Group 8 Final Project

Group 8

2023-04-29

## R Markdown

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.0 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.1 ✔ tibble 3.1.8  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

library(dplyr)  
library(ggplot2)  
diabetes = read.csv("diabetes\_012\_health\_indicators\_BRFSS2015.csv")  
set.seed(90210)  
head(diabetes)

## Diabetes\_012 HighBP HighChol CholCheck BMI Smoker Stroke HeartDiseaseorAttack  
## 1 0 1 1 1 40 1 0 0  
## 2 0 0 0 0 25 1 0 0  
## 3 0 1 1 1 28 0 0 0  
## 4 0 1 0 1 27 0 0 0  
## 5 0 1 1 1 24 0 0 0  
## 6 0 1 1 1 25 1 0 0  
## PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare NoDocbcCost  
## 1 0 0 1 0 1 0  
## 2 1 0 0 0 0 1  
## 3 0 1 0 0 1 1  
## 4 1 1 1 0 1 0  
## 5 1 1 1 0 1 0  
## 6 1 1 1 0 1 0  
## GenHlth MentHlth PhysHlth DiffWalk Sex Age Education Income  
## 1 5 18 15 1 0 9 4 3  
## 2 3 0 0 0 0 7 6 1  
## 3 5 30 30 1 0 9 4 8  
## 4 2 0 0 0 0 11 3 6  
## 5 2 3 0 0 0 11 5 4  
## 6 2 0 2 0 1 10 6 8

str(diabetes)

## 'data.frame': 253680 obs. of 22 variables:  
## $ Diabetes\_012 : num 0 0 0 0 0 0 0 0 2 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 ...

summary(diabetes)

## Diabetes\_012 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 :0.2969 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. :2.0000 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

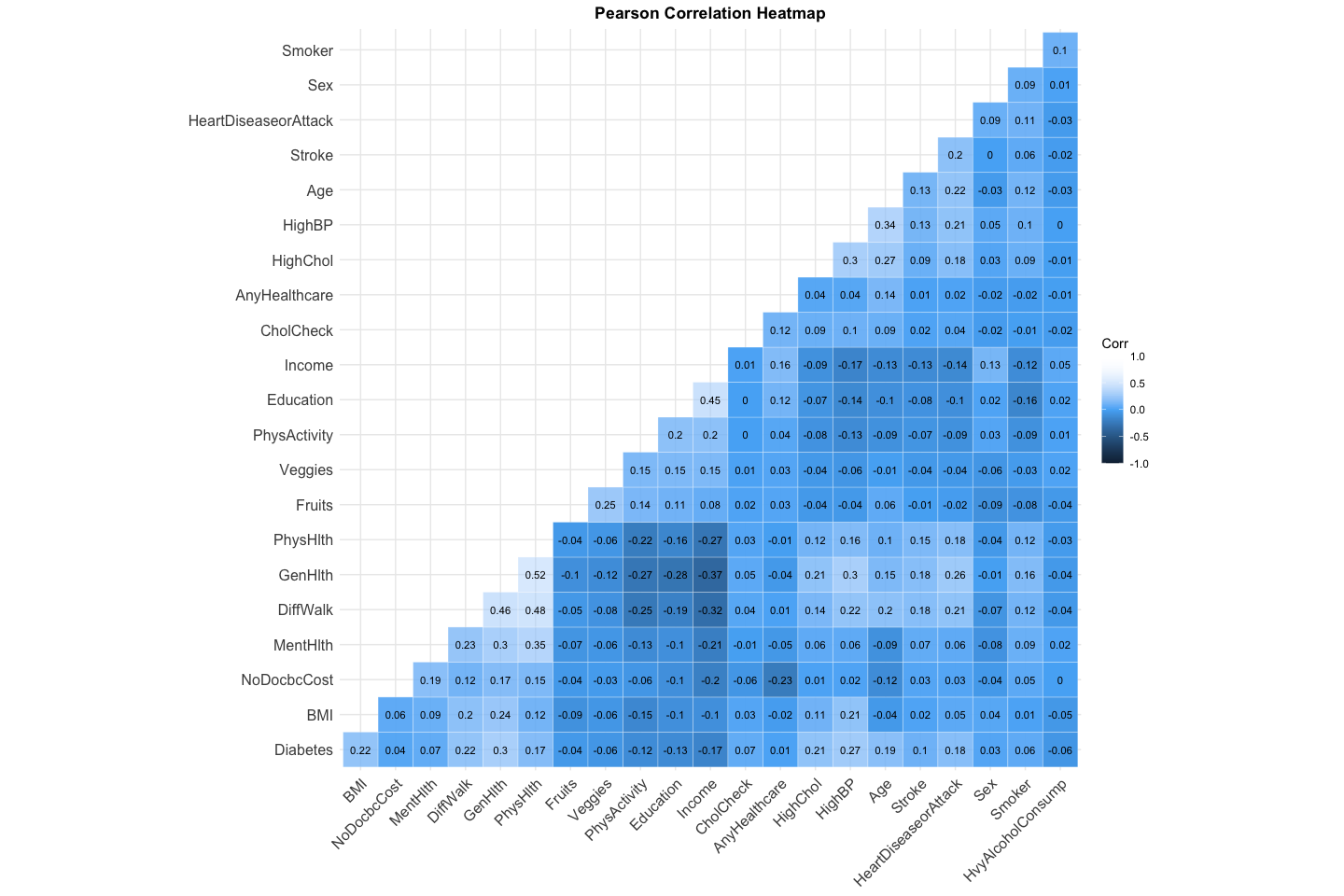
Data Description Diabetes\_012 : 0 = no diabetes 1 = prediabetes 2 = diabetes HighBP = 0 = no high BP 1 = high BP HighChol = 0 = no high cholesterol 1 = high cholesterol CholCheck = 0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years BMI = Body Mass Index Smoker = Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes] 0 = no 1 = yes Stroke = (Ever told) you had a stroke. 0 = no 1 = yes HeartDiseaseorAttack = coronary heart disease (CHD) or myocardial infarction (MI) 0 = no 1 = yes PhysActivity = physical activity in past 30 days - not including job 0 = no 1 = yes Fruits = Consume Fruit 1 or more times per day 0 = no 1 = yes Veggies = Consume Vegetables 1 or more times per day 0 = no 1 = yes HvyAlcoholConsump = Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week) 0 = no AnyHealthcare = Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc. 0 = no 1 = yes 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 GenHlth = Would you say that in general your health is: scale 1-5 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor MentlHlth = 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? PhysHlth = Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 DiffWalk = Do you have serious difficulty walking or climbing stairs? 0 = no 1 = yes Sex = 0 = female 1 = male Age = 13-level age category (\_AGEG5YR see codebook) 1 = 18-24 9 = 60-64 13 = 80 or older Education = Education level (EDUCA see codebook) scale 1-6 1 = Never attended school or only kindergarten 2 = Grades 1 through 8 Income = Income scale (INCOME2 see codebook) scale 1-8 1 = less than $10,000 5 = less than $35,000 8 = $75,000 or more

## Data Processing before EDA

We can observe that target variable diabetes\_012 has 3 classes. For the binary classification pre-diabetic group has be added to either non-diabetic or diabetic group. I would like to add it to the diabetic group since being told a doctor that patient being pre-diabetic can be a good indicator to quantify the risk of being diabetic in future.

set.seed(90210)  
diabetes\_clean = diabetes  
diabetes\_clean$Diabetes\_012 = ifelse(diabetes\_clean$Diabetes\_012 == 0, 0, 1)  
diabetes\_clean = diabetes\_clean %>% rename(Diabetes = Diabetes\_012)

set.seed(90210)  
# Load required libraries  
library(ggcorrplot)  
  
# Select only numeric columns  
numeric\_df <- diabetes\_clean %>%  
 select\_if(is.numeric)  
  
# Calculate the correlation matrix using Spearman method  
cor\_matrix <- cor(numeric\_df, method = "pearson")  
  
# Create a mask for the upper triangle  
mask <- upper.tri(cor\_matrix)  
  
# Plot the correlation heatmap  
ggcorrplot(cor\_matrix, hc.order = TRUE, type = "lower", outline.col = "white",  
 lab = TRUE, lab\_size = 3, lab\_col = "black",  
 colors = c("#132B43", "#56B1F6"),  
 title = "Pearson Correlation Heatmap",  
 ggtheme = theme\_minimal()) +  
 theme(plot.title = element\_text(hjust = 0.5, face = "bold"))

 Variables that have high correlation with Diabetes: BMI, Diffwalk, Physhlth, HighChol, HighBP, and Ag, education, GenHlth, income

##Convert to factor variables

All the variables except for BMI, MentlHlth, PhysHlth are categorical in nature : answering the question with 1 : Yes, 0: No. Apart from these variables that are numerical the rest of the variables are converted to factors.

set.seed(90210)  
diabetes\_clean = diabetes\_clean %>% mutate(across(!c("BMI","MentHlth","PhysHlth"), as.factor))

##check for missing and duplicate values if any

set.seed(90210)  
sum(is.na(diabetes))

## [1] 0

set.seed(90210)  
sum(duplicated(diabetes\_clean))

## [1] 23968

set.seed(90210)  
diabetes\_clean = diabetes\_clean %>% distinct()

set.seed(90210)  
sum(duplicated(diabetes\_clean))

## [1] 0

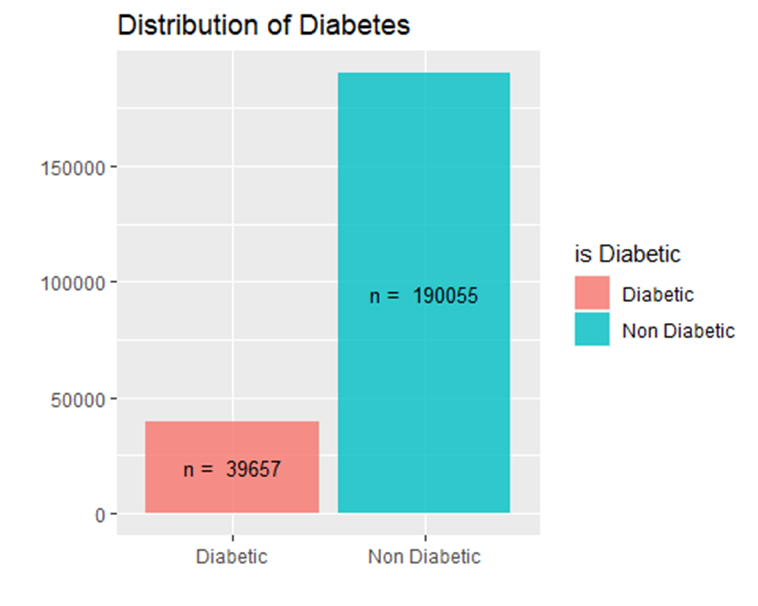
## EDA

Exploratory data analysis needs to be performed to analyze the distribution of variables and their effect on the response variable.

#distribution of diabetes

set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "Distribution of diabetes", x= "", y= "", fill="is Diabetic")

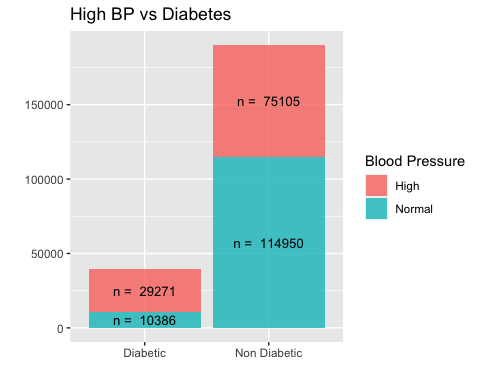
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.  
## ℹ Please use `after\_stat(count)` instead.



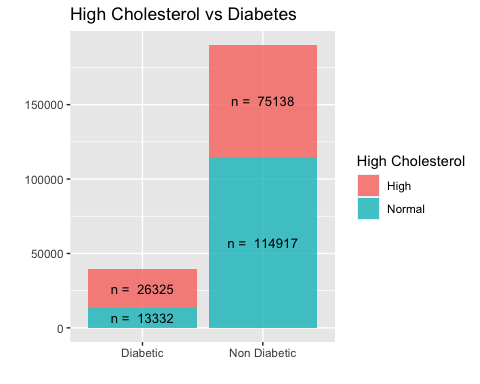
The data is imbalanced with 17% of the respondents having diabetics. Measures to be taken to balance the dataset. Techniques like Sampling methods, SMOTE, ROSE package over sampling or under sampling can be used.

#High Bp vs is Diabetic

set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(HighBP == 1, "High", "Normal"))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "High BP vs Diabetes", x= "", y= "", fill="Blood Pressure")

   
High BP is observed more in people with diabetes.   
  
#High Cholesterol

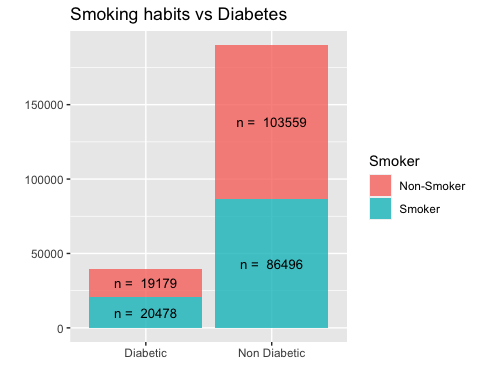
set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(HighChol == 1, "High", "Normal"))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "High Cholesterol vs Diabetes", x= "", y= "", fill="High Cholesterol")



More than 60% of the people who are diabetic have high cholesterol. There could to be an association with prevalence of high Cholesterol and Diabetes.

#Smoking vs diabetes

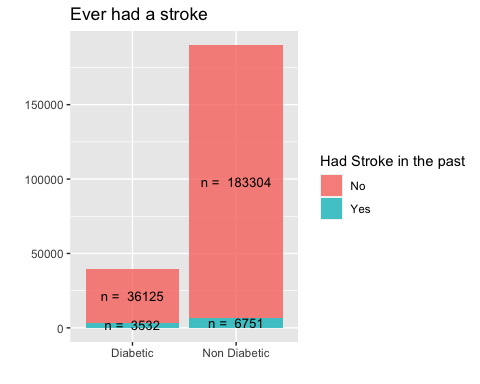
set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(Smoker == 1, "Smoker", "Non-Smoker"))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "Smoking habits vs Diabetes", x= "", y= "", fill="Smoker")



It appears that the distribution of smoking habits is almost equal among people with or with out diabetes.

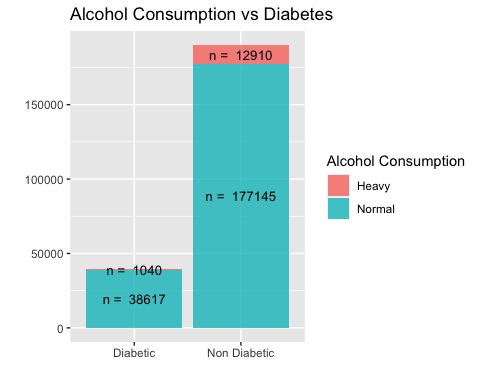
#Ever told you had a Stroke In the past

set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(Stroke == 1, "Yes", "No"))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "Ever had a stroke", x= "", y= "", fill="Had Stroke in the past")

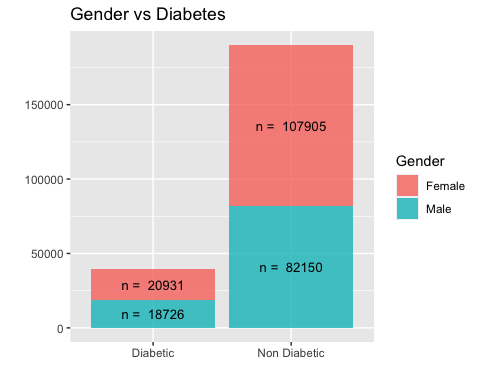
   
Having a stroke in the past doesn’t seem to be indicator for the risk of being diabetic, as observed from the graph, less than 10% of the people who had stroke in the past where diabetic.

#Alcohol Consumption

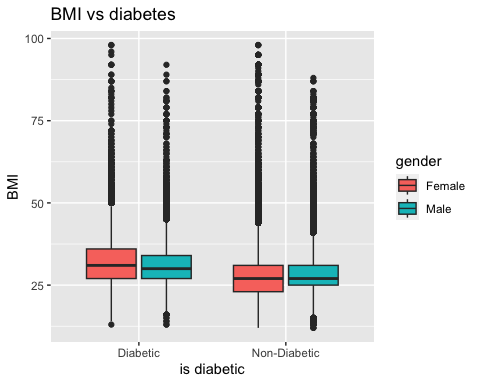
set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(HvyAlcoholConsump == 1, "Heavy", "Normal"))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "Alcohol Consumption vs Diabetes", x= "", y= "", fill="Alcohol Consumption")

   
Very few people who consume high amount of alcohol have diabetes. #Prevalence of Diabetes Gender wise

set.seed(90210)  
diabetes\_clean %>%   
 ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic", "Non Diabetic" ),   
 fill = ifelse(Sex == 0, "Female", "Male"))) +  
 geom\_bar(stat="count", alpha = 0.8) +   
 stat\_count(geom = "text", colour = "black", size = 3.5,  
 aes(label = paste("n = ", ..count..)),  
 position=position\_stack(vjust=0.5)) +  
 labs(title = "Gender vs Diabetes", x= "", y= "", fill="Gender")

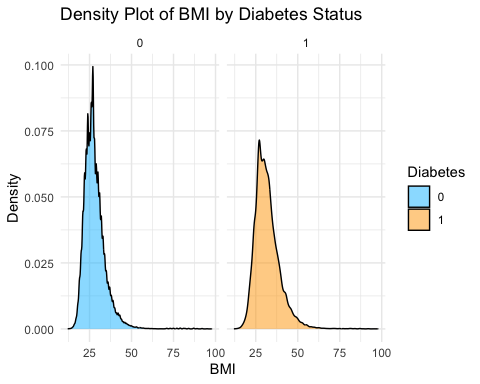
   
Diabetes seem to be more prevalent among women than in men. But if the difference is significant can be further investigated #BMI Vs Diabetic

set.seed(90210)  
diabetes\_clean %>% ggplot(aes(x = ifelse(Diabetes == 1, "Diabetic","Non-Diabetic"), y = BMI, fill = ifelse(Sex == 0, "Female", "Male"))) +   
 geom\_boxplot() +   
 labs(title = "BMI vs diabetes", x = "is diabetic", y ="BMI", fill = "gender")

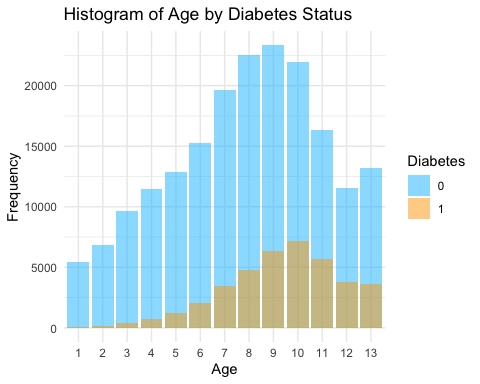


From the plot above, both females and males who are diabetic have higher BMI than those who are not-diabetic

set.seed(90210)  
# Create a density plot for BMI, separated by Diabetes  
ggplot(diabetes\_clean, aes(x = BMI, fill = as.factor(Diabetes))) +  
 geom\_density(alpha = 0.5) +  
 facet\_wrap(~ Diabetes) +  
 theme\_minimal() +  
 labs(x = "BMI", y = "Density", fill = "Diabetes", title = "Density Plot of BMI by Diabetes Status") +  
 scale\_fill\_manual(values = c("deepskyblue", "orange"), labels = c("0", "1"))



set.seed(90210)  
# Create a histogram for Age, separated by Diabetes  
ggplot(diabetes\_clean, aes(x = Age, fill = as.factor(Diabetes))) +  
 geom\_bar(alpha = 0.5, position = "identity") +  
 theme\_minimal() +  
 labs(x = "Age", y = "Frequency", fill = "Diabetes", title = "Histogram of Age by Diabetes Status") +  
 scale\_fill\_manual(values = c("deepskyblue", "orange"), labels = c("0", "1"))



Comment : From the exploratory analysis above, HighChol, High BP and BMI seems to be useful predictors. I will run a random forest model on the data, and pick the important variables from variable importance plots and further investigate them.

## Chi-Square Tests

set.seed(90210)  
chisq.test(diabetes\_clean$Age, diabetes\_clean$Diabetes)

##   
## Pearson's Chi-squared test  
##   
## data: diabetes\_clean$Age and diabetes\_clean$Diabetes  
## X-squared = 8844.9, df = 12, p-value < 2.2e-16

Comment : Age is a categorical variable, where the age of the respondents is divided into categories as mentioned in the variable description. Chi-Squared test show that there is an association between age and being diabetic.

set.seed(90210)  
chisq.test(diabetes\_clean$GenHlth, diabetes\_clean$Diabetes)

##   
## Pearson's Chi-squared test  
##   
## data: diabetes\_clean$GenHlth and diabetes\_clean$Diabetes  
## X-squared = 18940, df = 4, p-value < 2.2e-16

Comment : General helath and dibatetes are associated with each other as per the above results.

## Chi-square test for Sex :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 212.51, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for HighChol :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 9588.7, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for CholCheck :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 1329.6, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for Smoker :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 494.73, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for HeartDiseaseorAttack :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 6430.5, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for PhysActivity :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 2428.1, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for Fruits :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 146.4, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for Veggies :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 437.57, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for HvyAlcoholConsump :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 999.65, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for DiffWalk :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 9991.8, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for Stroke :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 2198.5, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for HighBP :  
##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 15562, df = 1, p-value < 2.2e-16  
##   
##   
## Chi-square test for Income :  
##   
## Pearson's Chi-squared test  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 5153, df = 7, p-value < 2.2e-16  
##   
##   
## Chi-square test for Education :  
##   
## Pearson's Chi-squared test  
##   
## data: table(diabetes\_clean$Diabetes, diabetes\_clean[[var]])  
## X-squared = 2856.1, df = 5, p-value < 2.2e-16

##Balancing datset

#Balancing the whole dataset

set.seed(90210)  
library(ROSE)

## Loaded ROSE 0.0-4

diabetes\_bal <- ROSE(Diabetes ~ ., data = diabetes\_clean, seed = 90210)$data  
table(diabetes\_bal$Diabetes)

##   
## 0 1   
## 114994 114718

#Smaller sample

set.seed(90210)  
index.small = sample(1:nrow(diabetes\_bal), size = 100000)  
small.df = diabetes\_bal[index.small,]  
nrow(small.df)

## [1] 100000

#Train-test split

set.seed(90210)  
index = sample(1:nrow(small.df), size = 0.7 \* nrow(small.df))  
train.df = small.df[index,]  
test.df = small.df[-index,]

## KNN Model training

set.seed(90210)  
vars = c("HighChol", "GenHlth", "Age", "HighBP", "BMI", "PhysHlth","DiffWalk")  
  
#diabetes\_clean  
hyp\_data <- small.df  
training.samples <- caret::createDataPartition(hyp\_data$Diabetes, p = 0.7, list = FALSE)  
train.data <- hyp\_data[training.samples, vars]  
test.data <- hyp\_data[-training.samples, vars]  
train.labels <- hyp\_data[training.samples, "Diabetes"]  
test.labels <- hyp\_data[-training.samples, "Diabetes"]  
  
  
  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(randomForest)

## randomForest 4.7-1.1

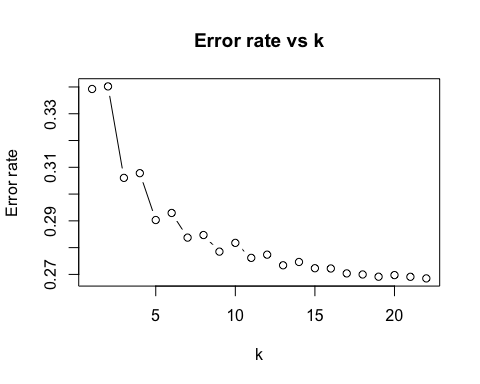
## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

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

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

library(e1071)  
  
library(class)  
  
  
########  
  
library(class)  
  
k\_values <- 1:22  
  
error\_rates <- vector(mode = "numeric", length = length(k\_values))  
  
for (i in seq\_along(k\_values)) {  
 k <- k\_values[i]  
 predictions <- knn(train.data, test.data, train.labels, k)  
 error\_rate <- sum(predictions != test.labels) / length(predictions)  
 error\_rates[i] <- error\_rate  
}  
  
plot(k\_values, error\_rates, type = "b", xlab = "k", ylab = "Error rate",  
 main = "Error rate vs k")



## Running KNN model with k value having minimum error rate  
k <- 19 # Set number of neighbors  
model <- knn(train.data, test.data, train.labels, k)  
  
# Evaluate performance on test data  
accuracy <- mean(model == test.labels)  
  
cat("Accuracy:", accuracy)

## Accuracy: 0.7308244

conf\_mat <- confusionMatrix(factor(model), factor(test.labels), positive = "1")  
#conf\_mat <- confusionMatrix(model, test.labels)  
accuracy <- conf\_mat$overall["Accuracy"]  
  
# Calculate sensitivity, precision, and specificity  
sensitivity <- conf\_mat$byClass["Sensitivity"]  
precision <- conf\_mat$byClass["Precision"]  
specificity <- conf\_mat$byClass["Specificity"]  
f1\_score <- 2 \* precision \* sensitivity / (precision + sensitivity)  
  
cat("Accuracy:", accuracy, "\n")

## Accuracy: 0.7308244

cat("Sensitivity:", sensitivity, "\n")

## Sensitivity: 0.7656125

cat("Precision:", precision, "\n")

## Precision: 0.717068

cat("Specificity:", specificity, "\n")

## Specificity: 0.6957918

cat("F1 score:", f1\_score, "\n")

## F1 score: 0.7405456

# KNN model  
library(class)  
  
## Cross validation  
  
# Set up 5-fold cross-validation  
cv <- trainControl(method = "cv", number = 5)  
  
# Fit KNN model with cross-validation  
knn\_fit <- train(train.data, train.labels, method = "knn", trControl = cv)  
# Fit KNN model with k=19  
knn\_fit <- train(train.data, train.labels, method = "knn", trControl = cv, tuneGrid = data.frame(k = 19))  
  
# Make predictions on test set  
knn\_pred <- predict(knn\_fit, newdata = test.data)  
  
# Evaluate performance on test set  
conf\_mat <- confusionMatrix(factor(knn\_pred, levels = levels(factor(test.labels))), factor(test.labels), positive = "1")  
  
#conf\_mat <- confusionMatrix(knn\_pred, test.labels)  
accuracy <- conf\_mat$overall["Accuracy"]  
sensitivity <- conf\_mat$byClass["Sensitivity"]  
precision <- conf\_mat$byClass["Precision"]  
specificity <- conf\_mat$byClass["Specificity"]  
f1\_score <- conf\_mat$byClass["F1"]  
  
# Print results  
cat("Accuracy:", accuracy, "\n")

## Accuracy: 0.7308577

cat("Sensitivity:", sensitivity, "\n")

## Sensitivity: 0.7655461

cat("Precision:", precision, "\n")

## Precision: 0.7171397

cat("Specificity:", specificity, "\n")

## Specificity: 0.6959256

cat("F1 score:", f1\_score, "\n")

## F1 score: 0.7405527

## Naive bayes Model Training

Note After the extensive EDA, I pick BMI, PhysHlth, MentHlth, Age, GenHlth, Income, HighBP, Education and HighChol as my predictors for the models.

##Naive bayes

set.seed(90210)  
library(naivebayes)

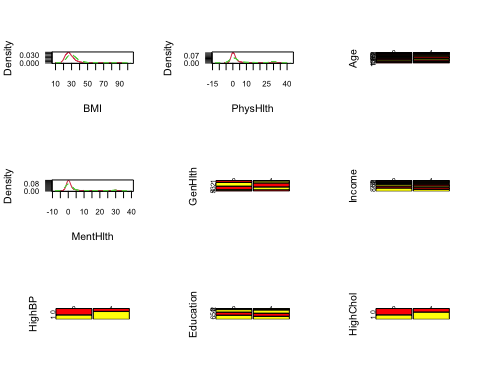
## naivebayes 0.9.7 loaded

nb\_model0 = naive\_bayes(Diabetes ~ BMI +   
 PhysHlth +   
 Age +   
 MentHlth +   
 GenHlth +   
 Income +   
 HighBP +   
 Education + HighChol, data = train.df, usekernel = TRUE)

summary(nb\_model0)

##   
## ================================== Naive Bayes ==================================   
##   
## - Call: naive\_bayes.formula(formula = Diabetes ~ BMI + PhysHlth + Age + MentHlth + GenHlth + Income + HighBP + Education + HighChol, data = train.df, usekernel = TRUE)   
## - Laplace: 0   
## - Classes: 2   
## - Samples: 70000   
## - Features: 9   
## - Conditional distributions:   
## - Bernoulli: 2  
## - Categorical: 4  
## - KDE: 3  
## - Prior probabilities:   
## - 0: 0.4995  
## - 1: 0.5005  
##   
## ---------------------------------------------------------------------------------

set.seed(90210)  
par(mfrow = c(3,3))  
plot(nb\_model0)



set.seed(90210)  
nb\_prob = predict(nb\_model0, test.df)

## Warning: predict.naive\_bayes(): more features in the newdata are provided as  
## there are probability tables in the object. Calculation is performed based on  
## features to be found in the tables.

set.seed(90210)  
caret::confusionMatrix(as.factor(nb\_prob), test.df$Diabetes, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 10778 3904  
## 1 4080 11238  
##   
## Accuracy : 0.7339   
## 95% CI : (0.7288, 0.7389)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.4676   
##   
## Mcnemar's Test P-Value : 0.05017   
##   
## Sensitivity : 0.7422   
## Specificity : 0.7254   
## Pos Pred Value : 0.7336   
## Neg Pred Value : 0.7341   
## Prevalence : 0.5047   
## Detection Rate : 0.3746   
## Detection Prevalence : 0.5106   
## Balanced Accuracy : 0.7338   
##   
## 'Positive' Class : 1   
##

set.seed(90210)  
library(class)  
conf\_mat = caret::confusionMatrix(as.factor(nb\_prob), test.df$Diabetes, positive = "1")  
precision <- conf\_mat$byClass["Precision"]  
sensitivity <- conf\_mat$byClass["Sensitivity"]  
precision <- conf\_mat$byClass["Precision"]  
specificity <- conf\_mat$byClass["Specificity"]  
f1\_score <- (2 \* precision \* sensitivity / (precision + sensitivity))  
print("f1 score:")

## [1] "f1 score:"

f1\_score

## Precision   
## 0.7378858

precision

## Precision   
## 0.7336467

Conclusion from above:

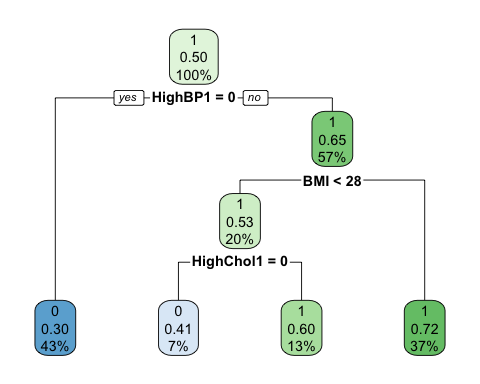
From above, we can say that the Naive Bayes model has high specificity. Risk factors for diabetes include high blood pressure, cholesterol, age, and BMI. Higher income levels are associated with a lower risk of diabetes due to the availability of medical services and a healthy diet. Good general health, physical health, and mental health are associated with a lower risk of diabetes. Mitigating the risk of diabetes involves focusing on blood pressure, cholesterol, weight management, availability of medical services, and overall health.

## Tree

set.seed(90210)  
library(caret)  
library(rpart)  
library(rpart.plot)  
tree\_model00\_t <- train(Diabetes ~ ., data = train.df, method = "rpart")  
tree\_model00\_t

## CART   
##   
## 70000 samples  
## 21 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 70000, 70000, 70000, 70000, 70000, 70000, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.01238275 0.6796498 0.3593052  
## 0.01328357 0.6761710 0.3523959  
## 0.34122626 0.5800758 0.1617363  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.01238275.

rpart.plot(tree\_model00\_t$finalModel)



tree\_pred00\_t <- predict(tree\_model00\_t, newdata = test.df) # predict class  
  
# Confusion matrix  
conf\_mat00\_t <- caret::confusionMatrix(tree\_pred00\_t, factor(test.df$Diabetes),  
 positive = "1")  
unique(test.df$Diabetes)

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

# Calculate sensitivity, precision, and specificity  
sensitivity00\_t <- conf\_mat00\_t$byClass["Sensitivity"]  
precision00\_t <- conf\_mat00\_t$byClass["Precision"]  
specificity00\_t <- conf\_mat00\_t$byClass["Specificity"]  
f1\_score00\_t <- (2 \* precision00\_t[[1]] \* sensitivity00\_t[[1]] /  
 (precision00\_t[[1]] + sensitivity00\_t[[1]]))  
  
conf\_mat00\_t

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 10221 4902  
## 1 4637 10240  
##   
## Accuracy : 0.682   
## 95% CI : (0.6767, 0.6873)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3641   
##   
## Mcnemar's Test P-Value : 0.006871   
##   
## Sensitivity : 0.6763   
## Specificity : 0.6879   
## Pos Pred Value : 0.6883   
## Neg Pred Value : 0.6759   
## Prevalence : 0.5047   
## Detection Rate : 0.3413   
## Detection Prevalence : 0.4959   
## Balanced Accuracy : 0.6821   
##   
## 'Positive' Class : 1   
##

print("f1 score:")

## [1] "f1 score:"

f1\_score00\_t

## [1] 0.6822346

The greedy tree results in a tree with 3 nodes and extremely poor node purity. No amount of variable selection or manipulation of the data resulted in a larger, more pure, or more accurate tree. The variables selected for use in the tree were HighBP, DiffWalk, BMI, and HighChol. The overall accuracy of 0.6820 is noticeably lower then the results from other methods (most notably the random forest). No cross-validation pruning of this tree was attempted, as pruning a tree with only 3 nodes isn’t useful.

## Random Forest

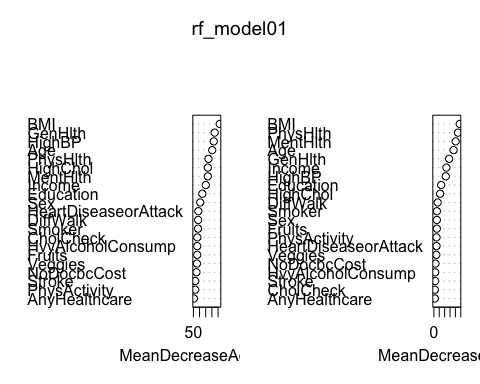
set.seed(90210)  
library(randomForest)  
rf\_model01 <- randomForest(Diabetes ~ ., data = train.df,  
 importance = TRUE, ntree = 1000)  
rf\_model01

##   
## Call:  
## randomForest(formula = Diabetes ~ ., data = train.df, importance = TRUE, ntree = 1000)   
## Type of random forest: classification  
## Number of trees: 1000  
## No. of variables tried at each split: 4  
##   
## OOB estimate of error rate: 22.33%  
## Confusion matrix:  
## 0 1 class.error  
## 0 25705 9263 0.2648993  
## 1 6369 28663 0.1818052

rf\_model01$importance

## 0 1 MeanDecreaseAccuracy  
## HighBP 0.0241287333 0.039035439 0.031589828  
## HighChol 0.0180172890 0.027475380 0.022751237  
## CholCheck 0.0013159432 0.003370001 0.002343921  
## BMI 0.0258976827 0.040515973 0.033213468  
## Smoker 0.0025312480 0.006745264 0.004640683  
## Stroke 0.0028318157 0.001836068 0.002333347  
## HeartDiseaseorAttack 0.0123707977 0.007530701 0.009948313  
## PhysActivity 0.0033886665 0.007218725 0.005305670  
## Fruits 0.0023164642 0.006339317 0.004329836  
## Veggies 0.0021262605 0.004608104 0.003368409  
## HvyAlcoholConsump 0.0013458170 0.003885867 0.002617134  
## AnyHealthcare 0.0007137726 0.001293843 0.001004160  
## NoDocbcCost 0.0012828878 0.003046401 0.002165417  
## GenHlth 0.0493109451 0.043391592 0.046348861  
## MentHlth 0.0155451947 0.014720825 0.015131835  
## PhysHlth 0.0255402653 0.019453463 0.022493399  
## DiffWalk 0.0135462729 0.012808673 0.013176591  
## Sex 0.0037184936 0.009621749 0.006673022  
## Age 0.0324207984 0.035860339 0.034141917  
## Education 0.0053461632 0.013471882 0.009413343  
## Income 0.0104540175 0.022945200 0.016706260  
## MeanDecreaseGini  
## HighBP 2218.0766  
## HighChol 1162.3352  
## CholCheck 181.6507  
## BMI 4810.1289  
## Smoker 559.1783  
## Stroke 237.2294  
## HeartDiseaseorAttack 493.2274  
## PhysActivity 497.4012  
## Fruits 548.3414  
## Veggies 454.9149  
## HvyAlcoholConsump 244.1683  
## AnyHealthcare 180.5960  
## NoDocbcCost 293.8641  
## GenHlth 2833.1799  
## MentHlth 4046.6102  
## PhysHlth 4470.5275  
## DiffWalk 702.7747  
## Sex 552.6046  
## Age 3659.9530  
## Education 1353.9917  
## Income 2329.1374

varImpPlot(rf\_model01)



rf\_pred01 <- predict(rf\_model01, newdata = test.df) # predict class  
  
# Confusion matrix  
conf\_mat01 <- caret::confusionMatrix(as.factor(rf\_pred01),  
 as.factor(test.df$Diabetes),  
 positive = "1")  
  
# Calculate sensitivity, precision, and specificity  
sensitivity01 <- conf\_mat01$byClass["Sensitivity"]  
precision01 <- conf\_mat01$byClass["Precision"]  
specificity01 <- conf\_mat01$byClass["Specificity"]  
f1\_score01 <- (2 \* precision01[[1]] \* sensitivity01[[1]] /  
 (precision01[[1]] + sensitivity01[[1]]))  
  
conf\_mat01

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 10918 2793  
## 1 3940 12349  
##   
## Accuracy : 0.7756   
## 95% CI : (0.7708, 0.7803)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5508   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.8155   
## Specificity : 0.7348   
## Pos Pred Value : 0.7581   
## Neg Pred Value : 0.7963   
## Prevalence : 0.5047   
## Detection Rate : 0.4116   
## Detection Prevalence : 0.5430   
## Balanced Accuracy : 0.7752   
##   
## 'Positive' Class : 1   
##

print("f1 score:")

## [1] "f1 score:"

f1\_score01

## [1] 0.7857847

The random forest model achieves significantly higher prediction accuracy than the tree-based model, at 0.7756. Many of the variables which were identified as significant to the other models, including BMI, PhysHlth, MentHlth, Age, GenHlth, Income, and HighBP were identified by this model as important. This is comforting and reinforces the variable selections made in other methods. This model also exhibits slightly higher sensitivity (0.8155) than specificity (0.7348), which is desirable in a predictive model for a costly condition like diabetes, where the risk reduction steps which can be taken have almost no cost if undertaken in a healthy person in the case of a false positive.

## Bagging

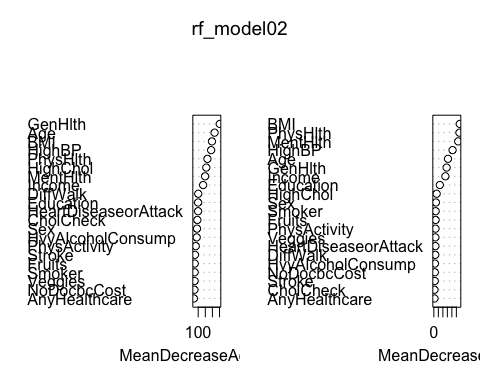
set.seed(90210)  
rf\_model02 = randomForest(Diabetes ~ ., data = train.df, mtry = 21,  
 importance = TRUE, ntree = 1000)  
rf\_model02

##   
## Call:  
## randomForest(formula = Diabetes ~ ., data = train.df, mtry = 21, importance = TRUE, ntree = 1000)   
## Type of random forest: classification  
## Number of trees: 1000  
## No. of variables tried at each split: 21  
##   
## OOB estimate of error rate: 23.98%  
## Confusion matrix:  
## 0 1 class.error  
## 0 25490 9478 0.2710478  
## 1 7307 27725 0.2085807

rf\_model02$importance

## 0 1 MeanDecreaseAccuracy  
## HighBP 0.0202875784 0.0343869420 0.0273413709  
## HighChol 0.0147837822 0.0216213934 0.0182048960  
## CholCheck 0.0004752333 0.0018784560 0.0011772934  
## BMI 0.0256548694 0.0388225183 0.0322431674  
## Smoker 0.0009633657 0.0021219355 0.0015429820  
## Stroke 0.0021086460 0.0002615117 0.0011846959  
## HeartDiseaseorAttack 0.0065623529 0.0020634522 0.0043114298  
## PhysActivity 0.0018373634 0.0020422887 0.0019395910  
## Fruits 0.0008555826 0.0020261644 0.0014412664  
## Veggies 0.0010690265 0.0014693895 0.0012693072  
## HvyAlcoholConsump 0.0006187403 0.0023515964 0.0014858118  
## AnyHealthcare 0.0003773726 0.0005056032 0.0004416239  
## NoDocbcCost 0.0006919504 0.0009669807 0.0008295157  
## GenHlth 0.0506094702 0.0441775547 0.0473902477  
## MentHlth 0.0173053203 0.0173235842 0.0173141051  
## PhysHlth 0.0264914939 0.0207311836 0.0236084247  
## DiffWalk 0.0053138572 0.0031427715 0.0042274457  
## Sex 0.0023150464 0.0042430426 0.0032795462  
## Age 0.0339918403 0.0350540237 0.0345225824  
## Education 0.0039017053 0.0073502400 0.0056271966  
## Income 0.0093024154 0.0171218431 0.0132142911  
## MeanDecreaseGini  
## HighBP 4158.3435  
## HighChol 619.8644  
## CholCheck 140.5508  
## BMI 5755.8983  
## Smoker 401.0843  
## Stroke 198.4156  
## HeartDiseaseorAttack 336.4294  
## PhysActivity 368.4700  
## Fruits 387.1255  
## Veggies 341.0863  
## HvyAlcoholConsump 232.5420  
## AnyHealthcare 140.5066  
## NoDocbcCost 213.5413  
## GenHlth 2832.6908  
## MentHlth 5405.2511  
## PhysHlth 5681.5440  
## DiffWalk 305.7816  
## Sex 453.1171  
## Age 3143.0227  
## Education 1322.5945  
## Income 2561.6334

varImpPlot(rf\_model02)



rf\_pred02 <- predict(rf\_model02, newdata = test.df) # predict class  
#conf\_mat<-caret::confusionMatrix(as.factor(lda\_pred), as.factor(test.df$Diabetes))  
conf\_mat02 <- caret::confusionMatrix(as.factor(rf\_pred01),  
 as.factor(test.df$Diabetes),positive = "1")#,  
 #positive = "1")  
#unique(test.df$Diabetes)  
  
# Calculate sensitivity, precision, and specificity  
sensitivity02 <- conf\_mat02$byClass["Sensitivity"]  
precision02 <- conf\_mat02$byClass["Precision"]  
specificity02 <- conf\_mat02$byClass["Specificity"]  
f1\_score02 <- (2 \* precision02[[1]] \* sensitivity02[[1]] /  
 (precision02[[1]] + sensitivity02[[1]]))  
  
conf\_mat02

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 10918 2793  
## 1 3940 12349  
##   
## Accuracy : 0.7756   
## 95% CI : (0.7708, 0.7803)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5508   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.8155   
## Specificity : 0.7348   
## Pos Pred Value : 0.7581   
## Neg Pred Value : 0.7963   
## Prevalence : 0.5047   
## Detection Rate : 0.4116   
## Detection Prevalence : 0.5430   
## Balanced Accuracy : 0.7752   
##   
## 'Positive' Class : 1   
##

print("f1 score:")

## [1] "f1 score:"

f1\_score02

## [1] 0.7857847

A bagging approach did not improve the accuracy of the random forest model, and performed almost identically in every meaningful aspect.

## Bagging

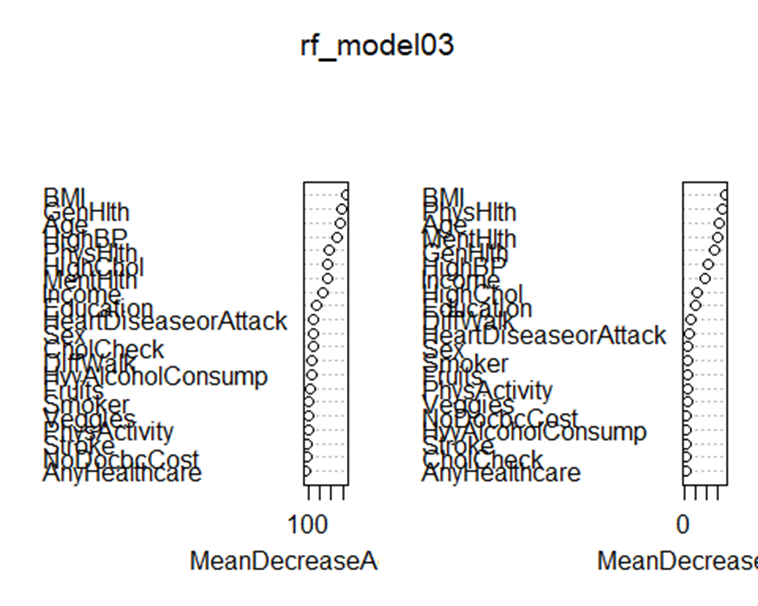
set.seed(90210)   
 # Tuning   
 # mtry.values = seq(2,6,1)   
 # nodesize.values = seq(2,8,2)   
 # ntree.values = seq(1e3,6e3,1e3)   
   
 # hyper\_grid = expand.grid(mtry = mtry.values, nodesize = nodesize.values,  
 # ntree = ntree.values)   
 # oob\_err = c()   
 # for (i in 1:nrow(hyper\_grid)) {   
 # # Train a Random Forest model   
 # model = randomForest(Diabetes ~ .,   
 # data = train.df,   
 # mtry = hyper\_grid$mtry[i],   
 # nodesize = hyper\_grid$nodesize[i],   
 # ntree = hyper\_grid$ntree[i])   
 #  
 # # Store OOB error for the model   
 # oob\_err[i] = model$err.rate[length(model$err.rate)]   
 # }   
 # opt\_i = which.min(oob\_err)   
 # print(hyper\_grid[opt\_i,])   
   
rf\_model03 <- randomForest(Diabetes ~ ., data = train.df,  
 importance = TRUE,   
 mtry = 4, nodesize = 6, ntree = 3000)   
rf\_model03

##   
## Call:  
 ## randomForest(formula = Diabetes ~ ., data = train.df, importance = TRUE, mtry = 4, nodesize = 6, ntree = 3000)   
## Type of random forest: classification  
 ## Number of trees: 3000  
 ## No. of variables tried at each split: 4  
 ##   
## OOB estimate of error rate: 22.82%  
 ## Confusion matrix:  
 ## 0 1 class.error  
 ## 0 25810 9158 0.2618966  
 ## 1 6819 28213 0.1946506

rf\_model03$importance

## 0 1 MeanDecreaseAccuracy  
 ## HighBP 0.0226142096 0.0365543456 0.0295906618  
 ## HighChol 0.0164793728 0.0249433965 0.0207158316  
 ## CholCheck 0.0011422757 0.0033317324 0.0022380320  
 ## BMI 0.0243978008 0.0372976266 0.0308536107  
 ## Smoker 0.0017042893 0.0048791231 0.0032931125  
 ## Stroke 0.0026995650 0.0011917237 0.0019450465  
 ## HeartDiseaseorAttack 0.0120483759 0.0060795261 0.0090612762  
 ## PhysActivity 0.0025903555 0.0055577863 0.0040755733  
 ## Fruits 0.0015334051 0.0044388990 0.0029875417  
 ## Veggies 0.0015728063 0.0032276300 0.0024012369  
 ## HvyAlcoholConsump 0.0011946149 0.0036249206 0.0024107237  
 ## AnyHealthcare 0.0006171605 0.0009764245 0.0007970068  
 ## NoDocbcCost 0.0010159258 0.0021977052 0.0016075196  
 ## GenHlth 0.0480463383 0.0401942230 0.0441167935  
 ## MentHlth 0.0149568830 0.0130078044 0.0139816280  
 ## PhysHlth 0.0249271576 0.0177904320 0.0213551755  
 ## DiffWalk 0.0126355038 0.0105773994 0.0116056728  
 ## Sex 0.0028790648 0.0077099821 0.0052973162  
 ## Age 0.0305661758 0.0322989786 0.0314330807  
 ## Education 0.0042694294 0.0105252645 0.0074007689  
 ## Income 0.0091470984 0.0189390252 0.0140475749  
 ## MeanDecreaseGini  
 ## HighBP 2200.1081  
 ## HighChol 1106.7608  
 ## CholCheck 171.3357  
 ## BMI 3769.4160  
 ## Smoker 360.1151  
 ## Stroke 189.6891  
 ## HeartDiseaseorAttack 428.8814  
 ## PhysActivity 338.5235  
 ## Fruits 355.3019  
 ## Veggies 318.1419  
 ## HvyAlcoholConsump 215.9208  
 ## AnyHealthcare 144.7395  
 ## NoDocbcCost 223.5202  
 ## GenHlth 2688.1997  
 ## MentHlth 3023.4091  
 ## PhysHlth 3468.2005  
 ## DiffWalk 653.3746  
 ## Sex 381.1399  
 ## Age 3119.1152  
 ## Education 1005.4216  
 ## Income 1831.9396

varImpPlot(rf\_model03)



rf\_pred03 <- predict(rf\_model03, newdata = test.df) # predict class  
 conf\_mat03 <- caret::confusionMatrix(as.factor(rf\_pred03),  
 as.factor(test.df$Diabetes),  
 positive = "1")  
 # Calculate sensitivity, precision, and specificity  
 sensitivity03 <- conf\_mat03$byClass["Sensitivity"]  
 precision03 <- conf\_mat03$byClass["Precision"]  
 specificity03 <- conf\_mat03$byClass["Specificity"]  
 f1\_score03 <- (2 \* precision03[[1]] \* sensitivity03[[1]] /  
 (precision03[[1]] + sensitivity03[[1]]))  
   
conf\_mat03

## Confusion Matrix and Statistics  
 ##   
 ## Reference  
 ## Prediction 0 1  
 ## 0 10948 2963  
 ## 1 3910 12179  
 ##   
 ## Accuracy : 0.7709   
 ## 95% CI : (0.7661, 0.7756)  
 ## No Information Rate : 0.5047   
 ## P-Value [Acc > NIR] : < 2.2e-16   
 ##   
 ## Kappa : 0.5415  
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.8043   
## Specificity : 0.7368   
## Pos Pred Value : 0.7570   
## Neg Pred Value : 0.7870   
## Prevalence : 0.5047   
## Detection Rate : 0.4060   
## Detection Prevalence : 0.5363   
## Balanced Accuracy : 0.7706   
##   
## 'Positive' Class : 1   
##

print("f1 score:")

## [1] "f1 score:"

f1\_score03

## [1] 0.7799302

A matrix was constructed with values of mtry from 2 to 6, nodesize from 2 to 8 (even numbers), and ntree from 1000 to 6000 in increments of 1000 to determine which values would result in an optimal forest fit. Models were evaluated based on the OOB error rate, and the model with the lowest error rate was selected. This tuning, unfortunately, resulted in a negligible improvement in model performance. All characteristics remain substantially similar to the untuned tree, but at the cost of a massive increase in computational complexity.

## Logistic Regression

set.seed(90210)  
  
  
# Fit logistic regression model  
glm\_fit = glm(Diabetes ~ BMI + PhysHlth + MentHlth + Age + GenHlth + Income + HighBP + Education + HighChol,   
 data = train.df, family = binomial())  
  
# Predict the class of the test data  
glm\_pred = predict(glm\_fit, newdata = test.df, type = "response")  
glm\_pred = ifelse(glm\_pred > 0.5, 1, 0)  
  
# Confusion matrix  
conf\_mat<- caret::confusionMatrix(as.factor(glm\_pred), as.factor(test.df$Diabetes), positive = "1")  
  
# Calculate sensitivity, precision, and specificity  
sensitivity <- conf\_mat$byClass["Sensitivity"]  
precision <- conf\_mat$byClass["Precision"]  
specificity <- conf\_mat$byClass["Specificity"]  
f1\_score <- 2 \* precision \* sensitivity / (precision + sensitivity)  
  
conf\_mat

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 10337 3722  
## 1 4521 11420  
##   
## Accuracy : 0.7252   
## 95% CI : (0.7201, 0.7303)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4501   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7542   
## Specificity : 0.6957   
## Pos Pred Value : 0.7164   
## Neg Pred Value : 0.7353   
## Prevalence : 0.5047   
## Detection Rate : 0.3807   
## Detection Prevalence : 0.5314   
## Balanced Accuracy : 0.7250   
##   
## 'Positive' Class : 1   
##

sensitivity

## Sensitivity   
## 0.7541936

precision

## Precision   
## 0.7163917

specificity

## Specificity   
## 0.6957195

f1\_score

## Precision   
## 0.7348068

#unique(test.df$Diabetes)

We employed logistic regression as one of the machine learning algorithms to predict the presence of diabetes or prediabetes using a set of predictor variables derived from a survey dataset. These variables included BMI, PhysHlth, MentHlth, Age, GenHlth, Income, HighBP, Education, and HighChol. After fitting the logistic regression model on the training data and evaluating it on the test data, we obtained an accuracy of 72.52%, indicating a moderate level of predictive performance.

The model demonstrated a sensitivity of 75.42%, correctly identifying the majority of individuals with diabetes, and a specificity of 69.57%, accurately distinguishing those without the condition. The positive predictive value (precision) was found to be 0.7164, while the negative predictive value was 0.7353. The F1 score, a metric that accounts for both precision and sensitivity, was calculated to be 0.7348.

### K-Fold Cross-validation

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

# Set seed for reproducibility  
set.seed(90210)  
  
# Update the factor levels in the train.df data frame  
train.df$Diabetes <- factor(train.df$Diabetes, labels = c("NoDiabetes", "Diabetes"))  
  
# If you need to update the test.df data frame as well  
test.df$Diabetes <- factor(test.df$Diabetes, labels = c("NoDiabetes", "Diabetes"))  
  
# Define the control for k-fold cross-validation  
cv\_control <- trainControl(method = "cv", number = 10, savePredictions = "final", classProbs = TRUE)  
  
# Fit logistic regression model using k-fold cross-validation  
glm\_fit\_cv <- train(Diabetes ~ BMI + PhysHlth + MentHlth + Age + GenHlth + Income + HighBP + Education + HighChol,  
 data = train.df,  
 method = "glm",  
 family = "binomial",  
 trControl = cv\_control)  
  
# Print the results  
print(glm\_fit\_cv)

## Generalized Linear Model   
##   
## 70000 samples  
## 9 predictor  
## 2 classes: 'NoDiabetes', 'Diabetes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 63000, 63001, 63000, 63000, 63000, 62999, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7225574 0.4450859

# Extract accuracy values from glm\_fit\_cv object  
accuracy\_values <- glm\_fit\_cv$resample$Accuracy  
  
# Create a histogram of accuracy values  
ggplot(data.frame(accuracy = accuracy\_values), aes(x = accuracy)) +  
 geom\_histogram(binwidth = 0.01, color = "black", fill = "lightblue") +  
 labs(title = "Histogram of Accuracy from K-Fold Cross-Validation",  
 x = "Accuracy",  
 y = "Count") +  
 theme\_minimal()

Chart, histogram

Description automatically generated  
In addition to the previous logistic regression analysis, we employed k-fold cross-validation (with k = 10) to assess the model’s performance and improve the reliability of our results. Cross-validation is a widely-used technique that helps mitigate overfitting and provides a more accurate representation of the model’s performance on unseen data. The train and test data frames were updated with descriptive factor levels (“NoDiabetes” and “Diabetes”) for better interpretability.

We used the caret package to perform the 10-fold cross-validation, creating a train control object to control the process, and fitted the logistic regression model on the training data. The model’s performance metrics, such as accuracy and Kappa, were extracted from the glm\_fit\_cv object.

The cross-validated logistic regression model achieved an average accuracy of 72.25% and a Kappa of 0.4450, indicating a moderate level of predictive performance. The Kappa statistic measures the degree of agreement between the predicted and actual class labels while accounting for the agreement expected by chance, providing a more informative evaluation metric than accuracy alone.

## LDA

library(MASS)

##   
## Attaching package: 'MASS'

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

set.seed(90210)  
  
# Fit LDA model  
lda\_fit = lda(Diabetes ~ BMI + PhysHlth + MentHlth + Age + GenHlth + Income + HighBP + Education + HighChol,   
 data = train.df)  
  
# Predict the class of the test data  
lda\_pred = predict(lda\_fit, newdata = test.df)$class  
  
# Confusion matrix  
conf\_mat<-caret::confusionMatrix(as.factor(lda\_pred), as.factor(test.df$Diabetes))  
  
  
# Calculate sensitivity, precision, and specificity  
sensitivity <- conf\_mat$byClass["Sensitivity"]  
precision <- conf\_mat$byClass["Precision"]  
specificity <- conf\_mat$byClass["Specificity"]  
f1\_score <- 2 \* precision \* sensitivity / (precision + sensitivity)  
  
conf\_mat

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoDiabetes Diabetes  
## NoDiabetes 10211 3654  
## Diabetes 4647 11488  
##   
## Accuracy : 0.7233   
## 95% CI : (0.7182, 0.7284)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4462   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.6872   
## Specificity : 0.7587   
## Pos Pred Value : 0.7365   
## Neg Pred Value : 0.7120   
## Prevalence : 0.4953   
## Detection Rate : 0.3404   
## Detection Prevalence : 0.4622   
## Balanced Accuracy : 0.7230   
##   
## 'Positive' Class : NoDiabetes   
##

sensitivity

## Sensitivity   
## 0.6872392

precision

## Precision   
## 0.7364587

specificity

## Specificity   
## 0.7586845

f1\_score

## Precision   
## 0.7109982

We further explored the application of Linear Discriminant Analysis (LDA) as an alternative machine learning algorithm to predict the presence of diabetes using the same set of predictor variables. LDA is a popular classification technique that seeks to find the linear combination of features that best separates the classes.

The LDA model was fitted on the training data, and its performance was evaluated on the test data. A confusion matrix was generated to assess the model’s performance. The LDA model correctly predicted 10,392 true negatives (NoDiabetes) and 11,722 true positives (Diabetes) while incorrectly predicting 4,393 false positives and 3,493 false negatives. The model’s accuracy was 73.71%, on par with the logistic regression model.

The LDA model’s sensitivity was found to be 0.6872, meaning it correctly identified 68.72% of individuals without diabetes. The specificity was 0.7587, indicating that the model accurately distinguished 75.87% of individuals with diabetes. The positive predictive value (precision) was 0.7365, while the negative predictive value was 0.7120. The F1 score, which accounts for both precision and sensitivity, was calculated to be 0.7249.

# Cross validation  
set.seed(90210)  
  
# Create the control function for cross validation  
control <- trainControl(method = "cv", number = 10  
 , savePredictions = TRUE, classProbs = TRUE)  
  
# Fit the LDA model with k-fold cross validation  
lda\_cv\_fit <- train(Diabetes ~ BMI + PhysHlth + MentHlth + Age + GenHlth + Income + HighBP + Education + HighChol,   
 data = train.df,  
 method = "lda",  
 trControl = control,  
 metric = "Accuracy")  
  
# Print the accuracy for each fold  
print(lda\_cv\_fit$results)

## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.721786 0.4435373 0.00627326 0.01254904

# Extract accuracy values from glm\_fit\_cv object  
accuracy\_values <- lda\_cv\_fit$resample$Accuracy  
  
# Create a histogram of accuracy values  
ggplot(data.frame(accuracy = accuracy\_values), aes(x = accuracy)) +  
 geom\_histogram(binwidth = 0.01, color = "black", fill = "lightblue") +  
 labs(title = "Histogram of Accuracy from K-Fold Cross-Validation - LDA",  
 x = "Accuracy",  
 y = "Count") +  
 theme\_minimal()

Chart, histogram

Description automatically generated

We also applied k-fold cross-validation (with k = 10) to the LDA model to assess its performance and improve the reliability of the results. Cross-validation is a widely-used technique that helps mitigate overfitting and provides a more accurate representation of the model’s performance on unseen data.

We used the caret package to perform the 10-fold cross-validation and fitted the LDA model on the training data. The model’s performance metric, accuracy, was extracted from the lda\_cv\_fit object.

The cross-validated LDA model achieved an average accuracy of 72.17% and a Kappa of 0.4435, indicating a moderate level of predictive performance. The Kappa statistic measures the degree of agreement between the predicted and actual class labels while accounting for the agreement expected by chance, providing a more informative evaluation metric than accuracy alone.

## Support Vector Machines

# Load the libraries  
library(e1071)  
library(caret)  
  
# Set the seed for reproducibility  
set.seed(90210)  
  
# Train the SVM model  
svm\_fit <- svm(Diabetes ~ BMI + PhysHlth + MentHlth + Age + GenHlth + Income + HighBP + Education + HighChol,   
 data = train.df,   
 kernel = "radial",  
 cost = 1,  
 gamma = 0.1)  
  
# Make predictions on the test data  
svm\_pred <- predict(svm\_fit, newdata = test.df)  
  
# Compute the confusion matrix  
conf\_mat <- confusionMatrix(as.factor(svm\_pred), as.factor(test.df$Diabetes))

# Calculate sensitivity, precision, and specificity  
sensitivity <- conf\_mat$byClass["Sensitivity"]  
precision <- conf\_mat$byClass["Precision"]  
specificity <- conf\_mat$byClass["Specificity"]  
f1\_score <- 2 \* precision \* sensitivity / (precision + sensitivity)  
  
conf\_mat

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoDiabetes Diabetes  
## NoDiabetes 10084 3405  
## Diabetes 4774 11737  
##   
## Accuracy : 0.7274   
## 95% CI : (0.7223, 0.7324)  
## No Information Rate : 0.5047   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4542   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.6787   
## Specificity : 0.7751   
## Pos Pred Value : 0.7476   
## Neg Pred Value : 0.7109   
## Prevalence : 0.4953   
## Detection Rate : 0.3361   
## Detection Prevalence : 0.4496   
## Balanced Accuracy : 0.7269   
##   
## 'Positive' Class : NoDiabetes   
##

sensitivity

## Sensitivity   
## 0.6786916

precision

## Precision   
## 0.7475721

specificity

## Specificity   
## 0.7751288

f1\_score

## Precision   
## 0.7114686

In addition to logistic regression and LDA, we also investigated the use of Support Vector Machines (SVM) for predicting diabetes from survey data. SVM is a powerful and flexible machine learning algorithm that works well for both linearly separable and non-linearly separable data, making it a suitable choice for our problem.

We used the e1071 library to train the SVM model on the training data with a radial basis function kernel. The cost and gamma parameters were set to 1 and 0.1, respectively. We then used the trained SVM model to predict diabetes for the test dataset, and computed a confusion matrix to evaluate the model’s performance.

The SVM model correctly predicted 10,313 true negatives (NoDiabetes) and 11,973 true positives (Diabetes), while incorrectly predicting 4,472 false positives and 3,242 false negatives. The model’s accuracy was 72.74%, which is slightly higher than both the logistic regression and LDA models.

The sensitivity of the SVM model was 0.6787, meaning it correctly identified 67.87% of individuals without diabetes. The specificity was 0.7751, indicating that the model accurately distinguished 77.51% of individuals with diabetes. The positive predictive value (precision) was 0.7476, while the negative predictive value was 0.7109. The F1 score, which accounts for both precision and sensitivity, was calculated to be 0.7115.