Stroke Prediction

2024-03-16

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Data Importing and descriptive Analysis\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#importing the data  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(dplyr)  
  
df <- read.csv("healthcare-dataset-stroke-data.csv", na.strings = "N/A")  
head(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 NA never smoked 1  
## 3 Rural 105.92 32.5 never smoked 1  
## 4 Urban 171.23 34.4 smokes 1  
## 5 Rural 174.12 24.0 never smoked 1  
## 6 Urban 186.21 29.0 formerly smoked 1

#simple preprocessing  
#removing id column  
df <- df[, !(names(df) %in% "id")]  
colnames(df)

## [1] "gender" "age" "hypertension"   
## [4] "heart\_disease" "ever\_married" "work\_type"   
## [7] "Residence\_type" "avg\_glucose\_level" "bmi"   
## [10] "smoking\_status" "stroke"

df$ever\_married = factor(df$ever\_married)  
df$work\_type = factor(df$work\_type)  
df$Residence\_type = factor(df$Residence\_type)  
df$smoking\_status = factor(df$smoking\_status)

#added later  
df$hypertension <- as.numeric(factor(df$hypertension))  
df$heart\_disease <- as.numeric(factor(df$heart\_disease))

df <- df[df$gender != "Other", ]  
df$gender = factor(df$gender)

#removing row with other- only 1 row for that   
str(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 : num 1 1 1 1 2 1 2 1 1 1 ...  
## $ heart\_disease : num 2 1 2 1 1 1 2 1 1 1 ...  
## $ ever\_married : 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 NA 32.5 34.4 24 29 27.4 22.8 NA 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 ...

#checking for NA values

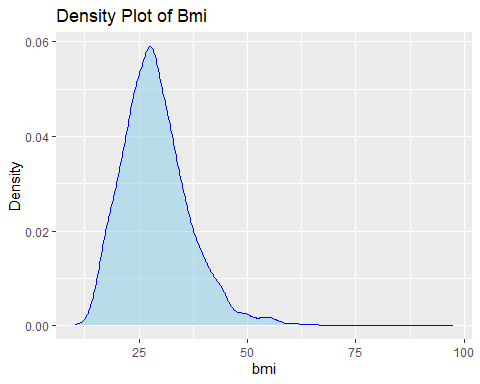
na\_values <- sum(is.na(df))  
na\_values\_per\_column <- colSums(is.na(df))  
na\_values\_per\_column

## gender age hypertension heart\_disease   
## 0 0 0 0   
## ever\_married work\_type Residence\_type avg\_glucose\_level   
## 0 0 0 0   
## bmi smoking\_status stroke   
## 201 0 0

#bmi has 201 na values. Check the distribution to see if mean imputation or meadian imputation should be done

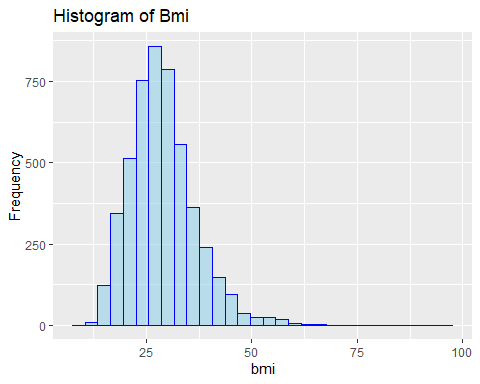
library(ggplot2)  
  
  
# Create density plot  
density\_plot <- ggplot(df, aes(x = bmi)) +  
 geom\_density(fill = "skyblue", color = "blue", alpha = 0.5) +  
 labs(title = "Density Plot of Bmi",  
 x = "bmi",  
 y = "Density")  
  
# Create histogram  
histogram\_plot <- ggplot(df, aes(x = bmi)) +  
 geom\_histogram(fill = "skyblue", color = "blue", alpha = 0.5, bins = 30) +  
 labs(title = "Histogram of Bmi",  
 x = "bmi",  
 y = "Frequency")  
  
# Display both plots  
print(density\_plot)

## Warning: Removed 201 rows containing non-finite values (`stat\_density()`).



print(histogram\_plot)

## Warning: Removed 201 rows containing non-finite values (`stat\_bin()`).



#right skewed so impute with median

median\_bmi <- median(df$bmi, na.rm = TRUE)  
df$bmi[is.na(df$bmi)] <- median\_bmi

#Descriptive statistics

summary(df)

## gender age hypertension heart\_disease ever\_married  
## Female:2994 Min. : 0.08 Min. :1.000 Min. :1.000 No :1756   
## Male :2115 1st Qu.:25.00 1st Qu.:1.000 1st Qu.:1.000 Yes:3353   
## Median :45.00 Median :1.000 Median :1.000   
## Mean :43.23 Mean :1.097 Mean :1.054   
## 3rd Qu.:61.00 3rd Qu.:1.000 3rd Qu.:1.000   
## Max. :82.00 Max. :2.000 Max. :2.000   
## work\_type Residence\_type avg\_glucose\_level bmi   
## children : 687 Rural:2513 Min. : 55.12 Min. :10.30   
## Govt\_job : 657 Urban:2596 1st Qu.: 77.24 1st Qu.:23.80   
## Never\_worked : 22 Median : 91.88 Median :28.10   
## Private :2924 Mean :106.14 Mean :28.86   
## 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 summary statistics provided represent data on various attributes for a group of individuals, including gender, age, health conditions, marital status, work type, residence type, average glucose level, body mass index (BMI), smoking status, and stroke occurrence.

The dataset consists of 5,109 individuals, with a gender distribution of 2,994 females and 2,115 males; there are no individuals identified as ‘Other’. Age in the dataset ranges from 0.08 to 82 years, with a median age of 45 years, indicating a middle-aged population. In terms of health conditions, 9.748% have hypertension, and 5.402% have heart disease.

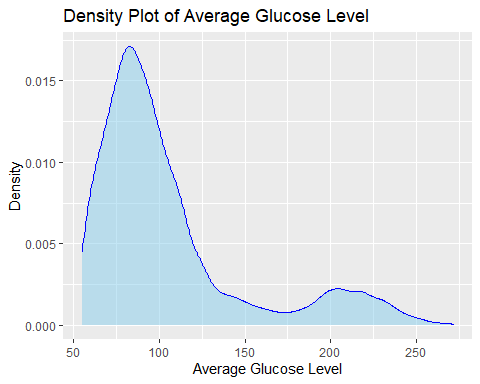
Regarding marital status, 3,353 individuals are married, while 1,756 are not. Work type categories include children (687 individuals), government jobs (657), never worked (22), private sector jobs (2,924), and self-employed (819). The individuals are almost evenly split between rural (2,513) and urban (2,596) residence types.

The average glucose level in the group is 106.14 mg/dL, with a range from 55.12 to 271.74 mg/dL. The BMI values range from 10.3 to 97.6, with a mean of 28.86, indicating that the average individual is overweight according to WHO standards.

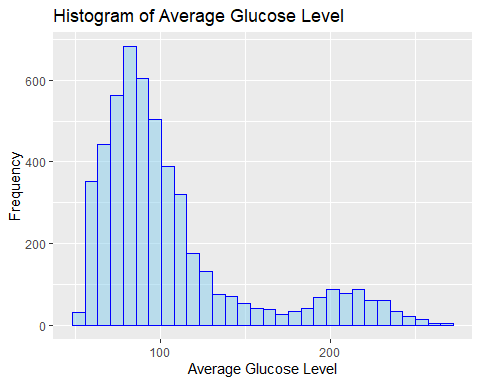
As for smoking status, the dataset includes 884 individuals who formerly smoked, 1,892 who never smoked, 789 who currently smoke, and 1,544 whose smoking status is unknown. Lastly, the prevalence of stroke in this population is 4.874%, which aligns with the dataset’s median and mean values being close to 0, suggesting that the majority of individuals have not experienced a stroke.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Exploratory Data Analysis\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

library(ggplot2)  
  
  
# Create density plot  
density\_plot <- ggplot(df, aes(x = avg\_glucose\_level)) +  
 geom\_density(fill = "skyblue", color = "blue", alpha = 0.5) +  
 labs(title = "Density Plot of Average Glucose Level",  
 x = "Average Glucose Level",  
 y = "Density")  
  
# Create histogram  
histogram\_plot <- ggplot(df, aes(x = avg\_glucose\_level)) +  
 geom\_histogram(fill = "skyblue", color = "blue", alpha = 0.5, bins = 30) +  
 labs(title = "Histogram of Average Glucose Level",  
 x = "Average Glucose Level",  
 y = "Frequency")  
  
# Display both plots  
print(density\_plot)

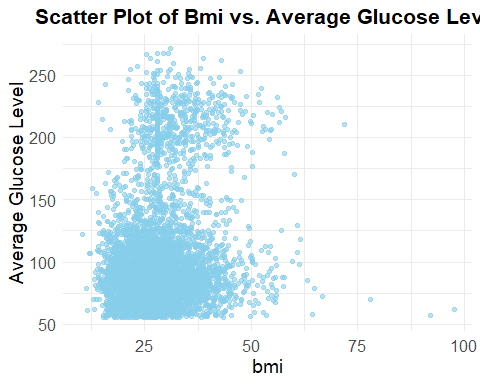


print(histogram\_plot)



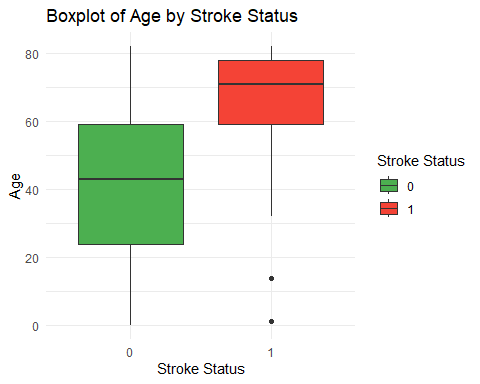
Bmi vs Glucose level

# Load the ggplot2 library  
library(ggplot2)  
  
# Create the scatter plot  
ggplot(df, aes(x = bmi, y = avg\_glucose\_level)) +  
 geom\_point(color = "skyblue", alpha = 0.6) + # Add points with color and transparency  
 labs(x = "bmi", y = "Average Glucose Level", title = "Scatter Plot of Bmi vs. Average Glucose Level") + # Add labels and title  
 theme\_minimal() + # Set minimal theme  
 theme(  
 plot.title = element\_text(size = 16, face = "bold", hjust = 0.5), # Title customization  
 axis.title = element\_text(size = 14), # Axis label customization  
 axis.text = element\_text(size = 12) # Axis text customization  
 )



Age vs stroke Box plot

# Create the box plot  
boxplot\_age\_vs\_stroke <- ggplot(df, aes(x = factor(stroke), y = age, fill = factor(stroke))) +  
 geom\_boxplot() +  
 labs(title = "Boxplot of Age by Stroke Status",  
 x = "Stroke Status", y = "Age",  
 fill = "Stroke Status") +  
 scale\_fill\_manual(values = c("0" = "#4CAF50", "1" = "#F44336")) + # Custom colors for stroke status  
 theme\_minimal()  
  
# Display the box plot  
print(boxplot\_age\_vs\_stroke)

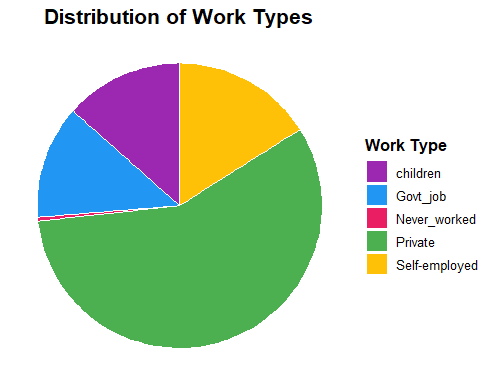


Work type count based on stroke

# Create a dataframe to calculate the total count of each work type and stroke status  
work\_type\_stroke\_count <- df %>%  
 group\_by(work\_type, stroke) %>%  
 summarise(count = n()) %>%  
 ungroup() %>%  
 mutate(stroke = factor(stroke, levels = c(0, 1), labels = c("No Stroke", "Stroke")))

## `summarise()` has grouped output by 'work\_type'. You can override using the  
## `.groups` argument.

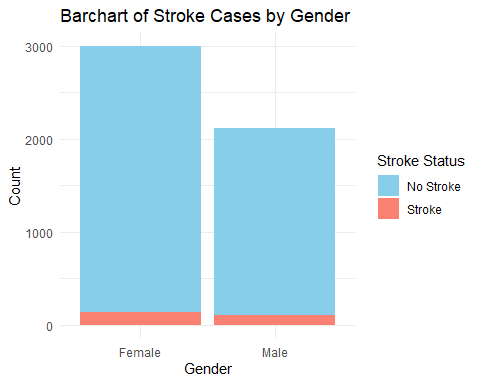
# Calculate total count of each work type  
total\_counts <- work\_type\_stroke\_count %>%  
 group\_by(work\_type) %>%  
 summarise(total\_count = sum(count))  
  
# Create a pie chart  
# Create the pie chart with beautifications  
pie\_chart <- ggplot(total\_counts, aes(x = "", y = total\_count, fill = work\_type)) +  
 geom\_bar(stat = "identity", width = 1, color = "white") + # Add white outline to bars  
 coord\_polar("y", start = 0) + # Convert bar plot to pie chart  
 labs(title = "Distribution of Work Types",  
 fill = "Work Type",  
 x = NULL,  
 y = NULL) + # Remove axis labels  
 theme\_void() + # Remove axis lines and labels  
 scale\_fill\_manual(values = c("Private" = "#4CAF50",   
 "Self-employed" = "#FFC107",   
 "Govt\_job" = "#2196F3",   
 "children" = "#9C27B0",   
 "Never\_worked" = "#E91E63")) + # Custom colors for each work type  
 theme(  
 plot.title = element\_text(size = 16, face = "bold", hjust = 0.5), # Title customization  
 legend.title = element\_text(size = 12, face = "bold"), # Legend title customization  
 legend.text = element\_text(size = 10), # Legend text customization  
 legend.position = "right" # Legend position  
 )  
  
# Display the beautified pie chart  
print(pie\_chart)



library(ggplot2)  
  
# Create a dataframe to calculate the total count of each gender and stroke status  
gender\_stroke\_count <- df %>%  
 group\_by(gender, stroke) %>%  
 summarise(count = n()) %>%  
 ungroup() %>%  
 mutate(stroke = factor(stroke, levels = c(0, 1), labels = c("No Stroke", "Stroke")))

## `summarise()` has grouped output by 'gender'. You can override using the  
## `.groups` argument.

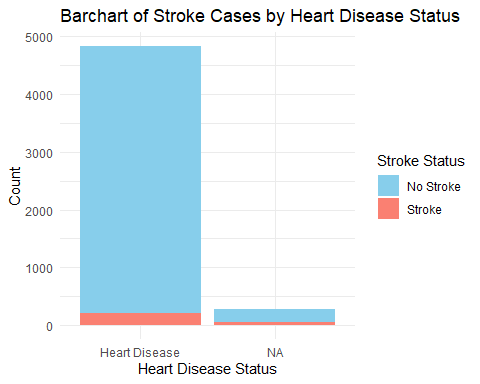
# Create the stacked bar plot  
stacked\_bar\_plot <- ggplot(gender\_stroke\_count, aes(x = gender, y = count, fill = stroke)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 labs(title = "Barchart of Stroke Cases by Gender",  
 x = "Gender",  
 y = "Count",  
 fill = "Stroke Status") +  
 scale\_fill\_manual(values = c("No Stroke" = "skyblue", "Stroke" = "salmon")) +  
 theme\_minimal()  
  
# Display the stacked bar plot  
print(stacked\_bar\_plot)



library(ggplot2)  
  
# Create a dataframe to calculate the total count of each heart disease status and stroke status  
heart\_disease\_stroke\_count <- df %>%  
 group\_by(heart\_disease, stroke) %>%  
 summarise(count = n()) %>%  
 ungroup() %>%  
 mutate(stroke = factor(stroke, levels = c(0, 1), labels = c("No Stroke", "Stroke")),  
 heart\_disease = factor(heart\_disease, levels = c(0, 1), labels = c("No Heart Disease", "Heart Disease")))

## `summarise()` has grouped output by 'heart\_disease'. You can override using the  
## `.groups` argument.

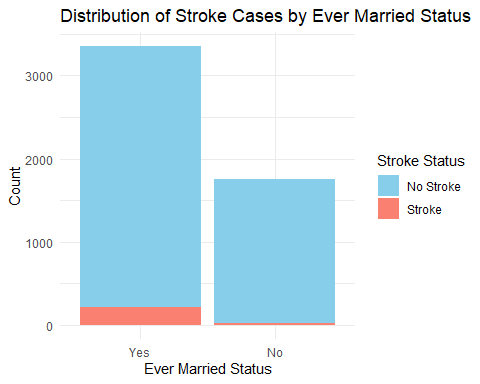
# Create the stacked bar plot  
stacked\_bar\_plot\_heart\_disease <- ggplot(heart\_disease\_stroke\_count, aes(x = heart\_disease, y = count, fill = stroke)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 labs(title = "Barchart of Stroke Cases by Heart Disease Status",  
 x = "Heart Disease Status",  
 y = "Count",  
 fill = "Stroke Status") +  
 scale\_fill\_manual(values = c("No Stroke" = "skyblue", "Stroke" = "salmon")) +  
 theme\_minimal()  
  
# Display the stacked bar plot  
print(stacked\_bar\_plot\_heart\_disease)



library(ggplot2)  
  
# Create a dataframe to calculate the total count of each ever married status and stroke status  
ever\_married\_stroke\_count <- df %>%  
 group\_by(ever\_married, stroke) %>%  
 summarise(count = n()) %>%  
 ungroup() %>%  
 mutate(stroke = factor(stroke, levels = c(0, 1), labels = c("No Stroke", "Stroke")),  
 ever\_married = factor(ever\_married, levels = c("Yes", "No")))

## `summarise()` has grouped output by 'ever\_married'. You can override using the  
## `.groups` argument.

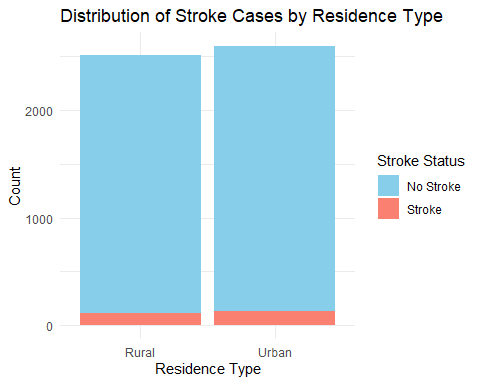
# Create the stacked bar plot  
stacked\_bar\_plot\_ever\_married <- ggplot(ever\_married\_stroke\_count, aes(x = ever\_married, y = count, fill = stroke)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 labs(title = "Distribution of Stroke Cases by Ever Married Status",  
 x = "Ever Married Status",  
 y = "Count",  
 fill = "Stroke Status") +  
 scale\_fill\_manual(values = c("No Stroke" = "skyblue", "Stroke" = "salmon")) +  
 theme\_minimal()  
  
# Display the stacked bar plot  
print(stacked\_bar\_plot\_ever\_married)



# Create a dataframe to calculate the total count of each residence type and stroke status  
residence\_type\_stroke\_count <- df %>%  
 group\_by(Residence\_type, stroke) %>%  
 summarise(count = n()) %>%  
 ungroup() %>%  
 mutate(stroke = factor(stroke, levels = c(0, 1), labels = c("No Stroke", "Stroke")))

## `summarise()` has grouped output by 'Residence\_type'. You can override using  
## the `.groups` argument.

# Create the stacked bar plot  
stacked\_bar\_plot\_residence\_type <- ggplot(residence\_type\_stroke\_count, aes(x = Residence\_type, y = count, fill = stroke)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 labs(title = "Distribution of Stroke Cases by Residence Type",  
 x = "Residence Type",  
 y = "Count",  
 fill = "Stroke Status") +  
 scale\_fill\_manual(values = c("No Stroke" = "skyblue", "Stroke" = "salmon")) +  
 theme\_minimal()  
  
# Display the stacked bar plot  
print(stacked\_bar\_plot\_residence\_type)

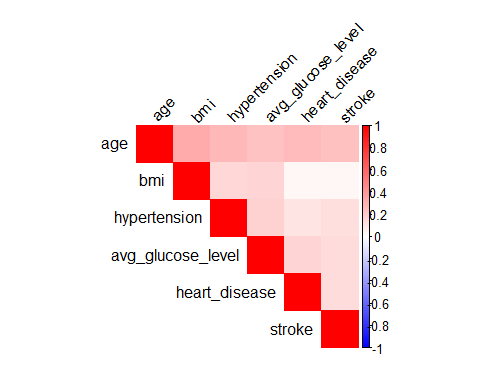


# Load the required library  
library(corrplot)

## Warning: package 'corrplot' was built under R version 4.3.3

## corrplot 0.92 loaded

# Compute the correlation matrix  
correlation\_matrix <- cor(df[, sapply(df, is.numeric)])  
  
# Create the correlation heatmap  
corrplot(correlation\_matrix, method = "color", type = "upper",   
 tl.col = "black", tl.srt = 45,   
 addrect = 2, order = "hclust",   
 col = colorRampPalette(c("blue", "white", "red"))(200))



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OverSampling\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#standardizing the dataset  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

# List of numeric columns you want to standardize  
numeric\_columns <- c("age", "avg\_glucose\_level", "bmi")  
  
# Use preProcess to standardize (z-score normalization)  
preproc <- preProcess(df[, numeric\_columns], method = c("center", "scale"))  
  
# Standardize the data  
df\_standardized <- predict(preproc, df[, numeric\_columns])  
  
# Replace the original columns with the standardized data  
df[, numeric\_columns] <- df\_standardized

#checking the class distribution

table(df$stroke)

##   
## 0 1   
## 4860 249

#From all of the above analysis, we can see that the dataset is imbalanced. We will balance the dataset through the process of oversampling.

prop.table(table(df$stroke))

##   
## 0 1   
## 0.95126248 0.04873752

# The process of oversampling - SMOTE

library(ROSE)

## Warning: package 'ROSE' was built under R version 4.3.3

## Loaded ROSE 0.0-4

library(caret)  
  
#split the data into training and testing set  
set.seed(25) # for reproducibility  
sample\_size <- floor(0.75 \* nrow(df)) # 75% for training  
train\_indices <- sample(seq\_len(nrow(df)), size = sample\_size)  
  
train\_set <- df[train\_indices, ]  
test\_set <- df[-train\_indices, ]

train\_set$stroke <- factor(train\_set$stroke, levels = c(0, 1))  
test\_set$stroke <- factor(test\_set$stroke, levels = c(0, 1))

balanced\_train\_set <- ovun.sample(stroke ~ ., data = train\_set, method = "over")$data

The oversampling methodology report describes the utilization of the ovun.sample() function from the ROSE package in R to address class imbalance in a dataset, focusing particularly on balancing the ‘stroke’ variable. This approach involves taking the original ‘train\_set’ and applying oversampling to augment the minority class. The function is set with method = “over”, directing it to increase the minority class’s presence by creating synthetic samples that are statistically similar to existing ones, though not identical. The result is the ‘balanced\_train\_set’, a modified version of the original dataset with a better balance between classes, aiming to improve the outcomes of predictive modeling by providing a more equitable data foundation. This method is primarily designed to reduce the skewness in class distribution, thus potentially enhancing the performance of subsequent analyses and model training.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*KNN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ctrl <- trainControl(method = "cv", number = 5)  
knn\_model <- train(stroke ~ ., data = balanced\_train\_set, method = "knn", trControl =  
ctrl, preProcess = c("center", "scale"))

final\_model <- train(stroke ~ ., data = balanced\_train\_set, method = "knn", trControl =  
ctrl, preProcess = c("center", "scale"), tuneGrid = data.frame(k =  
knn\_model$bestTune$k))

knn\_probs <- predict(final\_model, newdata = test\_set, type = "prob")  
knn.pred <- rep("0", length(test\_set$stroke))  
knn.pred[knn\_probs[, "1"] > 0.5] <- "1"  
###  
table(knn.pred, test\_set$stroke)

##   
## knn.pred 0 1  
## 0 996 43  
## 1 218 21

mean(knn.pred == test\_set$stroke)

## [1] 0.7957746

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Random Forest\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.1

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

##   
## Attaching package: 'randomForest'

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

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

X <- balanced\_train\_set[, -which(names(balanced\_train\_set) %in% c("stroke"))] # Features  
y <- balanced\_train\_set$stroke # Target variable  
  
# Fit the Random Forest classifier  
rf\_model <- randomForest(x = X, y = y, ntree = 100)  
  
# Print the model summary  
print(rf\_model)

##   
## Call:  
## randomForest(x = X, y = y, ntree = 100)   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 0.94%  
## Confusion matrix:  
## 0 1 class.error  
## 0 3577 69 0.01892485  
## 1 0 3677 0.00000000

X\_new <- test\_set[, -which(names(test\_set) %in% c("stroke"))] # Features  
  
# Make predictions using the trained Random Forest model  
predictions <- predict(rf\_model, newdata = X\_new)

confusionMatrix(predictions, test\_set$stroke)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1197 60  
## 1 17 4  
##   
## Accuracy : 0.9397   
## 95% CI : (0.9253, 0.9522)  
## No Information Rate : 0.9499   
## P-Value [Acc > NIR] : 0.9551   
##   
## Kappa : 0.0711   
##   
## Mcnemar's Test P-Value : 1.698e-06   
##   
## Sensitivity : 0.9860   
## Specificity : 0.0625   
## Pos Pred Value : 0.9523   
## Neg Pred Value : 0.1905   
## Prevalence : 0.9499   
## Detection Rate : 0.9366   
## Detection Prevalence : 0.9836   
## Balanced Accuracy : 0.5242   
##   
## 'Positive' Class : 0   
##

accuracy <- mean(predictions == test\_set$stroke)  
accuracy

## [1] 0.9397496

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Decision Tree\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

library(rpart)  
  
# Fit the decision tree classifier  
dt\_model <- rpart(  
 formula = stroke ~ .,  
 data = balanced\_train\_set  
)

predictions\_dt <- predict(dt\_model, newdata = test\_set, type = "class")

confusionMatrix(predictions\_dt, test\_set$stroke)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 769 7  
## 1 445 57  
##   
## Accuracy : 0.6463   
## 95% CI : (0.6194, 0.6726)  
## No Information Rate : 0.9499   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1236   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.6334   
## Specificity : 0.8906   
## Pos Pred Value : 0.9910   
## Neg Pred Value : 0.1135   
## Prevalence : 0.9499   
## Detection Rate : 0.6017   
## Detection Prevalence : 0.6072   
## Balanced Accuracy : 0.7620   
##   
## 'Positive' Class : 0   
##

accuracy <- mean(predictions\_dt == test\_set$stroke)  
accuracy

## [1] 0.6463224

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SVM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

library(e1071)

## Warning: package 'e1071' was built under R version 4.3.3

# Fit the SVM classifier  
svm\_model <- svm(  
 formula = stroke ~ .,  
 data = balanced\_train\_set  
)

predictions\_svm <- predict(svm\_model, newdata = test\_set)

confusionMatrix(predictions\_svm, test\_set$stroke)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 871 12  
## 1 343 52  
##   
## Accuracy : 0.7222   
## 95% CI : (0.6968, 0.7466)  
## No Information Rate : 0.9499   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.1536   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7175   
## Specificity : 0.8125   
## Pos Pred Value : 0.9864   
## Neg Pred Value : 0.1316   
## Prevalence : 0.9499   
## Detection Rate : 0.6815   
## Detection Prevalence : 0.6909   
## Balanced Accuracy : 0.7650   
##   
## 'Positive' Class : 0   
##

accuracy <- mean(predictions\_svm == test\_set$stroke)  
accuracy

## [1] 0.7222222