# Comparative Analysis of Predictive Models for Early GDM detection

## DATA EXPLORATION AND PREPARATION

``` r  
# Install required packages  
  
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(corrplot)

## corrplot 0.95 loaded

data <- read.csv("C:/Users/ERC/Downloads/assignment/patients.csv")

str(data)

## 'data.frame': 768 obs. of 9 variables:  
## $ Pregnancies : int 6 1 8 1 0 5 3 10 2 8 ...  
## $ Glucose : int 148 85 183 89 137 116 78 115 197 125 ...  
## $ BloodPressure: int 72 66 64 66 40 74 50 0 70 96 ...  
## $ SkinThickness: int 35 29 0 23 35 0 32 0 45 0 ...  
## $ Insulin : int 0 0 0 94 168 0 88 0 543 0 ...  
## $ BMI : num 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...  
## $ Pedigree : num 0.627 0.351 0.672 0.167 2.288 ...  
## $ Age : int 50 31 32 21 33 30 26 29 53 54 ...  
## $ Diagnosis : int 1 0 1 0 1 0 1 0 1 1 ...

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

cols\_to\_check <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")  
data[cols\_to\_check] <- lapply(data[cols\_to\_check], function(x) ifelse(x == 0, NA, x))  
  
colSums(is.na(data))

## Pregnancies Glucose BloodPressure SkinThickness Insulin   
## 0 5 35 227 374   
## BMI Pedigree Age Diagnosis   
## 11 0 0 0

Calculate and interpret basic statistics (mean, median, SD, quartiles) for each variable.  
basic\_stats <- data.frame(  
 Mean = sapply(data, mean, na.rm = TRUE),  
 Median = sapply(data, median, na.rm = TRUE),  
 SD = sapply(data, sd, na.rm = TRUE),  
 Q1 = sapply(data, function(x) quantile(x, 0.25, na.rm = TRUE)),  
 Q3 = sapply(data, function(x) quantile(x, 0.75, na.rm = TRUE))  
)   
  
print(basic\_stats)

## Mean Median SD Q1 Q3  
## Pregnancies 3.8450521 3.0000 3.3695781 1.00000 6.00000  
## Glucose 121.6867628 117.0000 30.5356411 99.00000 141.00000  
## BloodPressure 72.4051842 72.0000 12.3821582 64.00000 80.00000  
## SkinThickness 29.1534196 29.0000 10.4769824 22.00000 36.00000  
## Insulin 155.5482234 125.0000 118.7758552 76.25000 190.00000  
## BMI 32.4574637 32.3000 6.9249883 27.50000 36.60000  
## Pedigree 0.4718763 0.3725 0.3313286 0.24375 0.62625  
## Age 33.2408854 29.0000 11.7602315 24.00000 41.00000  
## Diagnosis 0.3489583 0.0000 0.4769514 0.00000 1.00000

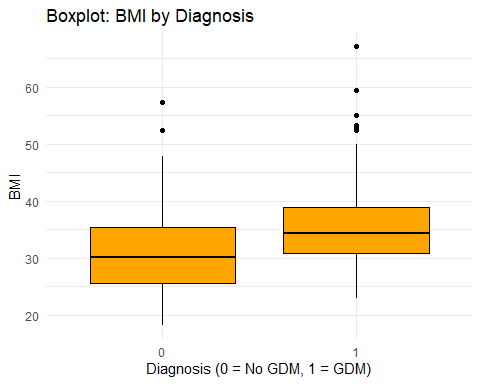
#Based on the summary, the dataset presents several notable data issues that literature consistently highlights as important to address before analysis. First, biologically implausible zero values appear frequently in key clinical variables specifically, SkinThickness and Insulin as well as smaller but nontrivial percentages of Glucose, BloodPressure, and BMI. These are not physiologically plausible to be zero. Overall, the dataset’s core statistical measures—means, medians, quartiles—are all biased by the zero-coded missing values and pronounced skewness/outliers. Literature strongly recommends preprocessing steps including median or mean-based imputation or normalization to manage skewed distributions, and outlier-handling via capping or robust scaling.

## VISUALIZATION: five visualizations using ggplot2

### **BOXPLOT: BMI BY DIAGNOSIS**

ggplot(data, aes(x = factor(Diagnosis), y = BMI)) +  
 geom\_boxplot(fill = "orange", color = "black") +  
 labs(title = "Boxplot: BMI by Diagnosis",  
 x = "Diagnosis (0 = No GDM, 1 = GDM)",  
 y = "BMI") +  
 theme\_minimal()

## Warning: Removed 11 rows containing non-finite outside the scale range  
## (`stat\_boxplot()`).

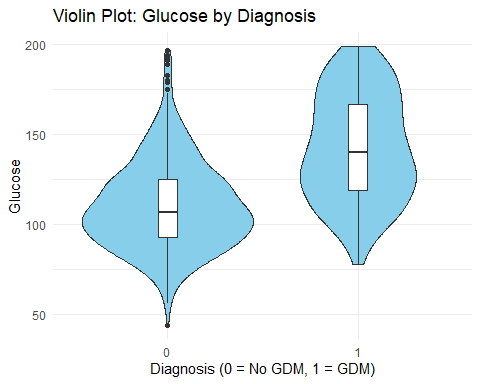


# Analysis of the plot shows that women diagnosed with GDM tend to have higher BMI values—both in median and upper range—compared to those without GDM (coded 0), indicating that the GDM group not only has a higher typical BMI, but also greater variability and more extreme BMI outliers. This visual finding aligns with clinical evidence showing that elevated early-pregnancy or pre-pregnancy BMI is a strong independent predictor of GDM.

**#Violin Plot: Glucose by Diagnosis**  
  
ggplot(data, aes(x = factor(Diagnosis), y = Glucose)) +  
 geom\_violin(fill = "skyblue") +  
 geom\_boxplot(width = 0.1, fill = "white") +  
 labs(title = "Violin Plot: Glucose by Diagnosis",  
 x = "Diagnosis (0 = No GDM, 1 = GDM)",  
 y = "Glucose") +  
 theme\_minimal()

## Warning: Removed 5 rows containing non-finite outside the scale range  
## (`stat\_ydensity()`).

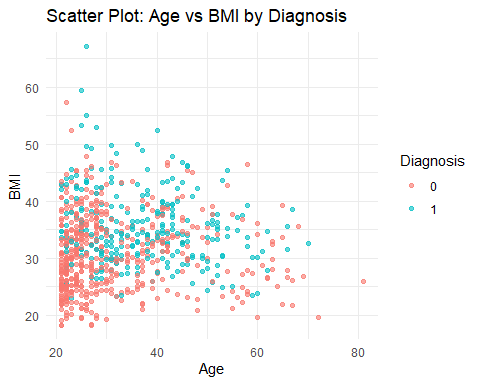
## Warning: Removed 5 rows containing non-finite outside the scale range  
## (`stat\_boxplot()`).



#Analysis of the violin plot explains the comparison of glucose levels without GDM (0) and with GDM (1), combining density distribution and boxplot elements into a single visual. The GDM group shows a notably wider and taller “violin,” especially above ~130 mg/dL, indicating a higher density of elevated glucose readings and several extreme values, while also having a higher median and wider interquartile range. In contrast, the non-GDM group’s violin is narrower and more concentrated around lower glucose values (~80–110 mg/dL), with fewer high outliers. Essentially, this plot reveals that not only do women with GDM tend to have higher typical glucose levels, but their glucose readings are more variable and skewed toward elevated values.  
  
SCATTER PLOT: AGE VS BMI BY DIAGNOSIS

ggplot(data, aes(x = Age, y = BMI, color = factor (Diagnosis))) +  
 geom\_point(alpha = 0.6) +  
 labs(title = "Scatter Plot: Age vs BMI by Diagnosis",  
 x = "Age",  
 y = "BMI",  
 color = "Diagnosis") +  
 theme\_minimal()

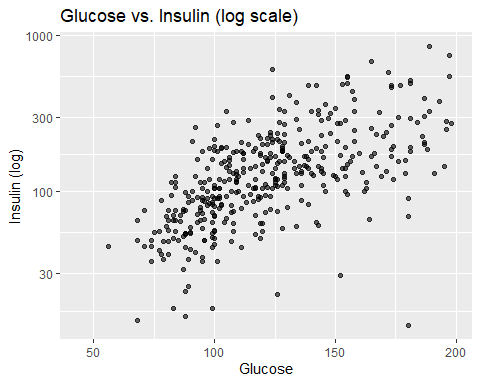
## Warning: Removed 11 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



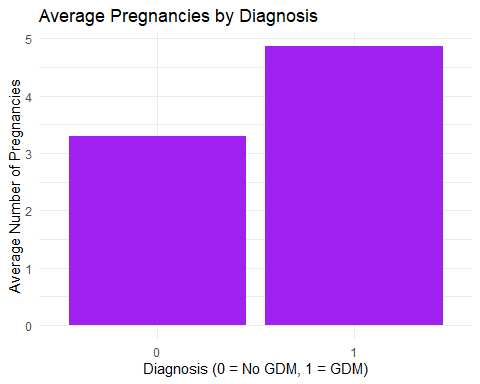
#The scatterplot illustrates how age and BMI jointly relate to GDM diagnosis, with red points (no GDM) and blue points (GDM). It reveals that GDM is more common in older individuals, especially those aged 40 and above, and among women with higher BMI,  
  
# Scatter plot: Glucose vs Insulin (log scale to reduce skew)

ggplot(data, aes(x = Glucose, y = Insulin)) +  
 geom\_point(alpha = 0.6) +  
 scale\_y\_log10() +  
 labs(title = "Glucose vs. Insulin (log scale)", x = "Glucose", y = "Insulin (log)")

## Warning: Removed 375 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

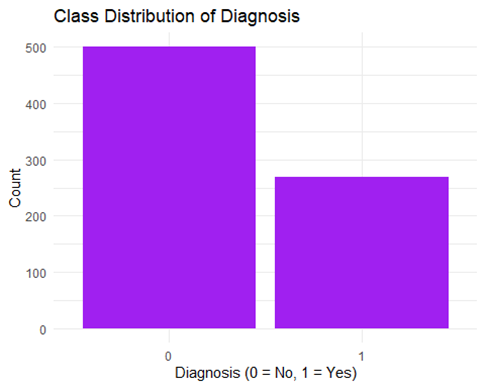


#Analysis of the scatter plot glucose vs. insulin (on a logarithmic scale) shows a clear positive relationship—with higher blood glucose levels generally associated with higher insulin levels—supporting the notion of insulin resistance: as glucose rises, the pancreas ramps up insulin production as described in the literature reviews. However, the relationship is not perfectly linear, and there's considerable vertical spread at higher glucose values indicating variability in individual insulin responses.   
  
**#Bar Plot: Average Pregnancies by Diagnosis**  
  
library(dplyr)  
  
# Calculate average pregnancies by diagnosis  
avg\_preg <- data %>%  
 group\_by(Diagnosis) %>%  
 summarise(avg\_pregnancies = mean(Pregnancies, na.rm = TRUE))  
  
# Bar plot  
ggplot(avg\_preg, aes(x = factor(Diagnosis), y = avg\_pregnancies)) +  
 geom\_bar(stat = "identity", fill = "purple") +  
 labs(title = "Average Pregnancies by Diagnosis",  
 x = "Diagnosis (0 = No GDM, 1 = GDM)",  
 y = "Average Number of Pregnancies") +  
 theme\_minimal()



# Analysis of this plot shows that the average pregnancies by Diagnosis. It shows that individuals diagnosed with GDM had more pregnancies than those with no diagnosis of the disease. Therefore, this plot shows a positive association between the number of pregnancies and the likelihood of developing GDM during pregnancy.

*# Bar chart of the classes*  
**ggplot**(patients, **aes**(x = **factor**(Diagnosis))) **+**  
 **geom\_bar**(fill = "purple") **+**  
 **labs**(title = "Class Distribution of Diagnosis",  
 x = "Diagnosis (0 = No, 1 = Yes)",  
 y = "Count") **+**  
 **theme\_minimal**()



# Slight class imbalance exists. Positive class is approximately 35% of the overall dataset. This could potentially mean that we have to use oversampling or undersampling techniques for models that are susceptible to class imbalance.

### MISSING VALUE ANALYSIS

**# Impute missing values**   
  
data[cols\_to\_check] <- lapply(data[cols\_to\_check], function(x) {  
 x[is.na(x)] <- median(x, na.rm = TRUE)  
 return(x)  
})  
colSums(is.na(data[cols\_to\_check]))

## Glucose BloodPressure SkinThickness Insulin BMI   
## 0 0 0 0 0

summary(data)

## Pregnancies Glucose BloodPressure SkinThickness   
## Min. : 0.000 Min. : 44.00 Min. : 24.00 Min. : 7.00   
## 1st Qu.: 1.000 1st Qu.: 99.75 1st Qu.: 64.00 1st Qu.:25.00   
## Median : 3.000 Median :117.00 Median : 72.00 Median :29.00   
## Mean : 3.845 Mean :121.66 Mean : 72.39 Mean :29.11   
## 3rd Qu.: 6.000 3rd Qu.:140.25 3rd Qu.: 80.00 3rd Qu.:32.00   
## Max. :17.000 Max. :199.00 Max. :122.00 Max. :99.00   
## Insulin BMI Pedigree Age   
## Min. : 14.0 Min. :18.20 Min. :0.0780 Min. :21.00   
## 1st Qu.:121.5 1st Qu.:27.50 1st Qu.:0.2437 1st Qu.:24.00   
## Median :125.0 Median :32.30 Median :0.3725 Median :29.00   
## Mean :140.7 Mean :32.46 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

#Median imputation is implemented to solve the missing values. Those key numerical variables such as Glucose,BloodPressure, SkinThickness, Insulin, and BMI were rectified during the median imputation. We chose Median over the mean, because it is robust to outliers and is a better representation of the central tendency of skewed distributions, particularly common in clinical data. The median is also less sensitive to extreme values and ensures a more realistic estimation for those outliers or missing entries.

**# Function to detect outliers using IQR**  
detect\_outliers <- function(x) {  
 Q1 <- quantile(x, 0.25, na.rm = TRUE)  
 Q3 <- quantile(x, 0.75, na.rm = TRUE)  
 IQR <- Q3 - Q1  
 lower <- Q1 - 1.5 \* IQR  
 upper <- Q3 + 1.5 \* IQR  
 return(which(x < lower | x > upper))  
}  
  
**# Variables to check**  
cols\_to\_check <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")

### **OUTLIER DETECTION AND HANDLING**

# Count of outliers per variable  
sapply(data[cols\_to\_check], function(x) length(detect\_outliers(x)))

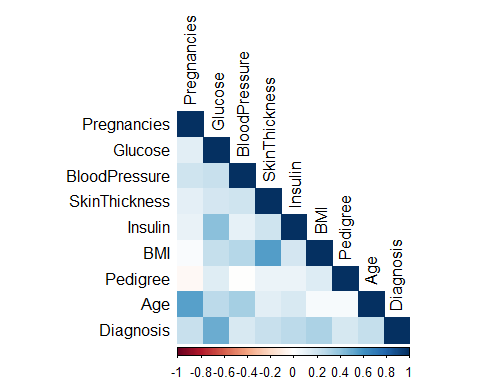
## Glucose BloodPressure SkinThickness Insulin BMI   
## 0 14 87 346 8

#Capping outliers  
cap\_outliers <- function(x) {  
 Q1 <- quantile(x, 0.25, na.rm = TRUE)  
 Q3 <- quantile(x, 0.75, na.rm = TRUE)  
 IQR <- Q3 - Q1  
 lower <- Q1 - 1.5 \* IQR  
 upper <- Q3 + 1.5 \* IQR  
 x[x < lower] <- lower  
 x[x > upper] <- upper  
 return(x)  
}  
  
data[cols\_to\_check] <- lapply(data[cols\_to\_check], cap\_outliers)  
  
#The technique of capping is performed to outliers without discarding datapoints. Instead of removing extreme values, you set upper and lower bounds and replace all values outside those bounds with the boundary values themselves. This method was selected because we wanted to manage outliers while preserving every data point and maintaining sample integrity. This method reduces the undue influence of outliers on summary statistics like the mean and variance, resulting in more robust model estimates.

# **FEATURE SELECTION AND DATA HANDLING**

**#Correlation matrix**

library(corrplot)  
  
**# Select numeric predictors only (excluding Diagnosis)**  
numeric\_vars <- data[, sapply(data, is.numeric)]  
cor\_matrix <- cor(numeric\_vars, use = "complete.obs")  
  
**# Visualization**  
corrplot(cor\_matrix, method = "color", type = "lower", tl.col = "black")



This is a correlation matrix heatmap showing the relationships between predictors and the dependent variable. The Darker blue color here shows a strong positive correlation (+1); the White color signifies no correlation (0); and the Dark Red shows a stronger negative correlation (-1). The strongest correlations are Glucose (someone with higher glucose levels is more likely to be diagnosed with GDM or diabetes), BMI (Body Mass Index) is moderately positive; Insulin and pregnancies are also moderately positive correlation with GDM as shown by the color of the correlation. The weak or no correlation are Blood pressure, Pedigree, and Skin thickness.

### Logistic Regression Training using the cleaned `data` from previous code  
  
# Ensure required packages are installed and loaded  
if (!require(caret)) install.packages("caret")

## Loading required package: caret

## Loading required package: lattice

if (!require(e1071)) install.packages("e1071")

## Loading required package: e1071

library(caret)  
library(e1071)  
  
# Prepare the data  
data$Diagnosis <- as.factor(data$Diagnosis)  
  
# Data partitioning ( training and testing set)  
set.seed(123) # For reproducibility  
trainIndex <- createDataPartition(data$Diagnosis, p = 0.8, list = FALSE)  
trainData <- data[trainIndex, ]  
testData <- data[-trainIndex, ]  
  
# Training the logistic regression model  
model <- glm(Diagnosis ~ ., data = trainData, family = "binomial")  
summary(model)

##   
## Call:  
## glm(formula = Diagnosis ~ ., family = "binomial", data = trainData)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -9.619262 1.807126 -5.323 1.02e-07 \*\*\*  
## Pregnancies 0.101988 0.036075 2.827 0.0047 \*\*   
## Glucose 0.035504 0.004182 8.489 < 2e-16 \*\*\*  
## BloodPressure -0.008883 0.009755 -0.911 0.3625   
## SkinThickness 0.004312 0.016952 0.254 0.7992   
## Insulin 0.008848 0.014067 0.629 0.5293   
## BMI 0.082961 0.020619 4.024 5.73e-05 \*\*\*  
## Pedigree 0.828420 0.334587 2.476 0.0133 \*   
## Age 0.011990 0.010441 1.148 0.2508   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 796.05 on 614 degrees of freedom  
## Residual deviance: 589.62 on 606 degrees of freedom  
## AIC: 607.62  
##   
## Number of Fisher Scoring iterations: 5

# Predictions  
pred\_probs <- predict (model, newdata = testData, type = "response")  
pred\_classes <- ifelse(pred\_probs > 0.5, 1, 0)

*RESULTS*  
  
# Confusion Matrix for 0.5 threshold  
conf\_matrix <- confusionMatrix(  
 factor(pred\_classes, levels = c(0, 1)),  
 factor(testData$Diagnosis, levels = c(0, 1))  
)  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 91 21  
## 1 9 32  
##   
## Accuracy : 0.8039   
## 95% CI : (0.7321, 0.8636)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 3.3e-05   
##   
## Kappa : 0.5426   
##   
## Mcnemar's Test P-Value : 0.04461   
##   
## Sensitivity : 0.9100   
## Specificity : 0.6038   
## Pos Pred Value : 0.8125   
## Neg Pred Value : 0.7805   
## Prevalence : 0.6536   
## Detection Rate : 0.5948   
## Detection Prevalence : 0.7320   
## Balanced Accuracy : 0.7569   
##   
## 'Positive' Class : 0   
##

# Threshold 0.4  
pred\_classes\_40 <- ifelse(pred\_probs > 0.4, 1, 0)  
conf\_matrix\_40 <- confusionMatrix(  
 factor(pred\_classes\_40, levels = c(0, 1)),  
 factor(testData$Diagnosis, levels = c(0, 1))  
)  
print(conf\_matrix\_40)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 83 14  
## 1 17 39  
##   
## Accuracy : 0.7974   
## 95% CI : (0.7249, 0.858)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 7.169e-05   
##   
## Kappa : 0.5584   
##   
## Mcnemar's Test P-Value : 0.7194   
##   
## Sensitivity : 0.8300   
## Specificity : 0.7358   
## Pos Pred Value : 0.8557   
## Neg Pred Value : 0.6964   
## Prevalence : 0.6536   
## Detection Rate : 0.5425   
## Detection Prevalence : 0.6340   
## Balanced Accuracy : 0.7829   
##   
## 'Positive' Class : 0   
##

# Threshold 0.45  
pred\_classes\_45 <- ifelse(pred\_probs > 0.45, 1, 0)  
conf\_matrix\_45 <- confusionMatrix(  
 factor(pred\_classes\_45, levels = c(0, 1)),  
 factor(testData$Diagnosis, levels = c(0, 1))  
)  
print(conf\_matrix\_45)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 89 18  
## 1 11 35  
##   
## Accuracy : 0.8105   
## 95% CI : (0.7393, 0.8692)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 1.46e-05   
##   
## Kappa : 0.568   
##   
## Mcnemar's Test P-Value : 0.2652   
##   
## Sensitivity : 0.8900   
## Specificity : 0.6604   
## Pos Pred Value : 0.8318   
## Neg Pred Value : 0.7609   
## Prevalence : 0.6536   
## Detection Rate : 0.5817   
## Detection Prevalence : 0.6993   
## Balanced Accuracy : 0.7752   
##   
## 'Positive' Class : 0   
##

*ANALYSIS*

*In binary classification, the common threshold is 0.5 and determines the probability cutoff for predicting class 1 vs class 0. Modifying this threshold will affect the model’s overall performance of trade-offs, particularly sensitivity and specificity. From those confusions' matrices, a conclusion can be made that lowering the threshold (from 0.5 to 0.4) leads to an increased specificity (model becoming more aware in predicting positive class), which leads to fewer false positives.*

*There is also a decreased sensitivity; the model becomes more likely to miss actual positives (people with no GDM diagnosis), resulting in an increase in false negatives. The model sensitivity shows a drop from 0.91 to 0.83, while specificity rises sharply (0.60 to 0.74). We notice a balance at the 0.45 threshold; the specificity is reduced from the 0.4 threshold but reflects a sharp increase from the 0.5 threshold. The sensitivity is also slightly reduced (0.91 to 0.89).*

# AUC & ROC Curve  
if (!require(pROC)) install.packages("pROC")

## Loading required package: pROC

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

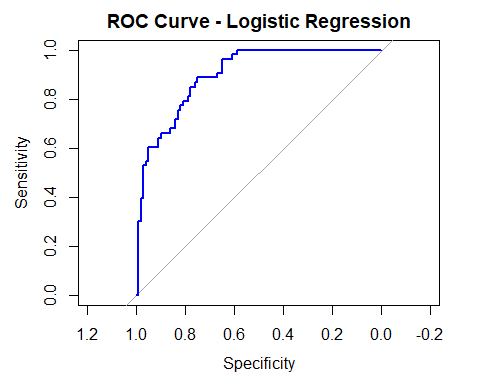
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(pROC)  
roc\_obj <- roc(testData$Diagnosis, pred\_probs)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

plot(roc\_obj, col = "blue", main = "ROC Curve - Logistic Regression")



*Area Under Curve (roc\_obj)*

**## Area under the curve: 0.8975**

# Cross-Validation  
ctrl <- trainControl(method = "cv", number = 10)  
cv\_model <- train(Diagnosis ~ .,   
 data = data,   
 method = "glm",   
 family = "binomial",   
 trControl = ctrl)  
print(cv\_model)

***The ROC (Receiver Operating Characteristic) curve is designed to show sensitivity (True Positive) vs 1 – Specificity (False Positive). It also helps to assess the model’s discriminative ability.***

## Generalized Linear Model   
##   
## 768 samples  
## 8 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10-fold)   
## Summary of sample sizes: 691, 691, 691, 692, 691, 691, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7616883 0.4489684

***We trained a GLM on sample data of 768 observations using 10-fold cross-validation. An accuracy of approximately 76.2% was obtained, with a Kappa score of 0.45, suggesting a moderate predictive power and fair agreement between the predicted and true outcomes. The model shows a somewhat acceptable performance with no sign of overfitting.***

##############################################################

**MODEL BUILDING AND VALIDATION: SUPPORT VECTOR MACHINE (SVM)**

##############################################################

**# LASSO Feature Selection**

if (!require(glmnet)) install.packages("glmnet")

## Loading required package: glmnet

## Loading required package: Matrix

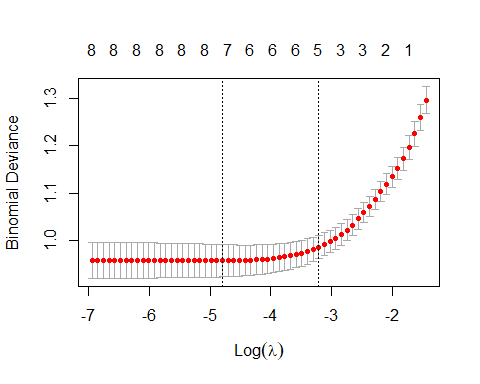
## Loaded glmnet 4.1-9

library(glmnet)  
  
x <- model.matrix(Diagnosis ~ ., data = data)[, -1]  
y <- data$Diagnosis  
  
lasso\_cv <- cv.glmnet(x, y, alpha = 1, family = "binomial")  
plot(lasso\_cv)

*LASSO (Least Absolute Shrinkage and Selection Operator) is a regularization technique commonly used in regression models. It is very useful for feature selection.*

*LASSO helps you choose the most important variables for a prediction model, by adding a penalty to the model, forcing it to use less variables. If a variable is not crucial for prediction, LASSO will shrink its coefficient to 0, prompting the model to ignore it.*

*The amount of penalty applied is controlled by Lambda; therefore, a bigger Lamba means more shrinking and fewer variables left in the model.*



lasso\_coef <- coef(lasso\_cv, s = "lambda.min")  
print(lasso\_coef)

## 9 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) -10.028806864  
## Pregnancies 0.106810799  
## Glucose 0.032636544  
## BloodPressure 0   
## SkinThickness 0.001647376  
## Insulin 0.012573503  
## BMI 0.079973451  
## Pedigree 0.673206607  
## Age 0.008678611

***ANALYSIS OF RESULTS***

***Here we can see that LASSO has selected 7 out of 8 features. All the non-zero coefficients' values were the ones selected by LASSO. Blood pressure dropped; therefore Pregnancies, Glucose, SkinThickness, Insulin, BMI, Pedigree, and Age were selected as important predictors for our target variable (GDM).***

**# --- SVM with SMOTE and Hyperparameter Tuning ---**

***SMOTE (Synthetic Minority Over-sampling Technique) is a common technique used in cases of imbalanced classification problems. It is mostly used in scenarios where the positive or minority class has fewer examples than the other class.***

if (!require(smotefamily)) install.packages("smotefamily")

## Loading required package: smotefamily

if (!require(DMwR2)) install.packages("DMwR2")

## Loading required package: DMwR2

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

if (!require(e1071)) install.packages("e1071")  
if (!require(caret)) install.packages("caret")  
library(smotefamily)  
library(DMwR2)  
library(e1071)  
library(caret)  
  
**# Prepare for SMOTE**  
data$Diagnosis <- as.factor(data$Diagnosis)  
x\_smote <- data[, -which(names(data) == "Diagnosis")]  
y\_smote <- data$Diagnosis  
smote\_result <- SMOTE(x\_smote, y\_smote, K = 5, dup\_size = 1)  
balanced\_data <- smote\_result$data  
names(balanced\_data)[ncol(balanced\_data)] <- "Diagnosis"  
balanced\_data$Diagnosis <- as.factor(balanced\_data$Diagnosis)  
  
**# Split balanced data**  
set.seed(123)  
index <- sample(1:nrow(balanced\_data), 0.7 \* nrow(balanced\_data))  
train\_data <- balanced\_data[index, ]  
test\_data <- balanced\_data[-index, ]  
  
**# Tuning SVM**  
tune\_result <- tune(svm,  
 Diagnosis ~ .,  
 data = train\_data,  
 kernel = "radial",  
 ranges = list(cost = c(0.1, 1, 10, 100),  
 gamma = c(0.001, 0.01, 0.1, 1)))  
summary(tune\_result)

***The tuning function was used with 10-fold cross-validation to find the best pairs of cost and gamma.***

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 10 0.1  
##   
## - best performance: 0.1945396   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 0.1 0.001 0.5296613 0.06509911  
## 2 1.0 0.001 0.2401256 0.03549553  
## 3 10.0 0.001 0.2318683 0.03771371  
## 4 100.0 0.001 0.2208714 0.04074741  
## 5 0.1 0.010 0.2332382 0.03633488  
## 6 1.0 0.010 0.2277207 0.04116878  
## 7 10.0 0.010 0.2249810 0.04409885  
## 8 100.0 0.010 0.2001142 0.04799709  
## 9 0.1 0.100 0.2277588 0.04388740  
## 10 1.0 0.100 0.2002473 0.05321718  
## 11 10.0 0.100 0.1945396 0.03864228  
## 12 100.0 0.100 0.2233067 0.05267301  
## 13 0.1 1.000 0.4824962 0.07807407  
## 14 1.0 1.000 0.2152017 0.04804639  
## 15 10.0 1.000 0.2165906 0.04927626  
## 16 100.0 1.000 0.2165906 0.04927626

***The best parameters found were cost = 10 and gamma = 0.1 as they correspond to the lowest error across all tested combinations.***

***The best performance = 0.1945 means that the model had a cross-validated error rate of ~19.45% or an accuracy of about 80.55%***

**# Train best model**  
best\_model <- svm(Diagnosis ~ .,   
 data = train\_data,   
 kernel = "radial",   
 cost = tune\_result$best.parameters$cost,  
 gamma = tune\_result$best.parameters$gamma,  
 probability = TRUE)

**# Predict**  
svm\_probs <- predict(best\_model, test\_data, probability = TRUE)  
probs <- attr(svm\_probs, "probabilities")[, "1"]  
custom\_pred <- ifelse(probs > 0.4, "1", "0")  
custom\_pred <- factor(custom\_pred, levels = c("0", "1"))

**# Evaluate**  
conf\_mat <- confusionMatrix(custom\_pred, test\_data$Diagnosis)  
print(conf\_mat)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 98 23  
## 1 46 144  
##   
## Accuracy : 0.7781   
## 95% CI : (0.7278, 0.8231)  
## No Information Rate : 0.537   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5489   
##   
## Mcnemar's Test P-Value : 0.008085   
##   
## Sensitivity : 0.6806   
## Specificity : 0.8623   
## Pos Pred Value : 0.8099   
## Neg Pred Value : 0.7579   
## Prevalence : 0.4630   
## Detection Rate : 0.3151   
## Detection Prevalence : 0.3891   
## Balanced Accuracy : 0.7714   
##   
## 'Positive' Class : 0   
##

***The results of the SVM are pretty good with an accuracy of 78% approximately and a sensitivity and specificity of 68% and 86%.***

# F1, F2 Scores  
precision <- conf\_mat$byClass["Pos Pred Value"]  
recall <- conf\_mat$byClass["Sensitivity"]  
f1 <- 2 \* (precision \* recall) / (precision + recall)  
f2 <- (5 \* precision \* recall) / ((4 \* precision) + recall)  
  
cat("\n--- Additional Metrics ---\n")

##   
## --- Additional Metrics ---

cat("Precision:", round(precision, 4), "\n")

## Precision: 0.8099

cat("Recall :", round(recall, 4), "\n")

## Recall : 0.6806

cat("F1 Score :", round(f1, 4), "\n")

## F1 Score : 0.7396

cat("F2 Score :", round(f2, 4), "\n")

## F2 Score : 0.703

***RESULTS***

***Precision (~81%) shows that about 81% of all model predictions made for the positive class were correct. This is pretty accurate, particularly in the medical field.***

***Recall (68.06%): percentage of actual positive cases in the model identified correctly. This measures the sensitivity of the model.***

***F1 Score (~74%) is the harmonic mean of precision and recall. This score suggests moderate performance.***

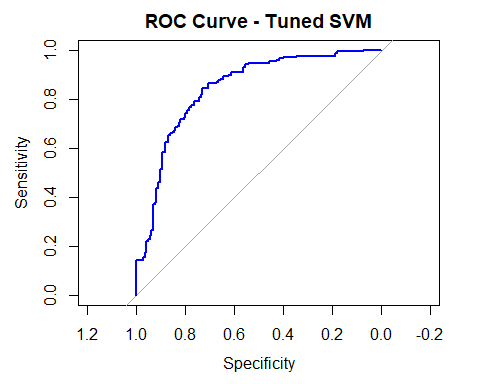
***F2 Score (70%) means that the model performance is reasonable, but still misses some true cases***

**# ROC Curve & AUC**  
if (!require(pROC)) install.packages("pROC")  
library(pROC)  
roc\_obj <- roc(test\_data$Diagnosis, probs)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

plot(roc\_obj, main = "ROC Curve - Tuned SVM", col = "blue", lwd = 2)



cat("AUC:", round(auc(roc\_obj), 4), "\n")

**## AUC: 0.8419**

library(caret)  
library(randomForest)

## randomForest 4.7-1.2

## 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

###############################################

# **MODEL BUILDING AND EVALUATION: RANDOM FOREST**

###############################################

# Load required libraries  
library(randomForest)  
library(caret)  
library(e1071) # For confusion matrix  
library(pROC) # For ROC curves  
  
# Ensure reproducibility  
set.seed(123)  
  
# Prepare data (assuming you've already done the preprocessing)  
# Make sure Diagnosis is a factor for classification  
data$Diagnosis <- as.factor(data$Diagnosis)  
  
# Create train-test split (80-20 split)  
train\_index <- createDataPartition(data$Diagnosis, p = 0.8, list = FALSE)  
train\_data <- data[train\_index, ]  
test\_data <- data[-train\_index, ]  
  
# Check class distribution in both sets  
print("Training set class distribution:")

## [1] "Training set class distribution:"

print(table(train\_data$Diagnosis))

##   
## 0 1   
## 400 215

print("Test set class distribution:")

## [1] "Test set class distribution:"

print(table(test\_data$Diagnosis))

##   
## 0 1   
## 100 53

# Train Random Forest model  
# Basic model  
rf\_model <- randomForest(  
 Diagnosis ~ .,  
 data = train\_data,  
 ntree = 500, # Number of trees  
 mtry = sqrt(ncol(train\_data) - 1), # Number of variables at each split  
 importance = TRUE, # Calculate variable importance  
 proximity = TRUE # Calculate proximity matrix  
)  
  
# Print model summary  
print(rf\_model)

##   
## Call:  
## randomForest(formula = Diagnosis ~ ., data = train\_data, ntree = 500, mtry = sqrt(ncol(train\_data) - 1), importance = TRUE, proximity = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 24.07%  
## Confusion matrix:  
## 0 1 class.error  
## 0 341 59 0.1475000  
## 1 89 126 0.4139535

# Make predictions on test set  
test\_predictions <- predict(rf\_model, test\_data)  
test\_probabilities <- predict(rf\_model, test\_data, type = "prob")  
  
# Evaluate model performance  
confusion\_matrix <- confusionMatrix(test\_predictions, test\_data$Diagnosis)  
print(confusion\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 90 16  
## 1 10 37  
##   
## Accuracy : 0.8301   
## 95% CI : (0.761, 0.8859)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 9.866e-07   
##   
## Kappa : 0.6145   
##   
## Mcnemar's Test P-Value : 0.3268   
##   
## Sensitivity : 0.9000   
## Specificity : 0.6981   
## Pos Pred Value : 0.8491   
## Neg Pred Value : 0.7872   
## Prevalence : 0.6536   
## Detection Rate : 0.5882   
## Detection Prevalence : 0.6928   
## Balanced Accuracy : 0.7991   
##   
## 'Positive' Class : 0   
##

# Calculate additional metrics  
accuracy <- confusion\_matrix$overall['Accuracy']  
sensitivity <- confusion\_matrix$byClass['Sensitivity']  
specificity <- confusion\_matrix$byClass['Specificity']  
  
print(paste("Accuracy:", round(accuracy, 3)))

## [1] "Accuracy: 0.83"

print(paste("Sensitivity (Recall):", round(sensitivity, 3)))

## [1] "Sensitivity (Recall): 0.9"

print(paste("Specificity:", round(specificity, 3)))

## [1] "Specificity: 0.698"

# ROC Curve and AUC  
roc\_curve <- roc(test\_data$Diagnosis, test\_probabilities[, 2])

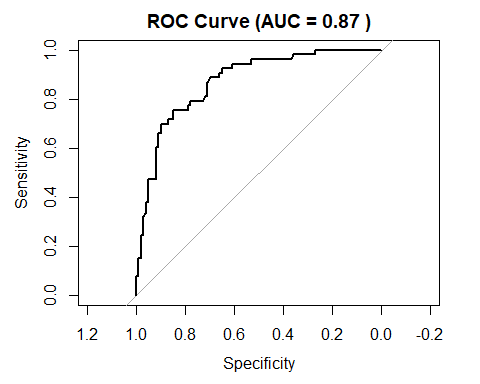
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

auc\_value <- auc(roc\_curve)  
print(paste("AUC:", round(auc\_value, 3)))

## [1] "AUC: 0.87"

# Plot ROC curve  
plot(roc\_curve, main = paste("ROC Curve (AUC =", round(auc\_value, 3), ")"))



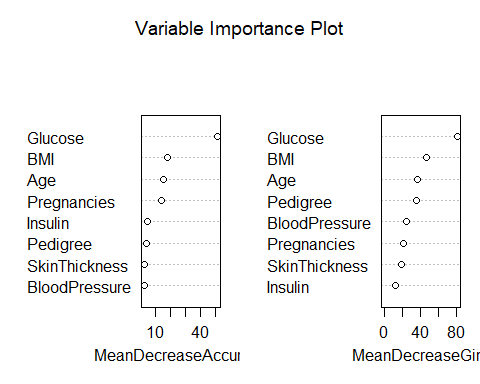
# Variable Importance  
importance\_scores <- importance(rf\_model)  
print("Variable Importance:")

## [1] "Variable Importance:"

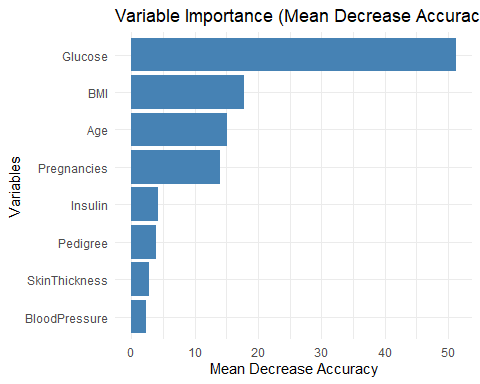
print(importance\_scores)

## 0 1 MeanDecreaseAccuracy MeanDecreaseGini  
## Pregnancies 13.486334 2.5382848 14.011730 21.29720  
## Glucose 36.445772 37.6716727 51.227687 81.53417  
## BloodPressure 3.614785 -0.5975323 2.374866 24.77932  
## SkinThickness 2.390173 1.4721779 2.806923 19.44625  
## Insulin 2.226522 3.7170994 4.281303 12.06789  
## BMI 7.173496 17.7605216 17.765903 47.28254  
## Pedigree 4.986104 0.2477997 3.854836 36.19771  
## Age 12.635518 5.8402756 15.112214 36.62751

# Plot variable importance  
varImpPlot(rf\_model, main = "Variable Importance Plot")



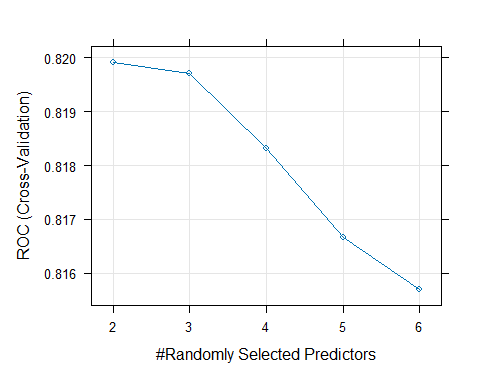
# Alternative importance plot using ggplot2  
importance\_df <- data.frame(  
 Variable = rownames(importance\_scores),  
 MeanDecreaseAccuracy = importance\_scores[, "MeanDecreaseAccuracy"],  
 MeanDecreaseGini = importance\_scores[, "MeanDecreaseGini"]  
)  
  
library(ggplot2)  
ggplot(importance\_df, aes(x = reorder(Variable, MeanDecreaseAccuracy), y = MeanDecreaseAccuracy)) +  
 geom\_col(fill = "steelblue") +  
 coord\_flip() +  
 labs(title = "Variable Importance (Mean Decrease Accuracy)",  
 x = "Variables", y = "Mean Decrease Accuracy") +  
 theme\_minimal()



# Model tuning with cross-validation  
# Define parameter grid for tuning  
tune\_grid <- expand.grid(  
 mtry = c(2, 3, 4, 5, 6) # Different values for mtry  
)  
  
# Set up cross-validation  
ctrl <- trainControl(  
 method = "cv",  
 number = 5, # 5-fold cross-validation  
 classProbs = TRUE,  
 summaryFunction = twoClassSummary,  
 savePredictions = TRUE  
)  
  
# Convert factor levels to valid names for caret  
levels(train\_data$Diagnosis) <- c("No", "Yes")  
levels(test\_data$Diagnosis) <- c("No", "Yes")  
  
# Train model with cross-validation  
rf\_tuned <- train(  
 Diagnosis ~ .,  
 data = train\_data,  
 method = "rf",  
 trControl = ctrl,  
 tuneGrid = tune\_grid,  
 metric = "ROC",  
 ntree = 500  
)  
  
# Print tuning results  
print(rf\_tuned)

## Random Forest   
##   
## 615 samples  
## 8 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 492, 492, 492, 492, 492   
## Resampling results across tuning parameters:  
##   
## mtry ROC Sens Spec   
## 2 0.8199128 0.8475 0.5488372  
## 3 0.8197093 0.8425 0.5581395  
## 4 0.8183140 0.8425 0.5534884  
## 5 0.8166570 0.8400 0.5767442  
## 6 0.8156977 0.8325 0.5767442  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

plot(rf\_tuned)



# Make predictions with tuned model  
tuned\_predictions <- predict(rf\_tuned, test\_data)  
tuned\_probabilities <- predict(rf\_tuned, test\_data, type = "prob")  
  
# Evaluate tuned model  
tuned\_confusion <- confusionMatrix(tuned\_predictions, test\_data$Diagnosis)  
print("Tuned Model Performance:")

## [1] "Tuned Model Performance:"

print(tuned\_confusion)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 88 17  
## Yes 12 36  
##   
## Accuracy : 0.8105   
## 95% CI : (0.7393, 0.8692)  
## No Information Rate : 0.6536   
## P-Value [Acc > NIR] : 1.46e-05   
##   
## Kappa : 0.5719   
##   
## Mcnemar's Test P-Value : 0.4576   
##   
## Sensitivity : 0.8800   
## Specificity : 0.6792   
## Pos Pred Value : 0.8381   
## Neg Pred Value : 0.7500   
## Prevalence : 0.6536   
## Detection Rate : 0.5752   
## Detection Prevalence : 0.6863   
## Balanced Accuracy : 0.7796   
##   
## 'Positive' Class : No   
##

# Compare models  
cat("\n=== Model Comparison ===\n")

##   
## === Model Comparison ===

cat("Basic RF Accuracy:", round(accuracy, 3), "\n")

## Basic RF Accuracy: 0.83

cat("Tuned RF Accuracy:", round(tuned\_confusion$overall['Accuracy'], 3), "\n")

## Tuned RF Accuracy: 0.81

# Feature selection based on importance (optional)  
# Select top features  
top\_features <- head(importance\_df[order(-importance\_df$MeanDecreaseAccuracy), ], 5)  
print("Top 5 most important features:")

## [1] "Top 5 most important features:"

print(top\_features$Variable)

## [1] "Glucose" "BMI" "Age" "Pregnancies" "Insulin"

# Train a model with only top features  
formula\_top <- as.formula(paste("Diagnosis ~", paste(top\_features$Variable, collapse = " + ")))  
rf\_top\_features <- randomForest(  
 formula\_top,  
 data = train\_data,  
 ntree = 500,  
 importance = TRUE  
)  
  
# Evaluate reduced model  
top\_predictions <- predict(rf\_top\_features, test\_data)  
top\_confusion <- confusionMatrix(top\_predictions, test\_data$Diagnosis)  
cat("Top Features Model Accuracy:", round(top\_confusion$overall['Accuracy'], 3), "\n")

## Top Features Model Accuracy: 0.797

############################################################################

# **MODEL BUILDING AND EVALUATION: XGBOOST**

#############################################################################

library(pROC)  
library(caret)  
library(xgboost)

##   
## Attaching package: 'xgboost'

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

**# Feature selection based on pearson correlation of threshold 0.1**  
cor\_vals <- cor\_matrix[, "Diagnosis"]  
**# View all correlations with Diagnosis**  
print(cor\_vals)

## Pregnancies Glucose BloodPressure SkinThickness Insulin   
## 0.2218982 0.4927824 0.1689712 0.2201109 0.2663816   
## BMI Pedigree Age Diagnosis   
## 0.3128112 0.1738441 0.2383560 1.0000000

selected\_features <- names(cor\_vals[abs(cor\_vals) > 0.1 & names(cor\_vals) != "Diagnosis"])  
print(selected\_features)

## [1] "Pregnancies" "Glucose" "BloodPressure" "SkinThickness"  
## [5] "Insulin" "BMI" "Pedigree" "Age"

***Pearson correlation was calculated between each feature and the target variable (Diagnosis). Then features with a correlation coefficient greater than 0.1 were selected. In this way, we filter out features that are weakly related and keep those that show a stronger linear association with the target.***

***We see clearly that 8 features were selected, excluding only the ones with correlation <= 0.1. This step was needed for preliminary filtering before the actual modeling, as it reduces noise while keeping contributing variables.***

**# Subset data with selected features + target**  
data\_selected <- data[, c(selected\_features, "Diagnosis")]  
  
# **Standardization - only on selected features (excluding target)**

**Z-score normalization**

features <- data\_selected[, -which(names(data\_selected) == "Diagnosis")]  
features\_scaled <- scale(features)

***z-score normalization was applied for feature selection. Helpful for models, as it ensures an equal contribution of all features to the model.***

**# Combine scaled features with Diagnosis**  
data\_scaled <- as.data.frame(cbind(features\_scaled, Diagnosis = data\_selected$Diagnosis))

***Target variable was added back to scaled data so it can be used for modeling***

**# Separate predictors and target**  
X <- as.matrix(data\_scaled[, -which(names(data\_scaled) == "Diagnosis")])  
y <- as.integer(data\_scaled$Diagnosis) - 1  
  
  
set.seed(123)  
train\_idx <- createDataPartition(y, p = 0.7, list = FALSE)  
train\_X <- X[train\_idx, ]  
train\_y <- y[train\_idx]  
test\_X <- X[-train\_idx, ]  
test\_y <- y[-train\_idx]

**# Define Parameters**

params <- list(  
 booster = "gbtree",  
 objective = "binary:logistic",  
 eval\_metric = "auc",  
 eta = 0.1,  
 max\_depth = 4,  
 subsample = 0.8,  
 colsample\_bytree = 0.8  
)

***Booster used is the gradient-boosted decision trees, with the goal of solving a binary classification problem. 80% of the data used in each boosting round (good for randomness). Also, overfitting is reduced by using colsample\_bytree of 80% (percentage of features used in each tree)***

**# Cross-Validation to Find Best nrounds**  
  
set.seed(123)  
cv\_results <- xgb.cv(  
 params = params,  
 data = train\_X,  
 label = train\_y,  
 nrounds = 100,  
 nfold = 10,  
 showsd = TRUE,  
 stratified = TRUE,  
 print\_every\_n = 10,  
 early\_stopping\_rounds = 10,  
 maximize = TRUE  
)

## [1] train-auc:0.838329+0.027840 test-auc:0.726991+0.068488   
## Multiple eval metrics are present. Will use test\_auc for early stopping.  
## Will train until test\_auc hasn't improved in 10 rounds.  
##   
## [11] train-auc:0.927224+0.005962 test-auc:0.825434+0.053667   
## [21] train-auc:0.940032+0.003996 test-auc:0.831974+0.052686   
## [31] train-auc:0.950921+0.004603 test-auc:0.830904+0.044413   
## Stopping. Best iteration:  
## [21] train-auc:0.940032+0.003996 test-auc:0.831974+0.052686

best\_nrounds <- cv\_results$best\_iteration  
cat("Best nrounds:", best\_nrounds, "\n")

## Best nrounds: 21

**# Final Model Training**  
  
final\_model <- xgboost(  
 data = train\_X,  
 label = train\_y,  
 objective = params$objective,  
 eval\_metric = params$eval\_metric,  
 nrounds = best\_nrounds,  
 eta = params$eta,  
 max\_depth = params$max\_depth,  
 subsample = params$subsample,  
 colsample\_bytree = params$colsample\_bytree,  
 verbose = 0  
)  
  
  
**# Predict on Test Set**  
  
**# Lowered threshold to 0.4 to improve sensitivity**  
pred\_probs <- predict(final\_model, test\_X)  
preds <- ifelse(pred\_probs >= 0.4, 1, 0)  
  
  
**# Evaluation**  
  
preds\_factor <- factor(preds, levels = c(0, 1))  
test\_y\_factor <- factor(test\_y, levels = c(0, 1))  
  
conf\_matrix <- confusionMatrix(preds\_factor, test\_y\_factor, positive = "1")  
print(conf\_matrix)

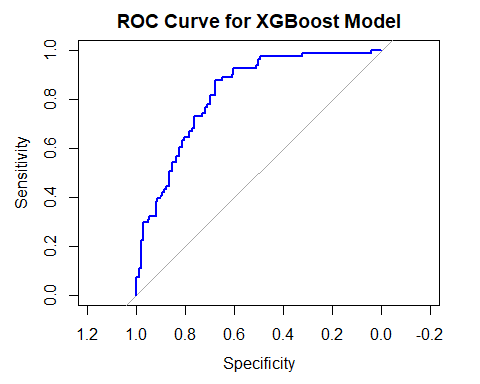
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 113 22  
## 1 36 59  
##   
## Accuracy : 0.7478   
## 95% CI : (0.6865, 0.8026)  
## No Information Rate : 0.6478   
## P-Value [Acc > NIR] : 0.0007366   
##   
## Kappa : 0.4683   
##   
## Mcnemar's Test P-Value : 0.0878251   
##   
## Sensitivity : 0.7284   
## Specificity : 0.7584   
## Pos Pred Value : 0.6211   
## Neg Pred Value : 0.8370   
## Prevalence : 0.3522   
## Detection Rate : 0.2565   
## Detection Prevalence : 0.4130   
## Balanced Accuracy : 0.7434   
##   
## 'Positive' Class : 1   
##

**# Generate ROC curve**  
roc\_obj <- roc(test\_y, pred\_probs)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

**# Plot ROC curve**  
plot(roc\_obj, main = "ROC Curve for XGBoost Model", col = "blue", lwd = 2)



**# Print AUC value**  
auc\_value <- auc(roc\_obj)  
cat("AUC:", auc\_value, "\n")

## AUC: 0.8230591

############################################################################

MODEL ANALYSIS AND LIMITATIONS

############################################################################

#### **Strengths**

Model diversity: We evaluated several models, covering a broad range from interpretable to complex models

Use of SMOTE: this balancing technique helped us address class imbalance, which is a common issue GDM prediction

Performance wise: all models achieved a decent AUC score (~84 – 90%). This clearly shows a strong discriminative power.

Feature relevance: the features we selected ( Glucose,BMI,Age,etc) exactly align with most of the ones mentioned in top studies. This clearly shows its biological potential.

#### **LIMITATIONS**

*Dataset scope: our dataset, while being clean, is limited in clinical depth as it lacks variables such as HbA1c, ethnicity, dietary behavior, or family medical history. Those were commonly used in robust models that we mentioned in our literature review.*

*Cross-sectional: our dataset is restricted to a single time point which makes the model’s ability to detect temporal changes harder.*

*Interpretability vs Accuracy Trade-off: it is true that advanced models like XGBoost perform better but are unfortunately harder to interpret than Logistic Regression. Interpretation is critical in clinical settings.*

*Generalizability: data sources like the PIMA dataset lack population diversity. This factor restricts the model’s applicability across diverse populations with different lifestyle styles globally.*

############################################################################

RECOMMENDATIONS

############################################################################

*Features improvement: we can include richer clinical and behavioral data such as: HbA1c, Dietary Intake, Physical activity, ethnicity, etc. Those features have shown relevance in recent studies.*

*Collecting Temporal data: collection of time-series data across pregnancy phases in order to capture early risks patterns and dynamic independent variables. This will allow timely intervention and prediction before the OGTT window.*

*Model Interpretation: tools such as SHAP ( SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) can be used to make complex models such as XGBoost more transparent for clinical users.*

*Advanced Preprocessing: we can improve data imputation with multiple imputation or predictive mean matching. We can also explore robust scaling or quantile transformation to improve our model’s learning on skewed variables.*

*External Validation: the use of independent datasets from a completely different population can help us test real-world generalizability. This testing practice has shown that model's performance can drop significantly due to the skipping of this step.*