Diabetes data analysis

Tanmay Gayen

May 2, 2025

1 Introduction

Feature Descriptions of the Diabetes Dataset

Feature	**Description**
Pregnancies	Number of times the individual has been pregnant
Glucose	Plasma glucose concentration measured two hours after an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (in mm Hg)
SkinThickness	Thickness of the triceps skinfold (in mm), a measure of subcutaneous fat
Insulin	2-hour serum insulin level (in mu U/ml)
BMI	Body Mass Index: weight (kg) divided by height squared (m^2); an indicator of body fat.
Diabetes Pedigree Function	A score indicating genetic risk of diabetes based on family history; higher values suggest greater predisposition
Age	Age of the individual (in years)
Outcome	Target variable (0 = no diabetes, 1 = diabetes present)

```r
library(readr)
library(caTools)
library(caret)
library(e1071)
data=read.csv("diabetes.csv")
head(data)
```

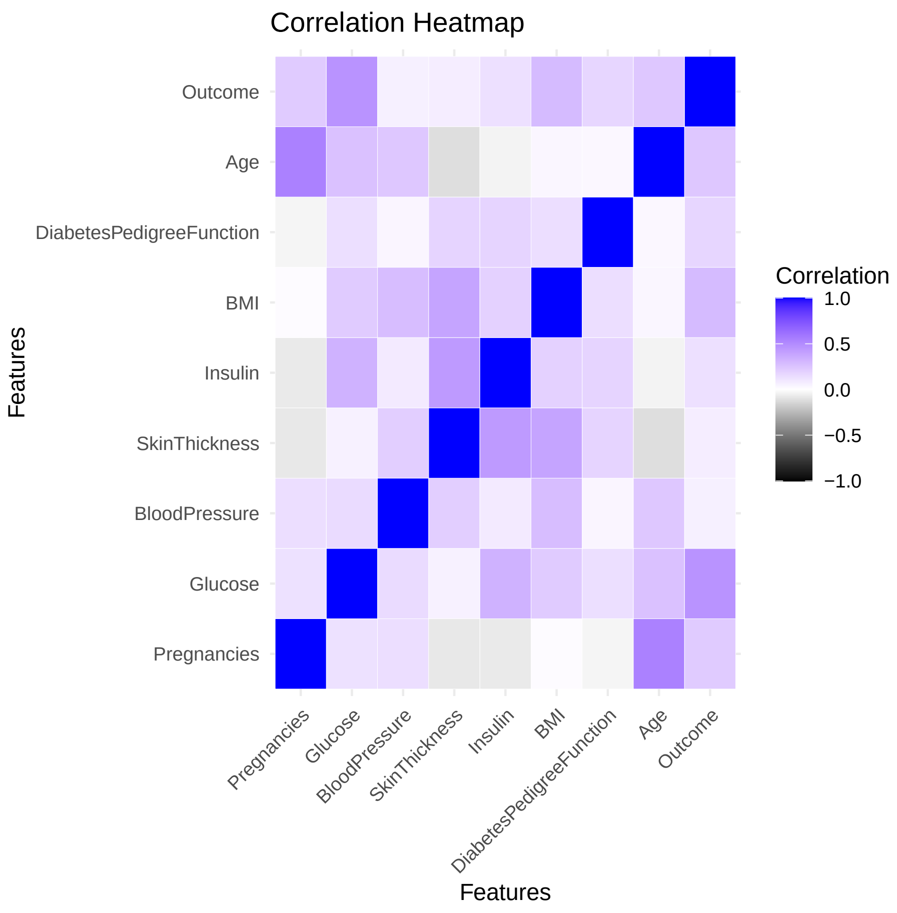
```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            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
```

2 Analyze the data

```
# Check data summary
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

# Check for missing values
colSums(is.na(data))

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

```r
sum(is.na(data))

## [1] 0
```

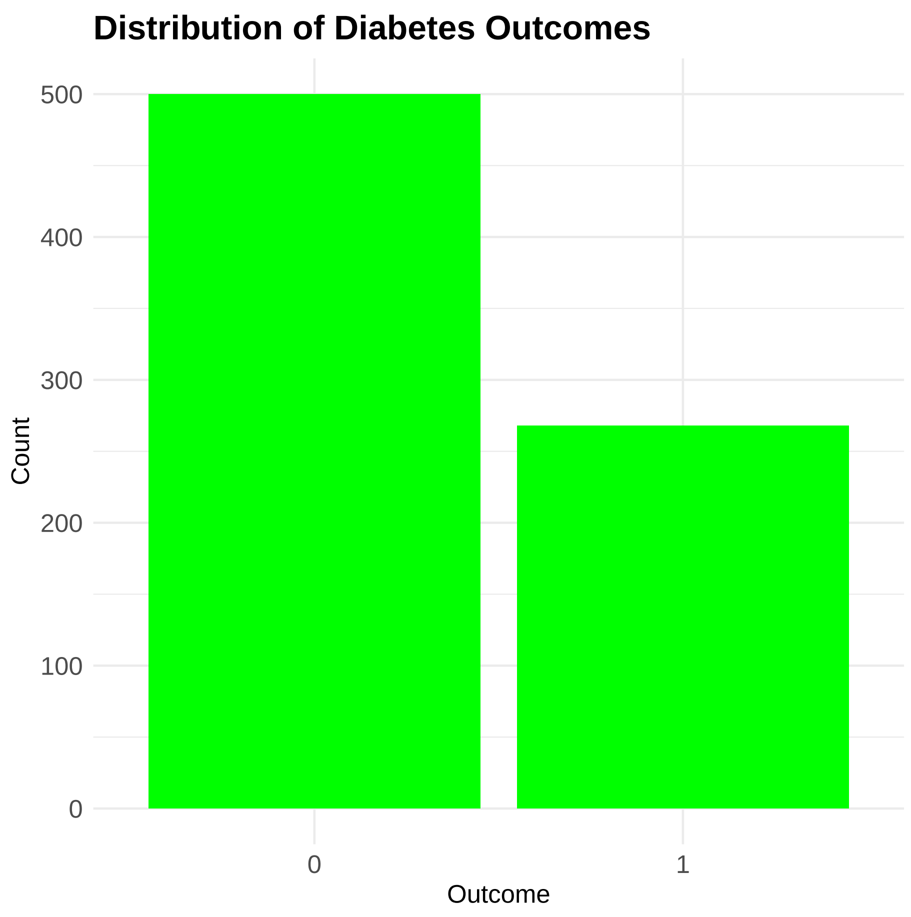
```r
library(ggplot2)
library(reshape2)

correlation_matrix <- cor(data)

# Convert correlation matrix to long format
correlation_melted <- melt(correlation_matrix)

# Plot heatmap
ggplot(correlation_melted, aes(Var1, Var2, fill=value)) +
  geom_tile(color="white") +
  scale_fill_gradient2(low="black", high="blue", mid="white", midpoint=0,
                       limit=c(-1,1), space="Lab", name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title="Correlation Heatmap", x="Features", y="Features")
```



3 Conclusion from the Correlation Heatmap:

The correlation heatmap visually presents the linear relationships between all features in the diabetes dataset. The color gradient from black (negative correlation) to blue (positive correlation), white indicating no correlation, helps to clearly identify these relationships.Glucose shows a strong positive correlation with outcome, suggesting that higher glucose levels are often associated with a higher likelihood of diabetes. Pregnancies and age show a moderate positive correlation, which makes sense as older individuals may have had more pregnancies. Skin thickness, insulin, and BMI are somewhat correlated with each other, indicating a possible pattern of body composition and metabolic characteristics.Some features, such as Diabetes Pedigree Function, show weak correlations with most others, indicating greater independence.Features with near-zero correlation (represented in white) indicate minimal or no linear relationship with each other. This heatmap is useful for identifying multicollinearity. If two features are highly correlated, one might be removed or transformed before model training.Strongly correlated variables with the Outcome can be prioritized for predictive modeling.Weakly correlated variables might still be valuable due to non-linear or interaction effects, which linear correlation won't capture.

```r
outcome_counts <- table(data$Outcome)
outcome_df <- data.frame(Outcome = names(outcome_counts),
                         Count = as.numeric(outcome_counts))

# Create bar plot
ggplot(outcome_df, aes(x=Outcome, y=Count)) +
  geom_bar(stat="identity", fill="GREEN") +
  labs(title="Distribution of Diabetes Outcomes", x="Outcome", y="Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=12),
        axis.title = element_text(size=12),
        plot.title = element_text(size=16, face="bold"))
```

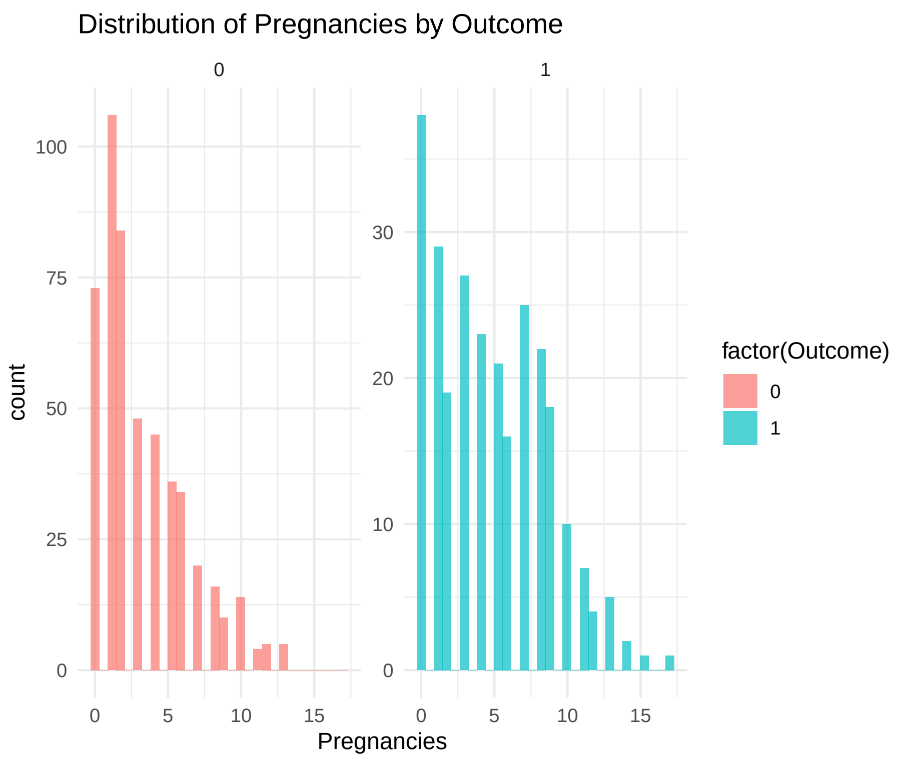


4 Conclusion from the Bar Plot:

The bar plot displays the frequency of diabetes diagnoses (0 = No diabetes, 1 = Diabetes) in the dataset.The majority of observations belong to the ”0” (non-diabetic) category.A smaller proportion of individuals are labeled with ”1” (diabetic).

```r
library(ggplot2)

# Select relevant columns
diabetes_subset <- data[, c("Pregnancies", "Glucose", "BloodPressure",
                            "BMI", "Age", "Outcome")]

# Histograms
```

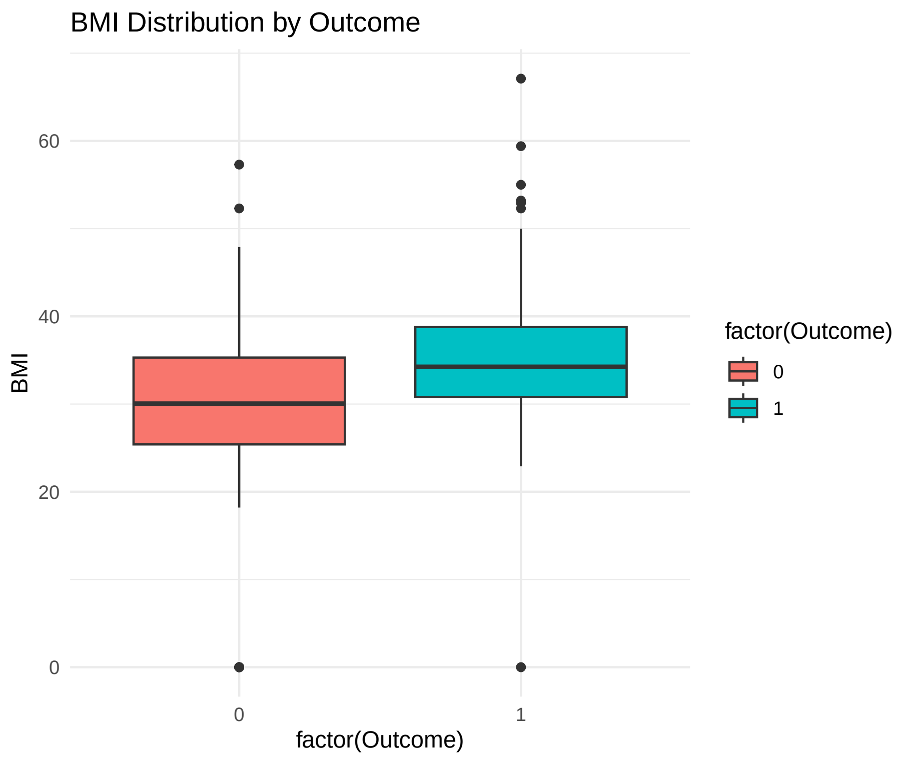
```r
ggplot(diabetes_subset, aes(x = Pregnancies, fill = factor(Outcome))) +
  geom_histogram(position = "identity", bins = 30, alpha = 0.7) +
  labs(title = "Distribution of Pregnancies by Outcome") +
  facet_wrap(~Outcome, scales = "free_y") +
  theme_minimal()
```



5 Conclusion from the Histogram:

The faceted histogram compares the number of pregnancies for individuals with (Outcome = 1) and without (Outcome = 0) diabetes. Outcome = 0 (Non-Diabetic): Most individuals had 0 to 2 pregnancies, with the highest frequency around 0.The distribution is right-skewed, indicating fewer individuals with many pregnancies. Outcome = 1 (Diabetic): The number of pregnancies is more evenly spread, with a higher proportion of diabetic individuals having 3 or more pregnancies. This suggests a trend where individuals with more pregnancies are more likely to be diabetic.

```r
library(ggplot2)

# Boxplot
ggplot(data, aes(x = factor(Outcome), y = BMI, fill = factor(Outcome))) +
  geom_boxplot() +
  labs(title = "BMI Distribution by Outcome") +
  theme_minimal()
```



6 Conclusion from the Boxplot

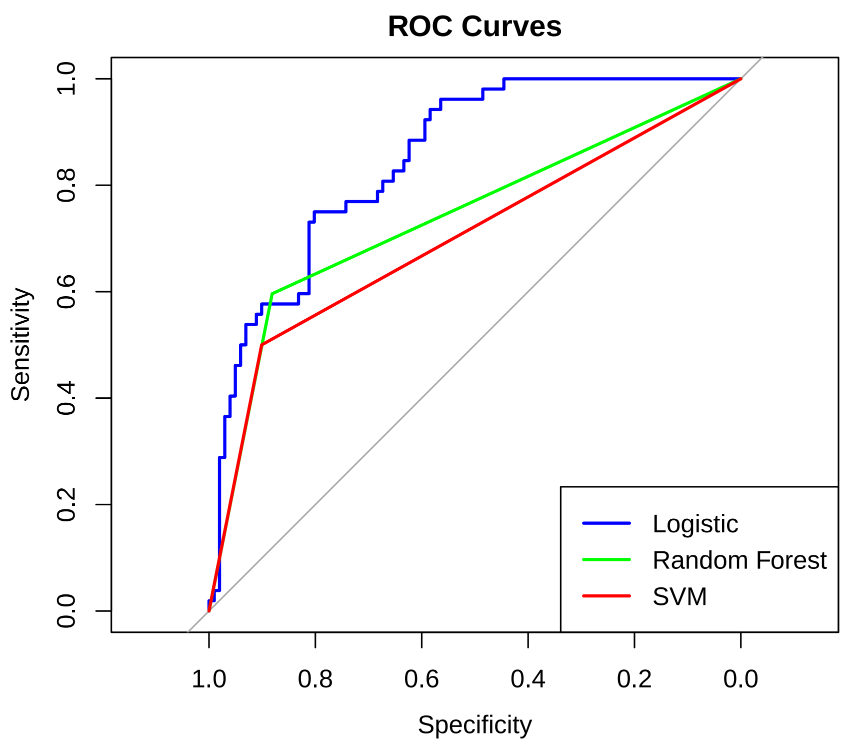
The box plot compares the Body Mass Index (BMI) distribution between individuals with and without diabetes: Outcome = 0 (Non-Diabetic): The BMI values are relatively lower on average.The median BMI is lower compared to the diabetic group.There's some spread, but fewer extreme outliers. Outcome = 1 (Diabetic): The BMI values tend to be higher overall.The median BMI is noticeably higher than that of non-diabetics.A wider interquartile range (IQR) suggests more variability in BMI among diabetics. A few outliers are present on the higher end, showing some individuals with very high BMI. Individuals diagnosed with diabetes generally have a higher BMI, suggesting a positive association between obesity and diabetes risk.BMI could be an important predictor when building a classification model for diabetes.

```
# Normalize
preProc <- preProcess(data[, -ncol(data)], method = c("center", "scale"))
data_norm <- predict(preProc, data)

# Add target
data_norm$Outcome <- data$Outcome

# Split
set.seed(123)
index <- createDataPartition(data_norm$Outcome, p = 0.8, list = FALSE)
train <- data_norm[index, ]
test <- data_norm[-index, ]
model_log <- glm(Outcome ~ ., data = train, family = "binomial")
pred_log <- predict(model_log, test, type = "response")
pred_class_log <- ifelse(pred_log > 0.5, 1, 0)
confusionMatrix(factor(pred_class_log), factor(test$Outcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 91 22
##          1 10 30
##
##                Accuracy : 0.7908
##                  95% CI : (0.7178, 0.8523)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 0.0002786
##
##                   Kappa : 0.5063
##
##  Mcnemar's Test P-Value : 0.0518299
##
##             Sensitivity : 0.9010
##             Specificity : 0.5769
##          Pos Pred Value : 0.8053
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6601
##          Detection Rate : 0.5948
##    Detection Prevalence : 0.7386
##       Balanced Accuracy : 0.7390
##
##        'Positive' Class : 0
```

```
##

library(randomForest)
model_rf <- randomForest(factor(Outcome) ~ ., data = train, ntree = 100)
pred_rf <- predict(model_rf, test)
confusionMatrix(pred_rf, factor(test$Outcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 89 21
##          1 12 31
##
##                Accuracy : 0.7843
##                  95% CI : (0.7106, 0.8466)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 0.0005461
##
##                   Kappa : 0.4983
##
##  Mcnemar's Test P-Value : 0.1637344
##
##             Sensitivity : 0.8812
##             Specificity : 0.5962
##          Pos Pred Value : 0.8091
##          Neg Pred Value : 0.7209
##              Prevalence : 0.6601
##          Detection Rate : 0.5817
##    Detection Prevalence : 0.7190
##       Balanced Accuracy : 0.7387
##
##        'Positive' Class : 0
##

library(e1071)
model_svm <- svm(factor(Outcome) ~ ., data = train, kernel = "radial")
pred_svm <- predict(model_svm, test)
confusionMatrix(pred_svm, factor(test$Outcome))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 91 26
##          1 10 26
```

```
##
##                Accuracy : 0.7647
##                  95% CI : (0.6894, 0.8294)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 0.003318
##
##                   Kappa : 0.4333
##
##  Mcnemar's Test P-Value : 0.012419
##
##             Sensitivity : 0.9010
##             Specificity : 0.5000
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7222
##              Prevalence : 0.6601
##          Detection Rate : 0.5948
##    Detection Prevalence : 0.7647
##       Balanced Accuracy : 0.7005
##
##        'Positive' Class : 0
##
```

```
library(pROC)

roc_log <- roc(test$Outcome, pred_log)
roc_rf <- roc(test$Outcome, as.numeric(pred_rf))
roc_svm <- roc(test$Outcome, as.numeric(pred_svm))

plot(roc_log, col = "blue", main = "ROC Curves")
lines(roc_rf, col = "green")
lines(roc_svm, col = "red")
legend("bottomright", legend=c("Logistic", "Random Forest", "SVM"), col=c("blue", "green", '
```



```
# AUC values
auc_log <- auc(roc_log)
auc_rf <- auc(roc_rf)
auc_svm <- auc(roc_svm)

# Print AUC values
auc_log

## Area under the curve: 0.8492

auc_rf

## Area under the curve: 0.7387

auc_svm

## Area under the curve: 0.7005

cat("AUC Values:\n")

## AUC Values:

cat("Logistic Regression:", round(auc_log, 4), "\n")
```

```
## Logistic Regression: 0.8492

cat("Random Forest:", round(auc_rf, 4), "\n")

## Random Forest: 0.7387

cat("SVM:", round(auc_svm, 4), "\n")

## SVM: 0.7005
```

7 conclusion

Logistic regression has the best discriminatory power among the three models, with an AUC of 0.8492, indicating good performance in distinguishing between diabetic and nondiabetic individuals.Random Forest has a fair performance with an AUC of 0.7387. While it performs decently, it is clearly outperformed by logistic regression in this case.SVM (Radial Kernel) shows the lowest AUC, 0.7005, which is still considered fair, but suggests that it struggles more than the other models in making accurate classifications. Despite Random Forest and SVM sometimes being more flexible and powerful in complex settings, in this case, simple logistic regression outperforms both, likely because: The data are relatively well behaved for linear separation after normalization.Logistic regression is well-suited for this binary classification problem.

```
##################
# Split data into X and y
X <- data[, 1:8]
y <- data[, 9]

# Scale only the features (X)
scaled_X <- as.data.frame(scale(X))

# Bind scaled X with y
scaled_data <- cbind(scaled_X, y)


# Split scaled data into X and y
X <- scaled_data[, 1:8]
y <- scaled_data[, 9]

# Split X and y into training and testing sets
set.seed(123)
sample <- sample.split(y, SplitRatio = 0.7)
X_train <- X[sample == TRUE, ]
y_train <- y[sample == TRUE]
```

```r
X_test <- X[sample == FALSE, ]
y_test <- y[sample == FALSE]
```

8 Conclusion:

The dataset is first divided into two parts the feature matrix X and the target variable y. The feature matrix X includes the first eight columns, while the target variable y corresponds to the ninth column, often labeled as Outcome. Next, the feature values are standardized this means they are transformed so that each feature has a mean of 0 and a standard deviation of 1. This normalization step ensures that all features contribute equally to the model and prevents variables with larger scales from dominating the learning process.
After scaling, the transformed features (let's call them `scaled_X`) are recombined with the original target variable `y` to form a new dataset called `scaled_data`. This preserves the association between inputs and outputs. Finally, the dataset is split into training and testing subsets using a 70:30 ratio. Specifically, 70% of the data is allocated to training (X_{train}, y_{train}) and the remaining 30% to testing (X_{test}, y_{test}). A random seed (e.g., `set.seed(123)`) is set to ensure the splitting process is reproducible.

```r
log_model <- glm(y_train ~ ., data = X_train, family = binomial)
# Make predictions
predictions <- predict(log_model, newdata = X_test, type = "response")

# Convert predicted outcome to factor with levels matching actual outcome
predictions <- factor(ifelse(predictions > 0.5, 1, 0),
                      levels = levels(as.factor(y_test)))

# Generate confusion matrix
confusionMatrix(predictions, as.factor(y_test))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 127  37
##          1  23  43
##
##                Accuracy : 0.7391
##                  95% CI : (0.6773, 0.7946)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 0.002949
##
##                   Kappa : 0.4005
```

```
##
##  Mcnemar's Test P-Value : 0.093290
##
##             Sensitivity : 0.8467
##             Specificity : 0.5375
##          Pos Pred Value : 0.7744
##          Neg Pred Value : 0.6515
##              Prevalence : 0.6522
##          Detection Rate : 0.5522
##    Detection Prevalence : 0.7130
##       Balanced Accuracy : 0.6921
##
##        'Positive' Class : 0
##
```

9 Conclusion

This code trains a logistic regression model (`log_model`) using the training data (X_{train} and Y_{train}). The formula $y_{train} \sim .$ indicates that the target variable (y_{train}) is modeled as a function of all the features in the training data (X_{train}). The `family = binomial` argument specifies that the logistic regression model is employed for binary classification, where the target variable follows a binomial distribution. Based on the input features, the model calculates the probability of the target variable belonging to a particular class.

The code then applies the trained logistic regression model (`log_model`) to the test data (X_{test}) to generate predictions.

By setting `type = "response"`, the model returns the predicted probabilities for the positive class. These probabilities are then converted into binary predictions, with values greater than 0.5 classified as positive and values less than or equal to 0.5 classified as negative. A confusion matrix is then created to evaluate the model's performance by comparing the predicted labels with the actual test labels (y_{test}).

- **Accuracy**: The percentage of correctly classified instances, representing the model's overall performance. In this case, the accuracy is 73.91%.
- **No Information Rate (NIR)**: The precision achieved by always predicting the most common class. The NIR here is 65.22%.
- **P-Value**: The p-value from a statistical test that evaluates whether the model's accuracy significantly exceeds the NIR. When the p-value is less than 0.05, it indicates that the model performs substantially better than the NIR. In this case, the p-value is 0.002949, indicating that the model's accuracy is statistically superior to the NIR.

- **Kappa**: A measure of agreement between the predicted and actual classes, with a value of 0.4005, indicating moderate agreement (values range from -1 to 1).
- **McNemar's Test P-Value**: A p-value of 0.093290, indicating that the difference between the predicted and actual classifications is not statistically significant.
- **Sensitivity**: The proportion of true positive cases correctly identified by the model, which is 84.67%.
- **Specificity**: The percentage of true negative cases correctly identified, which is 53.75%.
- **Positive Predictive Value (PPV)**: The percentage of predicted positive cases that are actually positive, at 77.44%.
- **Negative Predictive Value (NPV)**: The percentage of predicted negative cases that are truly negative, which is 65.15%.
- **Prevalence**: The proportion of positive cases in the dataset, at 65.22%.
- **Detection Rate**: The percentage of positive cases correctly identified by the model, irrespective of class imbalance, which is 55.22%.
- **Detection Prevalence**: The percentage of all cases predicted as positive, regardless of the actual class, which is 71.30%.
- **Balanced Accuracy**: The mean of specificity and sensitivity, which equals 69.21%.

```r
predict_diabetes <- function(pregnancies, glucose, bloodpressure, skinthickness,
                             insulin, bmi, diabetespedigreefunction, age) {
  input_data <- data.frame(
    Pregnancies = pregnancies,
    Glucose = glucose,
    BloodPressure = bloodpressure,
    SkinThickness = skinthickness,
    Insulin = insulin,
    BMI = bmi,
    DiabetesPedigreeFunction = diabetespedigreefunction,
    Age = age
  )
  input <- as.data.frame(input_data)
  prediction <- predict(log_model, newdata = input, type = "response")
  prediction <- factor(ifelse(prediction > 0.5, 1, 0),
                       levels = levels(as.factor(prediction)))
  return(prediction)
}

new_patient <- data.frame(
  pregnancies = 6,
  glucose = 148,
  bloodpressure = 72,
  skinthickness = 35,
  insulin = 0,
  bmi = 33.6,
  diabetespedigreefunction = 0.627,
  age = 50
)

prediction <- predict_diabetes(
  new_patient$pregnancies,
  new_patient$glucose,
  new_patient$bloodpressure,
  new_patient$skinthickness,
  new_patient$insulin,
  new_patient$bmi,
  new_patient$diabetespedigreefunction,
  new_patient$age
)

if (any(prediction == 1)) {
  cat("Based on the model's prediction, there is a higher chance of diabetes.")
} else {
  cat("Based on the model's prediction, the risk of diabetes appears lower.")
}

## Based on the model's prediction, there is a higher chance of diabetes.
```

10 output:

Based on the model's prediction, there is a higher chance of diabetes.

11 Conclusion:

This code defines a function, `predict_diabetes`, that predicts whether a new patient has diabetes based on their medical characteristics, such as pregnancy, glucose level, blood pressure, skin thickness, insulin level, BMI, diabetes pedigree function, and age. The function creates a data frame from the input variables and uses a pre-trained logistic regression model (`log_model`) to calculate the probability of diabetes.If the predicted probability is greater than 0.5, the function returns 1 (indicating diabetes); otherwise, it returns 0 (indicating no diabetes). The predicted result is returned as a factor.

A new patient's data is passed to the `predict_diabetes` function, and based on the prediction, a message is displayed: if the prediction is 1, the message indicates a higher chance of diabetes; if 0, the message suggests a lower risk of diabetes. This demonstrates the use of logistic regression for binary classification in predicting diabetes and interpreting the model's predictions to assess the patient's health status.