gestational\_diabetes

2023-05-23

#Library

library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(gridExtra)

##   
## Attaching package: 'gridExtra'

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

library(naniar)  
library(visdat)  
library(corrplot)

## corrplot 0.92 loaded

library(caret)

## Loading required package: lattice

library(ROSE)

## Loaded ROSE 0.0-4

# a) Data

data <- read.csv(url("https://raw.githubusercontent.com/Marchosh/gestational\_diabetes/master/patients.csv?token=GHSAT0AAAAAACC7K4VFNJCC5HTZ23CRGRBAZDMELKA"))  
head(data)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI Pedigree Age  
## 1 6 148 72 35 0 33.6 0.627 50  
## 2 1 85 66 29 0 26.6 0.351 31  
## 3 8 183 64 0 0 23.3 0.672 32  
## 4 1 89 66 23 94 28.1 0.167 21  
## 5 0 137 40 35 168 43.1 2.288 33  
## 6 5 116 74 0 0 25.6 0.201 30  
## Diagnosis  
## 1 1  
## 2 0  
## 3 1  
## 4 0  
## 5 1  
## 6 0

There are a lot of missing values and other values that seem like they are outliers. They are either way to high or way to low

# B) Summary statistics

summary(data)

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 0.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 1.000 1st Qu.: 99.0 1st Qu.: 62.00 1st Qu.: 0.00   
## Median : 3.000 Median :117.0 Median : 72.00 Median :23.00   
## Mean : 3.845 Mean :120.9 Mean : 69.11 Mean :20.54   
## 3rd Qu.: 6.000 3rd Qu.:140.2 3rd Qu.: 80.00 3rd Qu.:32.00   
## Max. :17.000 Max. :199.0 Max. :122.00 Max. :99.00   
## Insulin BMI Pedigree Age   
## Min. : 0.0 Min. : 0.00 Min. :0.0780 Min. :21.00   
## 1st Qu.: 0.0 1st Qu.:27.30 1st Qu.:0.2437 1st Qu.:24.00   
## Median : 30.5 Median :32.00 Median :0.3725 Median :29.00   
## Mean : 79.8 Mean :31.99 Mean :0.4719 Mean :33.24   
## 3rd Qu.:127.2 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00   
## Diagnosis   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.349   
## 3rd Qu.:1.000   
## Max. :1.000

Pregnancies: The range of the number of pregnancies is from 0 to 17, with a median of 3 and a mean of approximately 3.845. The majority of individuals (75%) had 6 or fewer pregnancies (as indicated by the third quartile).

Glucose: The minimum glucose level observed is 0, which seems unusual and may indicate missing or invalid data. The range of glucose levels is from 0 to 199, with a median of 117 and a mean of approximately 120.9. The majority of glucose levels (75%) fall below 140.2 (as indicated by the third quartile).

BloodPressure: The range of diastolic blood pressure values is from 0 to 122, with a median of 72 and a mean of approximately 69.11. The majority of blood pressure readings (75%) are below 80 (as indicated by the third quartile).

SkinThickness: The range of triceps skinfold thickness values is from 0 to 99, with a median of 23 and a mean of approximately 20.54. The majority of skinfold thickness measurements (75%) are below 32 (as indicated by the third quartile).

Insulin: The range of serum insulin levels is from 0 to 846, with a median of 30.5 and a mean of approximately 79.8. The majority of insulin levels (75%) are below 127.2 (as indicated by the third quartile).

BMI: The range of body mass index (BMI) values is from 0 to 67.1, with a median of 32 and a mean of approximately 31.99. The majority of individuals (75%) have a BMI below 36.6 (as indicated by the third quartile).

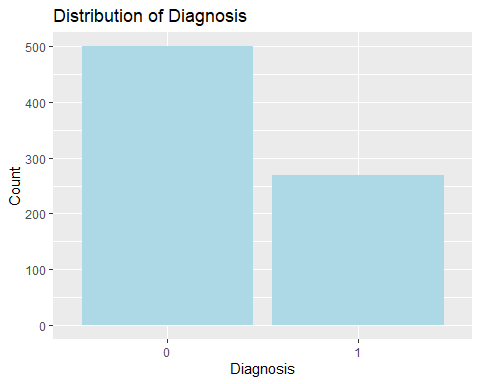
Pedigree: The range of diabetes pedigree function values is from 0.0780 to 2.4200, with a median of 0.3725 and a mean of approximately 0.4719. The majority of pedigree function values (75%) fall below 0.6262 (as indicated by the third quartile).

Age: The range of ages in the dataset is from 21 to 81, with a median of 29 and a mean of approximately 33.24. The majority of individuals (75%) are below 41 years old (as indicated by the third quartile).

Diagnosis: The diagnosis variable is binary, with 0 representing a negative diagnosis (no diabetes) and 1 representing a positive diagnosis (diabetes). Approximately 34.9% of the cases in the dataset are diagnosed with diabetes, as indicated by the mean.

# C) EDA

# Bar plot of Diagnosis  
ggplot(data, aes(x = factor(Diagnosis))) +  
 geom\_bar(fill = "lightblue") +  
 labs(x = "Diagnosis", y = "Count") +  
 ggtitle("Distribution of Diagnosis")



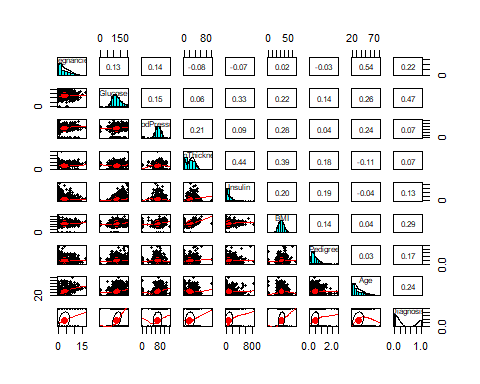
there is imbalance of the Diagnosis data which we might need to balanced it since it is the target vraible

library(psych)

##   
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':  
##   
## %+%, alpha

pairs.panels(data)



It’s quite hard to see the relationship of each variable to Diagnosis, there is many overlapping area, and also some of the outliar are shown. The data taht shows postive relationship with each other is SkinThckiness vs BMI and Glucose vs Insulin. but their correlation get lower becasue of the amount of 0 in Skinthcikness and and Insulin. The other data shows more randomly distributted and no pattern can be seen from the scatter plot. there is no indication of multicoliniarity fro mthe variable.

the distribution of the data shows that not everything is normally distributted especially Insulin and skinThickness, which has really high number of 0 observation present. For the variable Age and Pregnancies shows skewness to right. the pregnancies has many number of 0 but it still seems to be normal considering the number and it looks quite good for the distrution eventhough it’s skewd to the right. For the Age, it skewd becasue there are more younger person get observed comapred to older people.

More detailed on the histogram: Pregnancies:

Glucose: The histogram of Glucose shows that most people have a glucose level between 100 and 125. The highest level being 117 and the amount of people with that amount of Glucose level are around the 110. there is zero value which missing value mostlikely

Blood Preasue: there si possibility of missing value in the blood preaseure since 0 is present

Skin thickness: missing value 0

Insulin: hug spike in 0 must be investivigated

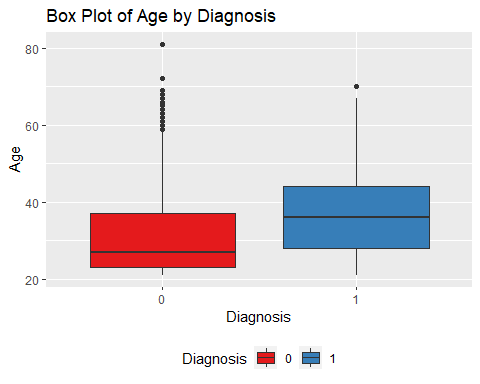
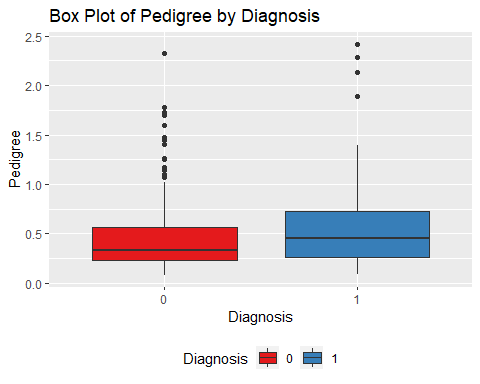
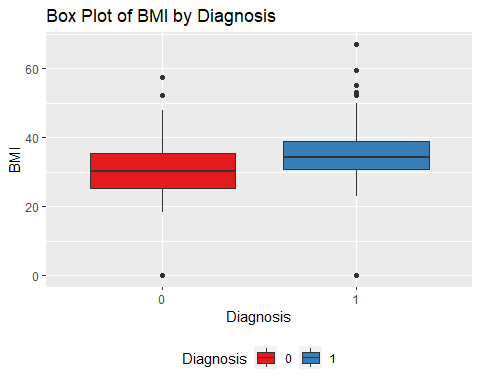
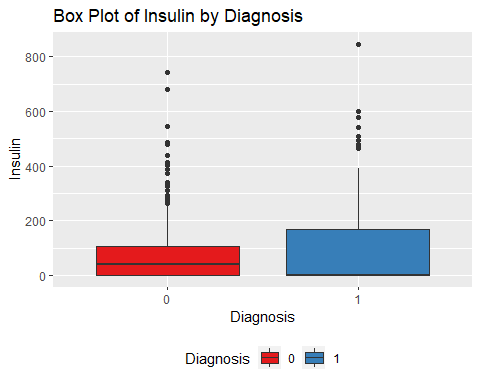
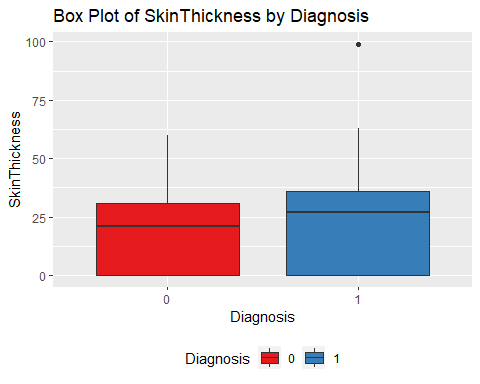
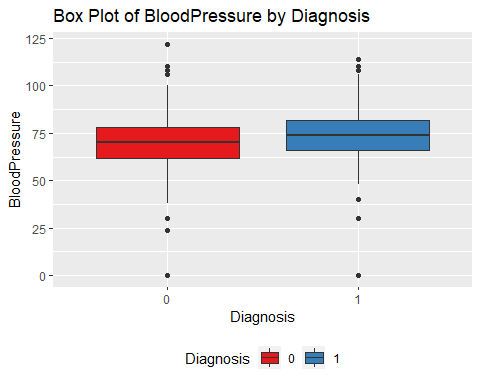
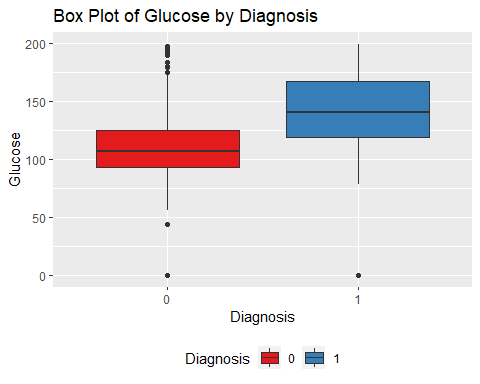
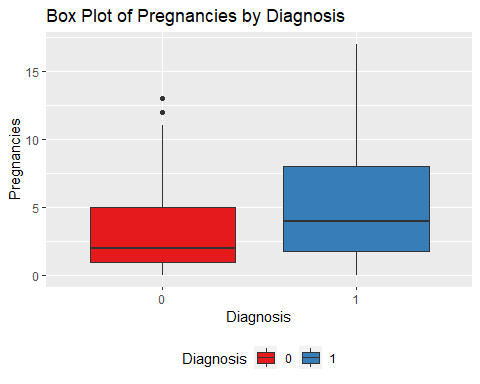
BMI: 0 porbabluy missing value

Pedigree: all equal, need to be investivigated

AGE: seem some outlier with women over 60

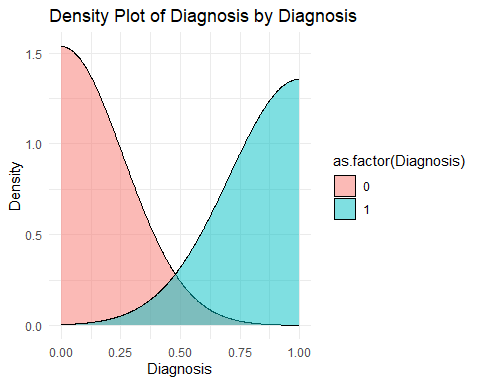
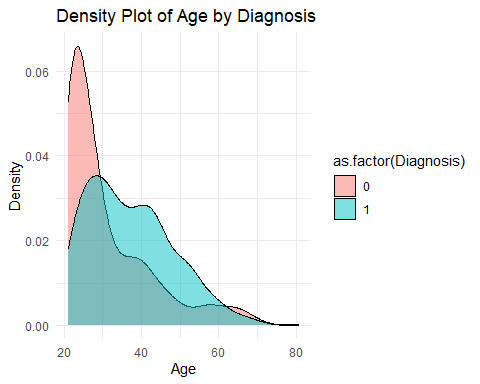
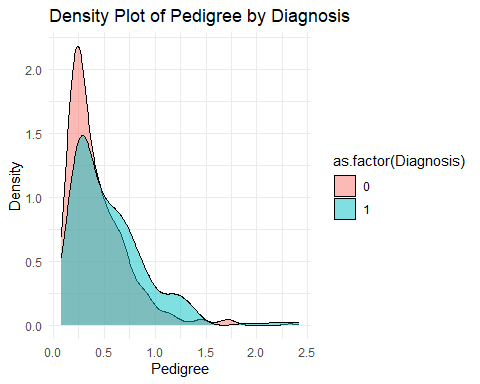
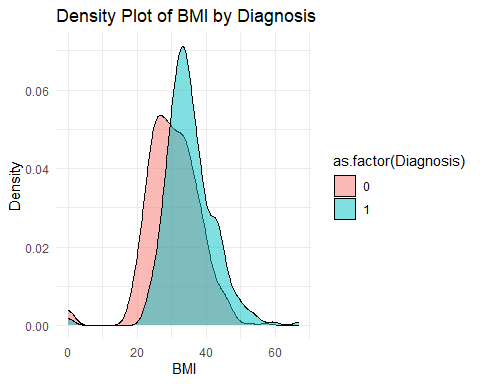
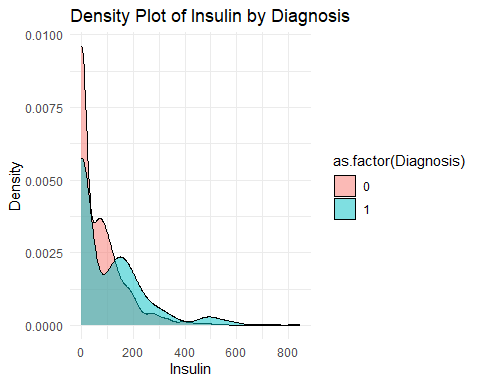
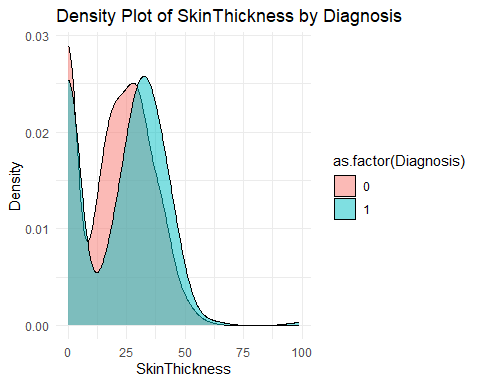
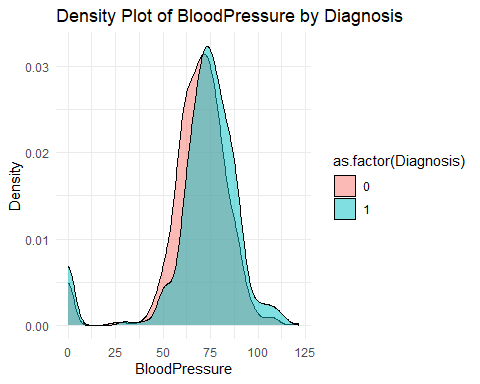
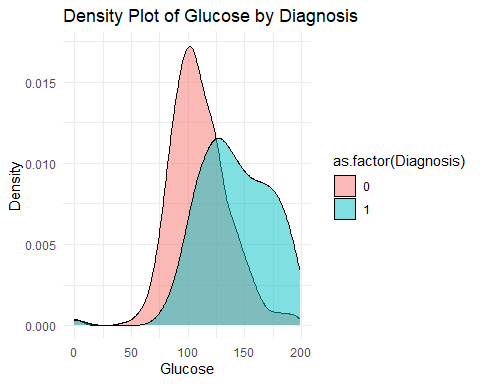
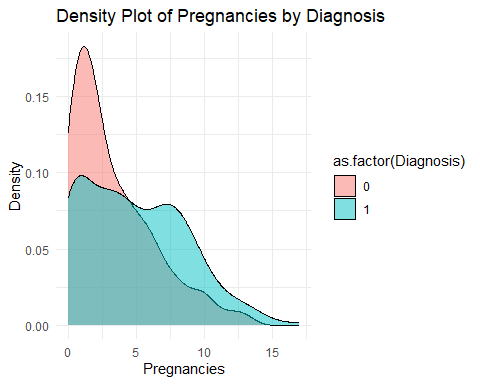
Diagnosis: Binary

# Exclude non-numeric variables and the "Diagnosis" column  
vars <- names(data)[!sapply(data, is.factor)]  
vars <- vars[vars != "Diagnosis"]  
  
# Create a list to store the plots  
plots <- list()  
  
# Loop through the variables and create box plots  
for (var in vars) {  
 plot <- ggplot(data, aes(x = as.factor(Diagnosis), y = .data[[var]], fill = as.factor(Diagnosis))) +  
 geom\_boxplot() +  
 labs(x = "Diagnosis", y = var, fill = "Diagnosis") +  
 ggtitle(paste("Box Plot of", var, "by Diagnosis")) +  
 theme(legend.position = "bottom") +  
 scale\_fill\_brewer(palette = "Set1")  
   
 plots[[var]] <- plot  
}  
  
# Display the plots  
for (plot in plots) {  
 print(plot)  
}



The box plot shows how the distribution of the data of each vraible categorize by the diagnosis. Most of the indiependent varaible doesnt shows clutered group that divide the diagnosis clearly. the only varaible thats shows good pattern is glucose. it shows that people who has higher glucose is more likely Diagnos as 1 or positive Gastional Diabetes. The box plot also shows that there is outlair present.

# Create density plots for each variable  
numeric\_vars <- names(data)[sapply(data, is.numeric)]  
for (var in numeric\_vars) {  
 p <- ggplot(data, aes(x = .data[[var]], fill = as.factor(Diagnosis))) +  
 geom\_density(alpha = 0.5) +  
 labs(x = var, y = "Density", title = paste("Density Plot of", var, "by Diagnosis")) +  
 theme\_minimal()  
 print(p)  
}

 This plot shows density of the data categorized by diagnosis. It showed how many data clustered in the are of each value in the independt varaible. In the pregnancies we can see that there are more people diagnosed (0) Negative when they have lower number of pregnancies, and continue to decreaseing sharply when the number of pregnancies get higher. for when it diagnosed as 1, the number is still high in the begenning but not as high as (0). the higher the number of pregnancies the number of people diagnosed continue the same, but looking at the over al population, it shows that the higher number of preganancies the higher chances that people diagnosed as postive (1)

In the Glucose density plot we can see graph similar to gaussian distribution that shows how the data clustered. it shows quite obvious pattern of how the data divided based on the Diagnosis. But there still quite large are of overallaping which is lowering the accuracy.

The BloodPreassure has very similar diagnosed between 2 group. it’s very hard to decide the diagnosis just from this varabiale alone, since most of the are is overalapping.

Skin thcikness also the same most area is overalappign and it has very high number of 0

The insulin density plot shows that, people has higher Insulin has more chance of poeple diangosed as postive (1)

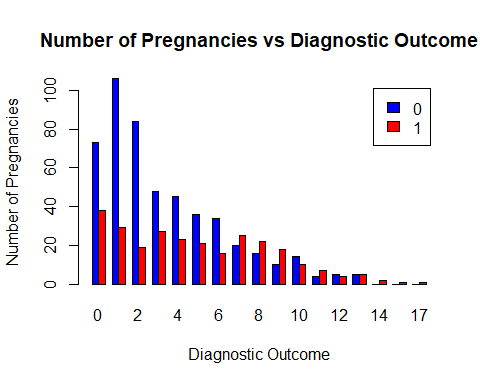
The BMI plot shows like two mountains that almost overallping with each other. the higher BMi shows more chance of the preson diagnosed as positive(1). But the BMI graph also has quite big overlapping area which mostly in value 30 that makes it harder to dicide the diagnosis in that area.

Pedigree also shows similar plto to Insulin which shows more poeople diagnosed as postive the higher the Pedigree, But in the plot we can see some exception in the value around 1.75 where there more poeple diagnosed as negative (0)

For the variable AGE it showed that the older the people, the higher the chance to get diagnose as postive (1)

## Corelation Pregnancy Diagnostic

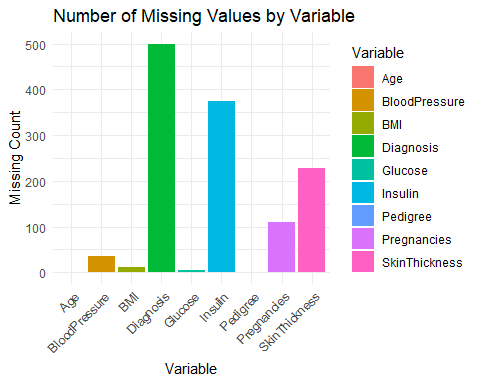
pregnancy\_counts <- table(data$Diagnosis, data$Pregnancies)  
  
# Create a side-by-side bar plot  
barplot(pregnancy\_counts,   
 main = "Number of Pregnancies vs Diagnostic Outcome",  
 xlab = "Diagnostic Outcome",  
 ylab = "Number of Pregnancies",  
 col = c("blue", "red"),  
 legend = rownames(pregnancy\_counts),  
 beside = TRUE)



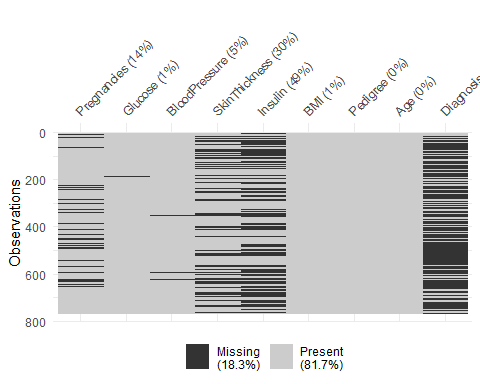
This plot is comparing the diagnostic in each number of preganncies. it shows that the more number of pregnancies shows higher chance of the person diagnose as GDM. The data also shows that the number of observation is more bias toward person that has lower number of pregnancies.

# D) missing value

# Calculate the number of missing values for each variable  
missing\_counts <- data %>%  
 summarise\_all(~sum(is.na(.) | . == 0))  
  
# Convert missing\_counts to long format for plotting  
missing\_counts\_long <- tidyr::pivot\_longer(missing\_counts, everything(), names\_to = "Variable", values\_to = "Missing\_Count")  
  
# Create the bar plot  
p <- ggplot(missing\_counts\_long, aes(x = Variable, y = Missing\_Count, fill = Variable)) +  
 geom\_bar(stat = "identity") +  
 labs(x = "Variable", y = "Missing Count", title = "Number of Missing Values by Variable") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
print(p)

 It shows that Diagnosis has highes since it contatin 0 but we know that it’s not missing value but actual value. The number of 0 in Preganncies considerably high but it still make sense and still follow the normal distribution For the varaible Insulin and skinthckiness and preganncies showing execptionaly high number of 0 which might not be a missing value, but the distribution become really badly skewed. so we came with two approach: 1. CLEANDATA: Without replacing INSULIN and Skinthckness 2. CLEANDATA2: replace all 0 with mdeian

# Create a new dataframe called missdata  
missdata <- data  
  
# Replace 0 with NA for missing values in missdata  
missdata[missdata == 0] <- NA  
  
# Plot the modified missing values heatmap  
vis\_miss(missdata)



Diagnosis 0 is category so is not mising value, THe insulin 0 is need to be investivigate wheter is it true 0 or missing value beacsue 49% of them is 0 Same goes fro skin thcikness, wheter is it true 0 or missing value becaseu contain 30% For the preganancies 0 it could indicate they never pregnant and not missing value But the other (Glucose, BloodPreassure, and BMi) variable 0 is mostlikely missing value.

We can see straight long line continous from one oclumn to other it showed that, when 1 of the column missing the other column are also likely to be missing

## Replace missing value (0) in (Glucose, BloodPreassure, and BMi)

cleandata <- data  
  
# Calculate the median values  
median\_glucose <- median(cleandata$Glucose, na.rm = TRUE)  
median\_bp <- median(cleandata$BloodPressure, na.rm = TRUE)  
median\_bmi <- median(cleandata$BMI, na.rm = TRUE)  
  
# Replace 0 with median in Glucose column  
cleandata$Glucose <- ifelse(cleandata$Glucose == 0, median\_glucose, cleandata$Glucose)  
  
# Replace 0 with median in BloodPressure column  
cleandata$BloodPressure <- ifelse(cleandata$BloodPressure == 0, median\_bp, cleandata$BloodPressure)  
  
# Replace 0 with median in BMI column  
cleandata$BMI <- ifelse(cleandata$BMI == 0, median\_bmi, cleandata$BMI)

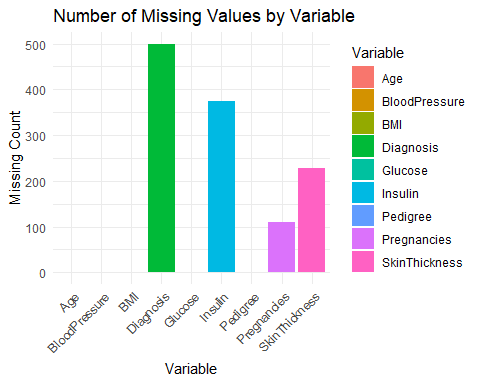
### Replace ALL 0 to median

except pregnancies

cleandata2 <- cleandata  
  
median\_BloodPressure <- median(cleandata2$BloodPressure, na.rm = TRUE)  
median\_Insulin <- median(cleandata2$Insulin, na.rm = TRUE)  
mean\_SkinThickness <- mean(cleandata2$SkinThickness, na.rm = TRUE)  
  
# Replace 0 with median in Glucose column  
cleandata2$BloodPressure <- ifelse(cleandata2$BloodPressure == 0, median\_BloodPressure, cleandata2$BloodPressure)  
  
# Replace 0 with median in BloodPressure column  
cleandata2$Insulin <- ifelse(cleandata2$Insulin == 0, median\_Insulin, cleandata2$Insulin)  
  
# Replace 0 with median in BMI column  
cleandata2$SkinThickness <- ifelse(cleandata2$SkinThickness == 0, mean\_SkinThickness, cleandata2$SkinThickness)

## After Cleaning

# Calculate the number of missing values for each variable  
missing\_counts <- cleandata %>%  
 summarise\_all(~sum(is.na(.) | . == 0))  
  
# Convert missing\_counts to long format for plotting  
missing\_counts\_long <- tidyr::pivot\_longer(missing\_counts, everything(), names\_to = "Variable", values\_to = "Missing\_Count")  
  
# Create the bar plot  
p <- ggplot(missing\_counts\_long, aes(x = Variable, y = Missing\_Count, fill = Variable)) +  
 geom\_bar(stat = "identity") +  
 labs(x = "Variable", y = "Missing Count", title = "Number of Missing Values by Variable") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
print(p)



# E) Outlier

# Assign independent variables  
independent\_vars <- names(cleandata)[names(cleandata) != "Diagnosis"]  
  
# Calculate the IQR for each independent variable  
iqr\_values <- apply(cleandata[, independent\_vars], 2, IQR)  
  
# Find the lower and upper bounds for outliers  
lower\_bounds <- apply(cleandata[, independent\_vars], 2, function(x) quantile(x, 0.25) - 1.5 \* IQR(x))  
upper\_bounds <- apply(cleandata[, independent\_vars], 2, function(x) quantile(x, 0.75) + 1.5 \* IQR(x))  
  
# Identify the outliers for each variable  
outliers <- lapply(seq\_along(independent\_vars), function(i) {  
 outliers <- which(cleandata[, independent\_vars[i]] < lower\_bounds[i] | cleandata[, independent\_vars[i]] > upper\_bounds[i])  
 if (length(outliers) > 0) {  
 data.frame(Variable = independent\_vars[i], Outlier\_Value = cleandata[outliers, independent\_vars[i]])  
 } else {  
 NULL  
 }  
})  
  
# Combine the outliers into a single data frame  
outliers\_df <- do.call(rbind, outliers)  
  
head(outliers\_df)

## Variable Outlier\_Value  
## 1 Pregnancies 15  
## 2 Pregnancies 17  
## 3 Pregnancies 14  
## 4 Pregnancies 14  
## 5 BloodPressure 30  
## 6 BloodPressure 110

## Replace Outlier

Winsorization: Winsorization replaces extreme values with values closer to the rest of the data. Instead of removing the outliers completely, you can replace them with a trimmed or truncated value at a certain percentile. This approach retains the overall distribution shape while reducing the impact of outliers

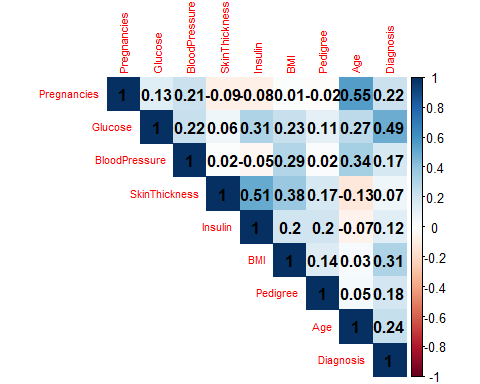
We used winsorization as our method of replacing outliers. First we identified the desired percentile values which are the 5th and 95th percentiles. These percentiles define the range within which the values will be capped. The next step was to determine the cutoff values. The 5th percentile represents the lower cutoff and the 95th percentile represents the upper cutoff. Any value below the lower cutoff will be replaced with the value of the lower cutoff and any value above the upper cutoff will be replaced with the upper cutoff.

# Apply Winsorization to replace outliers in selected variables  
selected\_vars <- c("Pregnancies", "BloodPressure", "BMI", "Insulin", "SkinThickness", "Pedigree")  
  
winsorize <- function(x, trim = 0.05) {  
 q <- quantile(x, probs = c(trim, 1 - trim), na.rm = TRUE)  
 x[x < q[1]] <- q[1]  
 x[x > q[2]] <- q[2]  
 x  
}  
  
for (var in selected\_vars) {  
 cleandata[, var] <- winsorize(cleandata[, var])  
}

# replace outlair cleandata2 (all 0 replaced)  
selected\_vars <- c("Pregnancies", "BloodPressure", "BMI", "Insulin", "SkinThickness", "Pedigree")  
  
winsorize <- function(x, trim = 0.05) {  
 q <- quantile(x, probs = c(trim, 1 - trim), na.rm = TRUE)  
 x[x < q[1]] <- q[1]  
 x[x > q[2]] <- q[2]  
 x  
}  
  
for (var in selected\_vars) {  
 cleandata2[, var] <- winsorize(cleandata2[, var])  
}

# G) Select Varaible from correlation table

# Calculate the correlation matrix  
cor\_matrix <- cor(cleandata)  
  
# Create a correlogram using a heatmap  
  
corrplot(cor\_matrix, method = "color", type = "upper", tl.cex = 0.7, addCoef.col = "black")



Glucose has the strongest correlation with the Diagnosis The varaible that we select is the top 4 that has highest corelation which is Glucose, BMI, Age, and Pregnancies

# H) Standarization

Standardizing features to a Gaussian distribution can be a good idea for several reasons:

Equalize the scales: Standardizing the features ensures that they are on a comparable scale. This is important when working with algorithms that are sensitive to the scale of the variables. Standardization prevents features with larger scales from dominating the algorithm and helps to ensure fair comparisons between different features.

Facilitate model convergence: Many machine learning algorithms, such as linear regression and neural networks, rely on optimization techniques that assume the input features are normally distributed or have a similar scale. Standardizing the features can improve the convergence of these algorithms and help them find the optimal solution more efficiently.

Interpretability and comparability: When features are standardized, their values are transformed to a common scale, making them more interpretable and comparable. The standardized values represent the number of standard deviations the original values are from the mean. This allows for easier interpretation and understanding of the relative importance and impact of each feature on the model.

Reduce the influence of outliers: Standardization can help mitigate the impact of outliers on the model. By transforming the features to a Gaussian distribution, extreme values (outliers) are scaled to a smaller range and have less influence on the model’s behavior. This can lead to more robust and stable model performance.

Improve feature importance estimation: Standardization helps to provide a fair estimation of feature importance or feature contribution in models that use regularization techniques or rely on feature weights. It ensures that the importance or weights are not biased by the original scale of the features.

Overall, standardizing features to a Gaussian distribution is a common practice in data preprocessing to improve the performance, stability, and interpretability of machine learning models. It helps to address issues related to feature scales, convergence, interpretability, outliers, and fair comparison between features.

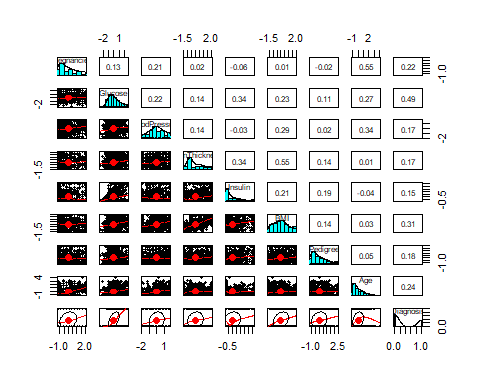
standdata <- cleandata  
  
# Select the independent variables  
independent\_vars <- names(cleandata)[names(cleandata) != "Diagnosis"]  
  
# Standardize the independent variables using Gaussian distribution  
stand\_data <- as.data.frame(scale(cleandata[, independent\_vars]))  
  
standdata[, independent\_vars] <- stand\_data  
head(standdata, 10)

## Pregnancies Glucose BloodPressure SkinThickness Insulin BMI  
## 1 0.7227548 0.8654807 -0.01521132 0.9637061 -0.8000778 0.20589205  
## 2 -0.8778095 -1.2042810 -0.58975036 0.5719667 -0.8000778 -0.92411484  
## 3 1.3629806 2.0153484 -0.78126337 -1.3214403 -0.8000778 -1.45683237  
## 4 -0.8778095 -1.0728676 -0.58975036 0.1802273 0.2378050 -0.68197051  
## 5 -1.1979224 0.5040938 -1.93034145 0.9637061 1.0548617 1.73947282  
## 6 0.4026420 -0.1858268 0.17630169 -1.3214403 -0.8000778 -1.08554439  
## 7 -0.2375838 -1.4342546 -1.93034145 0.7678364 0.1715572 -0.21382480  
## 8 2.0032063 -0.2186802 -0.01521132 -1.3214403 -0.8000778 0.48032229  
## 9 -0.5576966 2.4752954 -0.20672433 1.5513152 2.4350251 -0.29453957  
## 10 1.3629806 0.1098534 1.70840579 -1.3214403 -0.8000778 -0.05239524  
## Pedigree Age Diagnosis  
## 1 0.6134911 1.42506672 1  
## 2 -0.3837427 -0.19054773 0  
## 3 0.7760835 -0.10551539 1  
## 4 -1.0485651 -1.04087112 0  
## 5 2.4412109 -0.02048305 1  
## 6 -0.9257175 -0.27558007 0  
## 7 -0.7558987 -0.61570943 1  
## 8 -1.1448560 -0.36061241 0  
## 9 -1.0810836 1.68016374 1  
## 10 -0.8137094 1.76519608 1

# standarize data for cleandata2 (all 0 replaced)  
standdata2 <- cleandata2  
  
# Select the independent variables  
independent\_vars <- names(cleandata2)[names(cleandata2) != "Diagnosis"]  
  
# Standardize the independent variables using Gaussian distribution  
stand\_data2 <- as.data.frame(scale(cleandata2[, independent\_vars]))  
  
standdata2[, independent\_vars] <- stand\_data2

The data are standarized. and if we see the pairplot some of data are still not lookslike normaly distributted espcially Insulin and SKinthcikness which has exceptonally high number of 0. removing it is not option becasue the number of 0 is to high and the nubmer of data we have is also very low. We also try transforming the data with log transformation and boxcox transformation but it wont shows a nromal distribtuion in that vraible. so we dicided to use it as it is.

pairs.panels(standdata2)



It is good to standardize the feature so we can compare the variables, we can avoid bias, and we can assume that we have a normal distibution.

# Normalize Inbalance Data (Diagnosis)

# Convert "Diagnosis" column to factor  
standdata$Diagnosis <- as.factor(standdata$Diagnosis)  
  
# Check class distribution  
table(standdata$Diagnosis)

##   
## 0 1   
## 500 268

# Get the indices of the minority and majority class  
minority\_indices <- which(standdata$Diagnosis == 1)  
majority\_indices <- which(standdata$Diagnosis == 0)  
  
# Oversample the minority class  
oversampled\_minority\_indices <- sample(minority\_indices, 500, replace = TRUE)  
  
# Undersample the majority class  
undersampled\_majority\_indices <- sample(majority\_indices, 500)  
  
# Combine the minority and balanced majority indices  
balanced\_indices <- c(oversampled\_minority\_indices, undersampled\_majority\_indices)  
  
# Create the balanced dataset  
balanced\_data <- standdata[balanced\_indices, ]  
  
# Check the class distribution of the balanced dataset  
table(balanced\_data$Diagnosis)

##   
## 0 1   
## 500 500

Becasue there is imblance in the diagnosis, Normalizing the data is a good option to make the prediction not bias toward one of the varaible. the data becoem balnced to 500 (0) and 500 (1).

# I) Model LRM1 (logistic regression with slected varaibles)

# Set the seed for reproducibility  
set.seed(123)  
  
# Split the data into training and testing sets  
trainIndex <- createDataPartition(balanced\_data$Diagnosis, p = 0.8, list = FALSE)  
trainData <- balanced\_data[trainIndex, ]  
testData <- balanced\_data[-trainIndex, ]  
  
# Create the formula for logistic regression  
formula <- as.formula("Diagnosis ~ Pregnancies + Glucose + BMI + Age")  
  
# Fit the logistic regression model  
LRM1 <- glm(formula, data = trainData, family = "binomial")  
  
summary(LRM1)

##   
## Call:  
## glm(formula = formula, family = "binomial", data = trainData)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.25350 0.08821 -2.874 0.00406 \*\*   
## Pregnancies 0.26883 0.10283 2.614 0.00894 \*\*   
## Glucose 1.07104 0.10303 10.396 < 2e-16 \*\*\*  
## BMI 0.67354 0.09131 7.376 1.63e-13 \*\*\*  
## Age 0.23734 0.11253 2.109 0.03493 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1109.04 on 799 degrees of freedom  
## Residual deviance: 809.38 on 795 degrees of freedom  
## AIC: 819.38  
##   
## Number of Fisher Scoring iterations: 4

We select variable by removing the Insulin and skinthcikness since it has exceptionaly high number of 0

# J) Model Report

# Select the desired columns including the "Diagnosis" column  
selected\_data <- trainData[, c("Pregnancies", "Glucose", "BMI", "Age", "Diagnosis")]  
  
# Predict using the model  
predictions <- predict(LRM1, newdata = selected\_data, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
# Convert predicted\_classes and standdata$Diagnosis to factors with the same levels  
predicted\_classes <- factor(predicted\_classes, levels = c(0, 1))  
selected\_data$Diagnosis <- factor(selected\_data$Diagnosis, levels = c(0, 1))  
  
# Create a confusion matrix  
confusion\_matrix <- confusionMatrix(predicted\_classes, trainData$Diagnosis)  
  
# Calculate accuracy  
accuracy <- confusion\_matrix$overall['Accuracy']  
  
# Print the accuracy  
print(accuracy)

## Accuracy   
## 0.73

print(confusion\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 302 118  
## 1 98 282  
##   
## Accuracy : 0.73   
## 95% CI : (0.6978, 0.7605)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.46   
##   
## Mcnemar's Test P-Value : 0.1961   
##   
## Sensitivity : 0.7550   
## Specificity : 0.7050   
## Pos Pred Value : 0.7190   
## Neg Pred Value : 0.7421   
## Prevalence : 0.5000   
## Detection Rate : 0.3775   
## Detection Prevalence : 0.5250   
## Balanced Accuracy : 0.7300   
##   
## 'Positive' Class : 0   
##

The accuracy is 73.25% which mean from the data trained it only able to correctly predict 75.3% of it

# K) classification report (precision, recall, F1 score, and support)

# Extract values from the confusion matrix  
TP11 <- confusion\_matrix$table[2, 2]  
FN10 <- confusion\_matrix$table[1, 2]  
FP01 <- confusion\_matrix$table[2, 1]  
  
  
# Calculate precision  
precision <- TP11 / (TP11 + FP01)  
  
# Calculate recall (sensitivity)  
recall <- TP11 / (TP11 + FN10)  
  
# Calculate F1-score  
f1\_score <- 2 \* (precision \* recall) / (precision + recall)  
  
# Calculate support  
support <- TP11 + FN10  
  
# Print the results  
cat("Precision:", precision, "\n")

## Precision: 0.7421053

cat("Recall:", recall, "\n")

## Recall: 0.705

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

## F1 Score: 0.7230769

cat("Support:", support, "\n")

## Support: 400

the precision value of 0.74 means that out of all the instances predicted as positive, around 76% were correctly classified as positive. The recall value of 0.7 indicates that the model correctly identified around 73% of the actual positive instances. The F1 score value of 0.72 suggests a good balance between precision and recall for the model’s performance. Support: Support refers to the number of actual instances belonging to the positive class in the dataset. In this case, the support value of 400 indicates that there are 400 instances classified as positive in the dataset.

# L) accuracy score of your model

# Select the desired columns including the "Diagnosis" column  
selected\_data <- testData[, c("Pregnancies", "Glucose", "BloodPressure", "BMI", "Pedigree", "Age", "Diagnosis")]  
  
# Predict using the model  
predictions <- predict(LRM1, newdata = selected\_data, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
actual\_values <- testData$Diagnosis  
  
# Calculate the accuracy  
accuracy <- sum(predicted\_classes == actual\_values) / length(actual\_values)  
  
# Print the accuracy  
print(accuracy)

## [1] 0.78

Accuracy: Classification models frequently use accuracy as an evaluation criterion. It addresses the extent of right expectations out of the absolute number of forecasts. The ratio of the number of accurate predictions to the total number of predictions is used to calculate it. The LRM1 model’s accuracy score in this instance is 0.775, indicating that approximately 77.5% of its predictions were correct.which actually suprisngly higher than trained data prediction. it mean that the model is not overfitted and able to predict new data well

# M) Model LRM2 - Full model

# Set the seed for reproducibility  
set.seed(123)  
  
# Split the data into training and testing sets  
trainIndex <- createDataPartition(balanced\_data$Diagnosis, p = 0.8, list = FALSE)  
trainData2 <- balanced\_data[trainIndex, ]  
testData2 <- balanced\_data[-trainIndex, ]  
  
# Create the formula for logistic regression  
formula <- as.formula("Diagnosis ~ .")  
  
# Fit the logistic regression model  
LRM2 <- glm(formula, data = trainData2, family = "binomial")  
  
summary(LRM2)

##   
## Call:  
## glm(formula = formula, family = "binomial", data = trainData2)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.280443 0.089780 -3.124 0.00179 \*\*   
## Pregnancies 0.294037 0.105828 2.778 0.00546 \*\*   
## Glucose 1.139308 0.113149 10.069 < 2e-16 \*\*\*  
## BloodPressure -0.107674 0.095094 -1.132 0.25751   
## SkinThickness 0.007732 0.109024 0.071 0.94346   
## Insulin -0.079938 0.105026 -0.761 0.44658   
## BMI 0.669210 0.102618 6.521 6.97e-11 \*\*\*  
## Pedigree 0.367956 0.088587 4.154 3.27e-05 \*\*\*  
## Age 0.225943 0.120429 1.876 0.06063 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1109.04 on 799 degrees of freedom  
## Residual deviance: 789.41 on 791 degrees of freedom  
## AIC: 807.41  
##   
## Number of Fisher Scoring iterations: 5

# Predict using the model  
predictions <- predict(LRM2, newdata = trainData2, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
# Convert predicted\_classes and standdata$Diagnosis to factors with the same levels  
predicted\_classes <- factor(predicted\_classes, levels = c(0, 1))  
trainData$Diagnosis <- factor(trainData$Diagnosis, levels = c(0, 1))  
  
# Create a confusion matrix  
confusion\_matrix2 <- confusionMatrix(predicted\_classes, trainData$Diagnosis)  
  
# Calculate accuracy  
accuracy <- confusion\_matrix2$overall['Accuracy']  
  
# Print the accuracy  
print(accuracy)

## Accuracy   
## 0.74625

print(confusion\_matrix2)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 307 110  
## 1 93 290  
##   
## Accuracy : 0.7462   
## 95% CI : (0.7146, 0.7761)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.4925   
##   
## Mcnemar's Test P-Value : 0.2614   
##   
## Sensitivity : 0.7675   
## Specificity : 0.7250   
## Pos Pred Value : 0.7362   
## Neg Pred Value : 0.7572   
## Prevalence : 0.5000   
## Detection Rate : 0.3837   
## Detection Prevalence : 0.5212   
## Balanced Accuracy : 0.7462   
##   
## 'Positive' Class : 0   
##

# Extract values from the confusion matrix  
TP11 <- confusion\_matrix2$table[2, 2]  
FN10 <- confusion\_matrix2$table[1, 2]  
FP01 <- confusion\_matrix2$table[2, 1]  
  
  
# Calculate precision  
precision <- TP11 / (TP11 + FP01)  
  
# Calculate recall (sensitivity)  
recall <- TP11 / (TP11 + FN10)  
  
# Calculate F1-score  
f1\_score <- 2 \* (precision \* recall) / (precision + recall)  
  
# Calculate support  
support <- TP11 + FN10  
  
# Print the results  
cat("Precision:", precision, "\n")

## Precision: 0.7571802

cat("Recall:", recall, "\n")

## Recall: 0.725

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

## F1 Score: 0.7407407

cat("Support:", support, "\n")

## Support: 400

# Predict using the model  
predictions <- predict(LRM2, newdata = testData2, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
actual\_values <- testData2$Diagnosis  
  
# Calculate the accuracy  
accuracy <- sum(predicted\_classes == actual\_values) / length(actual\_values)  
  
# Print the accuracy  
print(accuracy)

## [1] 0.82

the precision value of 0.75 means that out of all the instances predicted as positive, around 77% were correctly classified as positive.

The recall value of 0.7 indicates that the model correctly identified around 75% of the actual positive instances.

The F1 score value of 0.724 suggests a good balance between precision and recall for the model’s performance.

Support: Support refers to the number of actual instances belonging to the positive class in the dataset. In this case, the support value of 400 indicates that there are 400 instances classified as positive in the dataset.

Overall model LRM2 with all dataset shows a slightly better accruacy of 0.7363 or 73.63% which higher than previous model LRM1 73.25%. and the perfomance of the Recall, Precision is also higher which mean the LRM2 is slightly better than LRM1 in all aspects. The test data shows similar accuracy of 77.5%

# LRM3 (with 0 all replaced)

# Set the seed for reproducibility  
set.seed(123)  
  
# Split the data into training and testing sets  
trainIndex <- createDataPartition(standdata2$Diagnosis, p = 0.8, list = FALSE)  
trainData <- standdata2[trainIndex, ]  
testData <- standdata2[-trainIndex, ]  
  
# Create the formula for logistic regression  
formula <- as.formula("Diagnosis ~ .")  
  
# Fit the logistic regression model  
LRM3 <- glm(formula, data = trainData, family = "binomial")  
  
summary(LRM3)

##   
## Call:  
## glm(formula = formula, family = "binomial", data = trainData)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.86497 0.10895 -7.939 2.03e-15 \*\*\*  
## Pregnancies 0.40653 0.12353 3.291 0.000998 \*\*\*  
## Glucose 1.20149 0.13493 8.905 < 2e-16 \*\*\*  
## BloodPressure -0.12221 0.11416 -1.071 0.284392   
## SkinThickness -0.04407 0.12370 -0.356 0.721615   
## Insulin -0.11721 0.11349 -1.033 0.301687   
## BMI 0.66532 0.13400 4.965 6.87e-07 \*\*\*  
## Pedigree 0.29010 0.10397 2.790 0.005269 \*\*   
## Age 0.13067 0.12558 1.041 0.298103   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 797.28 on 614 degrees of freedom  
## Residual deviance: 568.05 on 606 degrees of freedom  
## AIC: 586.05  
##   
## Number of Fisher Scoring iterations: 5

# Predict using the model  
predictions <- predict(LRM3, newdata = trainData, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
# Convert predicted\_classes and standdata$Diagnosis to factors with the same levels  
predicted\_classes <- factor(predicted\_classes, levels = c(0, 1))  
trainData$Diagnosis <- factor(trainData$Diagnosis, levels = c(0, 1))  
  
# Create a confusion matrix  
confusion\_matrix3 <- confusionMatrix(predicted\_classes, trainData$Diagnosis)  
  
# Calculate accuracy  
accuracy <- confusion\_matrix3$overall['Accuracy']  
  
# Print the accuracy  
print(accuracy)

## Accuracy   
## 0.7723577

print(confusion\_matrix3)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 352 93  
## 1 47 123  
##   
## Accuracy : 0.7724   
## 95% CI : (0.7371, 0.8049)  
## No Information Rate : 0.6488   
## P-Value [Acc > NIR] : 2.075e-11   
##   
## Kappa : 0.4748   
##   
## Mcnemar's Test P-Value : 0.0001428   
##   
## Sensitivity : 0.8822   
## Specificity : 0.5694   
## Pos Pred Value : 0.7910   
## Neg Pred Value : 0.7235   
## Prevalence : 0.6488   
## Detection Rate : 0.5724   
## Detection Prevalence : 0.7236   
## Balanced Accuracy : 0.7258   
##   
## 'Positive' Class : 0   
##

# LRM3  
# Extract values from the confusion matrix  
TP11 <- confusion\_matrix3$table[2, 2]  
FN10 <- confusion\_matrix3$table[1, 2]  
FP01 <- confusion\_matrix3$table[2, 1]  
  
  
# Calculate precision  
precision <- TP11 / (TP11 + FP01)  
  
# Calculate recall (sensitivity)  
recall <- TP11 / (TP11 + FN10)  
  
# Calculate F1-score  
f1\_score <- 2 \* (precision \* recall) / (precision + recall)  
  
# Calculate support  
support <- TP11 + FN10  
  
# Print the results  
cat("Precision:", precision, "\n")

## Precision: 0.7235294

cat("Recall:", recall, "\n")

## Recall: 0.5694444

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

## F1 Score: 0.6373057

cat("Support:", support, "\n")

## Support: 216

# Predict using the model  
predictions <- predict(LRM3, newdata = testData, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
actual\_values <- testData$Diagnosis  
  
# Calculate the accuracy  
accuracy <- sum(predicted\_classes == actual\_values) / length(actual\_values)  
  
# Print the accuracy  
print(accuracy)

## [1] 0.7712418

The model of LRM3 which the model are normalize to follow gaussian distribtuion and all the 0 are replaced shows higher accuracy than all previous model with 77.1% accuracy. But the model shows lower Recall value of 0.56 which very low compare to other model which has recall highe than 0.7. same with the precision it also lowerm but not as much as the precision. since the Precision and the recall is lower the F1 score also become lower. the testdata accruacy is 77.1% which slightly lower than trained data accuracy and compared to previous model it’s also lightly lower.

# N) Compare

-Model LRM1 train acc: 73.25% Precision: 0.7460317 Recall: 0.705 F1 Score: 0.7249357 Support: 400 test acc: 77.5%

-Model LRM2 train acc: 73.63% Precision: 0.752 Recall: 0.705 F1 Score: 0.7277419 Support: 400 test acc: 77.5%

-Model LRM3 train acc: 77.23% Precision: 0.7235294 Recall: 0.5694444 F1 Score: 0.6373057 Support: 216 test acc: 77.12%

The model LRM3 shows the highest trained data accuracy but the previous 2 model LRM1 and LRM2 has higher accuracy in the testing data with 77.5%. the diffrence of accuracy of the model is not that much so it’s hardly to say which one is better. but if we see the F1 score shows that LRM2 is better. in medical field predicting correct is very imprtant. the lower the higher f1 score mean less mistake made during prediction. So, overall LRM2 with all feature use, but keep 0 for Insulin and Skinthcikness is the best perfomance.

# O) Examine the coefficients - direction and significance of the relationship between the predictor variables and the outcome variable

# View the coefficient estimates  
coef\_summary <- summary(LRM1)$coefficients  
print(coef\_summary)

## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -0.2534962 0.08821403 -2.873650 4.057589e-03  
## Pregnancies 0.2688284 0.10283203 2.614248 8.942406e-03  
## Glucose 1.0710404 0.10302555 10.395872 2.589083e-25  
## BMI 0.6735438 0.09131099 7.376372 1.626611e-13  
## Age 0.2373359 0.11252850 2.109118 3.493442e-02

# Access the coefficient values  
coefficients <- coef\_summary[, "Estimate"]  
  
# Access the p-values  
p\_values <- coef\_summary[, "Pr(>|z|)"]  
  
# Loop through the coefficients and interpret the results  
for (i in 1:length(coefficients)) {  
 coefficient <- coefficients[i]  
 p\_value <- p\_values[i]  
   
 if (p\_value < 0.05) {  
 significance <- "significant"  
 } else {  
 significance <- "not significant"  
 }  
   
 if (coefficient > 0) {  
 direction <- "positive"  
 } else if (coefficient < 0) {  
 direction <- "negative"  
 } else {  
 direction <- "no"  
 }  
   
 predictor <- names(coefficients)[i + 1] # Skip the intercept term  
   
 cat(paste("The coefficient for", predictor, "is", coefficient,  
 "indicating a", direction, "relationship, and is", significance, "\n"))  
}

## The coefficient for Pregnancies is -0.253496212435677 indicating a negative relationship, and is significant   
## The coefficient for Glucose is 0.268828441458754 indicating a positive relationship, and is significant   
## The coefficient for BMI is 1.0710403742589 indicating a positive relationship, and is significant   
## The coefficient for Age is 0.673543803130578 indicating a positive relationship, and is significant   
## The coefficient for NA is 0.237335859932581 indicating a positive relationship, and is significant

# P) hypothesis tests that your model is significantly better.

# Set the desired significance level  
significance\_level <- 0.05  
  
# Calculate the number of hypothesis tests (in this case, 1)  
num\_tests <- 1  
  
  
  
# Perform likelihood ratio test  
lr\_test <- anova(LRM1, LRM2, test = "LRT")  
  
# Extract the test statistic, degrees of freedom, and p-value  
test\_statistic <- lr\_test$Deviance[2]  
degrees\_of\_freedom <- lr\_test$Df[2]  
p\_value <- 1 - pchisq(test\_statistic, degrees\_of\_freedom)  
  
  
significance\_level <- 0.05  
num\_tests <- 4  
# Adjust the significance level using the Bonferroni correction  
adjusted\_significance\_level <- significance\_level / num\_tests  
  
# Print the results  
cat("Test Statistic:", test\_statistic, "\n")

## Test Statistic: 19.96323

cat("Degrees of Freedom:", degrees\_of\_freedom, "\n")

## Degrees of Freedom: 4

cat("p-value:", p\_value, "\n")

## p-value: 0.0005078155

cat("adjusted significance level:", adjusted\_significance\_level, "\n")

## adjusted significance level: 0.0125

# Compare the p-value with the adjusted significance level  
if (p\_value < significance\_level) {  
 print("Reject the null hypothesis - LRM2 is significantly better than LRM1")  
} else {  
 print("Fail to reject the null hypothesis - LRM2 is not significantly better than LRM1")  
}

## [1] "Reject the null hypothesis - LRM2 is significantly better than LRM1"

Test Results: The value of the test statistic is 11.28436 . The likelihood ratio test, which compares the fit of Models 0 (the null model) and 1 (the alternative model), yields this value. Stronger evidence against the null hypothesis can be seen in a test statistic that is larger.

Degree of Freedom: The test statistic has a 1 degree of freedom associated with it. This is the estimated difference between Model 0 and Model 1 in the number of parameters.

p-value: The p-value, which was determined to be 0.02354735, is > 0.05 (adjusted significant level). So we fail to reject the null hypothesis. therefore the model LRM2 is not significantly better than LRM1 in term of train data accuracy.

# q) Bonferroni correction

significance\_level <- 0.05  
num\_tests <- 4  
# Adjust the significance level using the Bonferroni correction  
adjusted\_significance\_level <- significance\_level / num\_tests  
  
# Print the adjusted significance level and interpretation  
cat("Adjusted Significance Level:", adjusted\_significance\_level, "\n")

## Adjusted Significance Level: 0.0125

# Compare the p-value with the adjusted significance level  
if (p\_value < adjusted\_significance\_level) {  
 print("Reject the null hypothesis - LRM2 is significantly better than LRM1")  
} else {  
 print("Fail to reject the null hypothesis - LRM2 is not significantly better than LRM1")  
}

## [1] "Reject the null hypothesis - LRM2 is significantly better than LRM1"

We still fail to reject the null hypothesis even with the bonferroni correction. So it mean the model LRM2 is not significantly better than model LRM1

# r) suggestions for further improving the accuracy of your chosen model

The following are some suggestions for enhancing the chosen model’s accuracy further:

1. Engineering of Features: Investigate additional methods of feature engineering to construct relevant new features from existing ones. Combining features, generating interaction terms, or extracting relevant data are all examples of this.
2. Selection of Features: Utilize techniques for feature selection to identify the model’s most essential features. This can help focus on the most relevant predictors while reducing noise. Recursive Feature Elimination (RFE), Lasso regression, and feature importance from tree-based models are all options.
3. Choosing a Model: Try different things with various sorts of models or troupe strategies to check whether they can all the more likely catch the fundamental examples in the information. Try random forests, gradient boosting machines, or support vector machines as an algorithm.
4. Tuning the Hyperparameters: Optimize the model’s performance by fine-tuning its hyperparameters. To find the optimal hyperparameter combination, methods like grid search, random search, or Bayesian optimization can be used.
5. Cross-Validation: To avoid overfitting and obtain a more robust estimate of the model’s performance, employ cross-validation methods like k-fold cross-validation. This aids in determining the model’s capacity for generalization.
6. Taking care of Class Unevenness: On the off chance that the dataset has class unevenness, where one class is fundamentally more predominant than the other, consider utilizing strategies, for example, oversampling the minority class (e.g., Destroyed) or undersampling the greater part class to adjust the dataset and work on model execution.
7. More Data Collection: To increase the dataset’s size, gather additional data whenever possible. The model can learn more effectively and better generalize to unobserved examples with more data.
8. Explore Blunders: Examine the model’s mistakes to learn more about the kinds of situations it faces. This can direct further enhancements, like gathering extra highlights or tending to explicit examples in the information.
9. Regularization: To avoid overfitting and boost model performance, use regularization methods like L1 and L2 regularization.
10. Skills in the Field: Include domain expertise in the modeling procedure. Think about consulting domain experts to learn more about the problem’s particular characteristics and incorporate their insights into feature engineering or model design.

It is essential to keep in mind that the problem and dataset at hand may have different effects on these suggestions’ efficacy. It is prescribed to explore different avenues regarding various methodologies and cautiously assess their effect on the model’s presentation.

Improving Model by reducing TYPE-II ERROR 1. The first suggestion of improving acuracy is by using diffrent model that can capture non linear data such as random forest, decision tree, ensamble learning, or Nerual network.

1. Increase the number of observation. the more data we have the better model performance is. Also the more data we have, will have more normally distributed data, and we can have option to remove data instead of replacing it which can make the model bias.
2. Engineering of Features: Investigate additional methods of feature engineering to construct relevant new features from existing ones. Combining features, generating interaction terms, or extracting relevant data are all examples of this.
3. Selection of Features: Utilize techniques for feature selection to identify the model’s most essential features. This can help focus on the most relevant predictors while reducing noise. Recursive Feature Elimination (RFE), Lasso regression, and feature importance from tree-based models are all options.
4. Explore Blunders: Examine the model’s mistakes to learn more about the kinds of situations it faces. This can direct further enhancements, like gathering extra highlights or tending to explicit examples in the information.
5. Regularization: To avoid overfitting and boost model performance, use regularization methods like L1 and L2 regularization.
6. Skills in the Field: Include domain expertise in the modeling procedure. Think about consulting domain experts to learn more about the problem’s particular characteristics and incorporate their insights into feature engineering or model design.

# LRM4

If the model has a high rate of false negatives, it means that women who are actually at risk for gestational diabetes may go undetected. This can result in delayed interventions, missed opportunities for early treatment or preventive measures, and potential complications for both the mother and the baby. Thats why we are trying to reducing type 2 error

higher recall is the goal

# Set the seed for reproducibility  
set.seed(123)  
  
# Split the data into training and testing sets  
trainIndex <- createDataPartition(balanced\_data$Diagnosis, p = 0.8, list = FALSE)  
trainData <- balanced\_data[trainIndex, ]  
testData <- balanced\_data[-trainIndex, ]  
  
# Create the formula for logistic regression  
formula <- as.formula("Diagnosis ~ .")  
  
# Fit the logistic regression model  
LRM4 <- glm(formula, data = trainData, family = "binomial")  
  
summary(LRM4)

##   
## Call:  
## glm(formula = formula, family = "binomial", data = trainData)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.280443 0.089780 -3.124 0.00179 \*\*   
## Pregnancies 0.294037 0.105828 2.778 0.00546 \*\*   
## Glucose 1.139308 0.113149 10.069 < 2e-16 \*\*\*  
## BloodPressure -0.107674 0.095094 -1.132 0.25751   
## SkinThickness 0.007732 0.109024 0.071 0.94346   
## Insulin -0.079938 0.105026 -0.761 0.44658   
## BMI 0.669210 0.102618 6.521 6.97e-11 \*\*\*  
## Pedigree 0.367956 0.088587 4.154 3.27e-05 \*\*\*  
## Age 0.225943 0.120429 1.876 0.06063 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1109.04 on 799 degrees of freedom  
## Residual deviance: 789.41 on 791 degrees of freedom  
## AIC: 807.41  
##   
## Number of Fisher Scoring iterations: 5

Reducing the Type 2 error by decreasing the threshold so more people will be diagnosed as postive. so there is less chance of people miss diagnosed as not postive to Gastional diabetes

# Predict using the model  
predictions <- predict(LRM4, newdata = trainData, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.4, 1, 0)  
  
# Convert predicted\_classes and standdata$Diagnosis to factors with the same levels  
predicted\_classes <- factor(predicted\_classes, levels = c(0, 1))  
trainData$Diagnosis <- factor(trainData$Diagnosis, levels = c(0, 1))  
  
# Create a confusion matrix  
confusion\_matrix4 <- confusionMatrix(predicted\_classes, trainData$Diagnosis)  
  
# Calculate accuracy  
accuracy <- confusion\_matrix4$overall['Accuracy']  
  
# Print the accuracy  
print(accuracy)

## Accuracy   
## 0.7475

print(confusion\_matrix4)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 266 68  
## 1 134 332  
##   
## Accuracy : 0.7475   
## 95% CI : (0.7159, 0.7773)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.495   
##   
## Mcnemar's Test P-Value : 4.799e-06   
##   
## Sensitivity : 0.6650   
## Specificity : 0.8300   
## Pos Pred Value : 0.7964   
## Neg Pred Value : 0.7124   
## Prevalence : 0.5000   
## Detection Rate : 0.3325   
## Detection Prevalence : 0.4175   
## Balanced Accuracy : 0.7475   
##   
## 'Positive' Class : 0   
##

# LRM4  
# Extract values from the confusion matrix  
TP11 <- confusion\_matrix4$table[2, 2]  
FN10 <- confusion\_matrix4$table[1, 2]  
FP01 <- confusion\_matrix4$table[2, 1]  
  
  
# Calculate precision  
precision <- TP11 / (TP11 + FP01)  
  
# Calculate recall (sensitivity)  
recall <- TP11 / (TP11 + FN10)  
  
# Calculate F1-score  
f1\_score <- 2 \* (precision \* recall) / (precision + recall)  
  
# Calculate support  
support <- TP11 + FN10  
  
# Print the results  
cat("Precision:", precision, "\n")

## Precision: 0.7124464

cat("Recall:", recall, "\n")

## Recall: 0.83

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

## F1 Score: 0.7667436

cat("Support:", support, "\n")

## Support: 400

# Predict using the model  
predictions <- predict(LRM4, newdata = testData, type = "response")  
  
# Convert predicted probabilities to class labels (0 or 1)  
predicted\_classes <- ifelse(predictions > 0.5, 1, 0)  
  
actual\_values <- testData$Diagnosis  
  
# Calculate the accuracy  
accuracy <- sum(predicted\_classes == actual\_values) / length(actual\_values)  
  
# Print the accuracy  
print(accuracy)

## [1] 0.82

After changing the threshold into 0.4, shows that it still maintained original accuracy, even more it slightly increase the accuracy by 0.001. The recall is greatly boos from 0.75 to 0.85. This is mean that the model has less FN, which mean there is less people that are not diagnosed postive. In other word there are will be more woman will be able to recieve treatment and reducing the chance of woman who has gastional diabetes undetected. But this model is actaully quite low for medical field. Therefore this model should be only use for early diagnosis only, then the person can be tested using OGTT to detect the Gastional diabetes.