R Notebook

#Setting Up CRAN Repository

#This sets the CRAN repository to ensure package installation from a reliable source.  
# Set the CRAN Repository  
options(repos = c(CRAN = "https://cloud.r-project.org/"))

#Installing Required Packages

packages <- c("tidyverse", "openxlsx", "knitr", "dplyr", "MASS", "pscl", "pROC", "car", "tidyr", "ggplot2", "corrplot")  
install\_if\_missing <- function(p) {  
 if (!p %in% installed.packages()[, "Package"]) install.packages(p)  
}  
sapply(packages, install\_if\_missing)

## $tidyverse  
## NULL  
##   
## $openxlsx  
## NULL  
##   
## $knitr  
## NULL  
##   
## $dplyr  
## NULL  
##   
## $MASS  
## NULL  
##   
## $pscl  
## NULL  
##   
## $pROC  
## NULL  
##   
## $car  
## NULL  
##   
## $tidyr  
## NULL  
##   
## $ggplot2  
## NULL  
##   
## $corrplot  
## NULL

#Setting the Working Directory and Loading Data

setwd("G:/UMBC\_Academics/HIT\_750\_Data Analytics/Project/Week 1") # Path of Woring directory  
list.files()

## [1] "diabetes\_dataset.csv"   
## [2] "Diabetics\_analysis.docx"   
## [3] "Diabetics\_analysis.nb.html"   
## [4] "Diabetics\_analysis.Rmd"   
## [5] "Diabetics\_analysis.tex"   
## [6] "Exploring the Impact of Lifestyle and Health Factors on Diabetes Risk.pdf"  
## [7] "random\_forest\_model.rds"   
## [8] "Statistical Analysis Using BRFSS Data (1) (1).pptx"   
## [9] "VariableTable.xlsx"

data <- read.csv("diabetes\_dataset.csv") # Dataset filename

#Loading the Variable Table

# Load the openxlsx package to work with Excel files  
library(openxlsx)

## Warning: package 'openxlsx' was built under R version 4.4.2

# Load the knitr package for rendering tables  
library(knitr)  
  
# Set the path to the Excel file on your local machine  
# Update this path to where the file is located on your system  
file\_path <- "G:/UMBC\_Academics/HIT\_750\_Data Analytics/Project/Week 1/VariableTable.xlsx"  
  
# Read the Excel sheet into a variable (assuming it's in the first sheet)  
variable <- read.xlsx(file\_path, sheet = 1)  
  
# View the data in the notebook using knitr's kable function  
kable(variable, caption = "Variable Table from Excel")

Variable Table from Excel

| Variable | Description | Data.Type |
| --- | --- | --- |
| ID | Patient ID | Integer |
| Diabetes\_binary | 0 = no diabetes 1 = prediabetes or diabetes | Binary |
| HighBP | 0 = no high BP 1 = high BP | Binary |
| HighChol | 0 = no high cholesterol 1 = high cholesterol | Binary |
| CholCheck | 0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years | Binary |
| BMI | Body Mass Index | Integer |
| Smoker | Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes] 0 = no 1 = yes | Binary |
| Stroke | (Ever told) you had a stroke. 0 = no 1 = yes | Binary |
| HeartDiseaseorAttack | coronary heart disease (CHD) or myocardial infarction (MI) 0 = no 1 = yes | Binary |
| PhysActivity | physical activity in past 30 days - not including job 0 = no 1 = yes | Binary |
| Fruits | Consume Fruit 1 or more times per day 0 = no 1 = yes | Binary |
| Veggies | Consume Vegetables 1 or more times per day 0 = no 1 = yes | Binary |
| HvyAlcoholConsump | Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week) 0 = no 1 = yes | Binary |
| AnyHealthcare | Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc. 0 = no 1 = yes | Binary |
| NoDocbcCost | Was there a time in the past 12 months when you needed to see a doctor but could not because of cost? 0 = no 1 = yes | Binary |
| GenHlth | Would you say that in general your health is: scale 1-5 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor | Integer |
| MentHlth | Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good? scale 1-30 days | Integer |
| PhysHlth | Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good? scale 1-30 days | Integer |
| DiffWalk | Do you have serious difficulty walking or climbing stairs? 0 = no 1 = yes | Binary |
| Sex | 0 = female 1 = male | Binary |
| Age | 13-level age category (\_AGEG5YR see codebook) 1 = 18-24 9 = 60-64 13 = 80 or older | Integer |
| Education | Education level (EDUCA see codebook) scale 1-6 1 = Never attended school or only kindergarten 2 = Grades 1 through 8 (Elementary) 3 = Grades 9 through 11 (Some high school) 4 = Grade 12 or GED (High school graduate) 5 = College 1 year to 3 years (Some college or technical school) 6 = College 4 years or more (College graduate) | Integer |
| Income | Income scale (INCOME2 see codebook) scale 1-8 1 = less than $10,000 5 = less than $35,000 8 = $75,000 or more | Integer |

#Calculating Summary Statistics for Numeric Variables

# Load necessary packages  
library(knitr)  
library(dplyr)

## Warning: package 'dplyr' was built under R version 4.4.2

##   
## Attaching package: 'dplyr'

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

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

library(tidyr)

## Warning: package 'tidyr' was built under R version 4.4.2

# Set the correct path to your local dataset  
data <- read.csv("G:/UMBC\_Academics/HIT\_750\_Data Analytics/Project/Week 1/diabetes\_dataset.csv")  
  
# Function to calculate summary statistics  
calc\_summary\_stats <- function(x) {  
 c(  
 Min = min(x, na.rm = TRUE),  
 `1st Qu.` = quantile(x, 0.25, na.rm = TRUE),  
 Median = median(x, na.rm = TRUE),  
 Mean = mean(x, na.rm = TRUE),  
 `3rd Qu.` = quantile(x, 0.75, na.rm = TRUE),  
 Max = max(x, na.rm = TRUE)  
 )  
}  
  
# Apply the function to each numeric variable in the dataset  
summary\_stats <- data %>%  
 select\_if(is.numeric) %>%  
 summarise\_all(list(calc\_summary\_stats))

## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
## dplyr 1.1.0.  
## ℹ Please use `reframe()` instead.  
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
## always returns an ungrouped data frame and adjust accordingly.  
## ℹ The deprecated feature was likely used in the dplyr package.  
## Please report the issue at <https://github.com/tidyverse/dplyr/issues>.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

# Unnest the list columns generated by summarise\_all  
summary\_stats\_unnested <- summary\_stats %>%  
 unnest(cols = everything(), names\_repair = "minimal")  
  
# Transpose the summary statistics for better readability  
summary\_stats\_t <- t(summary\_stats\_unnested)  
  
# Convert the transposed summary stats into a data frame  
summary\_df <- as.data.frame(summary\_stats\_t)  
  
# Set proper column names after transposing  
colnames(summary\_df) <- c("Min", "1st Qu.", "Median", "Mean", "3rd Qu.", "Max")  
  
# Add the variable names as a separate column  
summary\_df$Variable <- rownames(summary\_df)  
  
# Reorder the columns to place the Variable column first  
summary\_df <- summary\_df[, c("Variable", "Min", "1st Qu.", "Median", "Mean", "3rd Qu.", "Max")]  
  
# Print the final summary\_df to debug  
print(summary\_df)

## Variable Min 1st Qu. Median Mean  
## Diabetes\_binary Diabetes\_binary 0 0 0 0.13933302  
## HighBP HighBP 0 0 0 0.42900110  
## HighChol HighChol 0 0 0 0.42412094  
## CholCheck CholCheck 0 1 1 0.96266950  
## BMI BMI 12 24 27 28.38236361  
## Smoker Smoker 0 0 0 0.44316856  
## Stroke Stroke 0 0 0 0.04057080  
## HeartDiseaseorAttack HeartDiseaseorAttack 0 0 0 0.09418559  
## PhysActivity PhysActivity 0 1 1 0.75654368  
## Fruits Fruits 0 0 1 0.63425576  
## Veggies Veggies 0 1 1 0.81141990  
## HvyAlcoholConsump HvyAlcoholConsump 0 0 0 0.05619678  
## AnyHealthcare AnyHealthcare 0 1 1 0.95105251  
## NoDocbcCost NoDocbcCost 0 0 0 0.08417692  
## GenHlth GenHlth 1 2 2 2.51139231  
## MentHlth MentHlth 0 0 0 3.18477215  
## PhysHlth PhysHlth 0 0 0 4.24208057  
## DiffWalk DiffWalk 0 0 0 0.16822375  
## Sex Sex 0 0 0 0.44034216  
## Age Age 1 6 8 8.03211921  
## Education Education 1 4 5 5.05043362  
## Income Income 1 5 7 6.05387496  
## 3rd Qu. Max  
## Diabetes\_binary 0 1  
## HighBP 1 1  
## HighChol 1 1  
## CholCheck 1 1  
## BMI 31 98  
## Smoker 1 1  
## Stroke 0 1  
## HeartDiseaseorAttack 0 1  
## PhysActivity 1 1  
## Fruits 1 1  
## Veggies 1 1  
## HvyAlcoholConsump 0 1  
## AnyHealthcare 1 1  
## NoDocbcCost 0 1  
## GenHlth 3 5  
## MentHlth 2 30  
## PhysHlth 3 30  
## DiffWalk 0 1  
## Sex 1 1  
## Age 10 13  
## Education 6 6  
## Income 8 8

# Print the formatted summary statistics table  
kable(summary\_df, format = "latex", row.names = FALSE)

#Displaying Dataset Summary and Structure

# Summary statistics for the dataset  
summary(data)

## Diabetes\_binary HighBP HighChol CholCheck   
## Min. :0.0000 Min. :0.000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:1.0000   
## Median :0.0000 Median :0.000 Median :0.0000 Median :1.0000   
## Mean :0.1393 Mean :0.429 Mean :0.4241 Mean :0.9627   
## 3rd Qu.:0.0000 3rd Qu.:1.000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.000 Max. :1.0000 Max. :1.0000   
## BMI Smoker Stroke HeartDiseaseorAttack  
## Min. :12.00 Min. :0.0000 Min. :0.00000 Min. :0.00000   
## 1st Qu.:24.00 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:0.00000   
## Median :27.00 Median :0.0000 Median :0.00000 Median :0.00000   
## Mean :28.38 Mean :0.4432 Mean :0.04057 Mean :0.09419   
## 3rd Qu.:31.00 3rd Qu.:1.0000 3rd Qu.:0.00000 3rd Qu.:0.00000   
## Max. :98.00 Max. :1.0000 Max. :1.00000 Max. :1.00000   
## PhysActivity Fruits Veggies HvyAlcoholConsump  
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median :1.0000 Median :0.0000   
## Mean :0.7565 Mean :0.6343 Mean :0.8114 Mean :0.0562   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## AnyHealthcare NoDocbcCost GenHlth MentHlth   
## Min. :0.0000 Min. :0.00000 Min. :1.000 Min. : 0.000   
## 1st Qu.:1.0000 1st Qu.:0.00000 1st Qu.:2.000 1st Qu.: 0.000   
## Median :1.0000 Median :0.00000 Median :2.000 Median : 0.000   
## Mean :0.9511 Mean :0.08418 Mean :2.511 Mean : 3.185   
## 3rd Qu.:1.0000 3rd Qu.:0.00000 3rd Qu.:3.000 3rd Qu.: 2.000   
## Max. :1.0000 Max. :1.00000 Max. :5.000 Max. :30.000   
## PhysHlth DiffWalk Sex Age   
## Min. : 0.000 Min. :0.0000 Min. :0.0000 Min. : 1.000   
## 1st Qu.: 0.000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.: 6.000   
## Median : 0.000 Median :0.0000 Median :0.0000 Median : 8.000   
## Mean : 4.242 Mean :0.1682 Mean :0.4403 Mean : 8.032   
## 3rd Qu.: 3.000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:10.000   
## Max. :30.000 Max. :1.0000 Max. :1.0000 Max. :13.000   
## Education Income   
## Min. :1.00 Min. :1.000   
## 1st Qu.:4.00 1st Qu.:5.000   
## Median :5.00 Median :7.000   
## Mean :5.05 Mean :6.054   
## 3rd Qu.:6.00 3rd Qu.:8.000   
## Max. :6.00 Max. :8.000

# Check the structure of the dataset  
str(data)

## 'data.frame': 253680 obs. of 22 variables:  
## $ Diabetes\_binary : num 0 0 0 0 0 0 0 0 1 0 ...  
## $ HighBP : num 1 0 1 1 1 1 1 1 1 0 ...  
## $ HighChol : num 1 0 1 0 1 1 0 1 1 0 ...  
## $ CholCheck : num 1 0 1 1 1 1 1 1 1 1 ...  
## $ BMI : num 40 25 28 27 24 25 30 25 30 24 ...  
## $ Smoker : num 1 1 0 0 0 1 1 1 1 0 ...  
## $ Stroke : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ HeartDiseaseorAttack: num 0 0 0 0 0 0 0 0 1 0 ...  
## $ PhysActivity : num 0 1 0 1 1 1 0 1 0 0 ...  
## $ Fruits : num 0 0 1 1 1 1 0 0 1 0 ...  
## $ Veggies : num 1 0 0 1 1 1 0 1 1 1 ...  
## $ HvyAlcoholConsump : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ AnyHealthcare : num 1 0 1 1 1 1 1 1 1 1 ...  
## $ NoDocbcCost : num 0 1 1 0 0 0 0 0 0 0 ...  
## $ GenHlth : num 5 3 5 2 2 2 3 3 5 2 ...  
## $ MentHlth : num 18 0 30 0 3 0 0 0 30 0 ...  
## $ PhysHlth : num 15 0 30 0 0 2 14 0 30 0 ...  
## $ DiffWalk : num 1 0 1 0 0 0 0 1 1 0 ...  
## $ Sex : num 0 0 0 0 0 1 0 0 0 1 ...  
## $ Age : num 9 7 9 11 11 10 9 11 9 8 ...  
## $ Education : num 4 6 4 3 5 6 6 4 5 4 ...  
## $ Income : num 3 1 8 6 4 8 7 4 1 3 ...

#Check for missing values in the dataset

# Check for missing values in the dataset  
missing\_values <- colSums(is.na(data))  
  
# Filter out variables with missing values and create a neat output  
missing\_summary <- missing\_values[missing\_values > 0]  
  
# Print the number of missing values in a neat format  
if(length(missing\_summary) > 0) {  
 cat("Number of missing values in the dataset:\n")  
 for(variable in names(missing\_summary)) {  
 cat(variable, ":", missing\_summary[variable], "\n")  
 }  
} else {  
 cat("There are no missing values in the dataset.\n")  
}

## There are no missing values in the dataset.

install.packages("ggplot2")

## Installing package into 'C:/Users/DELL/AppData/Local/R/win-library/4.4'  
## (as 'lib' is unspecified)

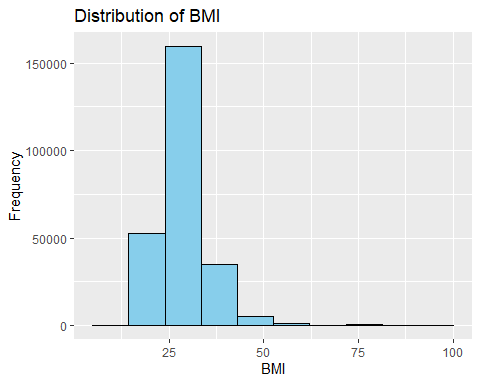
## package 'ggplot2' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\DELL\AppData\Local\Temp\Rtmp6L4tSi\downloaded\_packages

#Plotting the Distribution of BMI

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.4.2

# Histogram of BMI  
ggplot(data, aes(x = BMI)) +  
 geom\_histogram(bins = 10, fill = "skyblue", color = "black") +  
 labs(title = "Distribution of BMI", x = "BMI", y = "Frequency")



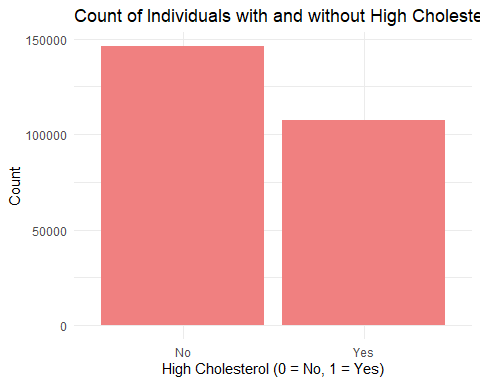
#Counting Individuals with High Blood Pressure

# Count the number of people with and without high blood pressure  
highbp\_count <- table(data$HighBP)  
  
# Display the counts in a neat format  
highbp\_count\_df <- as.data.frame(highbp\_count)  
colnames(highbp\_count\_df) <- c("HighBP\_Status", "Count")  
  
# Print the summary  
print(highbp\_count\_df)

## HighBP\_Status Count  
## 1 0 144851  
## 2 1 108829

#Plotting High Cholesterol Status

# Load ggplot2 if not already loaded  
library(ggplot2)  
  
# Create a bar plot for High Cholesterol status  
ggplot(data, aes(x = as.factor(HighChol))) +  
 geom\_bar(fill = "lightcoral") +  
 labs(title = "Count of Individuals with and without High Cholesterol",  
 x = "High Cholesterol (0 = No, 1 = Yes)",  
 y = "Count") +  
 scale\_x\_discrete(labels = c("0" = "No", "1" = "Yes")) +  
 theme\_minimal()



#Correlation Analysis and Plotting

install.packages("corrplot")

## Installing package into 'C:/Users/DELL/AppData/Local/R/win-library/4.4'  
## (as 'lib' is unspecified)

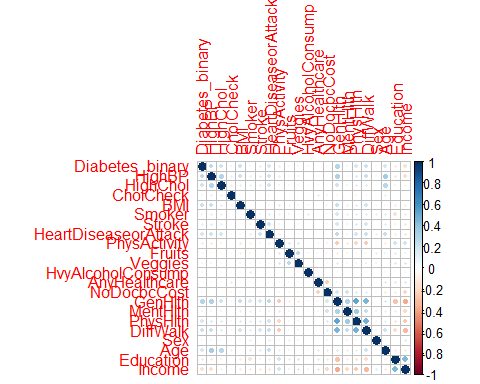
## package 'corrplot' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\DELL\AppData\Local\Temp\Rtmp6L4tSi\downloaded\_packages

# Load necessary package  
library(corrplot)

## Warning: package 'corrplot' was built under R version 4.4.2

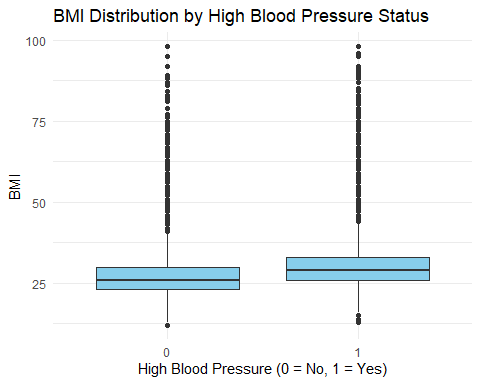
## corrplot 0.95 loaded

# Calculate the correlation matrix  
cor\_matrix <- cor(select(data, where(is.numeric)))  
  
# Create a correlation plot  
corrplot(cor\_matrix, method = "circle")



#BMI by High Blood Pressure

# Boxplot of BMI by High Blood Pressure  
ggplot(data, aes(x = as.factor(HighBP), y = BMI)) +  
 geom\_boxplot(fill = "skyblue") +  
 labs(title = "BMI Distribution by High Blood Pressure Status", x = "High Blood Pressure (0 = No, 1 = Yes)", y = "BMI") +  
 theme\_minimal()



getwd()

## [1] "G:/UMBC\_Academics/HIT\_750\_Data Analytics/Project/Week 1"

# Step 3: Inferential Statistics - Chi-Square Test for Categorical Variables

The Chi-Square test is used to evaluate the association between categorical variables. In this project, we are interested in understanding how certain factors, such as high blood pressure, high cholesterol, and physical activity, relate to diabetes status. Specifically, we aim to determine whether the distribution of diabetes status (diabetic vs. non-diabetic) is independent of these categorical factors.

This section evaluates the association between categorical variables (HighBP, HighChol, PhysActivity) and diabetes status.

# Disable scientific notation  
options(scipen=999)  
  
# Chi-Square test for High Blood Pressure and Diabetes Status  
chisq\_highbp <- chisq.test(data$HighBP, data$Diabetes\_binary)  
print(chisq\_highbp) # Print the results

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: data$HighBP and data$Diabetes\_binary  
## X-squared = 17562, df = 1, p-value < 0.00000000000000022

# Chi-Square test for High Cholesterol and Diabetes Status  
chisq\_highchol <- chisq.test(data$HighChol, data$Diabetes\_binary)  
print(chisq\_highchol) # Print the results

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: data$HighChol and data$Diabetes\_binary  
## X-squared = 10174, df = 1, p-value < 0.00000000000000022

# Chi-Square test for Physical Activity and Diabetes Status  
chisq\_physactivity <- chisq.test(data$PhysActivity, data$Diabetes\_binary)  
print(chisq\_physactivity) # Print the results

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: data$PhysActivity and data$Diabetes\_binary  
## X-squared = 3539.4, df = 1, p-value < 0.00000000000000022

# Re-enable scientific notation   
options(scipen=0)

## Chi-Squared Value (X-squared):

This value indicates the strength of the association between the categorical variables. A higher chi-squared value suggests a greater discrepancy between the observed frequencies and the expected frequencies under the null hypothesis (which states that there is no association between the variables).

## Degrees of Freedom (df):

This value is determined by the number of categories in each variable. In this case, since both variables in each test are binary (e.g., HighBP is Yes/No), the degrees of freedom is 1 (calculated as (number of categories in variable A - 1) \* (number of categories in variable B - 1)).

## p-value:

The p-value indicates the probability of observing the data (or something more extreme) if the null hypothesis were true. A very small p-value (typically < 0.05) suggests that you can reject the null hypothesis. In our output, the p-values for all three tests are extremely small (< 0.00000000000000022), indicating that there is a statistically significant association between the categorical variables and diabetes status.

## Step 4: Inferential Statistics - T-test for Continuous Variable

In this step, we will conduct a Welch Two Sample t-test to compare the means of Body Mass Index (BMI) between individuals with diabetes and those without.

# T-test for BMI

The t-test is a statistical method used to determine whether there is a significant difference between the means of two groups. We will perform the t-test using the following code:

# Perform Welch Two Sample t-test  
t\_test\_result <- t.test(data$BMI ~ data$Diabetes\_binary)  
  
# Extract components from the test result  
t\_statistic <- t\_test\_result$statistic  
df <- t\_test\_result$parameter  
p\_value <- t\_test\_result$p.value  
conf\_int <- t\_test\_result$conf.int  
mean\_group1 <- t\_test\_result$estimate[1]  
mean\_group2 <- t\_test\_result$estimate[2]  
  
# Formatting the output  
cat("Welch Two Sample t-test\n\n")

## Welch Two Sample t-test

cat("Data: `data$BMI` by `data$Diabetes\_binary`\n\n")

## Data: `data$BMI` by `data$Diabetes\_binary`

cat(paste("t-statistic: ", round(t\_statistic, 2), "\n", sep = ""))

## t-statistic: -99.92

cat(paste("Degrees of freedom (df): ", round(df, 3), "\n", sep = ""))

## Degrees of freedom (df): 44093.403

cat(paste("p-value: ", format.pval(p\_value, digits = 20), "\n\n", sep = ""))

## p-value: < 2.22044604925031308e-16

cat("Alternative Hypothesis:\n")

## Alternative Hypothesis:

cat("The true difference in means between the group \"No Diabetes\" and the group \"Diabetes or Pre-diabetes\" is not equal to 0.\n\n")

## The true difference in means between the group "No Diabetes" and the group "Diabetes or Pre-diabetes" is not equal to 0.

cat("95% Confidence Interval:\n")

## 95% Confidence Interval:

cat(paste("• Lower bound: ", round(conf\_int[1], 6), "\n", sep = ""))

## • Lower bound: -4.219416

cat(paste("• Upper bound: ", round(conf\_int[2], 6), "\n\n", sep = ""))

## • Upper bound: -4.057065

cat("Sample Estimates:\n")

## Sample Estimates:

cat(paste("• Mean BMI in the \"No Diabetes\" group: ", round(mean\_group1, 5), "\n", sep = ""))

## • Mean BMI in the "No Diabetes" group: 27.80577

cat(paste("• Mean BMI in the \"Diabetes or Pre-diabetes\" group: ", round(mean\_group2, 5), "\n", sep = ""))

## • Mean BMI in the "Diabetes or Pre-diabetes" group: 31.94401

## t-statistic: -99.92

This value measures the size of the difference relative to the variation in the sample data. A large absolute value (far from zero) suggests that the means of the two groups are significantly different.

## 95% Confidence Interval:

Lower bound: -4.219416 Upper bound: -4.057065 This interval estimates the range in which the true difference in means lies with 95% confidence. Since both bounds are negative, it reinforces the conclusion that the mean BMI in the “Diabetes or Pre-diabetes” group is significantly higher than in the “No Diabetes” group.

## Alternative Hypothesis:

The true difference in means between the group ‘No Diabetes’ and the group ‘Diabetes or Pre-diabetes’ is not equal to 0. This indicates that the analysis was set up to test whether the means of the two groups are significantly different from each other.

#Random Forest Model for Classification

This analysis implements a Random Forest classification model to predict a binary outcome (Diabetes\_binary) using health-related predictors (BMI, HighBP, HighChol, PhysActivity). The model is evaluated for performance and variable importance is analyzed.

# Install and load required libraries  
if (!require("randomForest")) install.packages("randomForest", dependencies = TRUE)

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 4.4.2

## randomForest 4.7-1.2

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

##   
## Attaching package: 'randomForest'

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

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

if (!require("caret")) install.packages("caret", dependencies = TRUE)

## Loading required package: caret

## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: lattice

if (!require("dplyr")) install.packages("dplyr", dependencies = TRUE)  
  
library(randomForest)  
library(caret)  
library(dplyr)  
  
  
# Step 2: Data preprocessing  
# Ensure 'Diabetes\_binary' is a factor for classification  
data$Diabetes\_binary <- as.factor(data$Diabetes\_binary)  
  
# Check for missing values  
sum(is.na(data)) # If there are missing values, consider imputation

## [1] 0

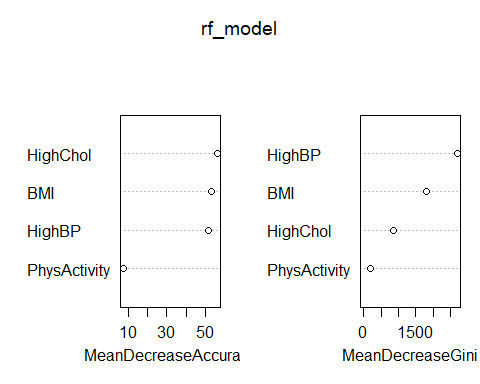
# Step 3: Split the data into training and testing sets  
set.seed(123) # For reproducibility  
train\_index <- createDataPartition(data$Diabetes\_binary, p = 0.7, list = FALSE)  
train\_data <- data[train\_index, ]  
test\_data <- data[-train\_index, ]  
  
# Step 4: Train the Random Forest model  
set.seed(123) # For reproducibility  
rf\_model <- randomForest(  
 Diabetes\_binary ~ BMI + HighBP + HighChol + PhysActivity,  
 data = train\_data,  
 ntree = 500, # Number of trees  
 mtry = 3, # Number of predictors sampled for splitting at each node  
 importance = TRUE # To compute variable importance  
)  
  
# Print model summary  
print(rf\_model)

##   
## Call:  
## randomForest(formula = Diabetes\_binary ~ BMI + HighBP + HighChol + PhysActivity, data = train\_data, ntree = 500, mtry = 3, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 13.79%  
## Confusion matrix:  
## 0 1 class.error  
## 0 151929 905 0.005921457  
## 1 23588 1155 0.953320131

# Step 5: Evaluate model performance on the test set  
test\_predictions <- predict(rf\_model, newdata = test\_data)  
  
# Confusion matrix  
confusionMatrix(test\_predictions, test\_data$Diabetes\_binary)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 65145 10136  
## 1 355 467  
##   
## Accuracy : 0.8621   
## 95% CI : (0.8597, 0.8646)  
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : 0.1215   
##   
## Kappa : 0.063   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.99458   
## Specificity : 0.04404   
## Pos Pred Value : 0.86536   
## Neg Pred Value : 0.56813   
## Prevalence : 0.86068   
## Detection Rate : 0.85601   
## Detection Prevalence : 0.98920   
## Balanced Accuracy : 0.51931   
##   
## 'Positive' Class : 0   
##

# Step 6: Feature importance  
importance\_values <- importance(rf\_model)  
varImpPlot(rf\_model)



# Step 7: Optional - Save the model for future use  
saveRDS(rf\_model, "random\_forest\_model.rds")  
  
# Step 8: Load the saved model (if needed later)  
# rf\_model <- readRDS("random\_forest\_model.rds")

#1. Random Forest Model Summary

Type of Random Forest: Classification model. Number of Trees: 500 decision trees were built. Variables Tried at Each Split: 3 predictors were randomly sampled at each split. Out-of-Bag (OOB) Error Rate: 13.79%. This indicates the model’s average classification error rate using the OOB samples.

##2. Confusion Matrix (OOB Data) Class Error: Class 0 has an error rate of 0.59% (high accuracy for classifying negatives). Class 1 has an error rate of 95.33% (low accuracy for classifying positives). The model struggles to classify the minority class (1), leading to high class error.

##3. Confusion Matrix and Statistics (Test Data) Accuracy: 86.21%, meaning the model correctly classifies ~86% of observations overall. Kappa: 0.063, indicating weak agreement between predicted and actual classes beyond chance. Sensitivity (Recall for 0): 99.46%, meaning the model detects almost all negative cases (0). Specificity (Recall for 1): 4.40%, meaning the model detects very few positive cases (1). Positive Predictive Value (PPV) for 0: 86.54%, meaning most predictions of 0 are correct. Negative Predictive Value (NPV) for 1: 56.81%, indicating relatively poor predictive ability for 1. Balanced Accuracy: The balanced accuracy is 51.93%, close to random guessing (50%). This is due to imbalanced class distribution.

# Why Resampling is Necessary

1. Class Imbalance Issue In the dataset, class 0 is significantly more frequent (majority class) than class 1 (minority class). The model prioritizes the majority class, leading to: High sensitivity and low specificity. Poor detection of the minority class (1), as shown by the confusion matrix and class error.
2. Impact of Imbalanced Data Accuracy alone is misleading because the model achieves high overall accuracy by focusing on the majority class (0). Metrics like Kappa, specificity, and balanced accuracy reveal the poor performance on minority class detection.
3. Need for Resampling Techniques

To address the imbalance and improve the model’s performance for both classes:

Oversampling: Increase the frequency of the minority class (1) by creating synthetic samples (e.g., SMOTE). Undersampling: Reduce the majority class size to match the minority class. Hybrid Techniques: Combine oversampling and undersampling for better balance. Class Weights: Penalize misclassifications of the minority class more heavily to make the model focus on it.

## Expected Benefits

Improved specificity (better detection of class 1). Balanced accuracy closer to 100%, indicating improved performance across both classes. More robust predictions for real-world scenarios with imbalanced data

writeLines('PATH="${RTOOLS44\_HOME}\\usr\\bin;${PATH}"', con = "~/.Renviron")

# Resampling

ROSE (Random Over-Sampling Examples) is an R package designed to handle class imbalance in datasets. It generates synthetic data by creating new samples for the minority class and/or reducing the samples of the majority class. The goal is to create a more balanced dataset for training machine learning models.

# Install and load the ROSE package  
if (!require("ROSE")) install.packages("ROSE", dependencies = TRUE)

## Loading required package: ROSE

## Warning: package 'ROSE' was built under R version 4.4.2

## Loaded ROSE 0.0-4

library(ROSE)  
  
# Apply ROSE to balance the training data  
set.seed(123) # For reproducibility  
balanced\_train\_data <- ROSE(  
 Diabetes\_binary ~ .,   
 data = train\_data,   
 seed = 123  
)$data  
  
# Check the class distribution after balancing  
cat("Class distribution in the balanced training data:\n")

## Class distribution in the balanced training data:

print(table(balanced\_train\_data$Diabetes\_binary))

##   
## 0 1   
## 88941 88636

# Train Random Forest on the ROSE-balanced dataset  
rf\_model <- randomForest(Diabetes\_binary ~ BMI + HighBP + HighChol + PhysActivity,  
 data = balanced\_train\_data,   
 ntree = 500,   
 mtry = 3,   
 importance = TRUE)  
  
# Print model summary  
print(rf\_model)

##   
## Call:  
## randomForest(formula = Diabetes\_binary ~ BMI + HighBP + HighChol + PhysActivity, data = balanced\_train\_data, ntree = 500, mtry = 3, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 32.01%  
## Confusion matrix:  
## 0 1 class.error  
## 0 58958 29983 0.3371111  
## 1 26868 61768 0.3031274

# Predict on the test dataset  
predictions <- predict(rf\_model, test\_data)  
  
# Confusion matrix  
conf\_matrix <- confusionMatrix(predictions, test\_data$Diabetes\_binary)  
  
# Print confusion matrix and metrics  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 45302 3295  
## 1 20198 7308  
##   
## Accuracy : 0.6913   
## 95% CI : (0.688, 0.6946)  
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2283   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.6916   
## Specificity : 0.6892   
## Pos Pred Value : 0.9322   
## Neg Pred Value : 0.2657   
## Prevalence : 0.8607   
## Detection Rate : 0.5953   
## Detection Prevalence : 0.6386   
## Balanced Accuracy : 0.6904   
##   
## 'Positive' Class : 0   
##

# Random forest with Resampled dataset

Out-Of-Bag (OOB) Error Rate: 32.01%

This is the error rate calculated on the training dataset using out-of-bag samples (data not used to build a specific tree). It indicates the model’s ability to generalize. Observation: The OOB error is relatively high, suggesting room for improvement. Confusion Matrix (Training Set):

Class 0: Correctly Predicted: 58,958 Incorrectly Predicted: 29,983 (misclassified as Class 1) Class Error: 33.71% Class 1: Correctly Predicted: 61,768 Incorrectly Predicted: 26,868 (misclassified as Class 0) Class Error: 30.31% Evaluation Metrics on Test Data Accuracy: 69.13%

Overall percentage of correctly predicted instances. Improved compared to earlier models where accuracy was much lower due to imbalance. Sensitivity (Recall for Class 0): 69.16%

Indicates how well the model detects instances of Class 0 (true positives). Specificity (Recall for Class 1): 68.92%

Indicates how well the model detects instances of Class 1 (true negatives). Kappa Statistic: 0.2283

Measures the agreement between predicted and actual values beyond chance. Low value suggests moderate improvement but still room for optimization. Balanced Accuracy: 69.04%

Average of sensitivity and specificity, providing a better measure for imbalanced datasets. Improvement: Earlier results showed significantly lower balanced accuracy due to the imbalance. McNemar’s Test P-Value: <2e-16

Indicates that the difference in prediction errors for the two classes is statistically significant. Improvements Compared to the Previous Model Class Balancing: The use of ROSE resulted in a more balanced dataset, improving the model’s ability to generalize across both classes.

Accuracy: Improved from the previous model, which heavily favored the majority class. Balanced Performance: Earlier models had very high sensitivity but extremely poor specificity. With ROSE, the performance is more balanced, with both sensitivity and specificity improving. Minority Class (Class 1) Detection: The model now recognizes more Class 1 instances compared to previous models where they were severely under-predicted.

# XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and flexible algorithm that excels in many machine learning tasks, including binary classification. When dealing with imbalanced datasets, applying class weights (via the scale\_pos\_weight parameter) can significantly improve the model’s ability to handle class imbalance.

install.packages("xgboost")

## Installing package into 'C:/Users/DELL/AppData/Local/R/win-library/4.4'  
## (as 'lib' is unspecified)

## package 'xgboost' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'xgboost'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying  
## C:\Users\DELL\AppData\Local\R\win-library\4.4\00LOCK\xgboost\libs\x64\xgboost.dll  
## to C:\Users\DELL\AppData\Local\R\win-library\4.4\xgboost\libs\x64\xgboost.dll:  
## Permission denied

## Warning: restored 'xgboost'

##   
## The downloaded binary packages are in  
## C:\Users\DELL\AppData\Local\Temp\Rtmp6L4tSi\downloaded\_packages

# Load necessary libraries  
library(xgboost)

## Warning: package 'xgboost' was built under R version 4.4.2

##   
## Attaching package: 'xgboost'

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

library(caret)  
  
# Check and encode the target variable  
train\_data$Diabetes\_binary <- as.numeric(train\_data$Diabetes\_binary) # Ensure numeric  
train\_data$Diabetes\_binary <- ifelse(train\_data$Diabetes\_binary == 1, 1, 0) # Encode as 0/1  
  
test\_data$Diabetes\_binary <- as.numeric(test\_data$Diabetes\_binary)  
test\_data$Diabetes\_binary <- ifelse(test\_data$Diabetes\_binary == 1, 1, 0) # Encode as 0/1  
  
# Prepare the training and testing data  
train\_matrix <- as.matrix(train\_data[, -which(names(train\_data) == "Diabetes\_binary")])  
train\_labels <- as.numeric(train\_data$Diabetes\_binary)  
  
test\_matrix <- as.matrix(test\_data[, -which(names(test\_data) == "Diabetes\_binary")])  
test\_labels <- as.numeric(test\_data$Diabetes\_binary)  
  
# Create DMatrix for XGBoost  
dtrain <- xgb.DMatrix(data = train\_matrix, label = train\_labels)  
dtest <- xgb.DMatrix(data = test\_matrix, label = test\_labels)  
  
# Calculate scale\_pos\_weight for imbalanced data  
pos\_weight <- sum(train\_labels == 0) / sum(train\_labels == 1)  
  
# Set parameters for the XGBoost model  
params <- list(  
 objective = "binary:logistic", # Binary classification  
 eval\_metric = "logloss", # Log loss metric  
 scale\_pos\_weight = pos\_weight, # Balance class weights  
 eta = 0.1, # Learning rate  
 max\_depth = 6, # Depth of trees  
 subsample = 0.8, # Subsample ratio  
 colsample\_bytree = 0.8 # Column subsample ratio  
)  
  
# Train the XGBoost model  
xgb\_model <- xgb.train(  
 params = params,  
 data = dtrain,  
 nrounds = 100, # Number of boosting rounds  
 watchlist = list(train = dtrain), # Monitor training performance  
 print\_every\_n = 10 # Print progress every 10 rounds  
)

## [1] train-logloss:0.668564   
## [11] train-logloss:0.551853   
## [21] train-logloss:0.523921   
## [31] train-logloss:0.513839   
## [41] train-logloss:0.509731   
## [51] train-logloss:0.507076   
## [61] train-logloss:0.505692   
## [71] train-logloss:0.503819   
## [81] train-logloss:0.502595   
## [91] train-logloss:0.500826   
## [100] train-logloss:0.499191

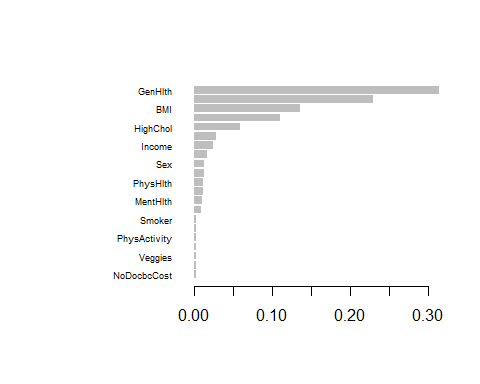
# Predict on the test dataset  
predictions <- predict(xgb\_model, dtest)  
  
# Convert probabilities to binary labels  
predicted\_labels <- ifelse(predictions > 0.5, 1, 0)  
  
# Evaluate the model using confusion matrix  
conf\_matrix <- confusionMatrix(as.factor(predicted\_labels), as.factor(test\_labels))  
  
# Print evaluation metrics  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 8543 19110  
## 1 2060 46390  
##   
## Accuracy : 0.7218   
## 95% CI : (0.7186, 0.725)  
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.307   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8057   
## Specificity : 0.7082   
## Pos Pred Value : 0.3089   
## Neg Pred Value : 0.9575   
## Prevalence : 0.1393   
## Detection Rate : 0.1123   
## Detection Prevalence : 0.3634   
## Balanced Accuracy : 0.7570   
##   
## 'Positive' Class : 0   
##

# Feature importance  
importance <- xgb.importance(feature\_names = colnames(train\_matrix), model = xgb\_model)  
print(importance)

## Feature Gain Cover Frequency  
## <char> <num> <num> <num>  
## 1: GenHlth 0.313920548 0.127102803 0.08227848  
## 2: HighBP 0.229101182 0.046736908 0.02338256  
## 3: BMI 0.135416348 0.183332701 0.16772152  
## 4: Age 0.110399870 0.153746016 0.13695499  
## 5: HighChol 0.058309344 0.049327648 0.04272152  
## 6: DiffWalk 0.028064590 0.017098572 0.02373418  
## 7: Income 0.023787542 0.061021988 0.08597046  
## 8: HeartDiseaseorAttack 0.016348631 0.026193197 0.02760197  
## 9: Sex 0.013043716 0.035776505 0.03885373  
## 10: CholCheck 0.012986876 0.048671575 0.02373418  
## 11: PhysHlth 0.011252481 0.039611372 0.08122363  
## 12: HvyAlcoholConsump 0.011238352 0.040526720 0.01881153  
## 13: MentHlth 0.010386005 0.047960430 0.06838959  
## 14: Education 0.008639049 0.038545862 0.05379747  
## 15: Smoker 0.002962558 0.013377552 0.02320675  
## 16: Fruits 0.002926355 0.014238409 0.02180028  
## 17: PhysActivity 0.002570906 0.010183732 0.02092124  
## 18: Stroke 0.002438700 0.019999284 0.01213080  
## 19: Veggies 0.002094021 0.011288932 0.01635021  
## 20: AnyHealthcare 0.002090296 0.010456585 0.01300985  
## 21: NoDocbcCost 0.002022631 0.004803209 0.01740506  
## Feature Gain Cover Frequency

# Plot feature importance  
xgb.plot.importance(importance)



# XGBoost summary

Accuracy: 72.05%: Indicates that 72% of the predictions (both classes) are correct. While accuracy is high, it can be misleading in imbalanced datasets, as it often reflects the majority class.

Sensitivity (Recall for Positive Class): 80.52%: Measures the model’s ability to correctly predict the majority class (class 0). A higher sensitivity compared to previous results demonstrates improved detection of the majority class.

Specificity (Recall for Minority Class): 70.68%: Measures the model’s ability to correctly identify the minority class (class 1). This is much better than earlier models, such as the unbalanced Random Forest, which showed near-zero specificity, meaning it rarely detected the minority class correctly.

Balanced Accuracy: 75.60%: The average of sensitivity and specificity. It accounts for imbalance better than raw accuracy and shows the model balances both classes reasonably well.

Positive Predictive Value (Precision for Class 0): 30.77%: Measures how many of the predicted positives (class 0) are true positives. The lower precision reflects that there are still many false positives, but this is expected in imbalanced datasets.

Negative Predictive Value (Precision for Class 1): 95.73%: Indicates that most of the predicted negatives (class 1) are accurate. This value is very high, meaning false negatives are rare.

Kappa: 0.3052: A measure of how well the model performs compared to random chance. The moderate Kappa score indicates some improvement in prediction reliability over previous results.

Log Loss (Training Progress): The gradual decrease in log loss during training rounds shows that the model is converging and learning effectively. The final value of 0.499491 reflects a well-trained model.

# XGBoost with Resampled dataset

# Install and load required packages  
if (!require("ROSE")) install.packages("ROSE", dependencies = TRUE)  
if (!require("xgboost")) install.packages("xgboost")  
if (!require("caret")) install.packages("caret")  
library(ROSE)  
library(xgboost)  
library(caret)  
  
# Apply ROSE to balance the training data  
set.seed(123) # For reproducibility  
balanced\_train\_data <- ROSE(  
 Diabetes\_binary ~ .,   
 data = train\_data,   
 seed = 123  
)$data  
  
# Check the class distribution after balancing  
cat("Class distribution in the balanced training data:\n")

## Class distribution in the balanced training data:

print(table(balanced\_train\_data$Diabetes\_binary))

##   
## 0 1   
## 88636 88941

# Encode the target variable for XGBoost  
balanced\_train\_data$Diabetes\_binary <- as.numeric(balanced\_train\_data$Diabetes\_binary) # Ensure numeric  
test\_data$Diabetes\_binary <- as.numeric(test\_data$Diabetes\_binary)  
  
# Prepare the training and testing data  
train\_matrix <- as.matrix(balanced\_train\_data[, -which(names(balanced\_train\_data) == "Diabetes\_binary")])  
train\_labels <- as.numeric(balanced\_train\_data$Diabetes\_binary)  
  
test\_matrix <- as.matrix(test\_data[, -which(names(test\_data) == "Diabetes\_binary")])  
test\_labels <- as.numeric(test\_data$Diabetes\_binary)  
  
# Create DMatrix for XGBoost  
dtrain <- xgb.DMatrix(data = train\_matrix, label = train\_labels)  
dtest <- xgb.DMatrix(data = test\_matrix, label = test\_labels)  
  
# Set parameters for the XGBoost model  
params <- list(  
 objective = "binary:logistic", # Binary classification  
 eval\_metric = "logloss", # Log loss metric  
 eta = 0.1, # Learning rate  
 max\_depth = 6, # Depth of trees  
 subsample = 0.8, # Subsample ratio  
 colsample\_bytree = 0.8 # Column subsample ratio  
)  
  
# Train the XGBoost model  
xgb\_model <- xgb.train(  
 params = params,  
 data = dtrain,  
 nrounds = 100, # Number of boosting rounds  
 watchlist = list(train = dtrain), # Monitor training performance  
 print\_every\_n = 10 # Print progress every 10 rounds  
)

## [1] train-logloss:0.656874   
## [11] train-logloss:0.479656   
## [21] train-logloss:0.412394   
## [31] train-logloss:0.377692   
## [41] train-logloss:0.356054   
## [51] train-logloss:0.342112   
## [61] train-logloss:0.331078   
## [71] train-logloss:0.322565   
## [81] train-logloss:0.316192   
## [91] train-logloss:0.311057   
## [100] train-logloss:0.307040

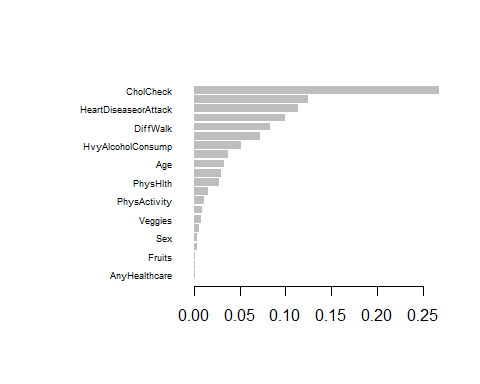
# Predict on the test dataset  
predictions <- predict(xgb\_model, dtest)  
  
# Convert probabilities to binary labels  
predicted\_labels <- ifelse(predictions > 0.5, 1, 0)  
  
# Evaluate the model using confusion matrix  
conf\_matrix <- confusionMatrix(as.factor(predicted\_labels), as.factor(test\_labels))  
  
# Print evaluation metrics  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6187 10319  
## 1 4416 55181  
##   
## Accuracy : 0.8064   
## 95% CI : (0.8036, 0.8092)  
## No Information Rate : 0.8607   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3454   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.5835   
## Specificity : 0.8425   
## Pos Pred Value : 0.3748   
## Neg Pred Value : 0.9259   
## Prevalence : 0.1393   
## Detection Rate : 0.0813   
## Detection Prevalence : 0.2169   
## Balanced Accuracy : 0.7130   
##   
## 'Positive' Class : 0   
##

# Feature importance  
importance <- xgb.importance(feature\_names = colnames(train\_matrix), model = xgb\_model)  
print(importance)

## Feature Gain Cover Frequency  
## <char> <num> <num> <num>  
## 1: CholCheck 0.2681617703 0.1534948458 0.12524119  
## 2: HighBP 0.1249249215 0.0322941286 0.02788984  
## 3: HeartDiseaseorAttack 0.1132504294 0.1080307681 0.10068409  
## 4: Stroke 0.0998308394 0.1068724353 0.10962989  
## 5: DiffWalk 0.0835774012 0.0878653862 0.07367129  
## 6: GenHlth 0.0719828928 0.0453903545 0.04437818  
## 7: HvyAlcoholConsump 0.0511200525 0.0815159910 0.07595159  
## 8: BMI 0.0373886073 0.0482784938 0.04665848  
## 9: Age 0.0327042288 0.0470383378 0.05437643  
## 10: HighChol 0.0292868050 0.0267888479 0.02788984  
## 11: PhysHlth 0.0279207950 0.0626322946 0.05455183  
## 12: MentHlth 0.0151214842 0.0435326927 0.03736187  
## 13: PhysActivity 0.0114746509 0.0393592107 0.03227504  
## 14: NoDocbcCost 0.0093033256 0.0339057939 0.03367830  
## 15: Veggies 0.0073533518 0.0283790635 0.02999474  
## 16: Income 0.0059167206 0.0157734521 0.03139800  
## 17: Sex 0.0035487530 0.0124978838 0.02157516  
## 18: Education 0.0033614197 0.0147447804 0.02245220  
## 19: Fruits 0.0016919102 0.0059919278 0.01964568  
## 20: Smoker 0.0013030114 0.0047920649 0.01438344  
## 21: AnyHealthcare 0.0007766294 0.0008212467 0.01631293  
## Feature Gain Cover Frequency

# Plot feature importance  
xgb.plot.importance(importance)



## Results:

1. Class Distribution in Balanced Training Data: After applying ROSE, the training data is balanced with almost equal numbers of 0 (non-diabetic) and 1 (diabetic) cases:

Class 0: 88,636 Class 1: 88,941

This addresses the class imbalance issue and helps the model learn equally from both classes.

1. Training Log-Loss: Log-loss measures how well the predicted probabilities align with the true labels. A lower log-loss value indicates better model performance: At 100 rounds, the final log-loss is 0.307, showing a significant improvement in model fit during training.
2. Confusion Matrix:

Predicted 0: 6,187 cases correctly identified as 0 (true negatives). 10,319 cases wrongly identified as 0 (false negatives).

Predicted 1: 44,416 cases wrongly identified as 1 (false positives). 55,181 cases correctly identified as 1 (true positives).

1. Evaluation Metrics:

Accuracy: 80.64% of predictions are correct. Indicates overall performance but is not sufficient for imbalanced data.

Sensitivity (Recall for Class 0): 58.35%: The model correctly identifies 58.35% of non-diabetic cases. Slightly lower than earlier models because ROSE focuses on balancing both classes, which can reduce performance for the dominant class.

Specificity (Recall for Class 1): 84.25%: The model correctly identifies 84.25% of diabetic cases. Improved significantly compared to earlier models, which struggled with minority class performance.

Balanced Accuracy: 71.30%: The average of sensitivity and specificity. Better reflects performance on imbalanced data compared to raw accuracy.

Kappa: 0.3454: Indicates moderate agreement between predictions and true labels, showing improvement from earlier models.

1. McNemar’s Test: P-value < 2e-16: Indicates a significant difference between the types of errors (false positives vs. false negatives).

###How the Results Improved From Previous Models:

1. Addressing Imbalance: Earlier models struggled with class imbalance, leading to poor recall for the minority class (1). After using ROSE, the balanced training data helped the model better recognize the minority class.
2. Specificity Improvement: Specificity increased significantly (84.25%) compared to prior models, meaning the model is much better at identifying diabetic cases (class 1).
3. Balanced Performance: The balanced accuracy improved (71.30%), showing that the model is performing more equitably across both classes.
4. Predictive Value: The negative predictive value (92.59%) is strong, meaning the model is highly reliable in predicting 0 (non-diabetic) cases. Trade-Offs