Comparative Analysis of Classification Techniques for Predicting Diabetes Onset in Pima Indian Women

Zeedan Shaikh

November 7, 2024

Introduction

Diabetes is a chronic health condition with significant long-term implications for individuals and healthcare systems worldwide. Early detection is crucial for effective management and prevention of complications. This project focuses on predicting the onset of diabetes in female Pima Indian individuals, using a dataset of medical records with key health indicators. The dataset allows for the application of various statistical and machine learning methods to assess their predictive capabilities.

Through this study, we aim to explore the effectiveness of multiple classification techniques, each with unique approaches to handling data patterns and making predictions. By comparing the performance of these classifiers, we seek to identify the most reliable and accurate model for early diabetes prediction in this specific population, ultimately contributing to the understanding of model selection in health prediction tasks.

Loading the required packages

```
pacman::p_load(class,FNN,MASS,caret,mda,randomForest,adabag,ipred,rpart,ROCR,dplyr, ggplot2, GGally, corrplot,Am
```

Methodology

- Data Preparation and Splitting

Load the diabetes.csv dataset containing 768 observations on 8 medical features. Randomly divide the data into training and test sets, ensuring a 2:1 class ratio. This balanced split helps in model training by preserving the class distribution across both sets.

- Perform EDA
- Model Training

k-Nearest Neighbour (k-NN) Perform three variations of k-NN classification:

1. Choose the optimal k value using leave-one-out cross-validation to maximize model performance.
2. Train using a condensed training set to reduce the number of data points while retaining essential decision boundaries.
3. Train using an edited training set obtained through the MULTIEDIT algorithm to remove noise and improve classification accuracy.

- Discriminant Analysis Models

1. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA): Apply LDA and QDA models to the training data to explore the impact of linear and quadratic decision boundaries.

- Flexible and Penalized Discriminant Analysis

1. Flexible Discriminant Analysis (FDA) with Multivariate Adaptive Regression Splines (MARS): Apply FDA to capture non-linearity in data using the MARS algorithm.
2. Penalized Discriminant Analysis (PDA) with Generalized Ridge Regression: Use PDA to apply regularization, reducing model complexity and improving performance.

- Mixture Discriminant Analysis (MDA)

  Apply MDA with two subclasses per class to allow for flexibility within each class distribution.

- Logistic Regression

  Train a logistic regression model as a baseline for comparison against the other models.

- Ensemble Models

1. Bagging: Use Breiman's bagging algorithm with classification trees as single classifiers to reduce variance and improve robustness.
2. Boosting (AdaBoost.M1): Implement the AdaBoost.M1 algorithm by Freund and Schapire, using classification trees, to iteratively improve model accuracy.
3. Random Forests: Train a random forest model, an ensemble of decision trees, to improve predictive accuracy and feature importance interpretation.

- Performance Evaluation

  For each classifier, predict class labels on both the training and test sets. Record and compare the following metrics: Confusion Matrix: Display true positives, true negatives, false positives, and false negatives. Accuracy: Calculate the percentage of correctly classified observations. Precision, Recall, and F1-Score: Report these metrics to evaluate the balance between precision and recall. Summarize the results in a comparative table to provide insights into each classifier's performance on both training and test data.

- ROC Curve Analysis

  Plot the ROC curves for each classifier on a single graph to visualize and compare their ability to distinguish between classes. This analysis will use the ROCR package to derive the ROC curves, allowing for easy performance comparison.

- Feature Importance Analysis with Random Forests

  From the random forest model, generate a variable importance plot to identify which features contribute most to diabetes onset prediction. Comment on how these features align with clinical expectations and the importance of each in model performance.

Loading the dataset

```
data=read.csv("C:\\Users\\zeeda\\OneDrive\\Desktop\\pg sem 3\\AP\\Assigments\\Data\\diabetes - diabetes.csv")
```

EDA

```
# View the first few rows and summary statistics
head(data)
```

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

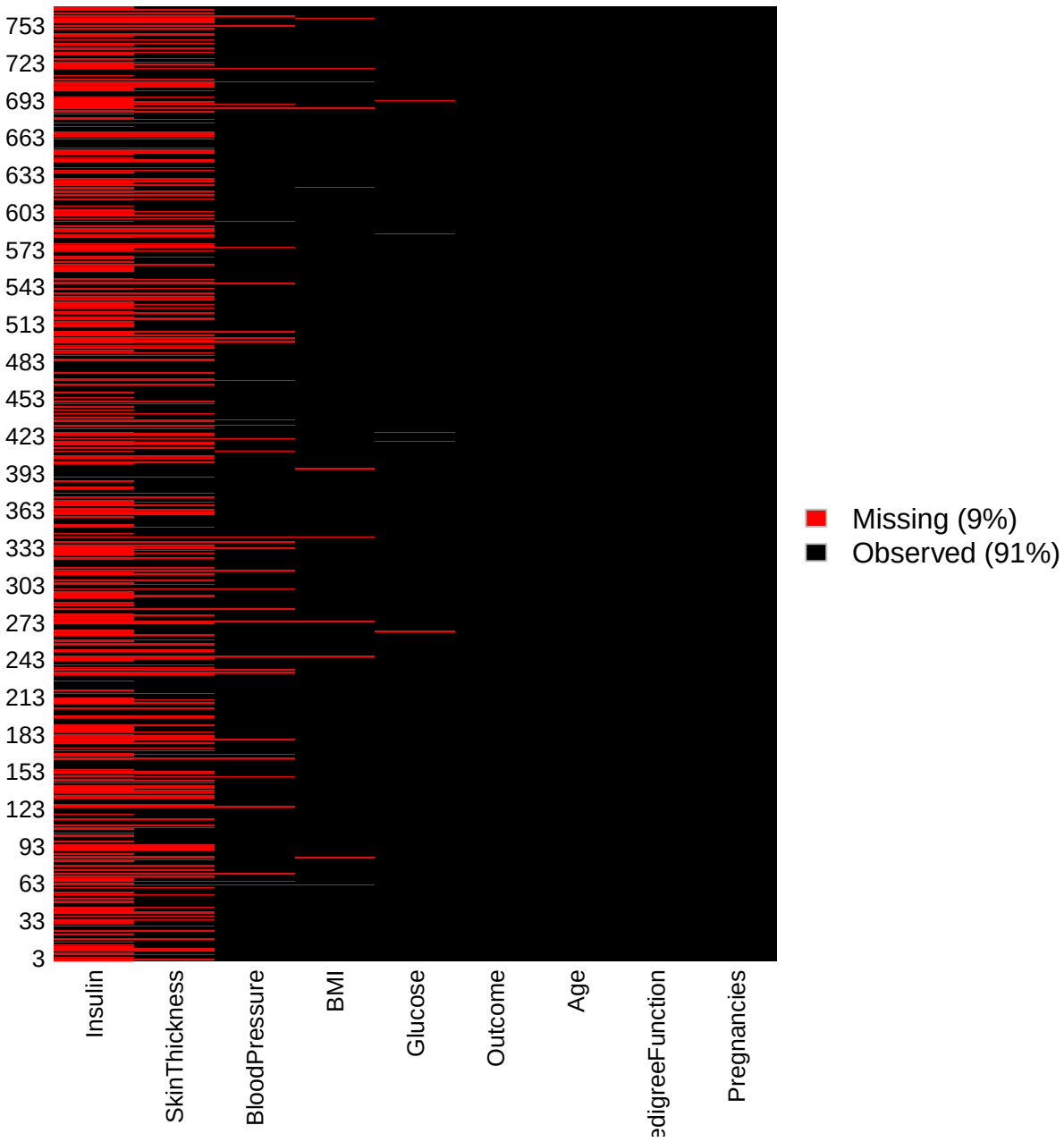
```
summary(data)
```

```
  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        DiabetesPedigreeFunction      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
    Outcome
 Min.   :0.000
 1st Qu.:0.000
 Median :0.000
 Mean   :0.349
 3rd Qu.:1.000
 Max.   :1.000
```

```r
# Check for missing values
missing_values <- colSums(is.na(data))
print(missing_values)
```

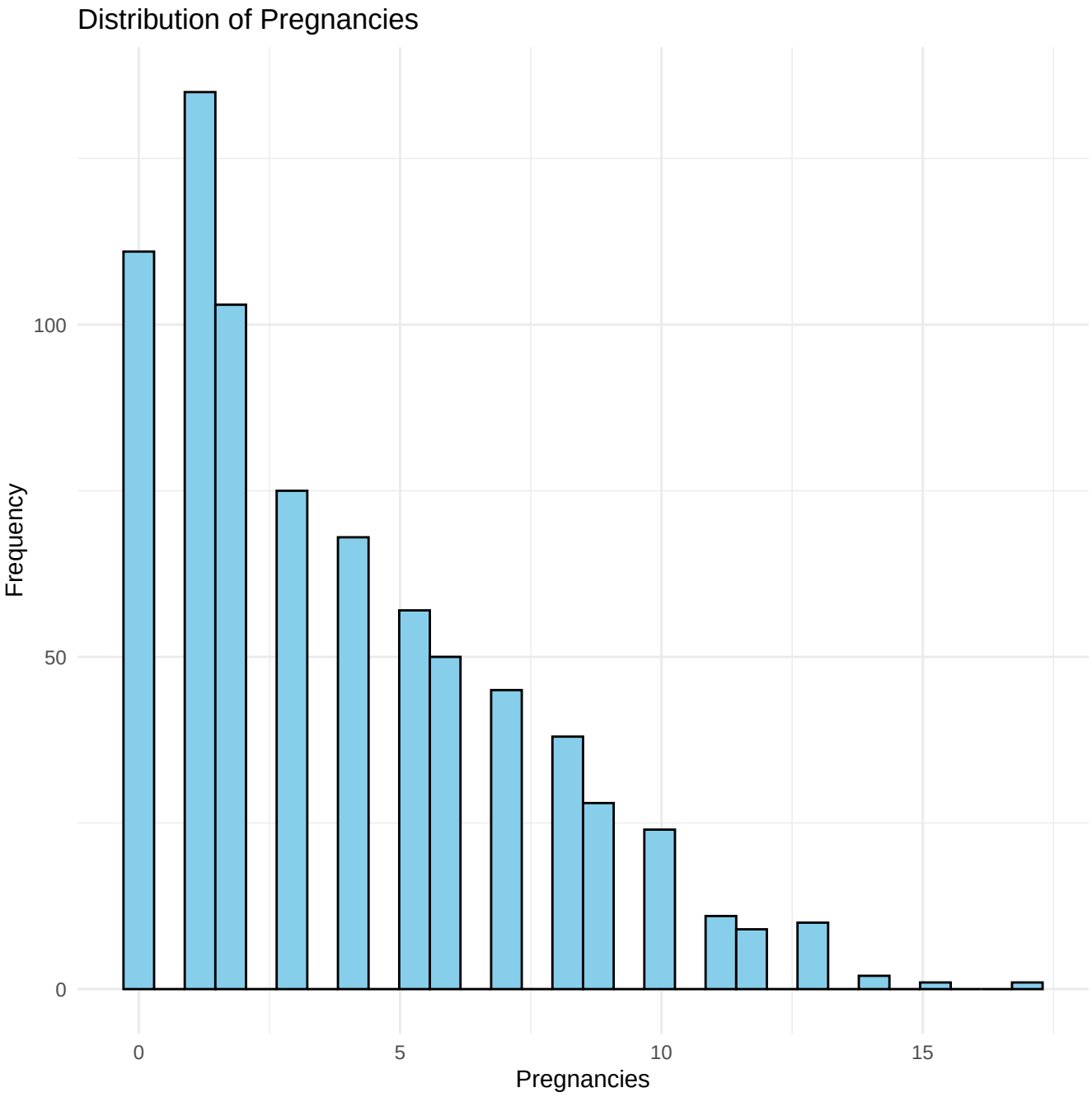
```
            Pregnancies                 Glucose           BloodPressure
                      0                       0                       0
          SkinThickness                 Insulin                     BMI
                      0                       0                       0
DiabetesPedigreeFunction                     Age                 Outcome
                      0                       0                       0
```

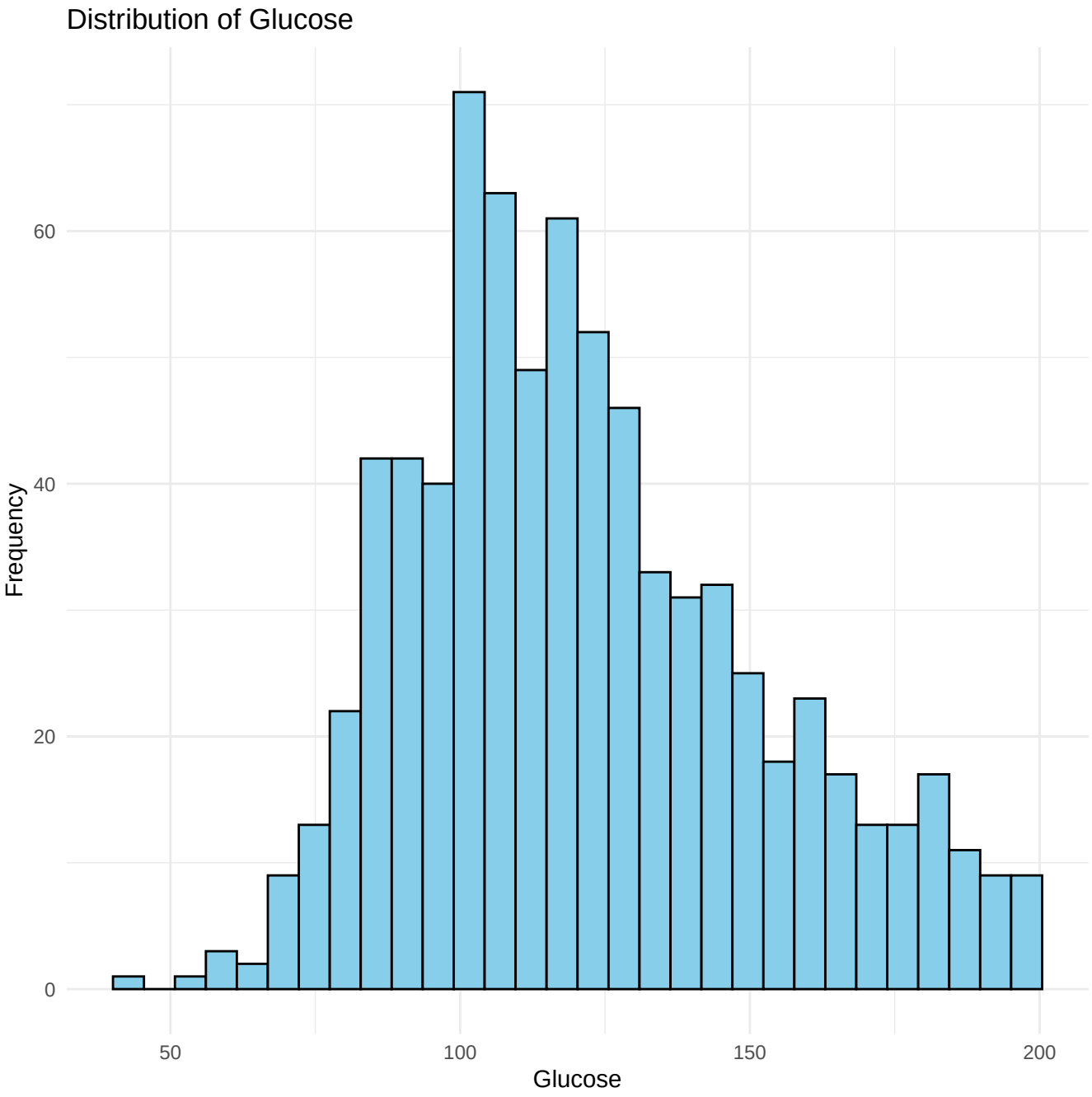
```r
# Replace 0 values in certain columns with NA (if 0 is considered invalid)
cols_with_zero <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")
data[cols_with_zero] <- lapply(data[cols_with_zero], function(x) ifelse(x == 0, NA, x))
# Visualize missing values pattern
missmap(data, main = "Missing values in the dataset", col = c("red", "black"))
```

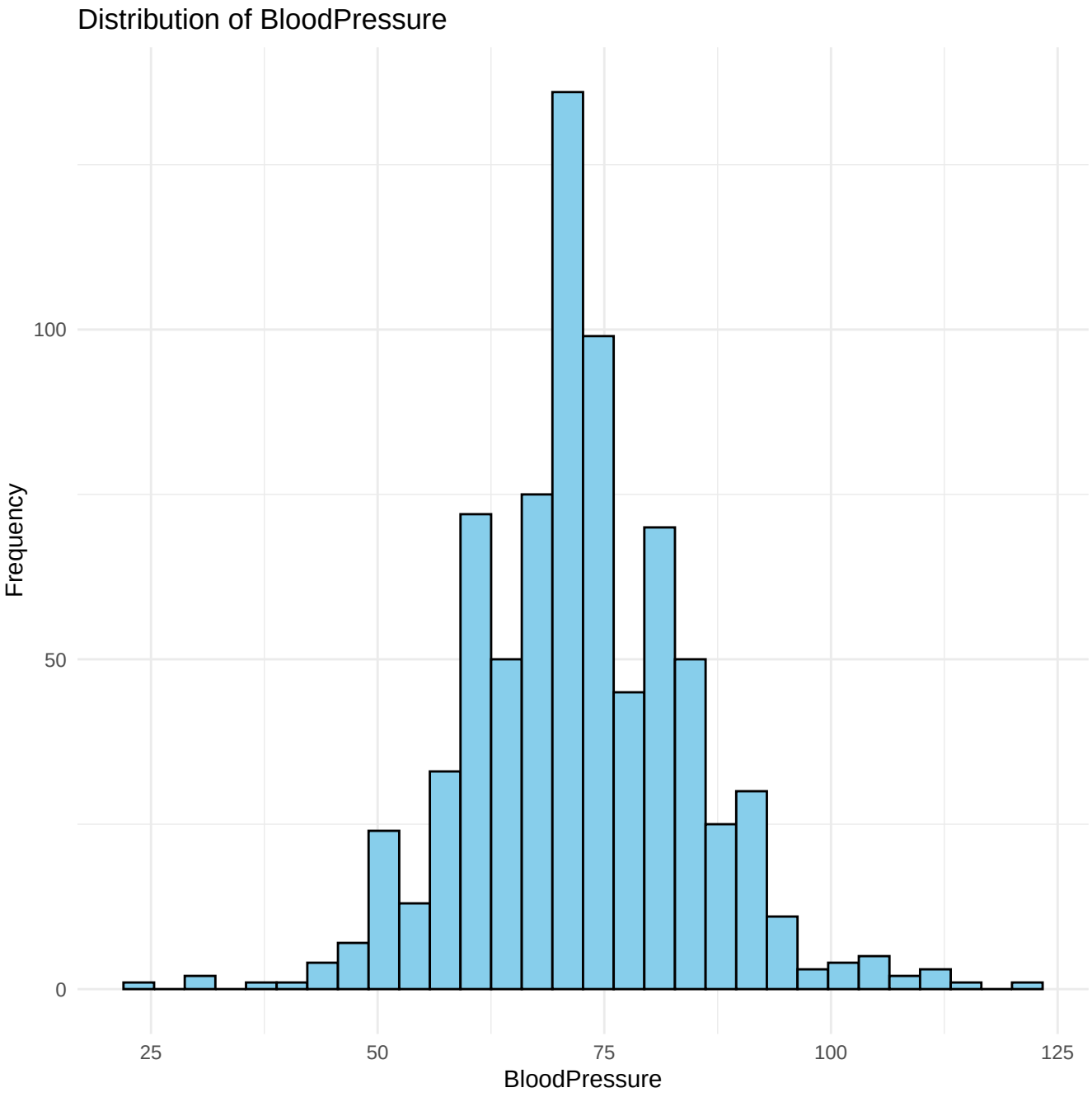


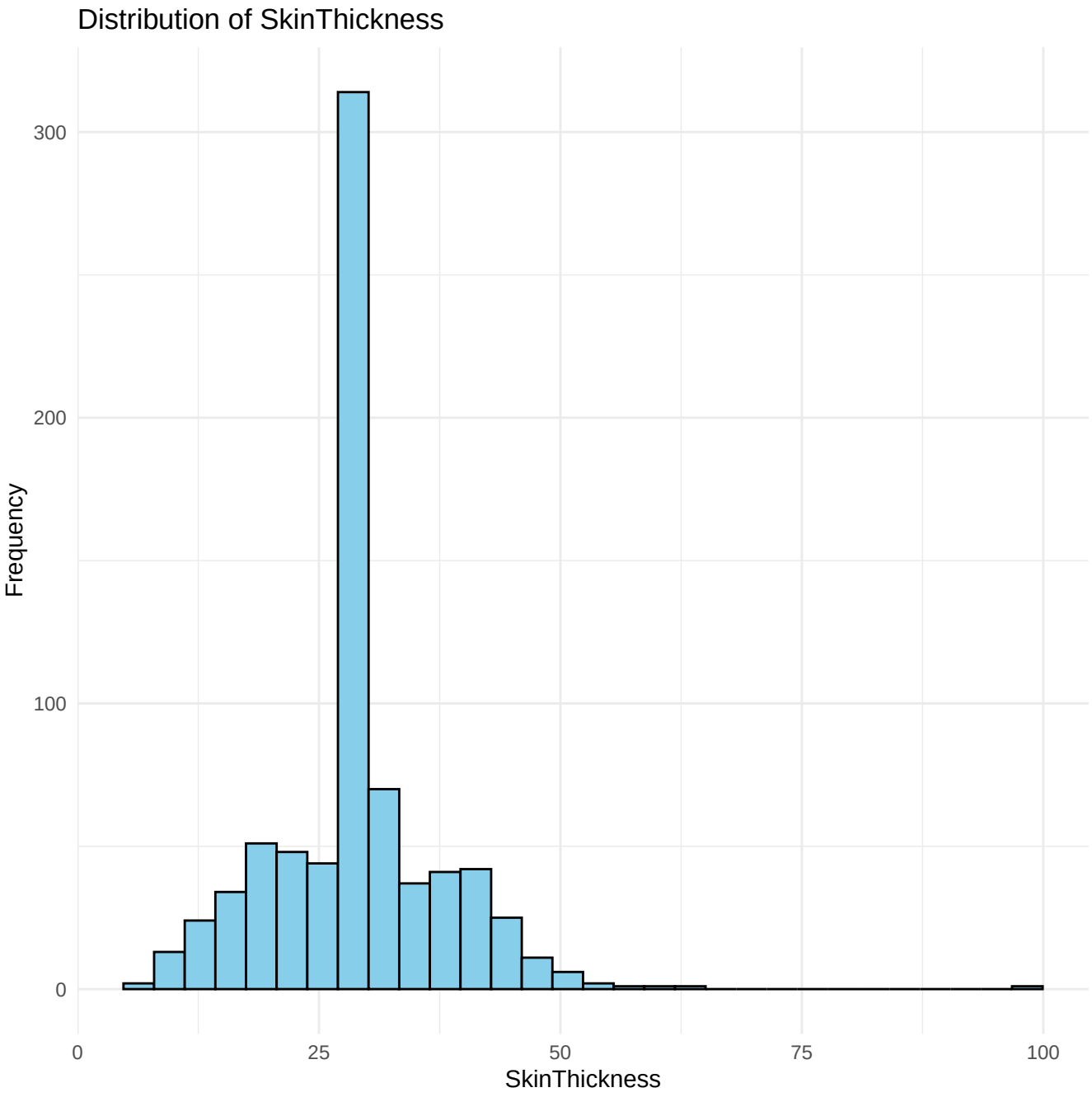

```
data[cols_with_zero] <- lapply(data[cols_with_zero], function(x) ifelse(is.na(x), median(x, na.rm = TRUE), x))
# Plot distributions for each variable
numeric_columns <- names(data)[sapply(data, is.numeric)]
for (col in numeric_columns)
{
print(     ggplot(data, aes_string(x = col)) +       geom_histogram(bins = 30, fill = "skyblue", color = "black"
```

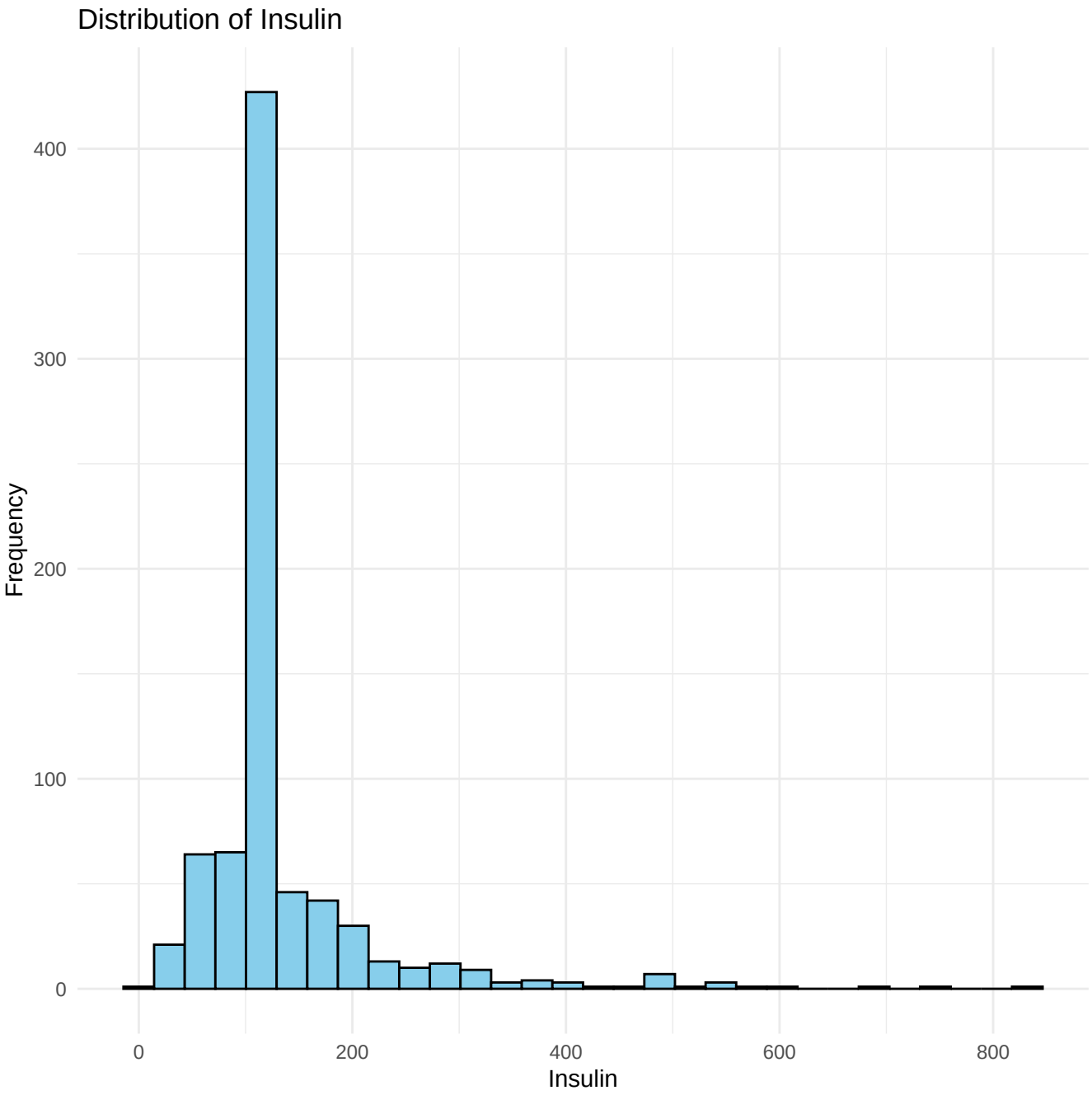
```
Warning:  ‘aes_string()‘ was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with ‘aes()‘.
i See also ‘vignette("ggplot2-in-packages")‘ for more information.
This warning is displayed once every 8 hours.
Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
generated.
```

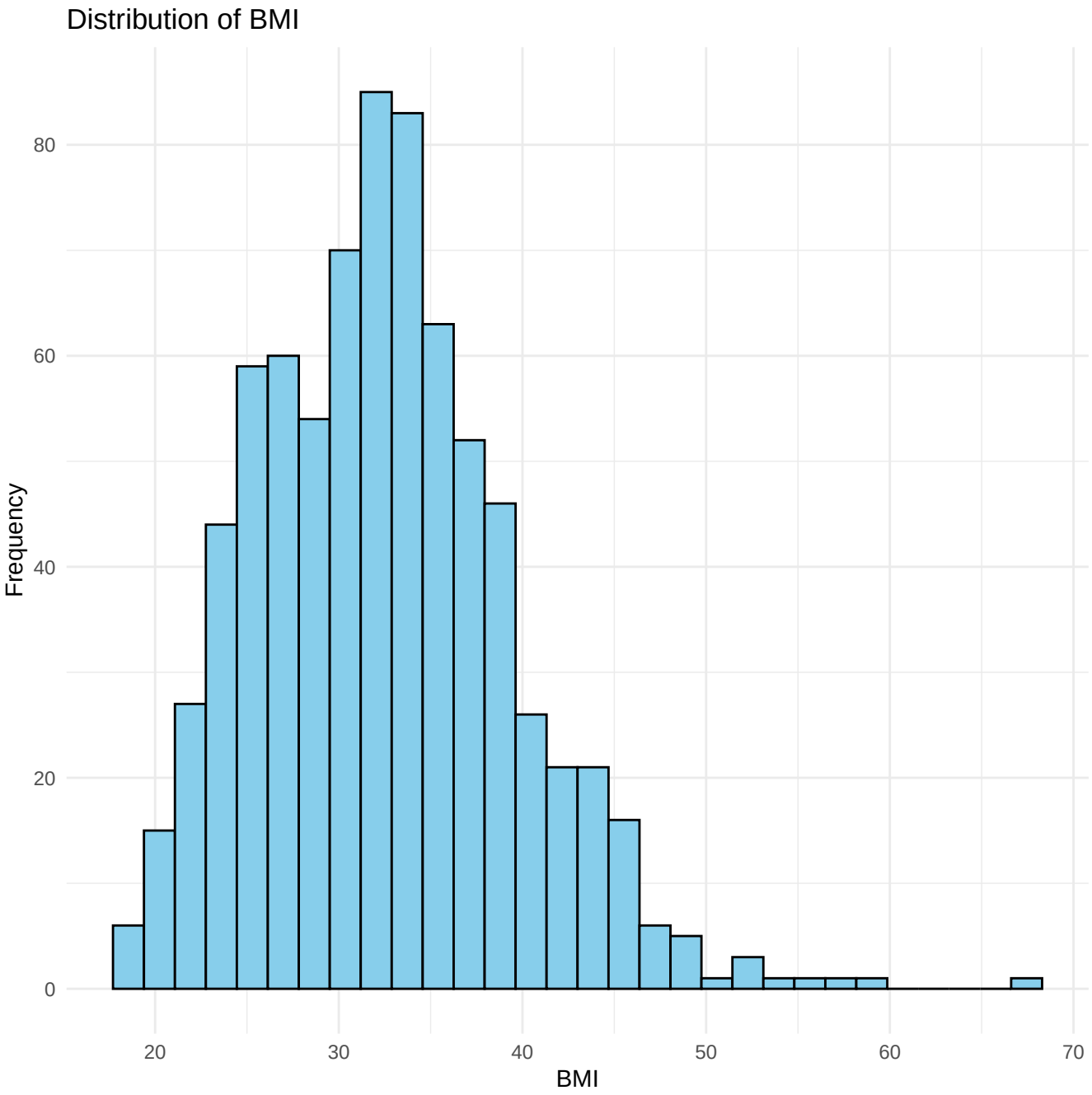

Distribution of Pregnancies
Frequency
100
50
0
0
5
10
15
Pregnancies

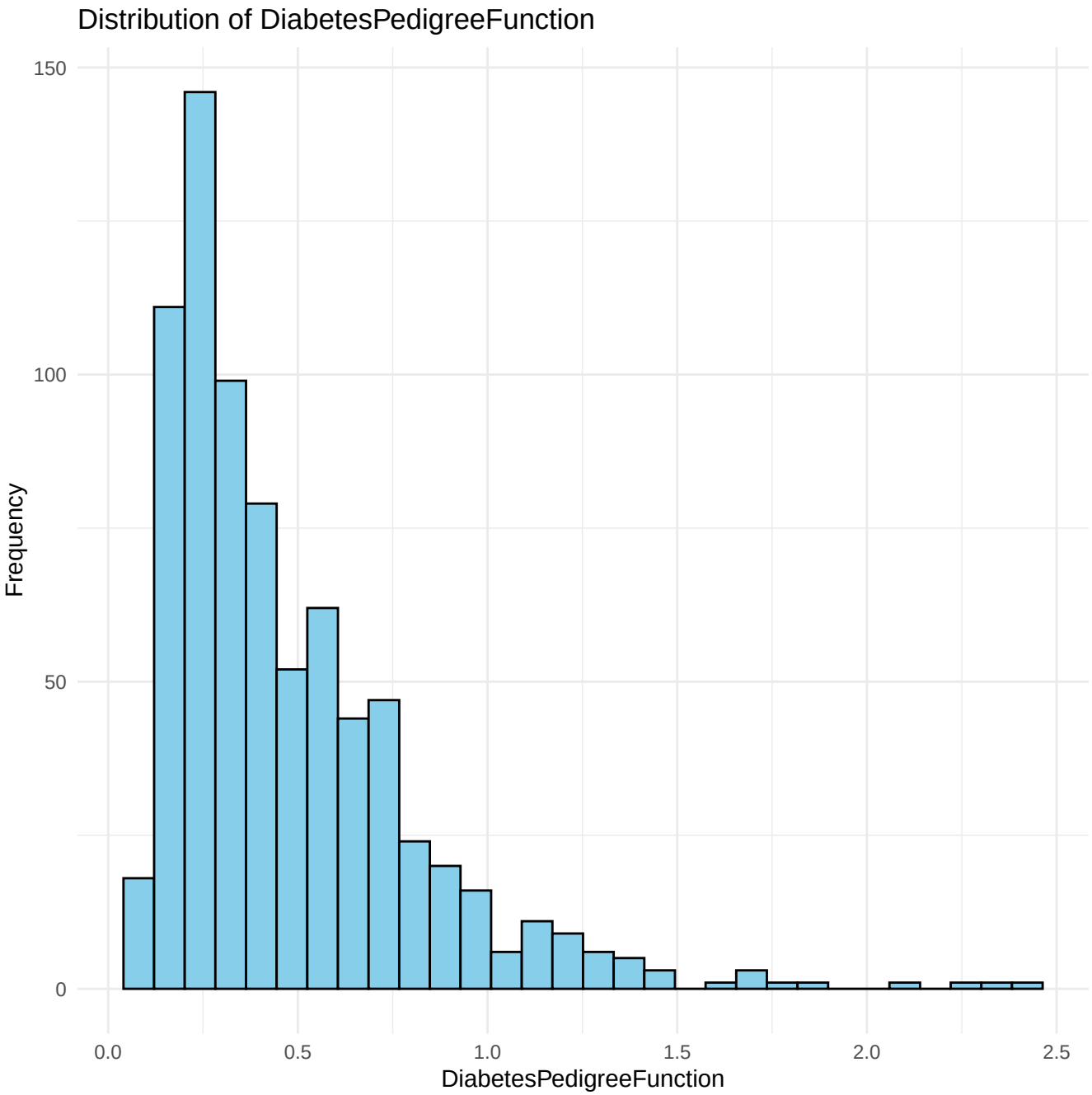

Distribution of Glucose
Frequency
60
40
20
0
50
100
150
200
Glucose

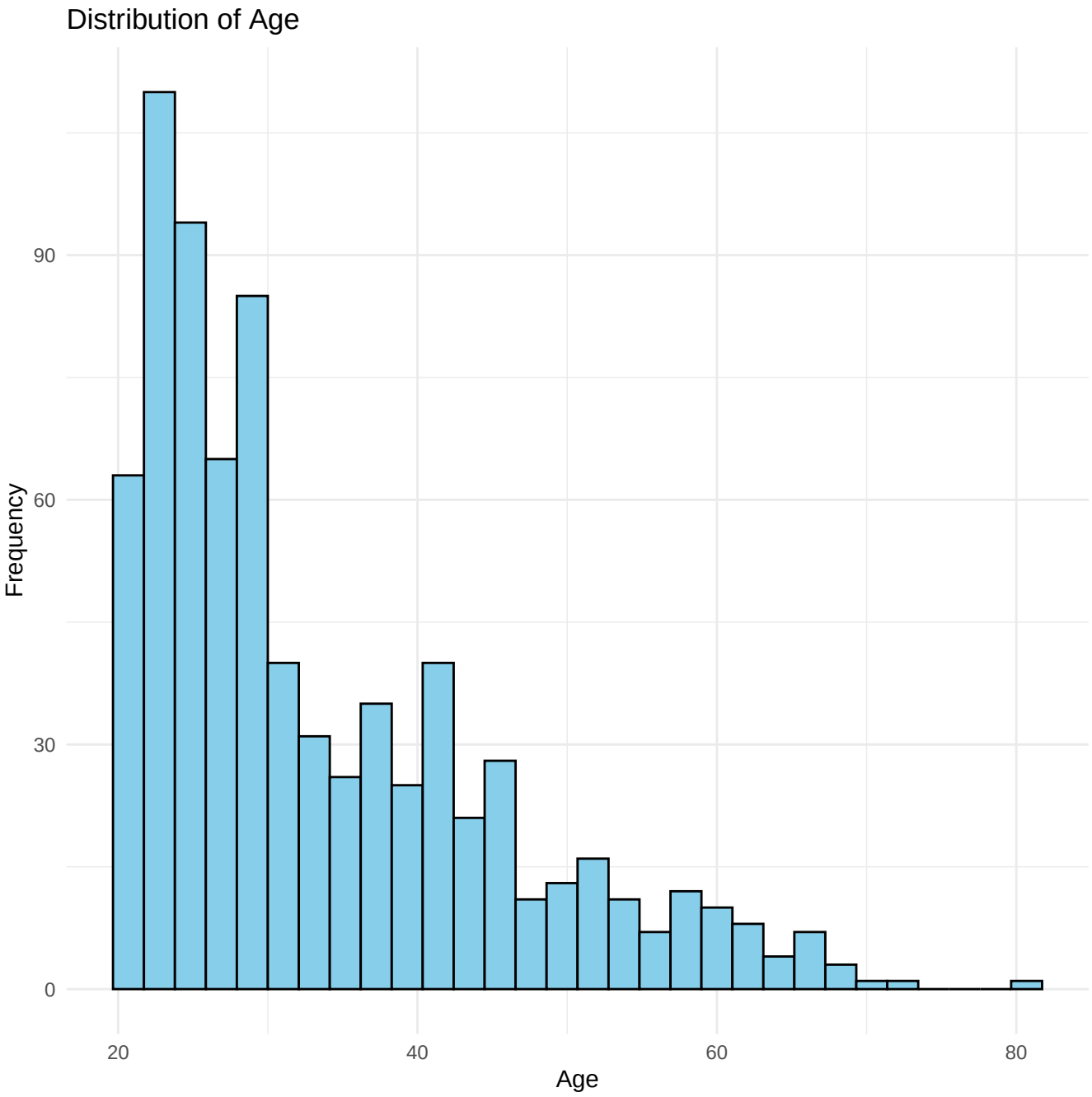

Distribution of BloodPressure
Frequency
100
50
0
25
50
75
100
125
BloodPressure

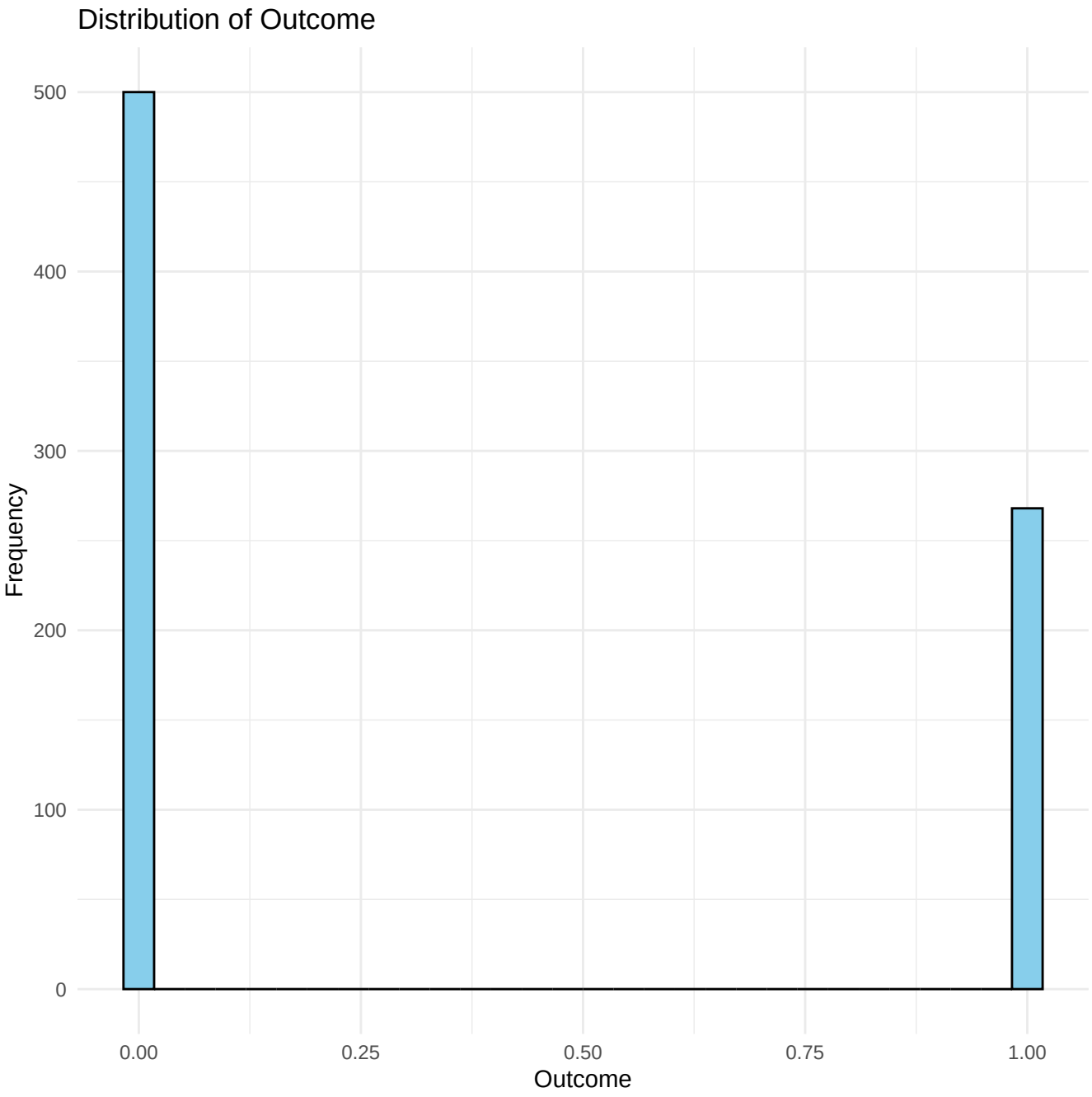

Distribution of SkinThickness
Frequency
300
200
100
0
0
25
50
75
100
SkinThickness


Distribution of Insulin
Frequency
400
300
200
100
0
0
200
400
600
800
Insulin


Distribution of BMI
Frequency
BMI
0
20
40
60
80
20
30
40
50
60
70


Distribution of DiabetesPedigreeFunction
150
100
50
0
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
DiabetesPedigreeFunction


Distribution of Age
Frequency
90
60
30
0
20
40
60
80
Age

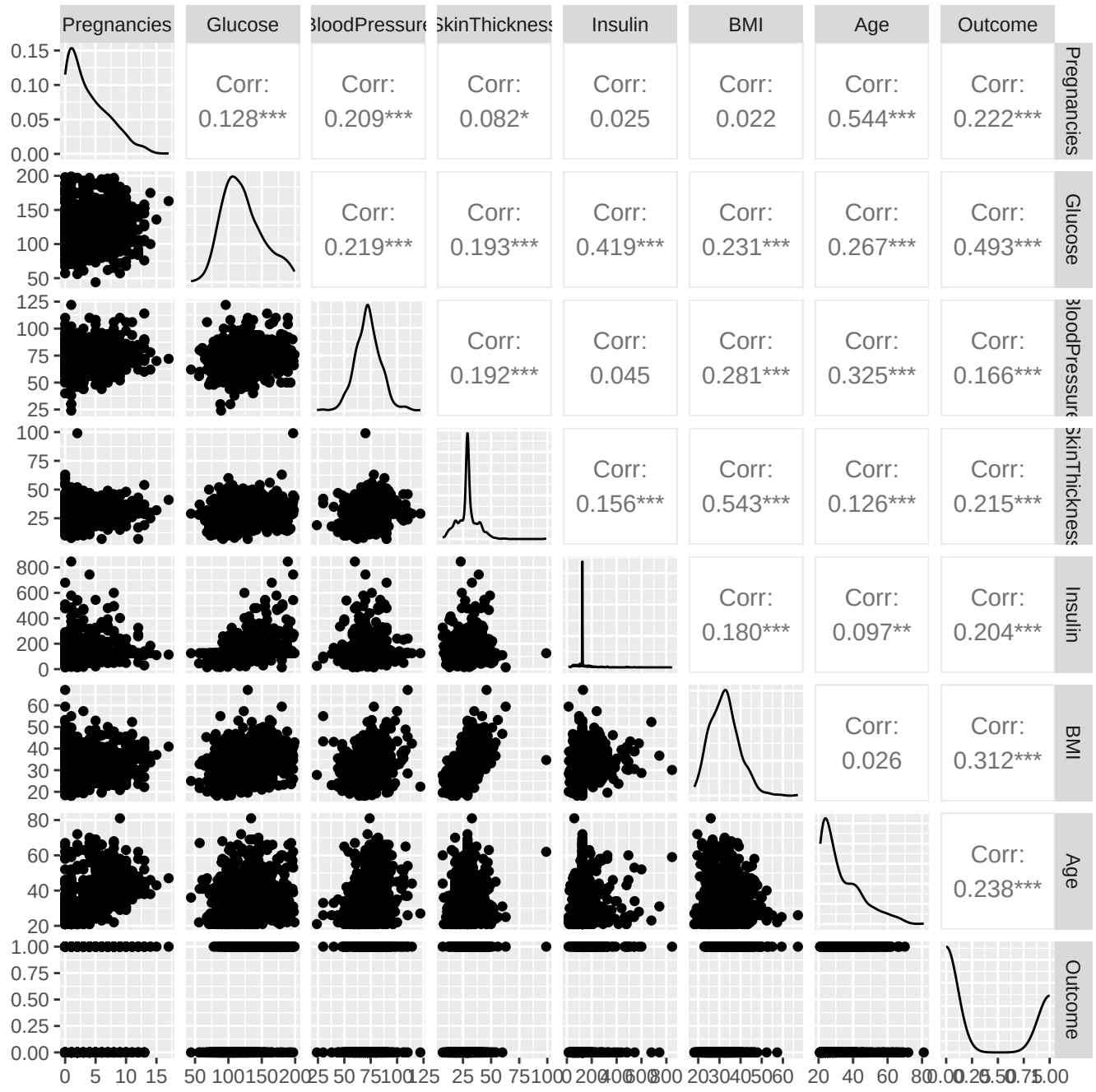



```r
# Check the correlation matrix
cor_matrix <- cor(data %>% select(-Outcome), use = "complete.obs")
print(cor_matrix)
```

```
                          Pregnancies   Glucose BloodPressure SkinThickness
Pregnancies                1.00000000 0.1282130   0.208615412    0.08176982
Glucose                    0.12821296 1.0000000   0.218937186    0.19261490
BloodPressure              0.20861541 0.2189372   1.000000000    0.19189239
SkinThickness              0.08176982 0.1926149   0.191892388    1.00000000
Insulin                    0.02504748 0.4194505   0.045363305    0.15561028
BMI                        0.02155873 0.2310486   0.281256564    0.54320507
DiabetesPedigreeFunction -0.03352267 0.1373269  -0.002378336    0.10218827
Age                        0.54434123 0.2669092   0.324915391    0.12610719
                             Insulin        BMI DiabetesPedigreeFunction
Pregnancies               0.02504748 0.02155873             -0.033522673
Glucose                   0.41945051 0.23104855              0.137326919
BloodPressure             0.04536330 0.28125656             -0.002378336
SkinThickness             0.15561028 0.54320507              0.102188267
```

```
Insulin                  1.00000000 0.18024114              0.126503086
BMI                      0.18024114 1.00000000              0.153437673
DiabetesPedigreeFunction 0.12650309 0.15343767              1.000000000
Age                      0.09710125 0.02559691              0.033561312
                                Age
Pregnancies              0.54434123
Glucose                  0.26690916
BloodPressure            0.32491539
SkinThickness            0.12610719
Insulin                  0.09710125
BMI                      0.02559691
DiabetesPedigreeFunction 0.03356131
Age                      1.00000000
```

```r
# Visualize the correlation matrix
cor_matrix <- cor(data[numeric_columns], use = "complete.obs")
corrplot(cor_matrix, method = "circle", type = "lower", diag = FALSE)
# Plot Outcome variable to see class distribution
ggplot(data, aes(x = factor(Outcome))) +   geom_bar(fill = "steelblue") +   labs(title = "Outcome Distribution",
ggpairs(data %>% select(Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, Age, Outcome))
```



Model Building:

```
data=read.csv("C:\\Users\\zeeda\\OneDrive\\Desktop\\pg sem 3\\AP\\Assigments\\Data\\diabetes - diabetes.csv")
#--partition
set.seed(12)
index=sample(nrow(data),floor((2/3)*nrow(data)),F)
train_data=data[index,]
test_data=data[-index,]
x_train=train_data[,-9]
y_train=train_data$Outcome
x_test=test_data[,-9]
y_test=test_data$Outcome
```

a) kk-nearest neighbour rule-

i. choosing kk optimally through leave-one-out cross-validation

```r
set.seed(12)
best_k <- 1
best_acc <- 0
for (k in 1:(nrow(x_train) - 1)) {
  pred <- knn.cv(x_train, y_train, k = k)
        acc <- mean(pred == y_train)
  if (acc > best_acc) {
best_acc <- acc
best_k <- k   } }
cat("Optimal value of k using LOOCV is:", best_k, "\n")
```

```
Optimal value of k using LOOCV is: 15
```

```r
cat("Best accuracy achieved with this k:", best_acc, "\n")
```

```
Best accuracy achieved with this k: 0.7714844
```

- So it is clear from here that the optimal value of k is 15

Now fitting a KNN model with k =15
for Training data

```r
set.seed(12)
knn_model_condense <- knn(x_train, x_train, y_train, k = 15,prob=TRUE)
conf_matrix <- confusionMatrix(as.factor(knn_model_condense), as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 322  78
         1  24  88

               Accuracy : 0.8008
                 95% CI : (0.7635, 0.8345)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 2.036e-10

                  Kappa : 0.5033

 Mcnemar's Test P-Value : 1.539e-07

            Sensitivity : 0.9306
            Specificity : 0.5301
         Pos Pred Value : 0.8050
         Neg Pred Value : 0.7857
             Prevalence : 0.6758
         Detection Rate : 0.6289
   Detection Prevalence : 0.7812
      Balanced Accuracy : 0.7304

       'Positive' Class : 0
```

```r
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 80.07812
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.805
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.9306358
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.8608391
```

for Test data

```
set.seed(12)
knn_model_condense <- knn(x_train, x_test, y_train, k = 15,prob=TRUE)
conf_matrix <- confusionMatrix(as.factor(knn_model_condense), as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 138  55
         1  16  47

               Accuracy : 0.7227
                 95% CI : (0.6635, 0.7766)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 3.471e-05

                  Kappa : 0.3815

 Mcnemar's Test P-Value : 6.490e-06

            Sensitivity : 0.8961
            Specificity : 0.4608
         Pos Pred Value : 0.7150
         Neg Pred Value : 0.7460
             Prevalence : 0.6016
         Detection Rate : 0.5391
   Detection Prevalence : 0.7539
      Balanced Accuracy : 0.6784

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 72.26562
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.7150259
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8961039
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.8000877
```

Comparison	Train	Test
Accuracy(%)	80	72
Precision	0.80	0.71
Recall	0.93	0.89
F score	0.86	0.80

Comment -

- KNN model shows good performance on the training set (80% accuracy, 0.80 precision) but slightly lower on the test set (72% accuracy, 0.71 precision), indicating some overfitting.
- The drop in precision suggests the model may be making more false positive predictions on unseen data.

ii. using a condensed training set

```
set.seed(12)
suppress_output <- capture.output({   condensed_indices <- condense(x_train, y_train) })
X_train_condensed <- x_train[condensed_indices, , drop = FALSE]
y_train_condensed <- y_train[condensed_indices]
```

for Training data

```
set.seed(12)
knn_model <- knn(X_train_condensed, x_train, y_train_condensed, k = 15,prob=TRUE)
conf_matrix <- confusionMatrix(as.factor(knn_model), as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 323  91
         1  23  75

               Accuracy : 0.7773
                 95% CI : (0.7388, 0.8127)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 2.596e-07

                  Kappa : 0.4313

 Mcnemar's Test P-Value : 3.494e-10

            Sensitivity : 0.9335
            Specificity : 0.4518
         Pos Pred Value : 0.7802
         Neg Pred Value : 0.7653
             Prevalence : 0.6758
         Detection Rate : 0.6309
   Detection Prevalence : 0.8086
      Balanced Accuracy : 0.6927

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 77.73438
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.7801932

cat("Recall with training set:",recall,"\n")

Recall with training set: 0.933526

cat("F Score with training set:",F_score,"\n")

F Score with training set: 0.8483061
```

for Test data

```
set.seed(12)
knn_model <- knn(X_train_condensed, x_test, y_train_condensed, k = 15,prob=TRUE)
conf_matrix <- confusionMatrix(as.factor(knn_model), as.factor(y_test))
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 137  62
         1  17  40

               Accuracy : 0.6914
                 95% CI : (0.6309, 0.7474)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 0.001794

                  Kappa : 0.3044

 Mcnemar's Test P-Value : 7.407e-07

            Sensitivity : 0.8896
            Specificity : 0.3922
         Pos Pred Value : 0.6884
         Neg Pred Value : 0.7018
             Prevalence : 0.6016
         Detection Rate : 0.5352
   Detection Prevalence : 0.7773
      Balanced Accuracy : 0.6409

       'Positive' Class : 0


accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")

Accuracy with test set: 69.14062

cat("Precision with test set:", precision, "\n")

Precision with test set: 0.6884422

cat("Recall with test set",recall,"\n")

Recall with test set 0.8896104

cat("F Score with test set",F_score,"\n")

F Score with test set 0.7780844
```

Comparison	Train	Test
Accuracy(%)	77	69
Precision	0.78	0.68
Recall	0.93	0.88
F score	0.84	0.77

Comment-

- The accuracy decline from 77% in training to 69% in testing suggests the model has some degree of overfitting. It performs reasonably well on the training data but struggles to generalize to unseen data.
- The precision drop indicates that the model is making more false positive predictions on the test set compared to the training set. This suggests that some patterns the model learned during training might not apply well to new data.
- The performance difference is not extreme, which indicates that the model is not overfitting heavily, but there is room for improvement

iii. using an edited training set obtained by the MULTIEDIT algorithm

```
multi_edit <- function(X_train, y_train, max_iter = 5, k = 3) {
for (i in 1:max_iter) {
cat("Iteration:", i, "\n")
preds <- knn(X_train, X_train, y_train, k = k)
errors <- which(preds != y_train)
if (length(errors) == 0) {
cat("No errors found. Stopping early.\n")
break     }
X_train <- X_train[-errors, , drop = FALSE]
y_train <- y_train[-errors]
cat(length(errors), "misclassified instances removed.\n")   }
return(list(X_train = X_train, y_train = y_train)) }
edited_data <- multi_edit(x_train, y_train)

Iteration: 1
77 misclassified instances removed.
Iteration: 2
7 misclassified instances removed.
Iteration: 3
No errors found. Stopping early.
```

for Training data

```
set.seed(12)
knn_model_edited <- knn(train =edited_data$X_train,test= x_train, edited_data$y_train, k = 15,prob=TRUE)
conf_matrix <- confusionMatrix(factor(knn_model_edited), as.factor(y_train))
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 323  91
         1  23  75

               Accuracy : 0.7773
                 95% CI : (0.7388, 0.8127)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 2.596e-07

                  Kappa : 0.4313

 Mcnemar's Test P-Value : 3.494e-10

            Sensitivity : 0.9335
```

```
            Specificity : 0.4518
         Pos Pred Value : 0.7802
         Neg Pred Value : 0.7653
             Prevalence : 0.6758
         Detection Rate : 0.6309
   Detection Prevalence : 0.8086
      Balanced Accuracy : 0.6927

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 77.73438
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.7801932
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.933526
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.8483061
```

for Test data

```
set.seed(12)
knn_model_edited <- knn(edited_data$X_train, x_test, edited_data$y_train, k = 15,prob=TRUE)
conf_matrix <- confusionMatrix(factor(knn_model_edited),as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 142  59
         1  12  43

               Accuracy : 0.7227
                 95% CI : (0.6635, 0.7766)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 3.471e-05

                  Kappa : 0.3726

 Mcnemar's Test P-Value : 4.783e-08

            Sensitivity : 0.9221
            Specificity : 0.4216
         Pos Pred Value : 0.7065
         Neg Pred Value : 0.7818
             Prevalence : 0.6016
         Detection Rate : 0.5547
   Detection Prevalence : 0.7852
      Balanced Accuracy : 0.6718
```

```
       'Positive' Class : 0


accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")

Accuracy with test set: 72.26562

cat("Precision with test set:", precision, "\n")

Precision with test set: 0.7064677

cat("Recall with test set",recall,"\n")

Recall with test set 0.9220779

cat("F Score with test set",F_score,"\n")

F Score with test set 0.8102773
```

Comparison	Train	Test
Accuracy(%)	77	72
Precision	0.78	0.70
Recall	0.93	0.92
F score	0.84	0.81

Comment-

- The test set acuuracy and precision improved in this case but the test set accuracy and precision remain highest where none of this methods used (case i.)

b) Using Linear Discriminant Analysis

for Training data

```
set.seed(12)
lda_model <- lda(Outcome ~ ., data = train_data)
lda_predictions <- predict(lda_model, newdata = x_train)
predicted_classes <-as.factor(lda_predictions$class)
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 142  59
         1  12  43

               Accuracy : 0.7227
                 95% CI : (0.6635, 0.7766)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 3.471e-05

                  Kappa : 0.3726

 Mcnemar's Test P-Value : 4.783e-08

            Sensitivity : 0.9221
            Specificity : 0.4216
         Pos Pred Value : 0.7065
         Neg Pred Value : 0.7818
```

```
             Prevalence : 0.6016
         Detection Rate : 0.5547
   Detection Prevalence : 0.7852
      Balanced Accuracy : 0.6718

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 72.26562
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.7064677
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.9220779
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.8102773
```

for Test data

```
set.seed(12)
lda_model <- lda(Outcome ~ ., data = train_data)
lda_predictions <- predict(lda_model, newdata = x_test)
predicted_classes <-as.factor(lda_predictions$class)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.04688
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.740113
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8506494
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	80	73
Precision	0.82	0.74
Recall	0.92	0.85
F score	0.81	0.78

Comment-

- The training accuracy (80%) and test accuracy (73%) suggest that the model performs reasonably well but shows some drop in performance on unseen data, which could indicate mild overfitting. The drop in precision from 0.82 (train) to 0.74 (test) implies that the model may be making more false positive predictions on the test data. This means that while the model learned the patterns well during training, it may struggle to generalize to unseen data
- The test set precision and accuracy here is better than KNN method.

c) Quadratic Discriminant Analysis

for Training data

```
set.seed(12)
qda_model <- qda(Outcome ~ ., data = train_data)
qda_predictions <- predict(qda_model, newdata = x_train)
predicted_classes <-as.factor(qda_predictions$class)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 306  64
         1  40 102

               Accuracy : 0.7969
                 95% CI : (0.7594, 0.8309)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 7.525e-10

                  Kappa : 0.5183

 Mcnemar's Test P-Value : 0.02411

            Sensitivity : 0.8844
```

```
            Specificity : 0.6145
         Pos Pred Value : 0.8270
         Neg Pred Value : 0.7183
             Prevalence : 0.6758
         Detection Rate : 0.5977
   Detection Prevalence : 0.7227
      Balanced Accuracy : 0.7494

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 79.6875
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.827027
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.8843931
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.8383562
```

for Test data

```
set.seed(12)
qda_model <- qda(Outcome ~ ., data = train_data)
qda_predictions <- predict(qda_model, newdata = x_test)
predicted_classes <-as.factor(qda_predictions$class)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 122  46
         1  32  56

               Accuracy : 0.6953
                 95% CI : (0.6349, 0.7511)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 0.001165

                  Kappa : 0.3493

 Mcnemar's Test P-Value : 0.141032

            Sensitivity : 0.7922
            Specificity : 0.5490
         Pos Pred Value : 0.7262
         Neg Pred Value : 0.6364
             Prevalence : 0.6016
         Detection Rate : 0.4766
   Detection Prevalence : 0.6562
```

```
      Balanced Accuracy : 0.6706

       'Positive' Class : 0


accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")

Accuracy with test set: 69.53125

cat("Precision with test set:", precision, "\n")

Precision with test set: 0.7261905

cat("Recall with test set",recall,"\n")

Recall with test set 0.7922078

cat("F Score with test set",F_score,"\n")

F Score with test set 0.7406043
```

Comparison	Train	Test
Accuracy(%)	79	69
Precision	0.82	0.72
Recall	0.88	0.79
F score	0.83	0.74

Comment-

- KNN has the highest test accuracy (72%), indicating better generalization to unseen data. QDA and LDA have slightly lower test accuracy (69% and 68%, respectively), suggesting these models may struggle more with unseen data.
- QDA has the highest test precision (0.72), indicating fewer false positives in predictions. KNN comes close with a test precision of 0.71, followed by LDA with 0.70.
- LDA and QDA show slightly larger gaps between training and test accuracy than KNN, suggesting KNN generalizes better overall.

d) Flexible Discriminant Analysis with Multivariate Adaptive Regression Splines (MARS)

for Training data

```
set.seed(12)
fda_model <- fda(Outcome ~ ., data = train_data)
fda_predictions <- predict(fda_model, newdata = x_train)
predicted_classes <-as.factor(fda_predictions)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_train))
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 315  69
         1  31  97

               Accuracy : 0.8047
                 95% CI : (0.7677, 0.8382)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 5.249e-11
```

```
                 Kappa : 0.5261

 Mcnemar's Test P-Value : 0.0002156

            Sensitivity : 0.9104
            Specificity : 0.5843
         Pos Pred Value : 0.8203
         Neg Pred Value : 0.7578
             Prevalence : 0.6758
         Detection Rate : 0.6152
   Detection Prevalence : 0.7500
      Balanced Accuracy : 0.7474

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 80.46875
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.8203125
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.9104046
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.8542879
```

for Test data

```
set.seed(12)
fda_model <- fda(Outcome ~ ., data = train_data)
fda_predictions <- predict(fda_model, newdata = x_test)
predicted_classes <-as.factor(fda_predictions)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
```

```
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.04688
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.740113
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8506494
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	80	73
Precision	0.82	0.74
Recall	0.91	0.85
F score	0.85	0.78

Comment-

- FDA performs better in terms of test accuracy and precision compared to KNN, LDA, and QDA, with the most stable performance across both training and test sets.
- KNN and QDA show signs of overfitting, with larger drops in performance from training to test sets.
- LDA performs reasonably well but still falls slightly behind FDA in both accuracy and precision.
- Overall, FDA seems to be the most suitable model, especially if consistent performance and fewer false positives are the priorities.

e) Using Penalized Discriminant Analysis with Generalized Ridge Regression

for Training data

```
set.seed(12)
pda_model <- fda(Outcome ~ ., data = train_data,lambda=0.1)
pda_predictions <- predict(pda_model, newdata = x_train)
predicted_classes <-as.factor(pda_predictions)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 315  69
         1  31  97
```

```
               Accuracy : 0.8047
                 95% CI : (0.7677, 0.8382)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 5.249e-11

                  Kappa : 0.5261

 Mcnemar's Test P-Value : 0.0002156

            Sensitivity : 0.9104
            Specificity : 0.5843
         Pos Pred Value : 0.8203
         Neg Pred Value : 0.7578
             Prevalence : 0.6758
         Detection Rate : 0.6152
   Detection Prevalence : 0.7500
      Balanced Accuracy : 0.7474

       'Positive' Class : 0


accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")

Accuracy with training set: 80.46875

cat("Precision with training set:", precision, "\n")

Precision with training set: 0.8203125

cat("Recall with training set:",recall,"\n")

Recall with training set: 0.9104046

cat("F Score with training set:",F_score,"\n")

F Score with training set: 0.8542879
```

for Test data

```
set.seed(12)
pda_model <- fda(Outcome ~ ., data = train_data,lambda=0.1)
pda_predictions <- predict(pda_model, newdata = x_test)
predicted_classes <-as.factor(pda_predictions)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_test))
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05
```

```
                 Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.04688
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.740113
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8506494
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	80	73
Precision	0.82	0.74
Recall	0.91	0.85
F score	0.85	0.78

Comment-

- Choosing lambda 0.1 does not change the results than the previous ones but if we keep on changing lambda and observe the results repeatedly then we might find improvements.

f) Mixture Discriminant Analysis

for Training data

```
set.seed(12)
mda_model <- mda(Outcome ~ ., data = train_data, subclasses = 2)
mda_predictions <- predict(mda_model, newdata = x_train)
predicted_classes <-as.factor(mda_predictions)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 319  71
         1  27  95
```

```
               Accuracy : 0.8086
                 95% CI : (0.7718, 0.8418)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 1.289e-11

                  Kappa : 0.5309

 Mcnemar's Test P-Value : 1.401e-05

            Sensitivity : 0.9220
            Specificity : 0.5723
         Pos Pred Value : 0.8179
         Neg Pred Value : 0.7787
             Prevalence : 0.6758
         Detection Rate : 0.6230
   Detection Prevalence : 0.7617
      Balanced Accuracy : 0.7471

       'Positive' Class : 0


accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")

Accuracy with training set: 80.85938

cat("Precision with training set:", precision, "\n")

Precision with training set: 0.8179487

cat("Recall with training set:",recall,"\n")

Recall with training set: 0.9219653

cat("F Score with training set:",F_score,"\n")

F Score with training set: 0.861566
```

for Test data

```
set.seed(12)
mda_model <- mda(Outcome ~ ., data = train_data, subclasses = 2)
mda_predictions <- predict(mda_model, newdata = x_test)
predicted_classes <-as.factor(mda_predictions)
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_test))
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 132  45
         1  22  57

               Accuracy : 0.7383
                 95% CI : (0.6799, 0.791)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 3.108e-06
```

```
                 Kappa : 0.4324

 Mcnemar's Test P-Value : 0.007194

            Sensitivity : 0.8571
            Specificity : 0.5588
         Pos Pred Value : 0.7458
         Neg Pred Value : 0.7215
             Prevalence : 0.6016
         Detection Rate : 0.5156
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.7080

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.82812
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.7457627
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8571429
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7932844
```

Comparison	Train	Test
Accuracy(%)	80	73
Precision	0.81	0.74
Recall	0.92	0.85
F score	0.86	0.79

Comment-

- Precision for test set slightly dipped rest is same as before and we do not receive any better improvements using Mixture Discriminant Analysis

g) Using logistic regression

for Training data

```
set.seed(12)
logi_model=glm(Outcome~.,data=train_data,family = "binomial")
logi_predictions <- predict(logi_model, newdata = x_train ,type="response")
predicted_classes <- as.factor(ifelse(logi_predictions > 0.5, 1, 0))
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 314  70
         1  32  96
```

```
               Accuracy : 0.8008
                 95% CI : (0.7635, 0.8345)
    No Information Rate : 0.6758
    P-Value [Acc > NIR] : 2.036e-10

                  Kappa : 0.5166

 Mcnemar's Test P-Value : 0.0002487

            Sensitivity : 0.9075
            Specificity : 0.5783
         Pos Pred Value : 0.8177
         Neg Pred Value : 0.7500
             Prevalence : 0.6758
         Detection Rate : 0.6133
   Detection Prevalence : 0.7500
      Balanced Accuracy : 0.7429

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 80.07812
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.8177083
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.9075145
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.8508135
```

for Test data

```
set.seed(12)
logi_model=glm(Outcome~.,data=train_data,family = "binomial")
logi_predictions <- predict(logi_model, newdata = x_test ,type="response")
predicted_classes <- as.factor(ifelse(logi_predictions > 0.5, 1, 0))
conf_matrix <- confusionMatrix(predicted_classes, as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05
```

```
                 Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.04688
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.740113
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8506494
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	80	73
Precision	0.81	0.74
Recall	0.90	0.85
F score	0.85	0.78

Comment-

- Results are almost same as the previous result no improvement found

h) Ensemble classification with

i. Bagging by Breiman's algorithm, using classification trees as single classifiers for Training data

```
set.seed(12)
bagging_model <- bagging(Outcome ~ ., data = train_data, nbagg = 50,coob = TRUE)
y_train <- factor(train_data$Outcome, levels = c(0, 1))
bagging_predictions <- predict(bagging_model, newdata = train_data)
bagging_predictions=ifelse(bagging_predictions>0.5,1,0)
conf_matrix_bagging <- confusionMatrix(as.factor(bagging_predictions), as.factor(y_train))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
```

```
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 73.04688
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.740113
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.8506494
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.7859916
```

for Test data

```
set.seed(12)
bagging_model <- bagging(Outcome ~ ., data = train_data, nbagg = 50,coob = TRUE)
y_test <- factor(test_data$Outcome, levels = c(0, 1))
bagging_predictions <- predict(bagging_model, newdata = test_data,type="probability")
bagging_predictions=ifelse(bagging_predictions>0.5,1,0)
conf_matrix_bagging <- confusionMatrix(as.factor(bagging_predictions), as.factor(y_test))
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
```

```
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.04688
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.740113
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8506494
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	89	73
Precision	0.89	0.74
Recall	0.85	0.85
F score	0.78	0.78

Comment-

- Only training set results have been improved so this actually caused over-fitted result

ii. Boosting by the AdaBoost.M1 algorithm of Freund and Schapire, using classification trees as single classifiers for Training data

```
set.seed(12)
train_data$Outcome <- as.factor(train_data$Outcome)
adaboost_model <- boosting(   Outcome ~ .,              data = train_data,          mfinal = 50,              c
adaboost_predictions <- predict(adaboost_model, newdata = train_data)
y_test <- factor(test_data$Outcome, levels = c(0, 1))
predicted_classes <- factor(adaboost_predictions$class, levels = c(0, 1))
conf_matrix_boosting <- confusionMatrix(predicted_classes, y_train)
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
```

```
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 73.04688
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.740113
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.8506494
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.7859916
```

for Test data

```
set.seed(12)
train_data$Outcome <- as.factor(train_data$Outcome)
adaboost_model <- boosting(   Outcome ~ .,              data = train_data,          mfinal = 50,              c
adaboost_predictions <- predict(adaboost_model, newdata = test_data)
y_test <- factor(test_data$Outcome, levels = c(0, 1))
predicted_classes <- factor(adaboost_predictions$class, levels = c(0, 1))
conf_matrix_boosting <- confusionMatrix(predicted_classes, y_test)
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
```

```
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0


accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")

Accuracy with test set: 73.04688

cat("Precision with test set:", precision, "\n")

Precision with test set: 0.740113

cat("Recall with test set",recall,"\n")

Recall with test set 0.8506494

cat("F Score with test set",F_score,"\n")

F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	80	73
Precision	0.82	0.74
Recall	0.85	0.85
F score	0.78	0.78

Comment-

- Again we observed improvements on training but not on test set which actually indicates the model has overfitted

iii. Random Forests
for Training data

```
set.seed(12)
rf_model <- randomForest(   Outcome ~ .,            data = train_data,       ntree = 100,            mtry = 3,
rf_predictions <- predict(rf_model, newdata = train_data)
y_train <- factor(train_data$Outcome, levels = levels(rf_predictions))
conf_matrix_rf <- confusionMatrix(rf_predictions, y_train)
print(conf_matrix)

Confusion Matrix and Statistics

          Reference
Prediction   0   1
```

```
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with training set:", accuracy*100, "\n")
```

```
Accuracy with training set: 73.04688
```

```
cat("Precision with training set:", precision, "\n")
```

```
Precision with training set: 0.740113
```

```
cat("Recall with training set:",recall,"\n")
```

```
Recall with training set: 0.8506494
```

```
cat("F Score with training set:",F_score,"\n")
```

```
F Score with training set: 0.7859916
```

for Test data

```
set.seed(12)
rf_model <- randomForest(   Outcome ~ .,            data = train_data,       ntree = 100,            mtry = 3,
rf_predictions <- predict(rf_model, newdata = test_data)
y_test <- factor(test_data$Outcome, levels = levels(rf_predictions))
conf_matrix_rf <- confusionMatrix(rf_predictions, y_test)
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  46
         1  23  56

               Accuracy : 0.7305
                 95% CI : (0.6717, 0.7838)
    No Information Rate : 0.6016
```

```
    P-Value [Acc > NIR] : 1.079e-05

                  Kappa : 0.4155

 Mcnemar's Test P-Value : 0.008085

            Sensitivity : 0.8506
            Specificity : 0.5490
         Pos Pred Value : 0.7401
         Neg Pred Value : 0.7089
             Prevalence : 0.6016
         Detection Rate : 0.5117
   Detection Prevalence : 0.6914
      Balanced Accuracy : 0.6998

       'Positive' Class : 0
```

```
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Precision"]
recall <- conf_matrix$byClass["Recall"]
F_score <- 2/((1/accuracy)+(1/recall))
cat("Accuracy with test set:", accuracy*100, "\n")
```

```
Accuracy with test set: 73.04688
```

```
cat("Precision with test set:", precision, "\n")
```

```
Precision with test set: 0.740113
```

```
cat("Recall with test set",recall,"\n")
```

```
Recall with test set 0.8506494
```

```
cat("F Score with test set",F_score,"\n")
```

```
F Score with test set 0.7859916
```

Comparison	Train	Test
Accuracy(%)	100	73
Precision	0.1	0.74
Recall	0.85	0.85
F score	0.78	0.78

Comment-

The models show tremendous overfitting as it acheives 100 percent accuracy on train data but no improvements on test data that the previous ones.

3. From the fitted random forest model, generate a plot of the variable importance measures, and comment.

```
cat("\nFeature Importance:\n")
```

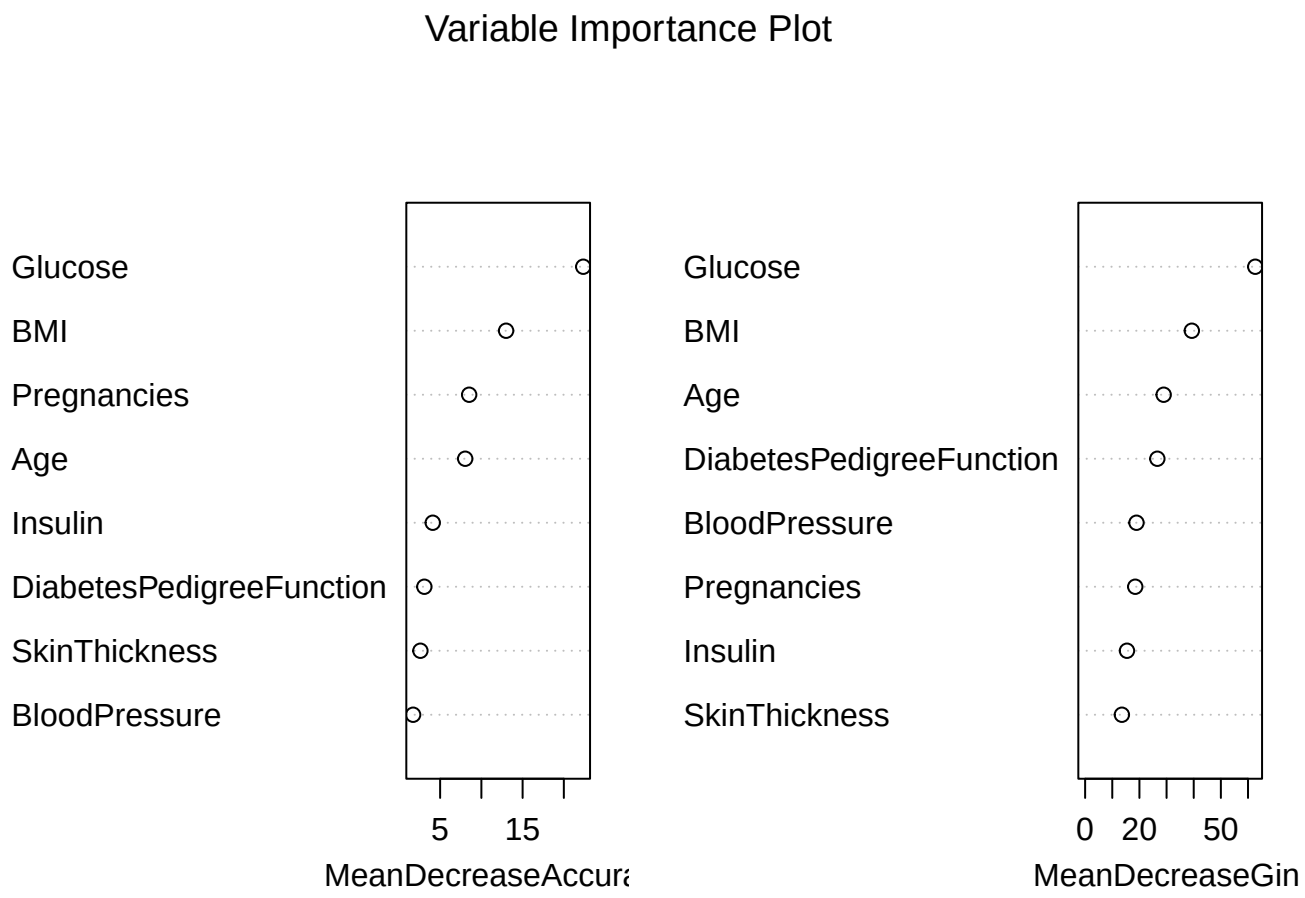
```
Feature Importance:
```

```
print(importance(rf_model))
```

```
                                0          1 MeanDecreaseAccuracy
Pregnancies              8.494195  2.2701527             8.519092
Glucose                 16.140532 14.2901652            22.351620
BloodPressure            3.328846 -0.9820783             1.723321
SkinThickness            1.942064  1.3969004             2.606678
Insulin                  4.508120  0.9531893             4.103802
```

```
BMI                       9.273941  8.8102378            12.998753
DiabetesPedigreeFunction  3.594465  0.8542759             3.084746
Age                       8.028372  3.0468146             8.038679
                        MeanDecreaseGini
Pregnancies                    18.46009
Glucose                        62.66804
BloodPressure                  18.92036
SkinThickness                  13.53971
Insulin                        15.41615
BMI                            39.30032
DiabetesPedigreeFunction       26.59962
Age                            28.93169
```

```
varImpPlot(rf_model, main = "Variable Importance Plot")
```



Comment

- Glucose has the highest importance in both Mean Decrease Accuracy and Mean Decrease Gini. This suggests that glucose levels are the most critical predictor for determining diabetes in your dataset. BMI (Body Mass Index) also has high importance, indicating a strong relationship between BMI and the onset of diabetes.
- Age and Pregnancies are also relatively important, suggesting that both age and pregnancy history play a role in predicting diabetes onset.
- Blood Pressure, Insulin, Skin Thickness, and Diabetes Pedigree Function appear to have lower contributions in both plots. This might mean they contribute less to improving the predictive performance of the model, though they are not entirely irrelevant.

Overall comparison

Method	Train Accuracy(%)	Test Accuracy(%)	Train Recall	Test Recall	Train Precision	Test Precisio
KNN- usual	80	72	0.93	0.89	0.80	0.71
KNN-condensed set	77	69	0.84	0.88	0.78	0.68
KNN-Multiedit algorithm	77	72	0.93	0.92	0.78	0.70
LDA	80	73	0.92	0.85	0.82	0.74
QDA	79	69	0.88	0.79	0.82	0.72
FDA	80	73	0.91	0.85	0.82	0.74
PDA with Ridge regression	80	73	0.91	0.85	0.82	0.74
MDA	80	73	0.92	0.85	0.81	0.74
Logistic Regression	80	73	0.90	0.85	0.81	0.74
Ensemble - Bagging	89	73	0.85	0.85	0.89	0.74
Ensemble - AdaBoost	80	73	0.85	0.85	0.82	0.74
Ensemble - Random Forest	100	0.1	0.85	0.85	73	0.74

Conclusion

The best performing models in terms of balanced accuracy and precision are FDA, LDA, MDA, Bagging, AdaBoost, and Logistic Regression, each achieving 73% test accuracy and 0.74 precision. These models generalize well to unseen data, making them suitable choices for this problem.

Random Forest shows signs of overfitting with 100% training accuracy, while KNN models underperform slightly, suggesting they might not be the optimal choice for this dataset.

Given the results, ensemble methods like Bagging and AdaBoost offer a slight edge, as they provide consistent performance with good generalization and robustness.

2. To compare the performance of all the classifiers trained in problem no. 1, show their ROC curves in a single plot and comment. You can use the ROCR package in R for this purpose.

The plot function

```
y_test <- factor(test_data$Outcome)
plot_roc <- function(probabilities, true_labels,color,hide_axis=F) {
pred <- prediction(probabilities, true_labels)
perf <- performance(pred, "tpr", "fpr")
plot(perf, col=color , lwd = 2, xaxt=ifelse(hide_axis,'n','s'),yaxt=ifelse(hide_axis,'n','s'),        xlab =ifel
```

Probabilities for all the models

```
#-random forest
rf_probs <- predict(rf_model, newdata = test_data, type = "prob")[, 2]
#--logistic
logistic_probs <- predict(logi_model, newdata = test_data, type = "response")
#--FDA
fda_probs <- predict(fda_model, newdata = test_data, type = "posterior")[, 2]
#--LDA
lda_probs <- predict(lda_model, newdata = test_data)$posterior[, 2]
#--MDA
mda_probs <- predict(mda_model, newdata = test_data, type = "posterior")[, 2]
#--QDA
qda_probs <- predict(qda_model, newdata = test_data)$posterior[, 2]
#--PDA
pda_probs <- predict(pda_model, newdata = test_data, type = "posterior")[, 2]
#--bagging
bagging_probs <- predict(bagging_model, newdata = test_data)
#--boosting
adaboost_probs <- predict(adaboost_model, newdata = test_data)$prob[, 2]
#--KNN-usual
knn_probs <- attr(knn_model, "prob")
knn_probs <- ifelse(knn_model == "1", knn_probs, 1 - knn_probs)
#--KNN condensed set
knn_probs_condense <- attr(knn_model_condense, "prob")
```

```
knn_probs_condense <- ifelse(knn_model_condense == "1", knn_probs_condense, 1 - knn_probs_condense)
#--KNN _edited
knn_probs_edited <- attr(knn_model_edited, "prob")
knn_probs_edited <- ifelse(knn_model_edited == "1", knn_probs_edited, 1 - knn_probs_edited)
```

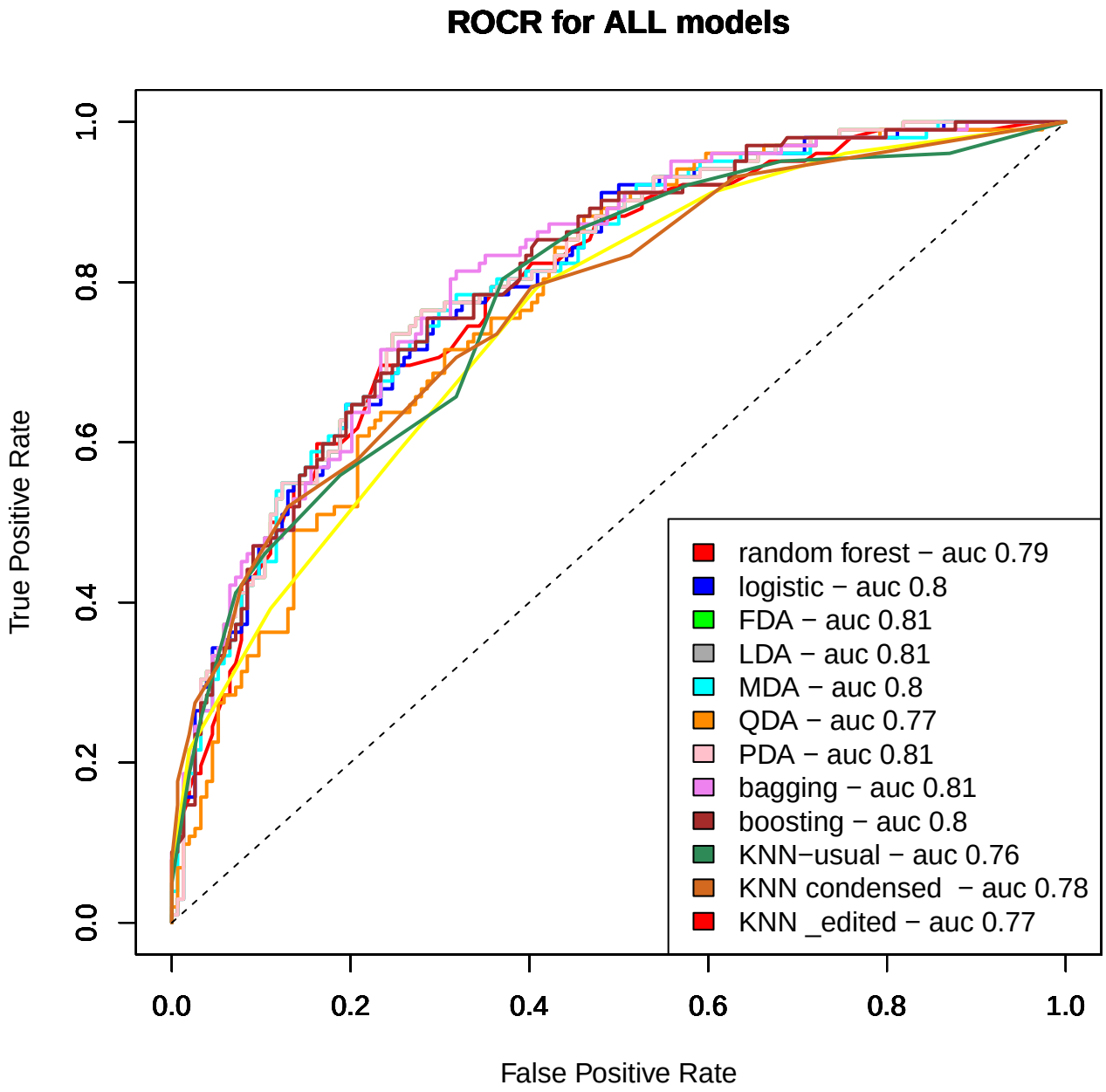
AUC of all the models

```
auc_function=function(probabilities,true_labels,model_name){   pred=prediction(probabilities,true_labels)
perf=performance(pred,"auc")
auc=perf@y.values[[1]]
return(paste(model_name,"- auc", round(auc,2))) }
#--legend
legend_vec=c(auc_function(model_name="random forest",  rf_probs,y_test),auc_function(model_name="logistic", logi
```

Plotting the ROCR curve

```
plot_roc(rf_probs, y_test,color="red")
par(new=T)
plot_roc(hide_axis=T,logistic_probs, y_test,color="blue")
par(new=T)
plot_roc(hide_axis=T,fda_probs, y_test,color="green")
par(new=T)
plot_roc(hide_axis=T,lda_probs, y_test,color="darkgrey")
par(new=T)
plot_roc(hide_axis=T,mda_probs, y_test,color="cyan")
par(new=T)
plot_roc(qda_probs, hide_axis=T,y_test,color="darkorange")
par(new=T)
plot_roc(pda_probs,hide_axis=T, y_test,color="pink")
par(new=T)
plot_roc(bagging_probs, hide_axis=T,y_test,color="violet")
par(new=T)
plot_roc(adaboost_probs,hide_axis=T, y_test,color="brown")
par(new=T)
plot_roc(knn_probs,hide_axis=T, y_test,color="yellow")
par(new=T)
plot_roc(knn_probs_condense, hide_axis=T,y_test,color="seagreen")
par(new=T)
plot_roc(hide_axis=T,knn_probs_edited, y_test,color="chocolate")
par(new=T)
curve(1*x+0,col="black",lty=2,xaxt='n',yaxt='n',,xlab="",ylab="")

legend("bottomright",legend =legend_vec,fill=c("red","blue","green","darkgrey","cyan","darkorange","pink","viole
```



Conclusion

- In this comparison, FDA, LDA, PDA, and Bagging demonstrate the best predictive performance with an AUC of 0.81. These models might be preferred when aiming for the highest accuracy and robustness. However, Logistic Regression and Boosting (AUC 0.80) also perform well and are good alternatives, especially if interpretability (in the case of logistic regression) or ensemble methods (boosting) are desired.
- Models like KNN (Usual) have comparatively lower performance, so they may not be the best choice for tasks where predictive power is critical