy Variable Transformation to the Dataset
stroke_data_dummies <- as.data.frame(predict(dummy_model, newdata = stroke_data_cleaned))

# Generating a Summary of the Transformed Data with Dummy Variables
summary(stroke_data_dummies)
```

```
##  gender.Female    gender.Male         age        hypertension.0
##  Min.   :0.000   Min.   :0.000   Min.   : 0.08   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:25.00   1st Qu.:1.0000
##  Median :1.000   Median :0.000   Median :45.00   Median :1.0000
##  Mean   :0.586   Mean   :0.414   Mean   :43.23   Mean   :0.9025
##  3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:61.00   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.000   Max.   :82.00   Max.   :1.0000
##  hypertension.1    heart_disease.0 heart_disease.1   ever_married.No
##  Min.   :0.00000   Min.   :0.000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :1.000   Median :0.00000   Median :0.0000
##  Mean   :0.09748   Mean   :0.946   Mean   :0.05402   Mean   :0.3437
##  3rd Qu.:0.00000   3rd Qu.:1.000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.000   Max.   :1.00000   Max.   :1.0000
##  ever_married.Yes work_type.children work_type.Govt_job work_type.Never_worked
##  Min.   :0.0000   Min.   :0.0000     Min.   :0.0000     Min.   :0.000000
##  1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.:0.0000     1st Qu.:0.000000
##  Median :1.0000   Median :0.0000     Median :0.0000     Median :0.000000
##  Mean   :0.6563   Mean   :0.1345     Mean   :0.1286     Mean   :0.004306
##  3rd Qu.:1.0000   3rd Qu.:0.0000     3rd Qu.:0.0000     3rd Qu.:0.000000
##  Max.   :1.0000   Max.   :1.0000     Max.   :1.0000     Max.   :1.000000
##  work_type.Private work_type.Self-employed Residence_type.Rural
##  Min.   :0.0000    Min.   :0.0000          Min.   :0.0000
##  1st Qu.:0.0000    1st Qu.:0.0000          1st Qu.:0.0000
##  Median :1.0000    Median :0.0000          Median :0.0000
##  Mean   :0.5723    Mean   :0.1603          Mean   :0.4919
##  3rd Qu.:1.0000    3rd Qu.:0.0000          3rd Qu.:1.0000
##  Max.   :1.0000    Max.   :1.0000          Max.   :1.0000
##  Residence_type.Urban avg_glucose_level      bmi
##  Min.   :0.0000       Min.   : 55.12    Min.   : 0.00
##  1st Qu.:0.0000       1st Qu.: 77.24    1st Qu.:22.90
##  Median :1.0000       Median : 91.88    Median :27.70
##  Mean   :0.5081       Mean   :106.14    Mean   :27.76
##  3rd Qu.:1.0000       3rd Qu.:114.09    3rd Qu.:32.80
##  Max.   :1.0000       Max.   :271.74    Max.   :97.60
##  smoking_status.formerly smoked smoking_status.never smoked
##  Min.   :0.000                  Min.   :0.0000
##  1st Qu.:0.000                  1st Qu.:0.0000
##  Median :0.000                  Median :0.0000
##  Mean   :0.173                  Mean   :0.3703
##  3rd Qu.:0.000                  3rd Qu.:1.0000
##  Max.   :1.000                  Max.   :1.0000
##  smoking_status.smokes smoking_status.Unknown     stroke
##  Min.   :0.0000        Min.   :0.0000         Min.   :0.00000
```

```
##  1st Qu.:0.0000       1st Qu.:0.0000        1st Qu.:0.00000
##  Median :0.0000       Median :0.0000        Median :0.00000
##  Mean   :0.1544       Mean   :0.3022        Mean   :0.04874
##  3rd Qu.:0.0000       3rd Qu.:1.0000        3rd Qu.:0.00000
##  Max.   :1.0000       Max.   :1.0000        Max.   :1.00000
```

We add dummy variables as the algorithms perform better with numerical variables

Normalization:

In this dataset, there's no requirement for binning or smoothing techniques.

Normalizing the data is beneficial as it ensures that all data points are scaled to a similar range, enhancing the consistency of the dataset.

Standardization:

This technique involves adjusting all features so that they are centered around zero, typically resulting in each feature having approximately unit variance. This standardization helps in balancing the scale of different variables, which is particularly important for many machine learning algorithms.

```r
# Setting a Random Seed for Reproducibility in Data Processing
set.seed(456)

# Excluding the Target Variable for Preprocessing
stroke_data_no_target <- stroke_data_dummies[,-c(23)]

# Standardizing Data: Centering and Scaling
data_preprocessor <- preProcess(stroke_data_no_target, method = c("center", "scale"))

# Applying Standardization to the Dataset
stroke_data_standardized <- predict(data_preprocessor, stroke_data_no_target)

# Reviewing the Summary of the Standardized Dataset
summary(stroke_data_standardized)
```

```
##  gender.Female      gender.Male          age            hypertension.0
##  Min.   :-1.1897   Min.   :-0.8404   Min.   :-1.90815   Min.   :-3.0426
##  1st Qu.:-1.1897   1st Qu.:-0.8404   1st Qu.:-0.80615   1st Qu.: 0.3286
##  Median : 0.8404   Median :-0.8404   Median : 0.07827   Median : 0.3286
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.8404   3rd Qu.: 1.1897   3rd Qu.: 0.78581   3rd Qu.: 0.3286
##  Max.   : 0.8404   Max.   : 1.1897   Max.   : 1.71446   Max.   : 0.3286
##  hypertension.1    heart_disease.0   heart_disease.1   ever_married.No
##  Min.   :-0.3286   Min.   :-4.1842   Min.   :-0.2389   Min.   :-0.7236
##  1st Qu.:-0.3286   1st Qu.: 0.2389   1st Qu.:-0.2389   1st Qu.:-0.7236
##  Median :-0.3286   Median : 0.2389   Median :-0.2389   Median :-0.7236
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.3286   3rd Qu.: 0.2389   3rd Qu.:-0.2389   3rd Qu.: 1.3817
##  Max.   : 3.0426   Max.   : 0.2389   Max.   : 4.1842   Max.   : 1.3817
##  ever_married.Yes  work_type.children work_type.Govt_job work_type.Never_worked
##  Min.   :-1.3817   Min.   :-0.3941    Min.   :-0.3841    Min.   :-0.06576
##  1st Qu.:-1.3817   1st Qu.:-0.3941    1st Qu.:-0.3841    1st Qu.:-0.06576
##  Median : 0.7236   Median :-0.3941    Median :-0.3841    Median :-0.06576
##  Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.0000    Mean   : 0.00000
##  3rd Qu.: 0.7236   3rd Qu.:-0.3941    3rd Qu.:-0.3841    3rd Qu.:-0.06576
```

```
##  Max.   : 0.7236   Max.   : 2.5368    Max.   : 2.6029    Max.   :15.20467
##  work_type.Private work_type.Self-employed Residence_type.Rural
##  Min.   :-1.1567   Min.   :-0.4369         Min.   :-0.9838
##  1st Qu.:-1.1567   1st Qu.:-0.4369         1st Qu.:-0.9838
##  Median : 0.8644   Median :-0.4369         Median :-0.9838
##  Mean   : 0.0000   Mean   : 0.0000         Mean   : 0.0000
##  3rd Qu.: 0.8644   3rd Qu.:-0.4369         3rd Qu.: 1.0163
##  Max.   : 0.8644   Max.   : 2.2885         Max.   : 1.0163
##  Residence_type.Urban avg_glucose_level      bmi
##  Min.   :-1.0163      Min.   :-1.1267   Min.   :-2.912632
##  1st Qu.:-1.0163      1st Qu.:-0.6382   1st Qu.:-0.509728
##  Median : 0.9838      Median :-0.3149   Median :-0.006063
##  Mean   : 0.0000      Mean   : 0.0000   Mean   : 0.000000
##  3rd Qu.: 0.9838      3rd Qu.: 0.1755   3rd Qu.: 0.529082
##  Max.   : 0.9838      Max.   : 3.6568   Max.   : 7.328565
##  smoking_status.formerly smoked smoking_status.never smoked
##  Min.   :-0.4574               Min.   :-0.7668
##  1st Qu.:-0.4574               1st Qu.:-0.7668
##  Median :-0.4574               Median :-0.7668
##  Mean   : 0.0000               Mean   : 0.0000
##  3rd Qu.:-0.4574               3rd Qu.: 1.3038
##  Max.   : 2.1860               Max.   : 1.3038
##  smoking_status.smokes smoking_status.Unknown
##  Min.   :-0.4273       Min.   :-0.658
##  1st Qu.:-0.4273       1st Qu.:-0.658
##  Median :-0.4273       Median :-0.658
##  Mean   : 0.0000       Mean   : 0.000
##  3rd Qu.:-0.4273       3rd Qu.: 1.519
##  Max.   : 2.3397       Max.   : 1.519
```

e. Clustering

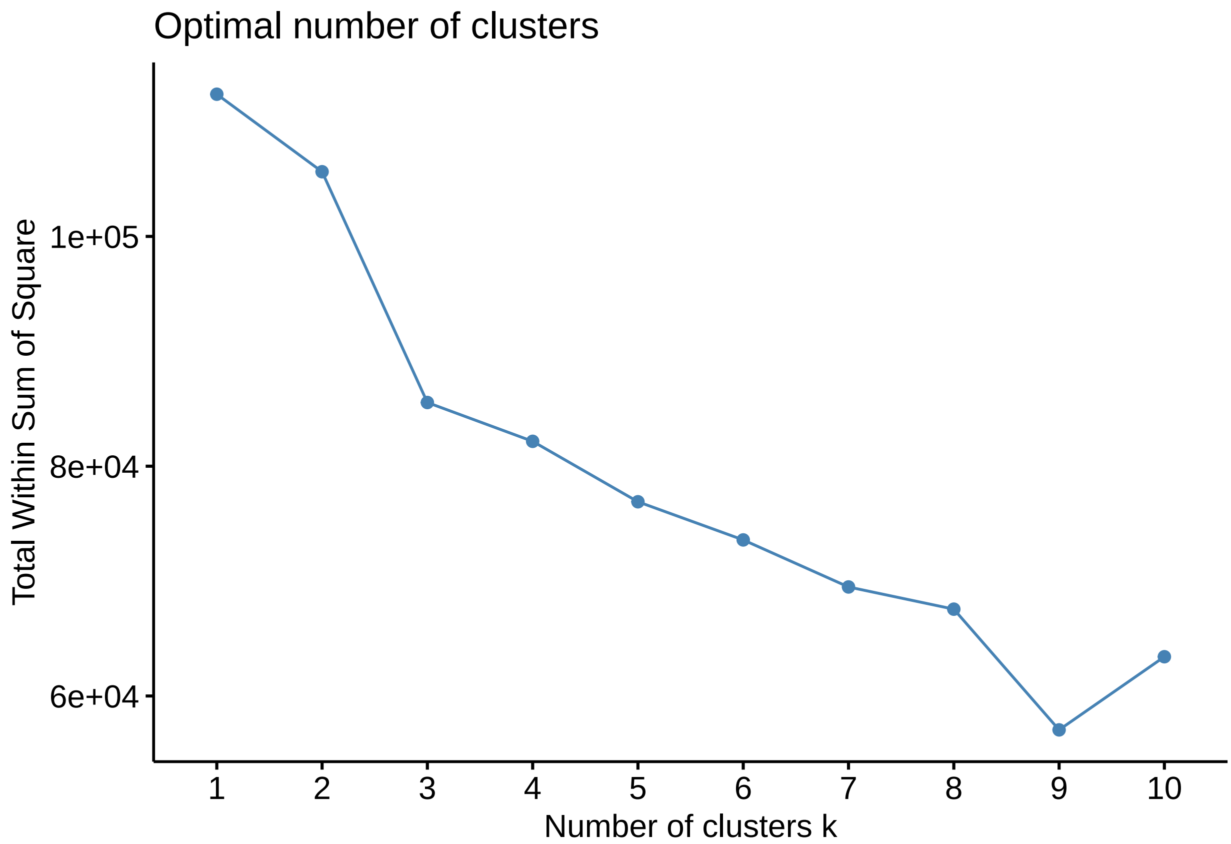
We plan to employ K-Means Clustering on our dataset, known as "stroke_data_standardized." This particular dataset is composed solely of predictor variables and does not include any class labels. It has undergone normalization, making it well-suited for clustering applications. Additionally, all categorical variables within the dataset have been transformed into dummy variables.

An essential prerequisite of K-Means clustering is pre-defining the number of clusters (k). To ascertain the optimal k value, we'll utilize two different methodologies. These methods will aid in identifying the most effective number of clusters for our analysis.

Determining the number of clusters -

1. Finding the knee

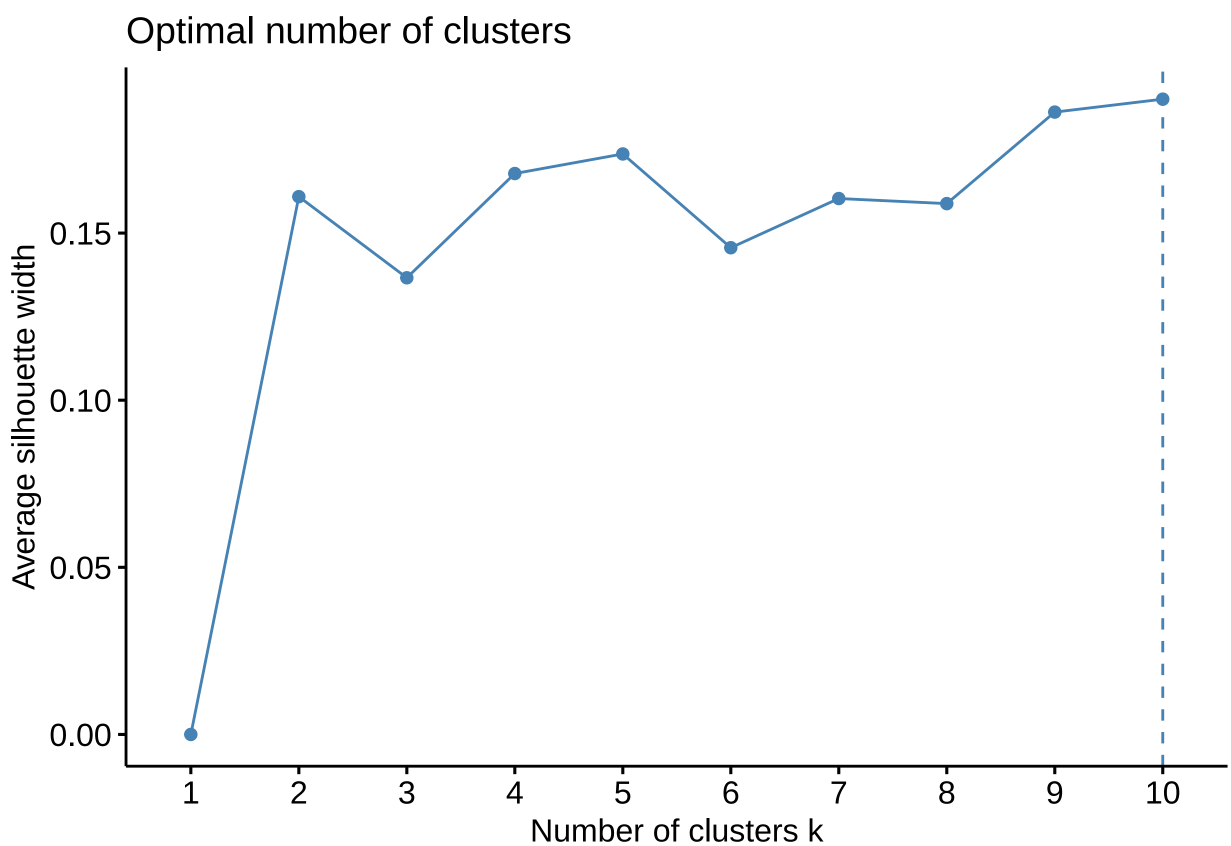
```
# Determining Optimal Number of Clusters Using the Elbow Method (Within-Sum-of-Squares)
fviz_nbclust(stroke_data_standardized, kmeans, method = "wss")
```



We observe that K = 2 and K = 3 represents the last non-flat slope The other option is comparing the average silhouette scores of different K values This technique is more straightforward because we will just be looking at the highest value

2. Silhouette

```
# Identifying the Optimal Number of Clusters Using the Silhouette Method
fviz_nbclust(stroke_data_standardized, kmeans, method = "silhouette")
```



The silhouette score suggests a K value of 10 Comparing both knee and silhouette suggestions, I will be using K = 3

Using K-means to fit the data with a k value of 3

```
# Fitting a K-Means Model with 3 Centers to the Standardized Data
kmeans_fit <- kmeans(stroke_data_standardized, centers = 3, nstart = 25)

# Displaying the Details of the Fitted K-Means Model
kmeans_fit
```

```
## K-means clustering with 3 clusters of sizes 3174, 1659, 276
##
## Cluster means:
##   gender.Female gender.Male        age hypertension.0 hypertension.1
## 1    0.05817615 -0.05817615  0.4536371     -0.1227963      0.1227963
## 2   -0.05165732  0.05165732 -1.0515148      0.3103169     -0.3103169
## 3   -0.35852031  0.35852031  1.1036919     -0.4531162      0.4531162
##   heart_disease.0 heart_disease.1 ever_married.No ever_married.Yes
## 1       0.2389481      -0.2389481      -0.6804931        0.6804931
## 2       0.2389481      -0.2389481       1.3816945       -1.3816945
## 3      -4.1841895       4.1841895      -0.4795144        0.4795144
##   work_type.children work_type.Govt_job work_type.Never_worked
## 1         -0.3941182        0.106186125            -0.06575639
## 2          0.8178281       -0.204068823             0.13674475
## 3         -0.3834989        0.005490642            -0.06575639
```

```
##   work_type.Private work_type.Self-employed Residence_type.Rural
## 1      0.1117158406              0.1306786          -0.01147982
## 2     -0.2137820400             -0.3103955           0.02408271
## 3      0.0002837914              0.3629432          -0.01274010
##   Residence_type.Urban avg_glucose_level        bmi
## 1           0.01147982        0.07210207  0.2114906
## 2          -0.02408271       -0.25065005 -0.3860131
## 3           0.01274010        0.67745095 -0.1118676
##   smoking_status.formerly smoked smoking_status.never smoked
## 1                      0.1147691                 0.09106078
## 2                     -0.2661723                -0.15897757
## 3                      0.2800823                -0.09160551
##   smoking_status.smokes smoking_status.Unknown
## 1            0.06697746             -0.2430001
## 2           -0.15879107              0.5113835
## 3            0.18423165             -0.2793588
##
## Clustering vector:
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
##    3    1    3    1    1    1    3    2    1    1    1    3    1    3    3    1
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
##    3    1    2    3    1    1    1    3    1    1    1    1    3    1    1    1
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
##    1    3    2    3    1    1    2    1    1    1    3    1    1    1    3    1
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
##    1    1    2    1    3    1    1    3    1    1    1    1    1    1    1    1
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
##    1    1    1    1    1    1    1    1    1    1    1    3    1    1    1    1
##   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95   96
##    1    1    2    1    1    1    3    3    1    1    3    1    2    1    1    1
##   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112
##    1    3    1    1    3    1    1    3    1    1    1    1    1    1    3    3
##  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128
##    1    1    2    3    1    1    2    1    1    1    1    1    1    1    1    1
##  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144
##    1    1    1    1    1    1    1    1    1    1    1    1    3    1    1    3
##  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160
##    1    1    1    1    1    3    1    3    1    1    1    3    1    1    1    1
##  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176
##    1    3    2    1    1    1    1    1    1    1    1    3    1    1    1    1
##  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192
##    1    1    1    1    1    1    1    2    3    1    3    3    1    3    1    1
##  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208
##    1    3    1    1    1    1    1    1    1    3    1    1    1    1    1    1
##  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224
##    1    1    1    1    1    1    1    3    1    3    3    3    3    1    1    1
##  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240
##    2    3    1    1    1    1    1    1    1    1    1    1    1    1    1    3
##  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256
##    1    1    1    3    1    2    1    1    1    2    1    2    1    2    1    1
##  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272
##    3    1    1    1    3    1    1    1    2    2    1    1    3    1    1    1
##  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288
##    1    1    1    1    2    1    1    1    1    1    2    1    2    1    2    1
```

```
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304
##   2   1   2   2   2   1   2   1   1   1   1   1   1   1   1   1
## 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   1   1   2   1   1   1   1   1   1   1   1   3   1   1   1   2
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336
##   2   2   2   1   2   1   1   2   2   2   1   1   1   1   1   2
## 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352
##   1   3   1   1   1   1   3   1   1   2   1   1   2   2   3   1
## 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368
##   2   1   1   1   1   2   1   1   1   1   1   2   1   1   1   2
## 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384
##   1   2   1   2   1   1   1   1   2   2   2   1   2   1   3   2
## 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
##   2   1   1   2   1   1   1   1   1   2   2   2   1   1   1   1
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416
##   2   2   2   1   1   3   1   1   1   1   2   1   2   1   2   1
## 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
##   2   1   2   2   1   2   1   1   1   2   1   1   1   1   3   1
## 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448
##   1   1   1   1   1   2   1   1   2   2   1   2   1   2   2   1
## 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464
##   1   1   3   2   2   1   1   2   3   1   1   2   2   2   2   1
## 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
##   1   1   1   1   1   1   1   2   1   1   2   1   2   1   1   1
## 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496
##   2   1   3   2   1   1   2   1   1   1   2   1   1   2   1   1
## 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512
##   2   2   2   2   1   1   1   1   1   1   1   1   1   1   1   1
## 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528
##   1   2   2   1   1   1   1   2   1   1   1   1   1   2   1   1
## 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544
##   1   1   1   2   1   1   2   2   1   1   2   1   1   1   1   1
## 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560
##   1   2   2   1   2   1   2   1   1   1   2   2   2   1   2   1
## 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576
##   2   3   2   1   2   1   2   2   2   2   1   1   1   1   2   1
## 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592
##   2   1   2   1   1   2   1   1   2   3   2   1   1   1   1   1
## 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608
##   1   1   1   1   1   2   2   1   2   1   1   2   1   2   1   2
## 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624
##   1   2   1   1   1   1   2   1   1   1   1   1   1   1   1   1
## 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640
##   3   1   3   1   2   1   1   1   1   1   2   1   3   1   1   2
## 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
##   1   2   1   3   1   1   1   2   1   1   1   1   1   1   1   1
## 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672
##   2   2   2   1   2   1   1   1   1   1   1   2   2   1   1   3
## 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688
##   2   1   2   1   1   1   1   2   1   2   2   2   2   2   2   1
## 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704
##   1   1   1   1   2   2   1   1   2   1   1   1   1   1   1   1
## 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720
##   2   1   1   1   1   2   1   2   1   1   2   1   2   2   1   1
```

```
##  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735  736
##    1    2    2    1    1    2    1    2    2    1    2    1    1    2    1    1
##  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751  752
##    1    1    1    1    2    1    2    2    1    3    1    1    1    1    2    1
##  753  754  755  756  757  758  759  760  761  762  763  764  765  766  767  768
##    2    2    1    1    2    2    1    1    1    2    2    1    1    1    2    1
##  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
##    1    1    2    1    1    1    1    2    1    1    1    2    2    1    1    2
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
##    1    3    1    1    1    1    1    2    2    2    1    1    2    2    2    2
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
##    1    1    1    1    1    2    1    2    1    1    1    1    1    2    2    1
##  817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832
##    1    2    2    1    2    1    1    1    1    1    1    1    1    3    2    1
##  833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848
##    1    2    1    1    2    2    1    1    2    1    2    2    2    1    1    1
##  849  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864
##    1    1    2    1    2    1    1    1    2    1    2    1    1    2    1    1
##  865  866  867  868  869  870  871  872  873  874  875  876  877  878  879  880
##    1    1    2    3    1    1    1    1    2    1    1    1    1    2    2    1
##  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896
##    1    1    1    2    1    1    2    1    1    3    3    1    2    2    1    2
##  897  898  899  900  901  902  903  904  905  906  907  908  909  910  911  912
##    1    1    1    1    3    1    1    2    1    1    1    2    2    1    1    1
##  913  914  915  916  917  918  919  920  921  922  923  924  925  926  927  928
##    1    2    1    1    1    1    1    2    2    2    2    3    1    2    2    1
##  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943  944
##    1    3    1    1    2    1    1    1    2    2    2    2    1    1    3    1
##  945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960
##    1    1    1    1    1    1    2    2    2    2    2    1    2    2    1    2
##  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975  976
##    2    2    2    2    1    2    2    2    1    2    3    3    2    2    2    1
##  977  978  979  980  981  982  983  984  985  986  987  988  989  990  991  992
##    1    2    1    2    2    1    2    1    1    1    2    1    2    1    3    1
##  993  994  995  996  997  998  999 1000 1001 1002 1003 1004 1005 1006 1007 1008
##    3    1    1    2    2    1    2    2    1    1    3    1    1    2    2    1
## 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024
##    1    2    1    1    2    1    1    2    1    1    1    1    2    1    2    1
## 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040
##    1    1    1    1    1    1    1    1    2    2    1    1    1    2    1    2
## 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056
##    1    1    1    1    1    2    1    2    3    2    1    2    1    1    1    1
## 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072
##    2    1    1    1    2    1    3    2    1    1    2    3    1    2    1    1
## 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088
##    1    1    2    1    2    1    2    1    1    1    1    1    1    1    2    2
## 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104
##    2    2    2    1    1    2    2    2    1    1    1    1    1    2    2    1
## 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120
##    1    1    1    2    1    2    1    2    1    1    1    2    1    1    1    1
## 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136
##    1    1    2    2    2    1    1    1    1    1    1    1    2    1    2    1
## 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152
##    1    2    1    1    1    1    1    2    2    2    1    1    1    1    1    1
```

```
## 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168
##    1    1    2    2    1    1    1    2    1    1    1    1    1    1    1    2
## 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184
##    1    1    1    2    2    1    2    1    2    2    1    1    1    2    2    1
## 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200
##    1    2    1    1    1    3    2    1    2    3    2    2    1    2    3    1
## 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216
##    1    1    1    2    2    1    2    1    1    1    1    1    2    1    1    1
## 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232
##    1    2    2    2    2    2    2    2    1    1    1    1    1    1    1    2
## 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248
##    2    1    1    1    1    1    2    2    1    1    2    2    2    1    2    1
## 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264
##    2    2    1    2    2    2    1    2    1    1    2    1    1    2    1    1
## 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280
##    1    2    1    2    2    1    2    2    2    1    1    1    1    2    2    2
## 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296
##    1    2    3    3    1    1    3    2    1    1    1    1    2    1    1    1
## 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312
##    1    1    2    1    1    2    1    1    2    1    1    2    1    3    2    1
## 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
##    2    1    1    1    2    2    3    3    3    2    1    1    1    1    2    1
## 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344
##    1    2    1    1    1    3    1    1    2    1    2    3    1    1    1    1
## 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
##    1    1    1    1    2    2    1    1    1    1    2    1    1    1    2    2
## 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
##    1    1    1    1    2    3    2    1    1    1    2    2    1    1    1    2
## 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
##    2    2    1    1    2    1    1    1    3    1    2    1    1    2    2    1
## 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408
##    2    1    2    1    2    1    1    1    1    1    1    1    2    2    2    1
## 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424
##    2    1    1    1    1    3    3    1    1    1    1    1    1    2    2    1
## 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440
##    2    2    2    1    1    2    2    1    2    1    1    3    1    1    1    1
## 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456
##    1    1    2    1    2    2    1    1    1    2    2    1    1    1    1    2
## 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472
##    1    2    1    1    1    2    2    2    2    2    1    2    1    2    2    2
## 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488
##    1    1    1    1    1    2    1    1    1    1    1    2    2    1    2    3
## 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
##    2    1    1    1    2    2    2    1    1    1    2    2    1    1    2    3
## 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520
##    1    1    3    1    1    1    2    3    2    2    1    1    1    1    2    2
## 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536
##    1    1    1    2    1    2    3    3    1    1    1    1    1    1    1    1
## 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552
##    2    1    1    1    1    1    1    1    1    2    1    3    2    1    2    1
## 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568
##    1    1    2    2    2    1    2    1    2    3    1    1    1    2    3    2
## 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584
##    1    1    2    1    2    2    1    2    1    1    1    2    1    1    2    1
```

```
## 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600
##    1    1    1    2    1    2    2    1    1    1    1    1    2    2    1    1
## 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616
##    2    2    1    1    1    1    2    1    1    2    2    3    2    1    2    1
## 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632
##    1    1    1    2    1    1    2    2    1    1    1    1    3    2    1    2
## 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648
##    2    2    1    1    1    1    1    1    3    2    1    1    2    2    3    2
## 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664
##    2    2    1    1    1    1    3    1    1    1    1    1    2    1    1    1
## 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680
##    1    1    2    1    2    1    3    1    2    1    1    1    1    1    1    1
## 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696
##    1    1    1    2    2    1    2    1    1    1    1    1    1    2    1    1
## 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712
##    1    1    2    1    1    2    2    1    2    2    1    2    2    1    1    1
## 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728
##    2    2    2    1    1    1    3    1    2    2    1    1    2    1    2    1
## 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744
##    2    1    1    2    1    1    2    1    1    3    1    2    1    1    1    1
## 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760
##    2    1    1    1    1    1    1    1    3    1    1    1    1    1    2    1
## 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776
##    1    1    2    2    1    2    1    2    2    3    1    1    1    2    2    1
## 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792
##    1    2    1    2    1    1    1    2    1    3    1    2    1    2    1    2
## 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808
##    2    1    2    1    1    2    1    1    2    2    2    1    1    1    2    2
## 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824
##    2    2    3    2    2    1    1    1    1    1    1    1    2    2    2    1
## 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840
##    2    1    2    1    1    1    2    2    1    2    2    3    2    1    1    1
## 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856
##    1    2    1    1    2    2    2    1    1    2    1    2    1    1    1    1
## 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872
##    1    1    1    1    1    1    3    2    1    1    2    1    1    2    1    1
## 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888
##    1    1    1    3    1    2    1    1    1    1    1    1    1    2    2    1
## 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904
##    2    1    1    1    1    3    2    1    2    2    1    2    2    2    2    1
## 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920
##    2    1    1    2    1    1    2    1    1    1    1    1    1    2    1    2
## 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936
##    3    1    2    1    1    1    1    2    2    2    1    1    1    1    1    1
## 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952
##    2    1    1    1    2    2    2    2    1    1    1    2    2    1    2    1
## 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968
##    1    3    3    1    2    2    2    2    2    1    1    2    1    2    1    1
## 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984
##    1    1    2    1    1    2    1    2    2    1    1    2    2    1    2    1
## 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
##    3    2    1    2    1    1    2    1    3    1    2    1    1    1    1    2
## 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016
##    1    1    1    1    1    1    2    1    2    2    1    1    2    2    1    2
```

```
## 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032
##    1    1    1    2    2    1    2    2    2    1    1    1    1    1    2    2
## 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048
##    1    1    2    1    2    1    1    1    1    2    2    2    1    2    1    1
## 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064
##    1    1    2    1    1    1    1    1    2    1    1    1    1    1    2    1
## 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080
##    1    2    1    1    1    1    1    1    1    2    2    1    2    1    1    2
## 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096
##    2    1    3    1    2    1    3    1    1    2    2    2    1    1    1    1
## 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112
##    1    2    1    1    1    2    3    1    1    2    1    1    2    2    3    1
## 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128
##    2    2    1    1    2    2    1    1    2    1    3    2    1    1    1    1
## 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144
##    1    1    1    2    3    2    2    1    1    2    1    1    2    1    1    1
## 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160
##    1    1    2    2    2    1    1    1    2    1    2    1    1    2    1    1
## 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176
##    1    1    1    2    1    2    1    1    1    2    1    2    1    1    1    2
## 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192
##    1    2    2    1    1    2    1    2    1    3    2    1    3    1    1    2
## 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208
##    1    2    2    2    1    1    1    2    1    3    3    1    3    1    1    1
## 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224
##    1    1    1    1    1    2    1    3    2    2    1    1    1    3    2    2
## 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240
##    1    1    1    1    1    1    2    1    1    1    1    1    1    1    1    1
## 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256
##    2    2    2    1    1    2    1    2    2    2    3    1    1    2    2    3
## 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272
##    1    1    2    2    1    1    2    2    1    1    1    2    2    1    1    1
## 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288
##    1    1    1    1    2    1    1    1    1    1    1    3    1    1    1    1
## 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304
##    2    2    1    2    1    1    1    1    1    1    1    1    2    1    1    1
## 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320
##    2    2    2    1    2    1    2    2    1    2    1    1    1    1    1    1
## 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336
##    1    3    1    2    1    1    2    1    1    2    1    1    3    2    2    1
## 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352
##    1    2    1    1    2    1    1    1    2    1    2    2    2    2    1    1
## 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368
##    1    2    2    1    1    1    2    1    3    1    3    1    2    2    1    1
## 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384
##    2    1    2    2    1    1    2    1    1    1    1    2    1    1    1    1
## 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400
##    2    2    2    2    2    2    1    2    1    1    2    1    1    2    1    1
## 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416
##    1    2    1    2    2    1    1    1    1    1    1    2    2    1    1    1
## 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432
##    1    1    3    1    1    1    1    2    1    2    2    2    2    2    1    1
## 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448
##    2    3    1    1    2    2    1    1    1    1    1    2    1    1    1    2
```

```
## 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464
##    2    1    2    1    2    1    1    2    1    2    1    1    1    1    1    2
## 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480
##    1    2    1    1    1    1    1    2    2    1    2    1    1    1    2    2
## 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496
##    2    2    1    1    2    2    1    1    2    1    2    1    1    1    3    1
## 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512
##    1    3    2    1    1    1    3    1    2    1    1    2    1    1    1    2
## 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528
##    1    2    2    3    2    1    1    3    2    3    1    3    1    2    2    3
## 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544
##    2    1    2    2    3    1    1    2    2    1    2    1    1    3    2    2
## 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560
##    3    1    2    1    1    1    1    1    2    1    2    1    1    1    1    1
## 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576
##    2    1    1    1    2    1    2    1    1    1    1    1    1    2    2    2
## 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592
##    3    1    1    2    1    1    1    3    1    1    2    1    2    1    2    1
## 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608
##    3    2    1    2    1    1    1    1    1    1    1    1    1    2    1    2
## 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624
##    1    1    1    1    1    2    1    1    1    1    1    2    2    1    1    1
## 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640
##    1    1    1    1    2    1    2    1    3    1    2    1    1    3    1    1
## 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656
##    1    1    1    1    2    1    1    1    1    2    1    1    2    2    1    1
## 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672
##    1    1    1    2    2    1    2    2    1    2    1    2    1    2    1    2
## 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688
##    2    1    1    1    1    1    1    2    1    2    1    1    1    1    1    2
## 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704
##    1    1    3    2    2    1    2    1    1    1    1    2    1    1    1    2
## 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717 2718 2719 2720
##    1    2    1    1    2    3    1    1    1    1    2    1    2    2    3    2
## 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736
##    3    2    1    2    1    2    2    1    2    2    2    1    1    2    1    1
## 2737 2738 2739 2740 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2752
##    2    1    1    2    2    1    1    2    2    2    1    2    1    1    1    1
## 2753 2754 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768
##    1    2    1    2    2    1    1    1    2    1    2    1    1    1    1    1
## 2769 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780 2781 2782 2783 2784
##    1    3    1    1    1    1    1    1    2    2    1    2    1    2    1    2
## 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795 2796 2797 2798 2799 2800
##    1    1    1    1    2    2    2    1    2    1    2    1    2    1    1    3
## 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815 2816
##    3    2    1    2    1    2    1    1    2    1    1    2    2    1    1    2
## 2817 2818 2819 2820 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830 2831 2832
##    1    1    3    2    2    3    1    1    1    1    1    2    1    1    1    1
## 2833 2834 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845 2846 2847 2848
##    2    1    2    1    1    1    1    2    1    1    2    2    1    2    1    1
## 2849 2850 2851 2852 2853 2854 2855 2856 2857 2858 2859 2860 2861 2862 2863 2864
##    1    1    2    1    2    2    2    1    2    1    2    1    1    2    2    1
## 2865 2866 2867 2868 2869 2870 2871 2872 2873 2874 2875 2876 2877 2878 2879 2880
##    1    1    1    3    1    1    1    2    1    1    2    2    1    1    1    2
```

```
## 2881 2882 2883 2884 2885 2886 2887 2888 2889 2890 2891 2892 2893 2894 2895 2896
##    3    1    1    1    1    2    1    2    1    2    1    3    2    1    1    1
## 2897 2898 2899 2900 2901 2902 2903 2904 2905 2906 2907 2908 2909 2910 2911 2912
##    1    3    2    1    1    2    3    1    1    1    1    1    1    2    1    2
## 2913 2914 2915 2916 2917 2918 2919 2920 2921 2922 2923 2924 2925 2926 2927 2928
##    2    1    1    1    1    2    3    2    2    2    1    1    2    1    1    2
## 2929 2930 2931 2932 2933 2934 2935 2936 2937 2938 2939 2940 2941 2942 2943 2944
##    2    2    2    1    1    1    3    1    2    1    3    1    3    1    2    2
## 2945 2946 2947 2948 2949 2950 2951 2952 2953 2954 2955 2956 2957 2958 2959 2960
##    1    1    2    2    1    1    2    1    2    1    2    1    2    1    2    1
## 2961 2962 2963 2964 2965 2966 2967 2968 2969 2970 2971 2972 2973 2974 2975 2976
##    1    3    1    1    1    1    1    1    1    1    2    2    2    2    1    1
## 2977 2978 2979 2980 2981 2982 2983 2984 2985 2986 2987 2988 2989 2990 2991 2992
##    1    1    1    1    1    1    3    1    2    3    3    2    1    2    1    1
## 2993 2994 2995 2996 2997 2998 2999 3000 3001 3002 3003 3004 3005 3006 3007 3008
##    1    1    1    1    1    3    1    2    1    2    1    2    2    1    1    2
## 3009 3010 3011 3012 3013 3014 3015 3016 3017 3018 3019 3020 3021 3022 3023 3024
##    1    2    1    2    1    1    1    1    1    1    1    1    1    1    1    1
## 3025 3026 3027 3028 3029 3030 3031 3032 3033 3034 3035 3036 3037 3038 3039 3040
##    1    2    1    1    1    1    1    3    1    1    2    1    1    2    2    1
## 3041 3042 3043 3044 3045 3046 3047 3048 3049 3050 3051 3052 3053 3054 3055 3056
##    2    1    2    1    2    1    1    2    1    1    2    2    1    1    3    1
## 3057 3058 3059 3060 3061 3062 3063 3064 3065 3066 3067 3068 3069 3070 3071 3072
##    3    1    2    1    1    1    1    2    2    1    2    2    1    3    1    1
## 3073 3074 3075 3076 3077 3078 3079 3080 3081 3082 3083 3084 3085 3086 3087 3088
##    1    1    1    3    1    2    1    2    1    1    1    1    1    3    2    2
## 3089 3090 3091 3092 3093 3094 3095 3096 3097 3098 3099 3100 3101 3102 3103 3104
##    1    1    1    1    2    1    2    2    2    2    1    2    1    1    1    1
## 3105 3106 3107 3108 3109 3110 3111 3112 3113 3114 3115 3116 3117 3118 3119 3120
##    1    1    1    2    1    1    2    1    1    1    2    1    2    1    2    2
## 3121 3122 3123 3124 3125 3126 3127 3128 3129 3130 3131 3132 3133 3134 3135 3136
##    2    1    1    1    2    1    2    1    2    1    2    2    2    3    1    1
## 3137 3138 3139 3140 3141 3142 3143 3144 3145 3146 3147 3148 3149 3150 3151 3152
##    2    1    1    1    1    1    1    2    2    1    1    1    1    1    1    1
## 3153 3154 3155 3156 3157 3158 3159 3160 3161 3162 3163 3164 3165 3166 3167 3168
##    1    1    1    1    1    2    1    2    1    1    1    2    1    1    1    1
## 3169 3170 3171 3172 3173 3174 3175 3176 3177 3178 3179 3180 3181 3182 3183 3184
##    1    2    1    1    2    1    2    1    1    1    1    2    3    1    2    1
## 3185 3186 3187 3188 3189 3190 3191 3192 3193 3194 3195 3196 3197 3198 3199 3200
##    2    1    2    1    1    1    2    1    2    1    1    2    1    1    1    2
## 3201 3202 3203 3204 3205 3206 3207 3208 3209 3210 3211 3212 3213 3214 3215 3216
##    1    2    1    1    1    1    1    2    1    1    1    1    3    1    1    1
## 3217 3218 3219 3220 3221 3222 3223 3224 3225 3226 3227 3228 3229 3230 3231 3232
##    1    1    1    1    1    1    1    1    1    2    1    1    2    1    1    1
## 3233 3234 3235 3236 3237 3238 3239 3240 3241 3242 3243 3244 3245 3246 3247 3248
##    2    1    1    1    1    1    2    2    1    2    1    1    1    2    1    1
## 3249 3250 3251 3252 3253 3254 3255 3256 3257 3258 3259 3260 3261 3262 3263 3264
##    1    1    2    2    1    1    2    2    1    1    1    1    2    1    1    1
## 3265 3266 3267 3268 3269 3270 3271 3272 3273 3274 3275 3276 3277 3278 3279 3280
##    1    1    1    2    1    1    1    1    1    1    1    1    1    1    1    2
## 3281 3282 3283 3284 3285 3286 3287 3288 3289 3290 3291 3292 3293 3294 3295 3296
##    2    2    2    2    2    1    2    1    1    1    1    2    1    1    2    2
## 3297 3298 3299 3300 3301 3302 3303 3304 3305 3306 3307 3308 3309 3310 3311 3312
##    1    1    3    2    2    1    1    2    2    1    1    1    1    1    2    1
```

```
## 3313 3314 3315 3316 3317 3318 3319 3320 3321 3322 3323 3324 3325 3326 3327 3328
##    2    1    2    1    1    3    2    1    2    2    2    2    1    2    1    3
## 3329 3330 3331 3332 3333 3334 3335 3336 3337 3338 3339 3340 3341 3342 3343 3344
##    1    1    2    1    1    2    1    1    1    2    1    2    1    1    2    2
## 3345 3346 3347 3348 3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3359 3360
##    3    1    1    2    1    1    1    1    1    2    2    2    2    3    2    3
## 3361 3362 3363 3364 3365 3366 3367 3368 3369 3370 3371 3372 3373 3374 3375 3376
##    1    1    1    1    2    1    2    1    1    2    1    1    1    2    1    1
## 3377 3378 3379 3380 3381 3382 3383 3384 3385 3386 3387 3388 3389 3390 3391 3392
##    1    2    1    1    1    2    1    1    2    2    1    1    1    1    1    2
## 3393 3394 3395 3396 3397 3398 3399 3400 3401 3402 3403 3404 3405 3406 3407 3408
##    1    2    1    1    2    2    3    3    2    2    2    1    1    2    1    1
## 3409 3410 3411 3412 3413 3414 3415 3416 3417 3418 3419 3420 3421 3422 3423 3424
##    1    1    1    2    2    2    2    1    1    2    1    2    1    1    2    1
## 3425 3426 3427 3428 3429 3430 3431 3432 3433 3434 3435 3436 3437 3438 3439 3440
##    2    2    1    1    1    1    1    1    1    2    2    3    1    2    1    2
## 3441 3442 3443 3444 3445 3446 3447 3448 3449 3450 3451 3452 3453 3454 3455 3456
##    1    1    1    1    1    2    2    1    1    1    1    1    1    1    1    1
## 3457 3458 3459 3460 3461 3462 3463 3464 3465 3466 3467 3468 3469 3470 3471 3472
##    2    1    1    2    3    1    1    2    1    1    1    2    1    1    1    2
## 3473 3474 3475 3476 3477 3478 3479 3480 3481 3482 3483 3484 3485 3486 3487 3488
##    1    1    1    1    2    1    3    1    2    1    2    1    1    2    1    2
## 3489 3490 3491 3492 3493 3494 3495 3496 3497 3498 3499 3500 3501 3502 3503 3504
##    1    1    2    1    1    1    1    2    1    1    1    1    2    3    3    1
## 3505 3506 3507 3508 3509 3510 3511 3512 3513 3514 3515 3516 3517 3518 3519 3520
##    1    2    1    1    1    2    1    1    2    1    1    1    1    2    1    2
## 3521 3522 3523 3524 3525 3526 3527 3528 3529 3530 3531 3532 3533 3534 3535 3536
##    1    1    1    1    1    2    2    1    1    2    1    2    2    1    1    1
## 3537 3538 3539 3540 3541 3542 3543 3544 3545 3546 3547 3548 3549 3550 3551 3552
##    2    1    2    2    2    1    2    1    2    2    1    1    1    1    1    1
## 3553 3554 3555 3556 3557 3558 3559 3560 3561 3562 3563 3564 3565 3566 3567 3568
##    3    1    2    1    1    1    1    2    2    1    3    3    1    1    1    1
## 3569 3570 3571 3572 3573 3574 3575 3576 3577 3578 3579 3580 3581 3582 3583 3584
##    1    1    1    2    2    2    1    1    2    1    2    1    1    1    1    2
## 3585 3586 3587 3588 3589 3590 3591 3592 3593 3594 3595 3596 3597 3598 3599 3600
##    1    1    3    1    1    1    1    1    1    1    1    1    1    1    1    1
## 3601 3602 3603 3604 3605 3606 3607 3608 3609 3610 3611 3612 3613 3614 3615 3616
##    2    2    1    1    2    1    1    2    2    2    1    2    1    1    1    2
## 3617 3618 3619 3620 3621 3622 3623 3624 3625 3626 3627 3628 3629 3630 3631 3632
##    1    2    2    1    1    2    2    2    3    2    1    1    1    1    1    2
## 3633 3634 3635 3636 3637 3638 3639 3640 3641 3642 3643 3644 3645 3646 3647 3648
##    1    1    1    2    1    1    1    1    3    1    1    2    2    1    2    2
## 3649 3650 3651 3652 3653 3654 3655 3656 3657 3658 3659 3660 3661 3662 3663 3664
##    1    1    2    2    2    1    2    1    2    1    1    1    2    2    1    1
## 3665 3666 3667 3668 3669 3670 3671 3672 3673 3674 3675 3676 3677 3678 3679 3680
##    1    1    1    1    2    1    2    1    1    1    1    2    1    2    1    3
## 3681 3682 3683 3684 3685 3686 3687 3688 3689 3690 3691 3692 3693 3694 3695 3696
##    2    1    3    2    2    1    1    1    1    1    1    2    2    2    2    1
## 3697 3698 3699 3700 3701 3702 3703 3704 3705 3706 3707 3708 3709 3710 3711 3712
##    1    2    1    1    1    2    1    2    2    1    1    1    2    1    1    2
## 3713 3714 3715 3716 3717 3718 3719 3720 3721 3722 3723 3724 3725 3726 3727 3728
##    2    2    2    2    2    1    2    1    2    1    1    1    1    2    1    2
## 3729 3730 3731 3732 3733 3734 3735 3736 3737 3738 3739 3740 3741 3742 3743 3744
##    1    1    2    1    1    2    1    1    1    1    2    1    1    1    1    2
```

```
## 3745 3746 3747 3748 3749 3750 3751 3752 3753 3754 3755 3756 3757 3758 3759 3760
##    2    2    1    1    2    1    2    1    3    2    1    3    1    2    1    2
## 3761 3762 3763 3764 3765 3766 3767 3768 3769 3770 3771 3772 3773 3774 3775 3776
##    1    2    1    1    2    2    1    1    1    1    3    1    1    2    1    2
## 3777 3778 3779 3780 3781 3782 3783 3784 3785 3786 3787 3788 3789 3790 3791 3792
##    2    2    1    1    3    1    1    3    2    2    2    1    2    1    1    1
## 3793 3794 3795 3796 3797 3798 3799 3800 3801 3802 3803 3804 3805 3806 3807 3808
##    2    1    1    2    1    2    1    2    1    1    1    1    3    2    2    2
## 3809 3810 3811 3812 3813 3814 3815 3816 3817 3818 3819 3820 3821 3822 3823 3824
##    2    1    1    2    1    1    1    1    2    2    2    1    1    1    2    1
## 3825 3826 3827 3828 3829 3830 3831 3832 3833 3834 3835 3836 3837 3838 3839 3840
##    1    1    1    1    2    1    2    2    1    1    2    2    2    1    2    2
## 3841 3842 3843 3844 3845 3846 3847 3848 3849 3850 3851 3852 3853 3854 3855 3856
##    1    1    1    2    1    2    1    1    2    1    2    2    1    2    1    1
## 3857 3858 3859 3860 3861 3862 3863 3864 3865 3866 3867 3868 3869 3870 3871 3872
##    2    2    2    1    1    1    1    2    1    1    2    2    1    2    1    2
## 3873 3874 3875 3876 3877 3878 3879 3880 3881 3882 3883 3884 3885 3886 3887 3888
##    2    2    1    2    2    1    1    2    1    3    2    1    3    1    2    2
## 3889 3890 3891 3892 3893 3894 3895 3896 3897 3898 3899 3900 3901 3902 3903 3904
##    1    1    1    1    1    2    1    2    2    1    1    1    2    1    1    1
## 3905 3906 3907 3908 3909 3910 3911 3912 3913 3914 3915 3916 3917 3918 3919 3920
##    1    1    1    2    1    1    1    1    1    1    1    1    2    1    1    1
## 3921 3922 3923 3924 3925 3926 3927 3928 3929 3930 3931 3932 3933 3934 3935 3936
##    2    1    1    2    1    1    1    2    2    1    1    2    1    1    1    1
## 3937 3938 3939 3940 3941 3942 3943 3944 3945 3946 3947 3948 3949 3950 3951 3952
##    2    2    1    2    1    2    1    2    3    2    1    1    1    2    1    2
## 3953 3954 3955 3956 3957 3958 3959 3960 3961 3962 3963 3964 3965 3966 3967 3968
##    1    1    2    2    1    2    1    1    1    1    1    2    1    2    1    2
## 3969 3970 3971 3972 3973 3974 3975 3976 3977 3978 3979 3980 3981 3982 3983 3984
##    2    2    2    2    1    1    1    1    1    2    1    1    1    1    1    1
## 3985 3986 3987 3988 3989 3990 3991 3992 3993 3994 3995 3996 3997 3998 3999 4000
##    1    2    1    1    1    2    1    1    1    3    1    1    3    1    1    1
## 4001 4002 4003 4004 4005 4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 4016
##    2    2    2    1    1    2    2    1    2    1    1    1    1    1    1    2
## 4017 4018 4019 4020 4021 4022 4023 4024 4025 4026 4027 4028 4029 4030 4031 4032
##    1    2    2    2    2    1    1    1    2    1    1    1    1    2    1    1
## 4033 4034 4035 4036 4037 4038 4039 4040 4041 4042 4043 4044 4045 4046 4047 4048
##    1    1    2    1    1    1    2    2    2    2    2    1    1    3    1    1
## 4049 4050 4051 4052 4053 4054 4055 4056 4057 4058 4059 4060 4061 4062 4063 4064
##    3    2    1    1    2    3    3    2    1    2    1    1    1    1    2    2
## 4065 4066 4067 4068 4069 4070 4071 4072 4073 4074 4075 4076 4077 4078 4079 4080
##    2    2    1    2    1    1    1    1    1    1    2    1    1    2    2    1
## 4081 4082 4083 4084 4085 4086 4087 4088 4089 4090 4091 4092 4093 4094 4095 4096
##    1    2    1    2    3    3    2    3    1    1    2    1    1    1    1    3
## 4097 4098 4099 4100 4101 4102 4103 4104 4105 4106 4107 4108 4109 4110 4111 4112
##    1    1    2    2    1    1    2    1    2    1    1    1    1    1    1    1
## 4113 4114 4115 4116 4117 4118 4119 4120 4121 4122 4123 4124 4125 4126 4127 4128
##    1    2    2    1    1    1    1    1    1    1    3    1    1    1    1    2
## 4129 4130 4131 4132 4133 4134 4135 4136 4137 4138 4139 4140 4141 4142 4143 4144
##    1    1    2    1    2    1    1    3    1    1    1    2    1    1    1    1
## 4145 4146 4147 4148 4149 4150 4151 4152 4153 4154 4155 4156 4157 4158 4159 4160
##    1    1    3    2    3    1    2    1    1    1    1    1    2    2    2    2
## 4161 4162 4163 4164 4165 4166 4167 4168 4169 4170 4171 4172 4173 4174 4175 4176
##    1    3    1    1    3    1    1    1    1    2    1    2    2    1    1    1
```

```
## 4177 4178 4179 4180 4181 4182 4183 4184 4185 4186 4187 4188 4189 4190 4191 4192
##    1    1    1    1    1    2    1    1    1    2    1    1    1    2    1    2
## 4193 4194 4195 4196 4197 4198 4199 4200 4201 4202 4203 4204 4205 4206 4207 4208
##    2    1    2    2    3    1    1    1    1    2    2    1    1    2    1    1
## 4209 4210 4211 4212 4213 4214 4215 4216 4217 4218 4219 4220 4221 4222 4223 4224
##    1    2    3    1    2    1    2    2    1    2    1    1    1    1    1    1
## 4225 4226 4227 4228 4229 4230 4231 4232 4233 4234 4235 4236 4237 4238 4239 4240
##    1    1    2    1    2    2    1    2    2    2    2    1    1    3    1    1
## 4241 4242 4243 4244 4245 4246 4247 4248 4249 4250 4251 4252 4253 4254 4255 4256
##    1    1    2    2    2    2    2    2    1    1    2    2    1    2    2    2
## 4257 4258 4259 4260 4261 4262 4263 4264 4265 4266 4267 4268 4269 4270 4271 4272
##    2    1    1    1    2    2    1    1    1    1    1    1    2    1    1    2
## 4273 4274 4275 4276 4277 4278 4279 4280 4281 4282 4283 4284 4285 4286 4287 4288
##    2    1    2    2    2    1    1    2    1    1    1    3    1    2    1    1
## 4289 4290 4291 4292 4293 4294 4295 4296 4297 4298 4299 4300 4301 4302 4303 4304
##    1    1    1    1    2    2    1    1    1    2    1    1    1    2    2    1
## 4305 4306 4307 4308 4309 4310 4311 4312 4313 4314 4315 4316 4317 4318 4319 4320
##    1    2    1    1    1    1    1    1    1    1    1    2    1    1    2    1
## 4321 4322 4323 4324 4325 4326 4327 4328 4329 4330 4331 4332 4333 4334 4335 4336
##    1    2    2    1    1    1    1    2    2    1    1    2    2    1    1    3
## 4337 4338 4339 4340 4341 4342 4343 4344 4345 4346 4347 4348 4349 4350 4351 4352
##    2    1    1    1    2    2    1    1    1    2    1    1    1    1    1    2
## 4353 4354 4355 4356 4357 4358 4359 4360 4361 4362 4363 4364 4365 4366 4367 4368
##    1    2    1    1    1    2    2    3    2    1    2    2    1    1    1    1
## 4369 4370 4371 4372 4373 4374 4375 4376 4377 4378 4379 4380 4381 4382 4383 4384
##    1    2    1    2    2    1    3    1    1    2    2    1    2    2    2    1
## 4385 4386 4387 4388 4389 4390 4391 4392 4393 4394 4395 4396 4397 4398 4399 4400
##    2    2    1    1    1    3    2    2    2    1    1    1    1    2    1    1
## 4401 4402 4403 4404 4405 4406 4407 4408 4409 4410 4411 4412 4413 4414 4415 4416
##    1    1    2    1    1    1    2    1    2    2    3    1    2    1    2    1
## 4417 4418 4419 4420 4421 4422 4423 4424 4425 4426 4427 4428 4429 4430 4431 4432
##    1    2    1    2    2    2    2    2    2    1    1    2    1    1    1    2
## 4433 4434 4435 4436 4437 4438 4439 4440 4441 4442 4443 4444 4445 4446 4447 4448
##    2    1    2    1    1    2    2    1    1    1    1    1    1    1    1    1
## 4449 4450 4451 4452 4453 4454 4455 4456 4457 4458 4459 4460 4461 4462 4463 4464
##    1    1    1    1    1    2    2    1    1    2    1    1    1    2    1    2
## 4465 4466 4467 4468 4469 4470 4471 4472 4473 4474 4475 4476 4477 4478 4479 4480
##    2    2    1    2    2    2    1    2    1    1    1    3    1    1    1    1
## 4481 4482 4483 4484 4485 4486 4487 4488 4489 4490 4491 4492 4493 4494 4495 4496
##    1    1    1    2    1    2    2    1    1    2    1    2    1    2    2    1
## 4497 4498 4499 4500 4501 4502 4503 4504 4505 4506 4507 4508 4509 4510 4511 4512
##    1    1    1    1    2    2    2    1    1    1    2    2    2    2    1    1
## 4513 4514 4515 4516 4517 4518 4519 4520 4521 4522 4523 4524 4525 4526 4527 4528
##    1    1    1    1    2    2    1    2    1    1    1    1    1    1    2    1
## 4529 4530 4531 4532 4533 4534 4535 4536 4537 4538 4539 4540 4541 4542 4543 4544
##    3    1    1    1    1    2    2    1    1    2    1    2    1    3    3    1
## 4545 4546 4547 4548 4549 4550 4551 4552 4553 4554 4555 4556 4557 4558 4559 4560
##    2    2    1    1    2    2    1    3    1    1    2    2    2    2    1    1
## 4561 4562 4563 4564 4565 4566 4567 4568 4569 4570 4571 4572 4573 4574 4575 4576
##    3    1    1    1    1    1    2    2    2    2    1    2    1    1    1    1
## 4577 4578 4579 4580 4581 4582 4583 4584 4585 4586 4587 4588 4589 4590 4591 4592
##    1    1    2    1    2    1    1    2    2    1    2    1    1    3    1    2
## 4593 4594 4595 4596 4597 4598 4599 4600 4601 4602 4603 4604 4605 4606 4607 4608
##    2    2    1    2    1    1    1    2    1    2    2    2    1    2    1    1
```

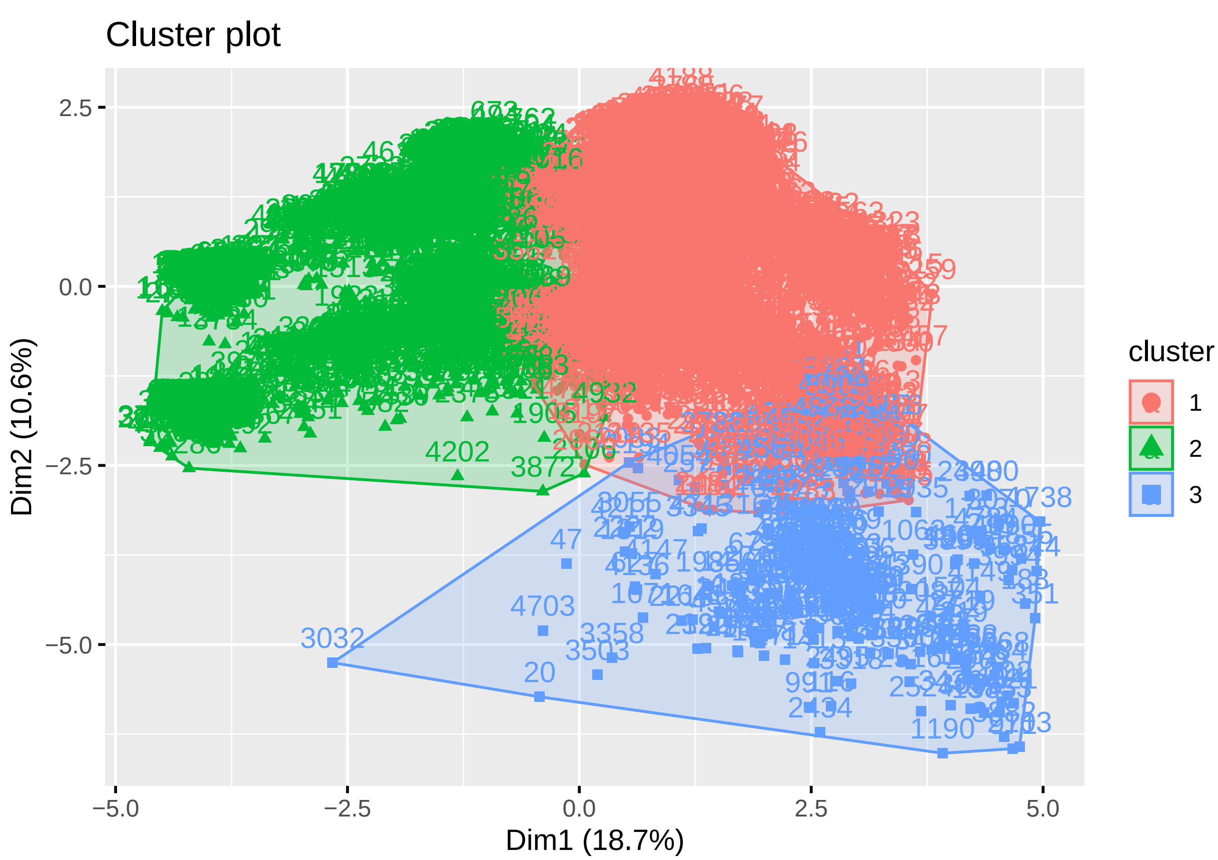
```
## 4609 4610 4611 4612 4613 4614 4615 4616 4617 4618 4619 4620 4621 4622 4623 4624
##    1    2    2    2    2    1    1    1    1    2    1    1    1    2    1    1
## 4625 4626 4627 4628 4629 4630 4631 4632 4633 4634 4635 4636 4637 4638 4639 4640
##    2    1    1    1    1    1    1    1    1    1    1    1    1    3    1    2
## 4641 4642 4643 4644 4645 4646 4647 4648 4649 4650 4651 4652 4653 4654 4655 4656
##    1    1    1    1    2    1    1    1    2    1    1    1    1    1    2    2
## 4657 4658 4659 4660 4661 4662 4663 4664 4665 4666 4667 4668 4669 4670 4671 4672
##    2    1    1    1    1    2    1    2    1    1    1    2    1    1    2    1
## 4673 4674 4675 4676 4677 4678 4679 4680 4681 4682 4683 4684 4685 4686 4687 4688
##    1    2    2    2    1    1    1    1    1    1    1    3    1    1    1    2
## 4689 4690 4691 4692 4693 4694 4695 4696 4697 4698 4699 4700 4701 4702 4703 4704
##    1    2    1    1    1    2    1    1    1    1    1    1    1    3    3    2
## 4705 4706 4707 4708 4709 4710 4711 4712 4713 4714 4715 4716 4717 4718 4719 4720
##    2    2    1    1    2    2    1    1    3    2    1    2    2    1    1    2
## 4721 4722 4723 4724 4725 4726 4727 4728 4729 4730 4731 4732 4733 4734 4735 4736
##    1    1    1    1    1    2    1    1    2    2    2    1    1    1    2    1
## 4737 4738 4739 4740 4741 4742 4743 4744 4745 4746 4747 4748 4749 4750 4751 4752
##    1    1    2    1    1    1    1    2    1    1    2    1    1    3    2    2
## 4753 4754 4755 4756 4757 4758 4759 4760 4761 4762 4763 4764 4765 4766 4767 4768
##    2    1    1    2    1    1    1    1    2    1    1    2    1    2    1    2
## 4769 4770 4771 4772 4773 4774 4775 4776 4777 4778 4779 4780 4781 4782 4783 4784
##    1    1    2    2    1    2    1    1    1    1    1    1    2    2    2    3
## 4785 4786 4787 4788 4789 4790 4791 4792 4793 4794 4795 4796 4797 4798 4799 4800
##    2    2    2    2    1    1    1    2    2    2    1    1    1    2    2    1
## 4801 4802 4803 4804 4805 4806 4807 4808 4809 4810 4811 4812 4813 4814 4815 4816
##    1    2    1    2    1    2    2    1    1    1    1    1    2    2    2    1
## 4817 4818 4819 4820 4821 4822 4823 4824 4825 4826 4827 4828 4829 4830 4831 4832
##    1    1    1    1    1    1    1    1    2    2    1    1    1    2    2    2
## 4833 4834 4835 4836 4837 4838 4839 4840 4841 4842 4843 4844 4845 4846 4847 4848
##    1    1    1    1    2    1    2    2    2    2    1    2    1    1    1    2
## 4849 4850 4851 4852 4853 4854 4855 4856 4857 4858 4859 4860 4861 4862 4863 4864
##    1    2    1    1    2    2    3    2    1    1    1    2    1    1    2    2
## 4865 4866 4867 4868 4869 4870 4871 4872 4873 4874 4875 4876 4877 4878 4879 4880
##    2    1    1    2    3    2    2    2    1    1    1    1    1    2    2    1
## 4881 4882 4883 4884 4885 4886 4887 4888 4889 4890 4891 4892 4893 4894 4895 4896
##    1    1    1    1    1    2    2    1    1    1    1    1    2    2    1    1
## 4897 4898 4899 4900 4901 4902 4903 4904 4905 4906 4907 4908 4909 4910 4911 4912
##    1    2    1    2    2    1    2    2    1    1    1    1    1    2    2    1
## 4913 4914 4915 4916 4917 4918 4919 4920 4921 4922 4923 4924 4925 4926 4927 4928
##    2    1    2    1    2    1    1    1    3    1    2    1    1    1    2    2
## 4929 4930 4931 4932 4933 4934 4935 4936 4937 4938 4939 4940 4941 4942 4943 4944
##    2    2    1    2    1    2    2    2    1    1    1    2    1    1    2    2
## 4945 4946 4947 4948 4949 4950 4951 4952 4953 4954 4955 4956 4957 4958 4959 4960
##    2    1    1    1    1    2    2    1    1    1    1    1    2    1    1    1
## 4961 4962 4963 4964 4965 4966 4967 4968 4969 4970 4971 4972 4973 4974 4975 4976
##    2    1    1    2    2    1    1    2    1    2    1    1    1    1    1    2
## 4977 4978 4979 4980 4981 4982 4983 4984 4985 4986 4987 4988 4989 4990 4991 4992
##    2    2    2    2    2    2    1    2    1    3    1    1    2    2    1    1
## 4993 4994 4995 4996 4997 4998 4999 5000 5001 5002 5003 5004 5005 5006 5007 5008
##    1    1    1    1    2    1    2    2    1    1    2    2    2    1    1    3
## 5009 5010 5011 5012 5013 5014 5015 5016 5017 5018 5019 5020 5021 5022 5023 5024
##    1    1    1    1    1    2    1    1    1    1    1    2    1    2    2    1
## 5025 5026 5027 5028 5029 5030 5031 5032 5033 5034 5035 5036 5037 5038 5039 5040
##    1    2    1    2    1    1    2    2    2    1    1    1    2    1    2    1
```

```
## 5041 5042 5043 5044 5045 5046 5047 5048 5049 5050 5051 5052 5053 5054 5055 5056
##    1    2    2    1    1    1    1    1    1    2    1    2    1    1    2    1
## 5057 5058 5059 5060 5061 5062 5063 5064 5065 5066 5067 5068 5069 5070 5071 5072
##    1    2    1    1    1    1    1    1    1    2    2    1    2    2    1    1
## 5073 5074 5075 5076 5077 5078 5079 5080 5081 5082 5083 5084 5085 5086 5087 5088
##    2    1    1    1    2    2    2    1    1    1    2    1    1    1    1    1
## 5089 5090 5091 5092 5093 5094 5095 5096 5097 5098 5099 5100 5101 5102 5103 5104
##    2    2    1    1    1    2    2    1    1    2    1    1    1    1    2    2
## 5105 5106 5107 5108 5109
##    1    1    1    1    1
##
## Within cluster sum of squares by cluster:
## [1] 53416.104 26498.175  5628.792
##  (between_SS / total_SS =  23.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

To understand the output we can visualize the data

Visualization of clusters -

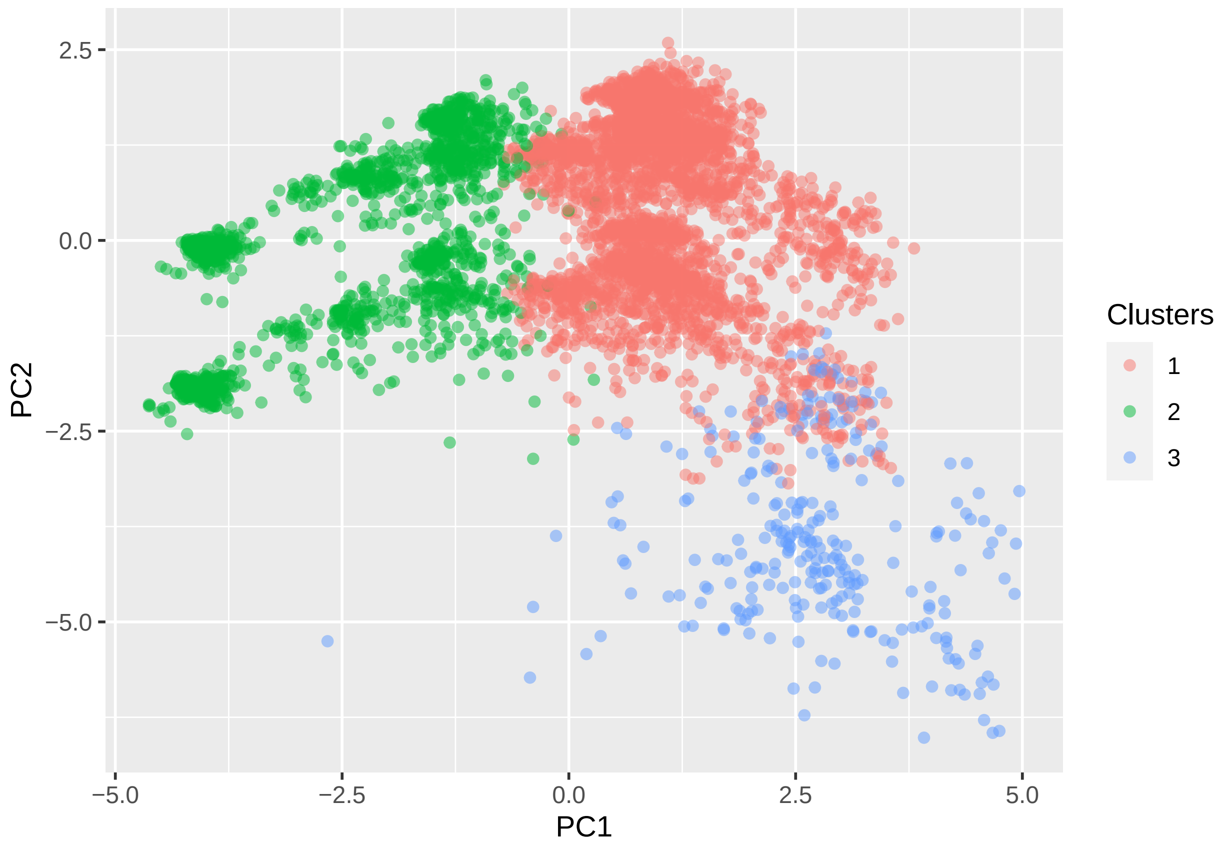
```r
# Visualizing Cluster Groups from the K-Means Model on the Standardized Data
fviz_cluster(kmeans_fit, data = stroke_data_standardized)
```



Visualizing clustering results by making a PCA projection -

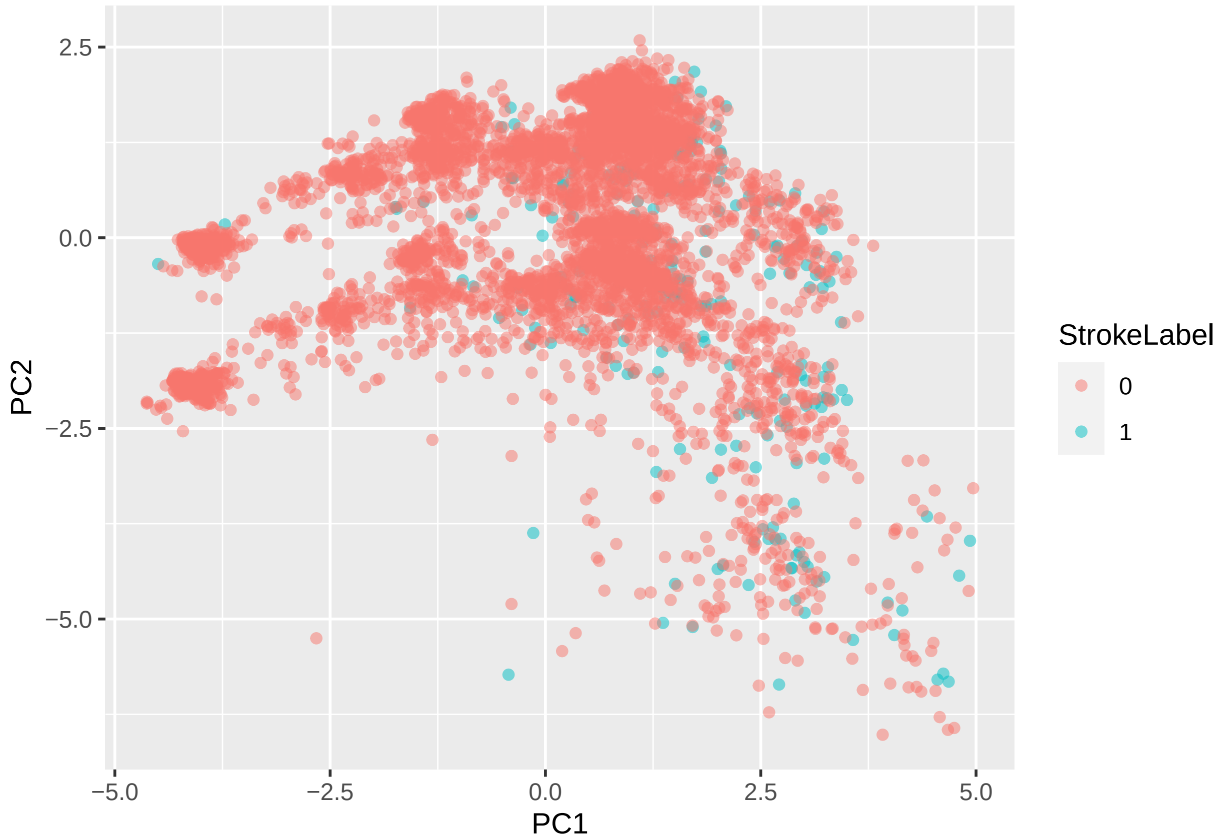
Comparing by generating a PCA plot and coloring the points by cluster assignment

```
# Performing Principal Component Analysis (PCA) on the Standardized Data
pca_result <- prcomp(stroke_data_standardized)

# Converting PCA Results into a Data Frame
pca_data_frame <- as.data.frame(pca_result$x)

# Adding the Cluster Assignments as a New Column
pca_data_frame$Clusters <- as.factor(kmeans_fit$cluster)

# Creating a Scatter Plot to Visualize PCA Results with Cluster Groupings
ggplot(data = pca_data_frame, aes(x = PC1, y = PC2, color = Clusters)) + geom_point(alpha=0.5)
```



Comparing by generating a PCA plot and coloring the points by class labels

```
# Incorporating Original Stroke Labels into the PCA Data Frame for Reference
pca_data_frame$StrokeLabel <- as.factor(stroke_data_dummies$stroke)

# Creating a Scatter Plot to Visualize PCA Results Highlighting Stroke Labels
ggplot(data = pca_data_frame, aes(x = PC1, y = PC2, color = StrokeLabel)) + geom_point(alpha=0.5)
```



Upon examining the cluster visualizations generated by the K-means algorithm, it's evident that this clustering method effectively groups the data in a manner that aligns closely with our actual class labels. Notably, the algorithm has formed three distinct clusters, which ideally would match three class labels. However, our dataset comprises only two class labels.

f. Classification

I will be using - SVM and KNN as my classifiers

1. KNN

Utilizing the stroke_data dataset, which has already undergone normalization/scaling and includes dummy variables for its categorical elements, we'll apply two distinct distance metrics to determine the optimal k value. These distance metrics are the Manhattan and Euclidean distances. Additionally, we plan to reintroduce class labels into the dataset, a step necessary for the development of classification models.

a. Manhattan distance -

The range for K was given as 3 to 10 and the distance function which was used was Manhattan The best value reported for K is 10 and the kernel used was cos The algorithm reported an accuracy of 95%

```
# Reintroducing Stroke Class Labels into the Normalized Dataset
stroke_data_standardized$stroke <- stroke_data_dummies$stroke
```

```r
# Converting the Stroke Variable to a Factor for Classification
stroke_data_standardized$stroke <- as.factor(stroke_data_standardized$stroke)

# Setting a Seed for Reproducible Results in Model Training
set.seed(456)

# Configuring 10-Fold Cross-Validation for Model Training
cross_val_control <- trainControl(method = "cv", number = 10, allowParallel = TRUE)

# Defining the Tuning Grid for Hyperparameter Optimization
tuning_parameters <- expand.grid(kmax = 3:10,                 # Testing k values from 3 to 10
                                 kernel = c("rectangular", "cos"),
                                 distance = 1)                # Using Manhattan distance

# Training the K-Nearest Neighbors Model with Manhattan Distance
kknn_model <- train(stroke ~ .,
                    data = stroke_data_standardized,
                    method = 'kknn',
                    trControl = cross_val_control,
                    tuneGrid = tuning_parameters)

# Outputting the Trained K-Nearest Neighbors Model
kknn_model
```

```
## k-Nearest Neighbors
##
## 5109 samples
##   22 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4598, 4599, 4598, 4598, 4598, 4598, ...
## Resampling results across tuning parameters:
##
##   kmax  kernel       Accuracy   Kappa
##    3    rectangular  0.9183811  0.05926769
##    3    cos          0.9291459  0.05874761
##    4    rectangular  0.9450002  0.02946761
##    4    cos          0.9363885  0.05402459
##    5    rectangular  0.9450002  0.02946761
##    5    cos          0.9424562  0.05157528
##    6    rectangular  0.9473485  0.03634192
##    6    cos          0.9461744  0.05567348
##    7    rectangular  0.9479356  0.03740140
##    7    cos          0.9479360  0.03670222
##    8    rectangular  0.9485227  0.03891468
##    8    cos          0.9491102  0.03966628
##    9    rectangular  0.9485227  0.03891468
##    9    cos          0.9495016  0.04071220
##   10    rectangular  0.9489141  0.03329186
##   10    cos          0.9504800  0.04406092
##
```

```
## Tuning parameter 'distance' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 10, distance = 1 and kernel
##  = cos.
```

b. Euclidean Distance -

The range for K was given as 3 to 10 and the distance function which was used was Euclidean The best value reported for K is 10 and the kernel used was rectangular The algorithm reported an accuracy of 94%

```r
# Setting a Seed for Consistent Results in Model Training
set.seed(456)

# Configuring 10-Fold Cross-Validation for the Training Process
cross_validation_control <- trainControl(method = "cv", number = 10, allowParallel = TRUE)

# Creating a Tuning Grid for Hyperparameter Optimization
# Testing k values from 3 to 10, using Euclidean distance
hyperparameter_grid <- expand.grid(kmax = 3:10,
                                   kernel = c("rectangular", "cos"),
                                   distance = 2)

# Training the K-Nearest Neighbors Model with Euclidean Distance
kknn_model_euclidean <- train(stroke ~ .,
                              data = stroke_data_standardized,
                              method = 'kknn',
                              trControl = cross_validation_control,
                              tuneGrid = hyperparameter_grid)

# Outputting the Details of the Trained KNN Model
kknn_model_euclidean
```

```
## k-Nearest Neighbors
##
## 5109 samples
##   22 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4598, 4599, 4598, 4598, 4598, 4598, ...
## Resampling results across tuning parameters:
##
##   kmax  kernel       Accuracy   Kappa
##    3    rectangular  0.9174030  0.05645689
##    3    cos          0.9264061  0.04319806
##    4    rectangular  0.9448045  0.04526663
##    4    cos          0.9356057  0.05278211
##    5    rectangular  0.9459787  0.04295981
##    5    cos          0.9399114  0.04522040
##    6    rectangular  0.9495016  0.06005579
##    6    cos          0.9440213  0.04871442
##    7    rectangular  0.9496972  0.04820595
```

```
##    7    cos          0.9467614  0.05006930
##    8    rectangular  0.9504800  0.04345584
##    8    cos          0.9481313  0.05371260
##    9    rectangular  0.9504800  0.04345584
##    9    cos          0.9481313  0.04859740
##   10    rectangular  0.9506757  0.05079944
##   10    cos          0.9487188  0.04503789
##
## Tuning parameter 'distance' was held constant at a value of 2
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 10, distance = 2 and kernel
##  = rectangular.
```

In summary, the accuracy levels achieved by both algorithms, each utilizing a different distance function, were nearly identical. I have decided to opt for the algorithm that employs the Manhattan distance function, which demonstrated an impressive accuracy of 95%. Moving forward, we will proceed to create a confusion matrix for the K-Nearest Neighbors (KNN) model that uses the Manhattan distance.

Generating the confusion matrix for KNN with Manhattan Distance -

```r
# Generating Predictions Using the K-Nearest Neighbors Model
predicted_knn <- predict(kknn_model, stroke_data_standardized)

# Creating a Confusion Matrix to Evaluate Model Performance
confusionMatrix(stroke_data_standardized$stroke, predicted_knn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4857    3
##          1  228   21
##
##                Accuracy : 0.9548
##                  95% CI : (0.9487, 0.9603)
##     No Information Rate : 0.9953
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1465
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.95516
##             Specificity : 0.87500
##          Pos Pred Value : 0.99938
##          Neg Pred Value : 0.08434
##              Prevalence : 0.99530
##          Detection Rate : 0.95068
##    Detection Prevalence : 0.95126
##       Balanced Accuracy : 0.91508
##
##        'Positive' Class : 0
##
```

After looking at the confusion matrix we understand that the algorithm does a good job at classifying true negatives but struggles to classify true positives

2. SVM

We will tune the parameters using Grid Search and then fit our model

```r
# Setting a Seed for Consistent Model Training Results
set.seed(456)

# Configuring 10-Fold Cross-Validation for the SVM Model Training
cross_validation_control_svm <- trainControl(method = "cv", number = 10, allowParallel = TRUE)

# Defining a Grid for Tuning the 'C' Parameter in SVM
tuning_grid_svm <- expand.grid(C = 10^seq(-5, 2, 0.5))

# Training the Support Vector Machine (SVM) Model with Linear Kernel
svm_model <- train(stroke ~ .,
                   data = stroke_data_standardized,
                   method = "svmLinear",
                   trControl = cross_validation_control_svm,
                   tuneGrid = tuning_grid_svm)

# Outputting the Details of the Trained SVM Model
svm_model
```

```
## Support Vector Machines with Linear Kernel
##
## 5109 samples
##   22 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4598, 4599, 4598, 4598, 4598, 4598, ...
## Resampling results across tuning parameters:
##
##   C             Accuracy   Kappa
##   1.000000e-05  0.9512628  0
##   3.162278e-05  0.9512628  0
##   1.000000e-04  0.9512628  0
##   3.162278e-04  0.9512628  0
##   1.000000e-03  0.9512628  0
##   3.162278e-03  0.9512628  0
##   1.000000e-02  0.9512628  0
##   3.162278e-02  0.9512628  0
##   1.000000e-01  0.9512628  0
##   3.162278e-01  0.9512628  0
##   1.000000e+00  0.9512628  0
##   3.162278e+00  0.9512628  0
##   1.000000e+01  0.9512628  0
##   3.162278e+01  0.9512628  0
##   1.000000e+02  0.9512628  0
```

```
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 1e-05.
```

After running the model, we observe that the accuracy with SVM is 95%

Generating a confusion matrix fo SVM -

```r
# Generating Predictions with the SVM Model
predicted_svm <- predict(svm_model, stroke_data_standardized)

# Creating a Confusion Matrix to Evaluate the SVM Model
confusionMatrix(stroke_data_standardized$stroke, predicted_svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4860    0
##          1  249    0
##
##                Accuracy : 0.9513
##                  95% CI : (0.945, 0.957)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9513
##             Specificity :     NA
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 1.0000
##          Detection Rate : 0.9513
##    Detection Prevalence : 0.9513
##       Balanced Accuracy :     NA
##
##        'Positive' Class : 0
##
```

Upon examining the confusion matrix for the SVM (Support Vector Machine) model, it becomes evident that SVM is not particularly effective in this classification task. It primarily classifies patients as not having had a stroke, failing to accurately classify those who have had a stroke.

While both SVM and KNN (K-Nearest Neighbors) models report an accuracy rate of 95%, this metric alone can be misleading. A closer analysis, particularly of the confusion matrices for both classifiers, reveals that KNN performs more effectively in terms of classification. Therefore, KNN emerges as the more suitable classifier for this model, demonstrating superior performance in distinguishing between the different classes.

g. Evaluation

We concluded that KNN is a better classifier for this dataset so we will be using that for further advanced evaluations

Creating a 70-30 train test split

```
# Initializing a Seed for Reproducible Data Partitioning
set.seed(456)

# Creating Indices for Data Partitioning Based on Stroke Label
partition_indices <- createDataPartition(y = stroke_data_standardized$stroke, p = 0.7, list = FALSE)

# Creating the Training Dataset Using the Specified Indices
training_data <- stroke_data_standardized[partition_indices, ]

# Forming the Test Dataset Excluding the Training Data Indices
testing_data <- stroke_data_standardized[-partition_indices, ]
```

Building a KNN model using train dataset

```
# Setting a Seed for Consistent Results in KNN Model Training
set.seed(456)

# Configuring 10-Fold Cross-Validation for KNN Training
cross_validation_control_knn <- trainControl(method = "cv", number = 10, allowParallel = TRUE)

# Training the K-Nearest Neighbors (KNN) Model on the Training Data
knn_model <- train(stroke ~ .,
                   data = training_data,
                   method = "knn",
                   trControl = cross_validation_control_knn,
                   tuneLength = 20)

# Outputting the Details of the Trained KNN Model
knn_model
```

```
## k-Nearest Neighbors
##
## 3577 samples
##   22 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 3219, 3220, 3219, 3219, 3219, 3220, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9477223  -0.0060867382
##    7  0.9496800  -0.0026585525
##    9  0.9507973  -0.0005319149
##   11  0.9510774   0.0000000000
##   13  0.9510774   0.0000000000
##   15  0.9510774   0.0000000000
##   17  0.9510774   0.0000000000
##   19  0.9510774   0.0000000000
##   21  0.9510774   0.0000000000
```

```
##   23  0.9510774   0.0000000000
##   25  0.9510774   0.0000000000
##   27  0.9510774   0.0000000000
##   29  0.9510774   0.0000000000
##   31  0.9510774   0.0000000000
##   33  0.9510774   0.0000000000
##   35  0.9510774   0.0000000000
##   37  0.9510774   0.0000000000
##   39  0.9510774   0.0000000000
##   41  0.9510774   0.0000000000
##   43  0.9510774   0.0000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 43.
```

We see that the best accuracy of the model is with k=43 and the reported accuracy is 95%

1. Confusion Matrix -

```r
# Predicting Stroke Labels on the Test Dataset Using the Trained KNN Model
predictions_knn_test <- predict(knn_model, testing_data)

# Creating a Confusion Matrix to Evaluate the KNN Model's Performance
confusion_matrix_knn <- confusionMatrix(testing_data$stroke, predictions_knn_test)

# Displaying the Confusion Matrix for the KNN Model
confusion_matrix_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1458    0
##          1   74    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9517
##             Specificity :     NA
##          Pos Pred Value :     NA
##          Neg Pred Value :     NA
##              Prevalence : 1.0000
##          Detection Rate : 0.9517
##    Detection Prevalence : 0.9517
##       Balanced Accuracy :     NA
##
```

```
##        'Positive' Class : 0
##
```

We notice that the model performs worse with train data as it does not classify any true negatives, hence there is no specificity mentioned as specificity determines the true negative rate The reported accuracy is 94%

2. Precision and Recall

Calculating Precision and Recall manually -

Formula for Precision is equal to True Positive/(True Positive + False Positive) Formula for Recall is equal to True Positive/(True Positive + False Negative)

```
# Calculating Precision: Proportion of Correct Positive Predictions
precision_score <- 1450 / (1450 + 0)

# Calculating Recall: Proportion of Actual Positives Correctly Identified
recall_score <- 1450 / (1450 + 74)

# Displaying the Calculated Precision and Recall Scores
print("The Precision score is:")
```

```
## [1] "The Precision score is:"
```

```
print(precision_score)
```

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

```
print("The Recall score is:")
```

```
## [1] "The Recall score is:"
```

```
print(recall_score)
```

```
## [1] 0.9514436
```

We can get the values for Precision and Recall by using the byClass object of confusionMatrix

```
# Extracting and Storing Performance Metrics from the Confusion Matrix as a DataFrame
performance_metrics <- as.data.frame(confusion_matrix_knn$byClass)

# Displaying the DataFrame Containing Performance Metrics
performance_metrics
```

```
##                         confusion_matrix_knn$byClass
## Sensitivity                                0.9516971
## Specificity                                       NA
## Pos Pred Value                                    NA
## Neg Pred Value                                    NA
```

```
## Precision                             1.0000000
## Recall                                0.9516971
## F1                                    0.9752508
## Prevalence                            1.0000000
## Detection Rate                        0.9516971
## Detection Prevalence                  0.9516971
## Balanced Accuracy                            NA
```

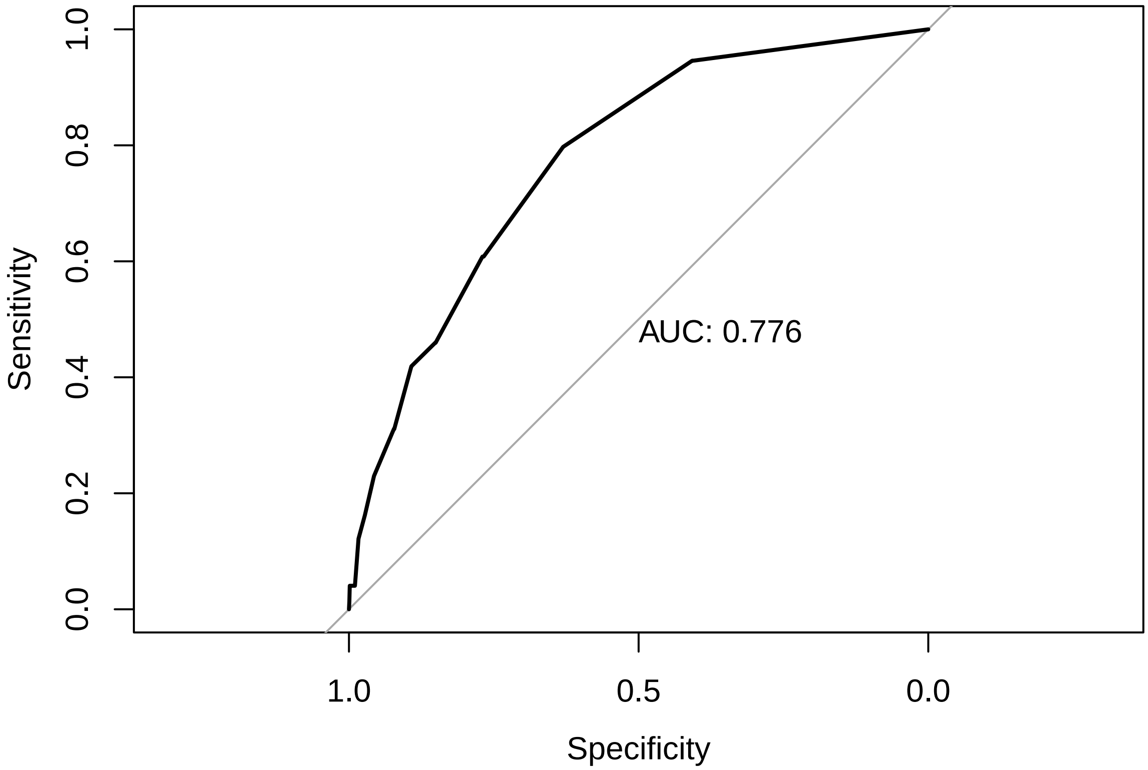
3. ROC Plot

```
# Generating Class Probabilities for Each Test Instance Using the KNN Model
predicted_probabilities_knn <- predict(knn_model, testing_data, type = "prob")

# Displaying the First Few Rows of the Predicted Probabilities DataFrame
head(predicted_probabilities_knn)
```

```
##           0          1
## 1 0.7906977 0.20930233
## 2 0.9534884 0.04651163
## 3 0.9767442 0.02325581
## 4 0.7906977 0.20930233
## 5 0.8837209 0.11627907
## 6 0.7906977 0.20930233
```

Creating the ROC Curve for the KNN model

```
# Building a ROC Curve to Evaluate Model Performance
roc_curve <- roc(testing_data$stroke, predicted_probabilities_knn[,1])

# Plotting the ROC Curve with Area Under the Curve (AUC) Displayed
plot(roc_curve, print.auc = TRUE)
```



An AUC (Area Under the Curve) of 0.7 suggests that there's a more than 50% chance that the model will rank a randomly chosen positive instance higher than a randomly chosen negative one.

The usefulness of performance evaluation metrics is multifold: - The confusion matrix provides a detailed comparison between actual cases and the classifier's predictions, offering insights into the specific areas where the classifier errs. - Precision reflects the accuracy of positive predictions, indicating the proportion of instances correctly identified as positive out of all labeled as positive. - Recall, or sensitivity, measures the ability of the model to identify all relevant instances, showing the percentage of actual positives that are correctly classified. - Depending on the specific model and dataset, we may prioritize different metrics like Precision, Recall, or Specificity to evaluate our model's effectiveness. - The ROC curve is valuable for assessing the model's capacity to differentiate between true positives and false positives.

In our scenario, while the accuracy appears high, a deeper dive into these performance metrics reveals that the model is actually biased in classifying only one class effectively, neglecting the other.

h. Report

Observing the substantial impact of dataset imbalance was a striking revelation. Without advanced evaluation methods like examining the confusion matrix or ROC curve, we might be misled by high accuracy figures that don't truly reflect the model's performance. Each phase in the pipeline significantly influences the subsequent stages, underscoring the importance of meticulous attention from the outset. Missteps at any stage can potentially compromise the entire model. This experience has underscored the critical importance of every step in the data mining process. To conduct a thorough analysis of this dataset, it's essential to have a balanced distribution in the class label column.

i. Reflection

This course has been enlightening in several ways: - It deepened my understanding of various preprocessing methods. - It highlighted the significance of both univariate and bivariate data analysis. - It provided insights into different classifiers and the underlying principles guiding them.

The section on clustering was particularly fascinating and emerged as my favorite topic. I've always admired intricate data visualizations online and been curious about the creative process behind them. This course not only fueled my interest but also brought me closer to understanding the core concepts of data science. It made me realize that data science isn't just about crafting striking visualizations; it's more about the journey of applying correct methodologies to achieve those results.