Statistical Learning Lab

Assignment - 3

LDA, QDA and KNN Assignment

NAME : SUNNY KUMAR ROLL NO: 22IM10040

Show the code snippets and the corresponding output for the following:

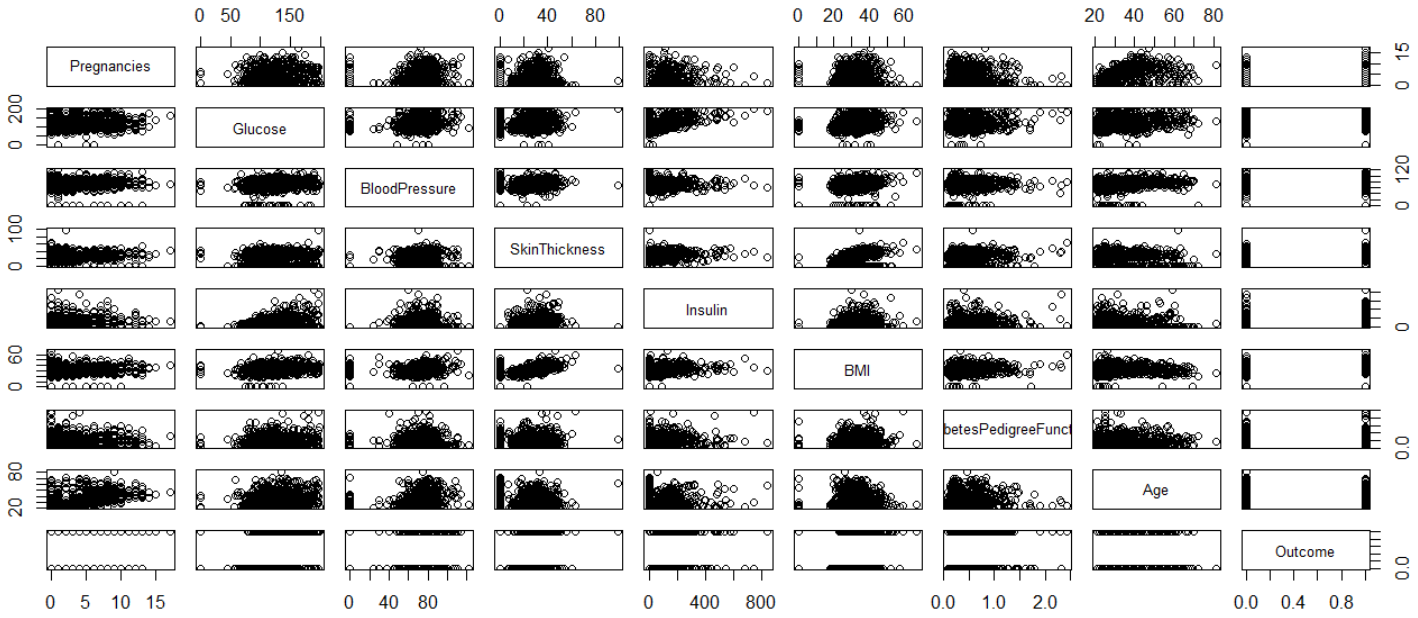
1. **Load the dataset "diabetes.csv". Display first few rows of the dataset.**

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

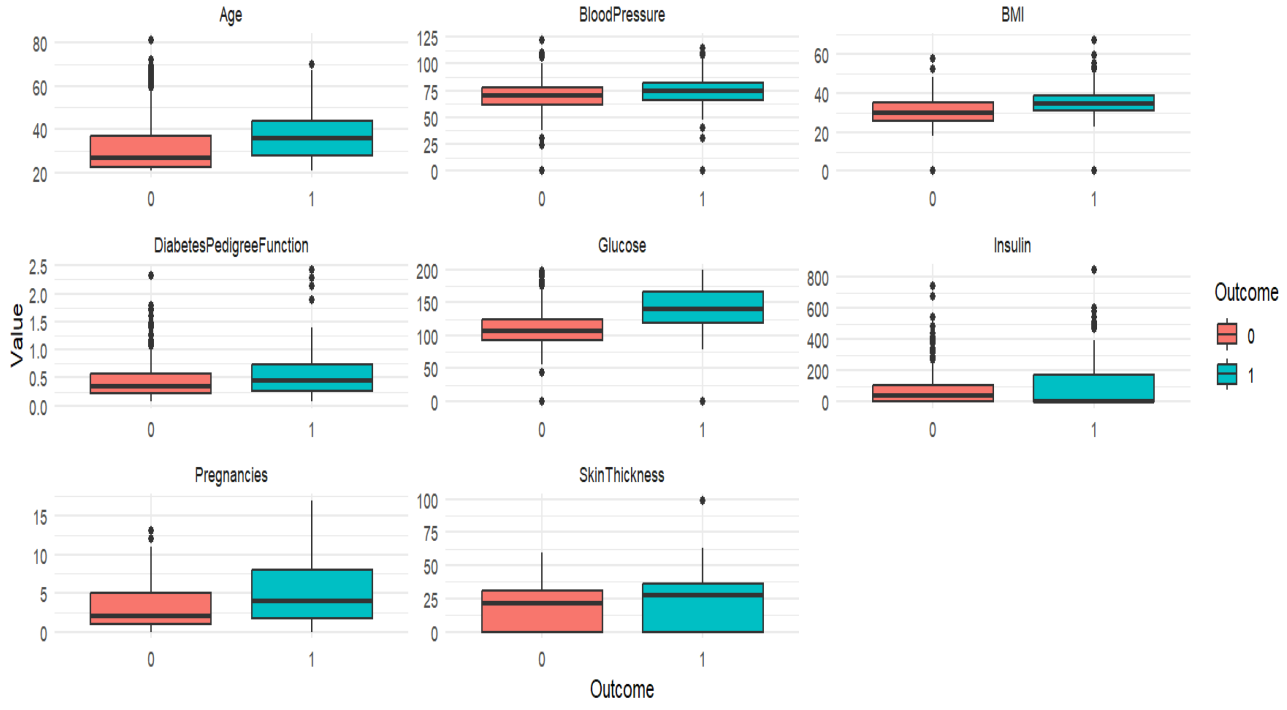
2. **Perform preliminary analysis to show how the variables are related to each other. Use scatter plot, box plot etc. to visualize how different variables impact the "Outcome" variable.**

```
> str(df)
'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 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
> sum(is.na(df))
[1] 0
> dim(df)
[1] 768   9
```

Scatter plot between different variables is given below:



My Inference : The median BMI for individuals with Outcome = 1 (diabetic) is higher than for individuals with Outcome = 0 (non-diabetic). This suggests that diabetics tend to have a higher BMI on average. BMI appears to have a relationship with the Outcome variable. Higher BMI values are more associated with diabetes (Outcome = 1). Similarly, All the plots are shown below.

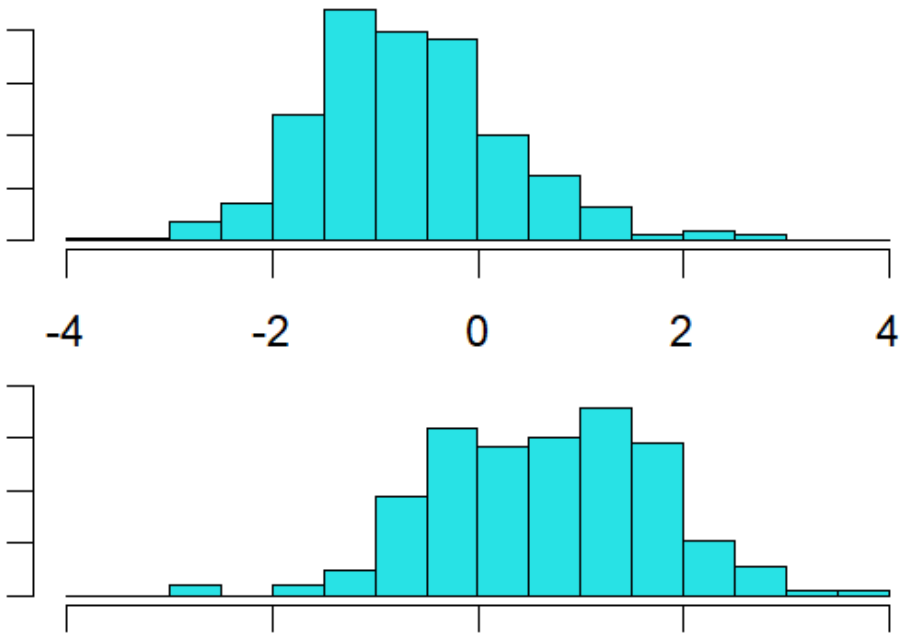


3. Randomly sample 80% of the data as training data and rest as test data. Fit a LDA model and interpret the result.

Code and corresponding output :

```
plot(df)
df_index = sample(nrow(df),615)
df_train = df[df_index,]
df_test = df[-df_index,]
library(ISLR)
library(ggplot2)
library(MASS)
attach(df)
lda_fit1 <- lda(Outcome ~ ., family = binomial , data = df_train)
lda_fit1
```

```
> lda_fit1
Call:
lda(Outcome ~ ., data = df_train, family = binomial)

Prior probabilities of groups:
        0         1
0.6536585 0.3463415

Group means:
  Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI DiabetesPedigreeFunction      Age
0    3.276119 110.2264      68.03483      19.47015  70.8806 30.32438                0.4460000 31.51244
1    4.826291 141.3850      71.38967      21.92488 106.6901 35.13146                0.5513615 37.16432

Coefficients of linear discriminants:
                                LD1
Pregnancies               0.0964983927
Glucose                   0.0279329606
BloodPressure            -0.0090709365
SkinThickness             0.0027881850
Insulin                  -0.0008403882
BMI                       0.0562163963
DiabetesPedigreeFunction  0.4879746563
Age                       0.0087153068
> plot(lda_fit1)
```



Interpretation from the result :

Diabetics (Outcome = 1) tend to have:

- Higher **Pregnancies**
- Higher **Glucose levels** (a strong indicator)
- Higher **BMI**

4. From the model fitted in problem 3, derive confusion matrix, accuracy, and F1-score on test data.

   Code:

```
predictions <- predict(lda_model, test_diabetes)
predicted_classes <- predictions$class
actual_classes <- test_diabetes$Outcome

# Confusion Matrix
conf_matrix <- table(Predicted = predicted_classes, Actual = actual_classes)
conf_matrix

TP <- conf_matrix[2,2]
TN <- conf_matrix[1,1]
FP <- conf_matrix[2,1]
FN <- conf_matrix[1,2]

# Accuracy
accuracy <- (TP + TN) / sum(conf_matrix)
accuracy

# F1-score
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)

f1_score <- 2 * (precision * recall) / (precision + recall)
f1_score
```

   Output:

```
> predictions <- predict(lda_model, test_diabetes)
> predicted_classes <- predictions$class
> actual_classes <- test_diabetes$Outcome
> # Confusion Matrix
> conf_matrix <- table(Predicted = predicted_classes, Actual = actual_classes)
> conf_matrix
         Actual
Predicted  0  1
        0 94 19
        1 12 29
> TP <- conf_matrix[2,2]
> TN <- conf_matrix[1,1]
> FP <- conf_matrix[2,1]
> FN <- conf_matrix[1,2]
> # Accuracy
> accuracy <- (TP + TN) / sum(conf_matrix)
> accuracy
[1] 0.7987013
> # F1-score
> precision <- TP / (TP + FP)
> recall <- TP / (TP + FN)
> f1_score <- 2 * (precision * recall) / (precision + recall)
> f1_score
[1] 0.6516854
```

5. Fit QDA and KNN (K = 5) models on training data. Compare the metrics in problem 4 for LDA, QDA and KNN models for test data and discuss the results.

   Code :

```r
library(MASS)      # For LDA & QDA
library(class)     # For KNN
library(caret)     # For confusion matrix, accuracy, and F1-score

# QDA Model
qda_model <- qda(Outcome ~ ., data = train_diabetes)
qda_predictions <- predict(qda_model, test_diabetes)$class

# KNN Model (K = 5)
train_scaled <- scale(train_diabetes[, -ncol(train_diabetes)])
test_scaled <- scale(test_diabetes[, -ncol(test_diabetes)])

knn_predictions <- knn(train = train_scaled, test = test_scaled, cl = train_diabetes$Outcome, k = 5)

# Function to evaluate Models
evaluate_model <- function(actual, predicted, model_name) {
  conf_matrix <- confusionMatrix(as.factor(predicted), as.factor(actual))

  cat("\n=== Model:", model_name, "===\n")
  print(conf_matrix$table)
  print(conf_matrix$overall["Accuracy"])

  precision <- conf_matrix$byClass["Precision"]
  recall <- conf_matrix$byClass["Recall"]
  f1_score <- 2 * (precision * recall) / (precision + recall)
  print(paste("F1-score:", round(f1_score, 4)))
}

# Evaluate LDA
evaluate_model(test_diabetes$Outcome, predictions$class, "LDA")

# Evaluate QDA
evaluate_model(test_diabetes$Outcome, qda_predictions, "QDA")

# Evaluate KNN
evaluate_model(test_diabetes$Outcome, knn_predictions, "KNN (K=5)")
```

   Output:

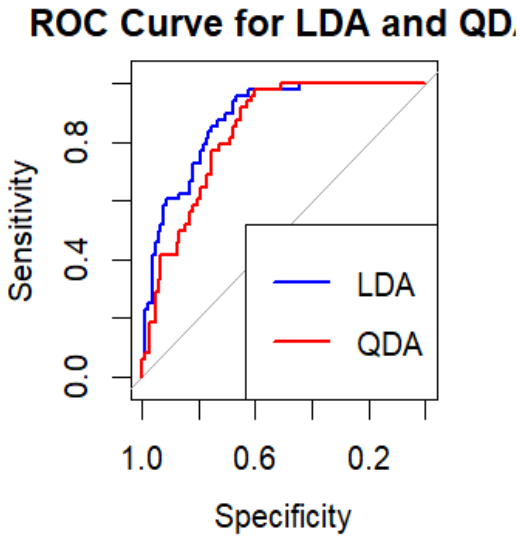
```
=== Model: LDA ===
          Reference
Prediction  0  1
         0 94 19
         1 12 29
 Accuracy
0.7987013
[1] "F1-score: 0.8584"
> # Evaluate QDA
> evaluate_model(test_diabetes$Outcome, qda_predictions, "QDA")

=== Model: QDA ===
          Reference
Prediction  0  1
         0 87 21
         1 19 27
 Accuracy
0.7402597
[1] "F1-score: 0.8131"
> # Evaluate KNN
> evaluate_model(test_diabetes$Outcome, knn_predictions, "KNN (K=5)")

=== Model: KNN (K=5) ===
          Reference
Prediction  0  1
         0 88 24
         1 18 24
 Accuracy
0.7272727
[1] "F1-score: 0.8073"
```

6. Plot ROC curve for LDA and QDA models using the test data.

Code:

```
library(pROC)

lda_probs <- predict(lda_model, test_diabetes)$posterior[,2]

qda_probs <- predict(qda_model, test_diabetes)$posterior[,2]

lda_roc <- roc(test_diabetes$Outcome, lda_probs)
qda_roc <- roc(test_diabetes$Outcome, qda_probs)

plot(lda_roc, col="blue", main="ROC Curve for LDA and QDA", lwd=2)
plot(qda_roc, col="red", add=TRUE, lwd=2)

legend("bottomright", legend=c("LDA", "QDA"), col=c("blue", "red"), lwd=2)
```

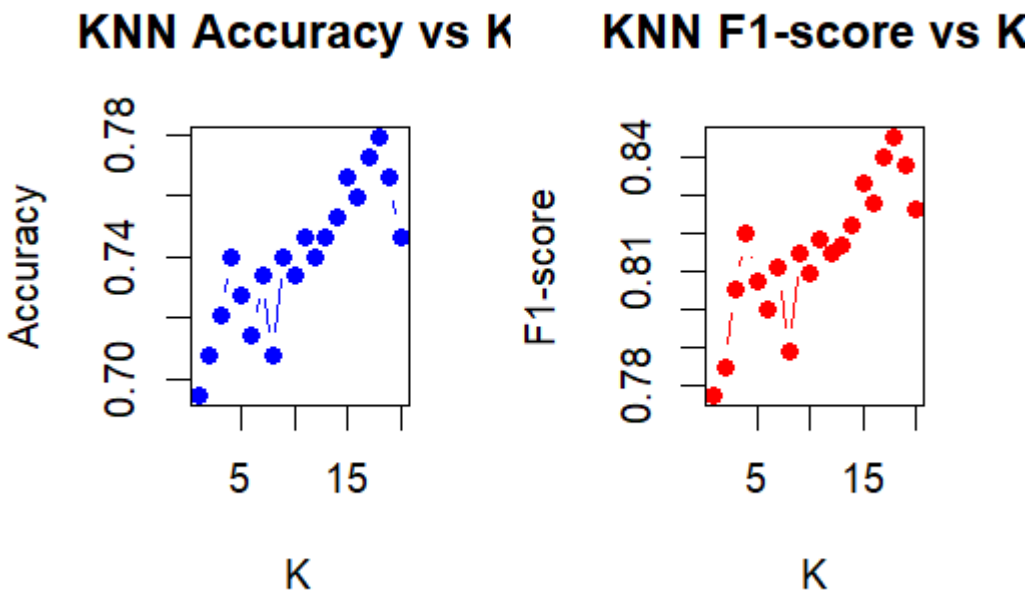
Output:



7. Plot accuracy and f1-score by varying the neighbourhood size from K=1 to K=20 and interpret the results.

```
# Vectors to store accuracy and F1-score
k_values <- 1:20
accuracy_values <- numeric(length(k_values))
f1_values <- numeric(length(k_values))

# Function to compute F1-score
compute_f1 <- function(conf_matrix) {
  precision <- conf_matrix$byClass["Precision"]
  recall <- conf_matrix$byClass["Recall"]
  f1_score <- 2 * (precision * recall) / (precision + recall)
  return(f1_score)
}

for (i in seq_along(k_values)) {
  k <- k_values[i]

  knn_preds <- knn(train = train_scaled, test = test_scaled, cl = train_diabetes$Outcome, k = k)

  # Confusion matrix
  conf_matrix <- confusionMatrix(as.factor(knn_preds), as.factor(test_diabetes$Outcome))

  # Accuracy and F1-score
  accuracy_values[i] <- conf_matrix$overall["Accuracy"]
  f1_values[i] <- compute_f1(conf_matrix)
}

par(mfrow=c(1,2))

plot(k_values, accuracy_values, type="b", col="blue", pch=19, xlab="K", ylab="Accuracy", main="KNN Accuracy vs K")
plot(k_values, f1_values, type="b", col="red", pch=19, xlab="K", ylab="F1-score", main="KNN F1-score vs K")
```

Output:



Interpretation:

- For smaller K values (e.g., K = 1 to 5), the model overfits to the training data, resulting in lower accuracy and F1-score. Also, predictions are more sensitive to noise.
- For higher K values (e.g., K = 15 to 20), accuracy and F1-score stabilize, indicating that the model generalizes well. The performance is better balanced, meaning less overfitting and better predictions.
- The best value of K is likely between 15 and 20, where both accuracy and F1-score are highest.
- Choosing K too high (e.g., K > 20) may start to decrease performance due to underfitting.

Data can be downloaded from:

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Description of the study:

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988, November). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care* (p. 261). American Medical Informatics Association.