Statistical Learning Lab

Assignment – 3

LDA, QDA and KNN

Name: Semanti Ghosh

Roll No.: 22IM10036

Loading the dataset

Code snippet

```
#changing the directory
setwd("C:/Study/Semester_6/Statistical_Learning_Lab")
getwd()

#loading the dataset
data <- read.csv("diabetes.csv")
head(data)
```

Output

```
> #changing the directory
> setwd("C:/Study/Semester_6/Statistical_Learning_Lab")
> getwd()
[1] "C:/Study/Semester_6/Statistical_Learning_Lab"
> #loading the dataset
> data <- read.csv("diabetes.csv")
> head(data)
  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
```

Analysing the data (and finding the correlation between the parameters)

Code snippet

```
#obtaining a summary about the different parameters as well as the outcome
summary(data)

#obtaining and printing the correlation matrix
corr_matrix <- cor(data)
print(corr_matrix)
```

Output

```
> summary(data)
  Pregnancies        Glucose      BloodPressure    SkinThickness      Insulin          BMI        DiabetesPedigreeFunction      Age
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   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
> #obtaining and printing the correlation matrix
> corr_matrix <- cor(data)
> print(corr_matrix)
                          Pregnancies    Glucose BloodPressure SkinThickness     Insulin        BMI DiabetesPedigreeFunction         Age
Pregnancies                1.00000000 0.12945867    0.14128198   -0.08167177 -0.07353461 0.01768309              -0.03352267  0.54434123
Glucose                    0.12945867 1.00000000    0.15258959    0.05732789  0.33135711 0.22107107               0.13733730  0.26351432
BloodPressure              0.14128198 0.15258959    1.00000000    0.20737054  0.08893338 0.28180529               0.04126495  0.23952795
SkinThickness             -0.08167177 0.05732789    0.20737054    1.00000000  0.43678257 0.39257320               0.18392757 -0.11397026
Insulin                   -0.07353461 0.33135711    0.08893338    0.43678257  1.00000000 0.19785906               0.18507093 -0.04216295
BMI                        0.01768309 0.22107107    0.28180529    0.39257320  0.19785906 1.00000000               0.14064695  0.03624187
DiabetesPedigreeFunction  -0.03352267 0.13733730    0.04126495    0.18392757  0.18507093 0.14064695               1.00000000  0.03356131
Age                        0.54434123 0.26351432    0.23952795   -0.11397026 -0.04216295 0.03624187               0.03356131  1.00000000
Outcome                    0.22189815 0.46658140    0.06506836    0.07475223  0.13054795 0.29269466               0.17384407  0.23835598
                            Outcome
Pregnancies              0.22189815
Glucose                  0.46658140
BloodPressure            0.06506836
SkinThickness            0.07475223
Insulin                  0.13054795
BMI                      0.29269466
DiabetesPedigreeFunction 0.17384407
Age                      0.23835598
Outcome                  1.00000000
```

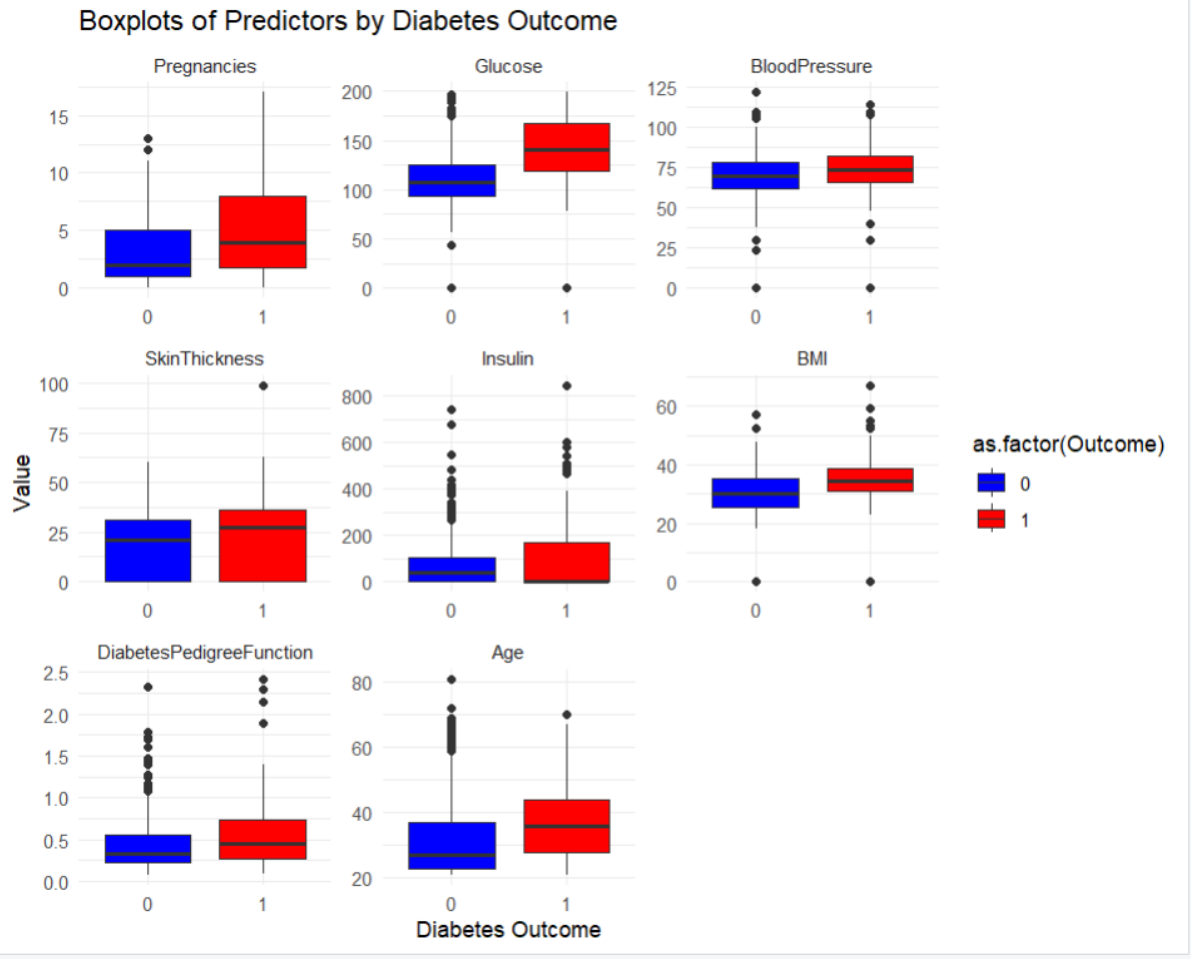
Obtaining the boxplot

Code snippet

```r
#obtaining the box plot
install.packages("reshape2")
library(reshape2) #this library will plot box plots for all parameters at one
library(ggplot2)

data_long <- melt(data, id.vars = "Outcome")
ggplot(data_long, aes(x = as.factor(Outcome), y = value, fill = as.factor(Outcome))) +
  geom_boxplot() +
  facet_wrap(~variable, scales = "free") +
  theme_minimal() +
  labs(title = "Boxplots of Predictors by Diabetes Outcome",
       x = "Diabetes Outcome",
       y = "Value") +
  scale_fill_manual(values = c("blue", "red"))
```

Output

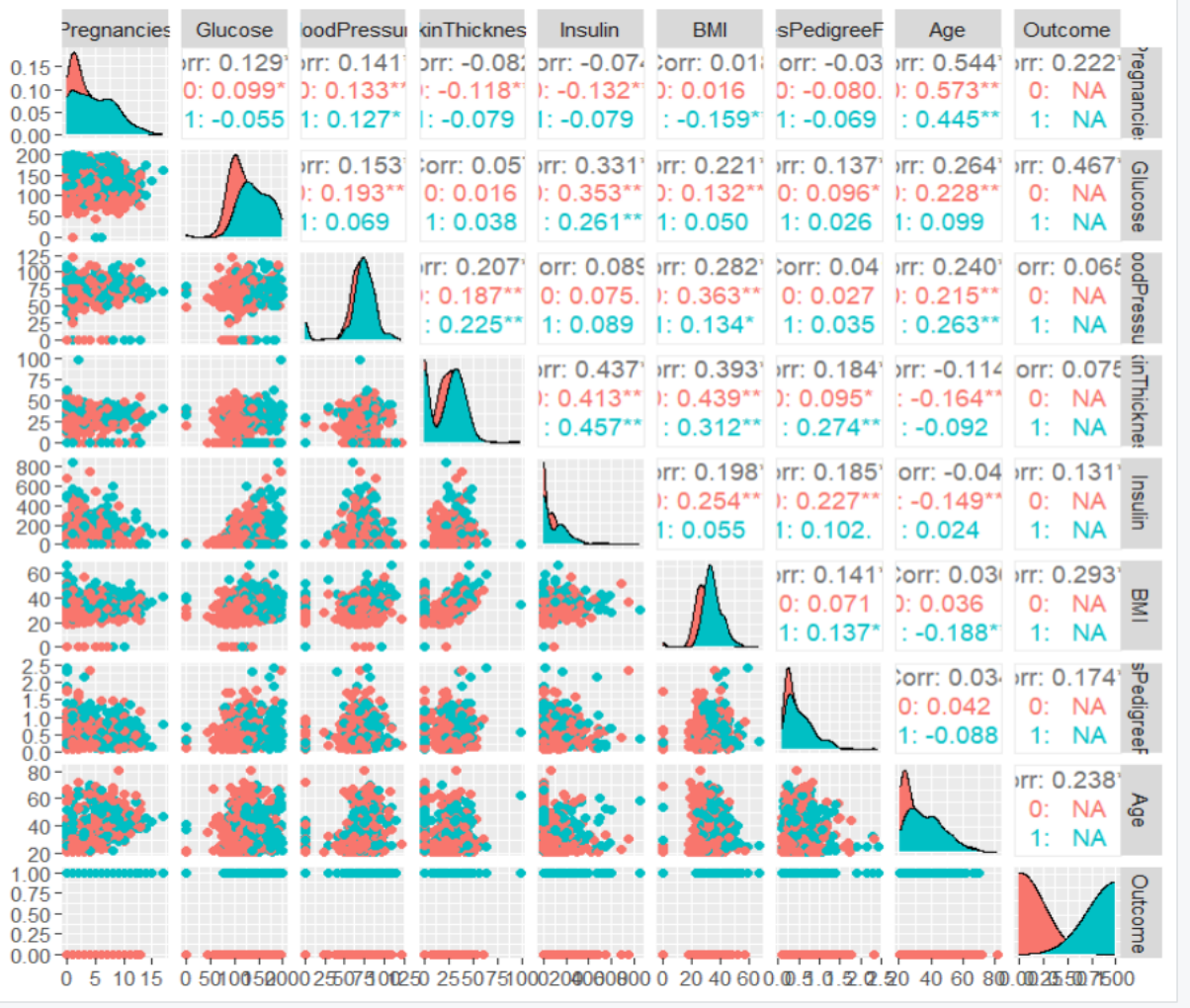


Obtaining the scatter plot

Code snippet

```
#obtaining the scatter plot
install.packages("GGally") # to get all scatter plots at once
library(GGally)
ggpairs(data, aes(color = as.factor(Outcome)))
```

Output



Splitting into train and test set and fitting an LDA

Code snippet

```
#splitting into train and test data
set.seed(97)
sample.size <- floor(0.8*nrow(data))
train.indices <- sample(seq_len(nrow(data)), size = sample.size)
train.data <- data[train.indices, ]
test.data <- data[-train.indices, ]

#fitting an lda
lda.model <- lda(Outcome ~ ., data=train.data)
print(lda.model)
```

Output

```
> library(MASS)
> #splitting into train and test data
> set.seed(97)
> sample.size <- floor(0.8*nrow(data))
> train.indices <- sample(seq_len(nrow(data)), size = sample.size)
> train.data <- data[train.indices, ]
> test.data <- data[-train.indices, ]
> lda.model <- lda(Outcome ~ ., data=train.data)
> print(lda.model)
Call:
lda(Outcome ~ ., data = train.data)

Prior probabilities of groups:
        0         1
0.6465798 0.3534202

Group means:
  Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI DiabetesPedigreeFunction      Age
0    3.342569 110.7683      69.08312      19.88917 70.13854 30.46826                0.4204811 31.58438
1    4.912442 141.0369      70.45622      22.08756 99.13364 35.00230                0.5504562 37.19816

Coefficients of linear discriminants:
                              LD1
Pregnancies           0.114541888
Glucose               0.027089454
BloodPressure        -0.011745197
SkinThickness         0.001084467
Insulin              -0.001059625
BMI                   0.057395953
DiabetesPedigreeFunction  0.749747023
Age                   0.006975956
```

Predicting the labels for the test data using the model fitted

Code snippet

```
#predicting for the test data from the lda model fitted
lda.pred <- predict(lda.model, test.data)

# Printing few predictions as well as the posterior probabilities for each class
head(lda.pred$class)
head(lda.pred$posterior)
```

Output

```
> #predicting for the test data from the lda model fitted
> lda.pred <- predict(lda.model, test.data)
> # Printing few predictions as well as the posterior probabilities for each class
> head(lda.pred$class)
[1] 0 1 1 0 1 0
Levels: 0 1
> head(lda.pred$posterior)
           0          1
7  0.93478101 0.06521899
8  0.28976325 0.71023675
15 0.36508145 0.63491855
16 0.53637133 0.46362867
23 0.05989864 0.94010136
33 0.94809500 0.05190500
>
```

Deriving the confusion matrix, accuracy, F1-score on the test data (for LDA)

Code snippet

```
#getting the confusion matrix
library(caret)

#converted them to factor because they had to be of the same "level"
test.data$Outcome <- as.factor(test.data$Outcome)
lda.pred$class <- as.factor(lda.pred$class)

#Confusion matrix
conf_mat <- confusionMatrix(lda.pred$class, test.data$Outcome)
print(conf_mat)

#extracting accuracy from the confusion matrix
accuracy <- conf_mat$overall["Accuracy"]
print(accuracy)

#obtaining the precision and recall (we need them for the confusion matrix)
precision <- as.numeric(conf_mat$byClass["Precision"])
recall <- as.numeric(conf_mat$byClass["Recall"])

# Compute F1-score
F1_score <- 2 * (precision * recall) / (precision + recall)
print(F1_score)
```

Output

```
> print(conf_mat)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 95 23
         1  8 28

               Accuracy : 0.7987
                 95% CI : (0.7266, 0.8589)
    No Information Rate : 0.6688
    P-Value [Acc > NIR] : 0.0002606

                  Kappa : 0.5092

 Mcnemar's Test P-Value : 0.0119210

            Sensitivity : 0.9223
            Specificity : 0.5490
         Pos Pred Value : 0.8051
         Neg Pred Value : 0.7778
             Prevalence : 0.6688
         Detection Rate : 0.6169
   Detection Prevalence : 0.7662
      Balanced Accuracy : 0.7357

       'Positive' Class : 0

> #extracting accuracy from the confusion matrix
> accuracy <- conf_mat$overall["Accuracy"]
> print(accuracy)
 Accuracy
0.7987013
> #obtaining the precision and recall (we need them for the confusion matrix)
> precision <- as.numeric(conf_mat$byClass["Precision"])
> recall <- as.numeric(conf_mat$byClass["Recall"])
> # Compute F1-score
> F1_score <- 2 * (precision * recall) / (precision + recall)
> print(F1_score)
[1] 0.8597285
```

Fitting a QDA model and printing its metrics (confusion matrix, accuracy and F1 score)

Code snippet

```
install.packages("class")
library(class)

#fitting the QDA model and predicting the class on the test data
qda.model <- qda(Outcome ~ ., data = train.data)
print(qda.model)
qda_pred <- predict(qda.model, test.data)

#extracting predicted class labels
qda_class <- qda_pred$class
qda_class <- as.factor(qda_class)

#the confusion matrix, accuracy and F1 score of QDA
qda_conf_matrix <- confusionMatrix(qda_class, test.data$Outcome)
print(qda_conf_matrix)
qda_accuracy <- qda_conf_matrix$overall["Accuracy"]
print(qda_accuracy)
qda_precision <- as.numeric(qda_conf_matrix$byClass["Precision"])
qda_recall <- as.numeric(qda_conf_matrix$byClass["Recall"])
qda_f1_score <- 2 * (qda_precision * qda_recall) / (qda_precision + qda_recall)
print(qda_f1_score)
```

Output

```
> library(class)
> #fitting the QDA model and predicting the class on the test data
> qda.model <- qda(Outcome ~ ., data = train.data)
> print(qda.model)
Call:
qda(Outcome ~ ., data = train.data)

Prior probabilities of groups:
        0         1
0.6465798 0.3534202

Group means:
  Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI DiabetesPedigreeFunction      Age
0    3.342569 110.7683      69.08312      19.88917 70.13854 30.46826                0.4204811 31.58438
1    4.912442 141.0369      70.45622      22.08756 99.13364 35.00230                0.5504562 37.19816
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 88 21
         1 15 30

               Accuracy : 0.7662
                 95% CI : (0.6914, 0.8306)
    No Information Rate : 0.6688
    P-Value [Acc > NIR] : 0.005473

                  Kappa : 0.4562

 Mcnemar's Test P-Value : 0.404657

            Sensitivity : 0.8544
            Specificity : 0.5882
         Pos Pred Value : 0.8073
         Neg Pred Value : 0.6667
             Prevalence : 0.6688
         Detection Rate : 0.5714
   Detection Prevalence : 0.7078
      Balanced Accuracy : 0.7213

       'Positive' Class : 0

> qda_accuracy <- qda_conf_matrix$overall["Accuracy"]
> print(qda_accuracy)
 Accuracy
0.7662338
> qda_precision <- as.numeric(qda_conf_matrix$byClass["Precision"])
> qda_recall <- as.numeric(qda_conf_matrix$byClass["Recall"])
> qda_f1_score <- 2 * (qda_precision * qda_recall) / (qda_precision + qda_recall)
> print(qda_f1_score)
[1] 0.8301887
>
```

Fitting a KNN model with K=5 and printing the necessary metrics

Code snippet

```r
#obtaining the X and y for KNN
X.train <- train.data[, -which(names(train.data) == "Outcome")]
X.test <- test.data[, -which(names(test.data) == "Outcome")]
train.data$Outcome <- as.factor(train.data$Outcome)
test.data$Outcome <- as.factor(test.data$Outcome)
y.train <- train.data$Outcome
y.test <- test.data$Outcome

#KNN with K = 5 (and then printing the result)
knn.pred <- knn(train = X.train, test = X.test, cl = y.train, k = 5)
head(knn.pred)

# Computing confusion matrix, accuracy and F1 score for KNN
knn_conf_matrix <- confusionMatrix(knn.pred, y.test)
print(knn_conf_matrix)
knn_accuracy <- knn_conf_matrix$overall["Accuracy"]
print(knn_accuracy)
knn_precision <- as.numeric(knn_conf_matrix$byClass["Precision"])
knn_recall <- as.numeric(knn_conf_matrix$byClass["Recall"])
knn_f1_score <- 2 * (knn_precision * knn_recall) / (knn_precision + knn_recall)
print(knn_f1_score)
```

Output

```
> #obtaining the X and y for KNN
> X.train <- train.data[, -which(names(train.data) == "Outcome")]
> X.test <- test.data[, -which(names(test.data) == "Outcome")]
> train.data$Outcome <- as.factor(train.data$Outcome)
> test.data$Outcome <- as.factor(test.data$Outcome)
> y.train <- train.data$Outcome
> y.test <- test.data$Outcome
> #KNN with K = 5 (and then printing the result)
> knn.pred <- knn(train = X.train, test = X.test, cl = y.train, k = 5)
> head(knn.pred)
[1] 0 0 1 0 1 0
Levels: 0 1
```

```
> print(knn_conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 85 22
         1 18 29

               Accuracy : 0.7403
                 95% CI : (0.6635, 0.8075)
    No Information Rate : 0.6688
    P-Value [Acc > NIR] : 0.03415

                  Kappa : 0.4018

 Mcnemar's Test P-Value : 0.63526

            Sensitivity : 0.8252
            Specificity : 0.5686
         Pos Pred Value : 0.7944
         Neg Pred Value : 0.6170
             Prevalence : 0.6688
         Detection Rate : 0.5519
   Detection Prevalence : 0.6948
      Balanced Accuracy : 0.6969

       'Positive' Class : 0

> knn_accuracy <- knn_conf_matrix$overall["Accuracy"]
> print(knn_accuracy)
 Accuracy
0.7402597
> knn_precision <- as.numeric(knn_conf_matrix$byClass["Precision"])
> knn_recall <- as.numeric(knn_conf_matrix$byClass["Recall"])
> knn_f1_score <- 2 * (knn_precision * knn_recall) / (knn_precision + knn_recall)
> print(knn_f1_score)
[1] 0.8095238
>
```

Interpretation from the different classifiers fitted (LDA, QDA and KNN)

Both the accuracy and F1 score are maximum in case of LDA and minimum in case of KNN. From this, we can say that the separation boundary is more linear than curved (that's why LDA performs better than QDA). Also, we can say that since KNN performs poorly, the data does not need a highly flexible, non-linear decision boundary. Besides, there are too many parameters for KNN to perform well.

Plotting the ROC curve for LDA and QDA using the test data

Code Snippet

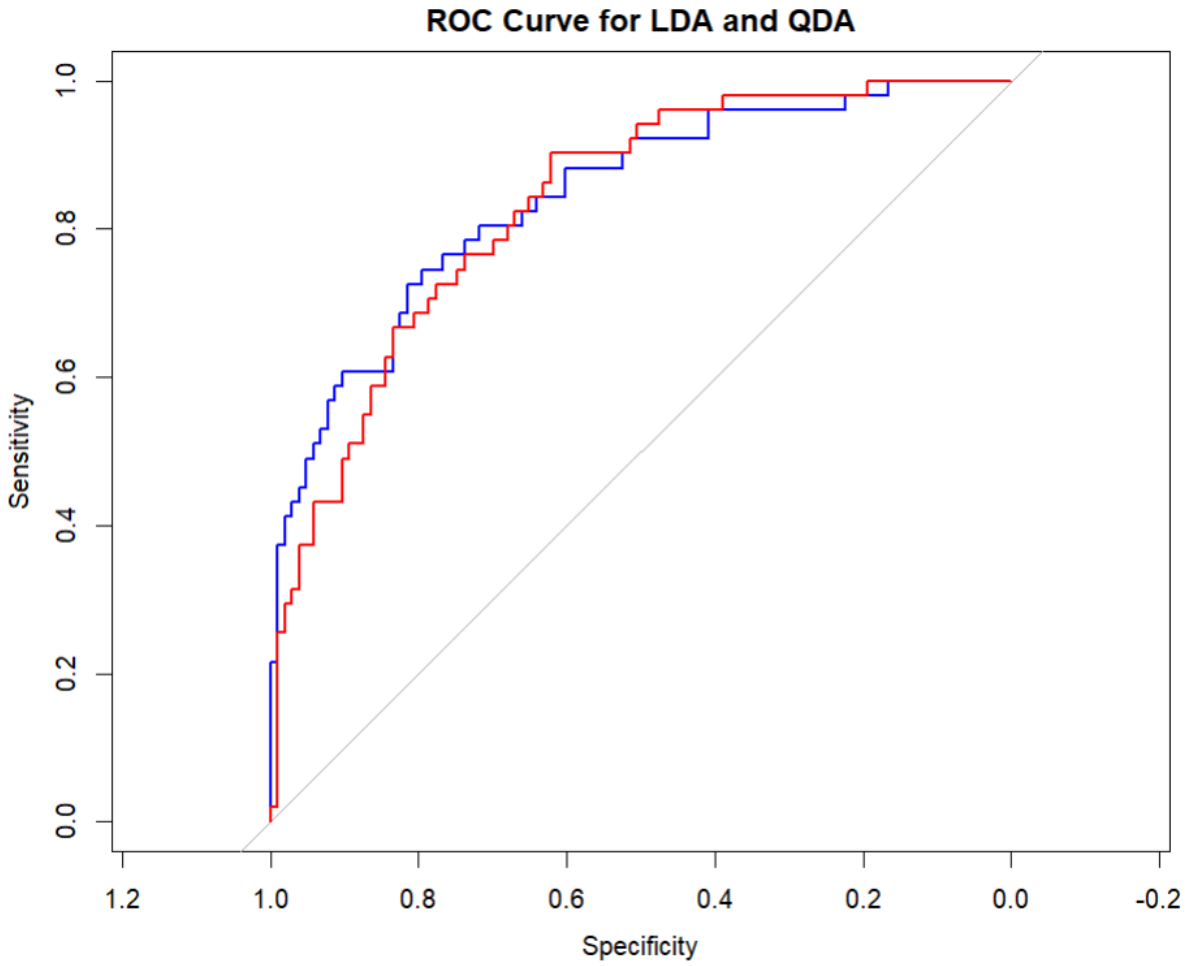
```
#installing packages for the ROC curve
install.packages("pROC")
library(pROC)

#predictions using the LDA and the QDA models (probabilities for having diabetes)
lda_probs <- predict(lda.model, test.data)$posterior[, 2]
qda_probs <- predict(qda.model, test.data)$posterior[, 2]

#Computing the ROC for the LDA and QDA
lda_roc <- roc(test.data$Outcome, lda_probs)
qda_roc <- roc(test.data$Outcome, qda_probs)

# Plotting the ROC curves
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


ROC Curve for LDA and QDA
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
Specificity

Fitting KNN models for k=1 to k=20 and then plotting the accuracy and F1-scores

Code snippet

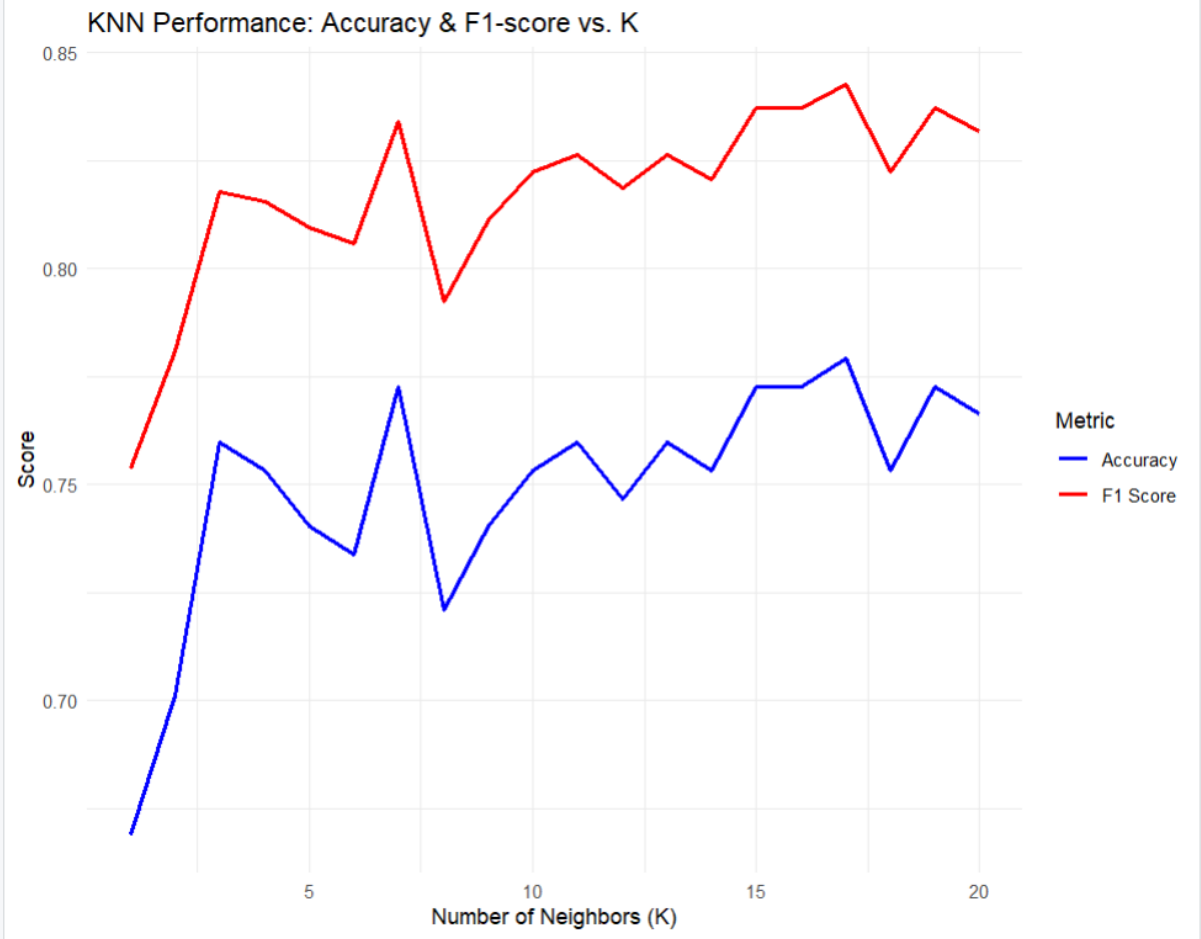
```
#KNN taking k values 1 to 20
k_values <- 1:20 #initializing vector
accuracy_values <- c()
f1_values <- c()

#looping through each value, training KNN model and calculating accuracy and F1 score for each k
for (k in k_values) {
  knn_pred <- knn(train = X.train, test = X.test, cl = y.train, k = k)
  conf_matrix <- confusionMatrix(knn_pred, y.test)
  accuracy <- conf_matrix$overall["Accuracy"]
  accuracy_values <- c(accuracy_values, as.numeric(accuracy))
  precision <- as.numeric(conf_matrix$byClass["Precision"])
  recall <- as.numeric(conf_matrix$byClass["Recall"])
  f1_score <- 2 * (precision * recall) / (precision + recall)
  f1_values <- c(f1_values, f1_score)
}

#Creating a data frame for plotting
results_df <- data.frame(K = k_values, Accuracy = accuracy_values, F1_Score = f1_values)

#Plot the accuracy and F1-score
ggplot(results_df, aes(x = K)) +
  geom_line(aes(y = Accuracy, color = "Accuracy"), size = 1) +
  geom_line(aes(y = F1_Score, color = "F1 Score"), size = 1) +
  labs(title = "KNN Performance: Accuracy & F1-score vs. K",
       x = "Number of Neighbors (K)",
       y = "Score") +
  scale_color_manual(name = "Metric", values = c("Accuracy" = "blue", "F1 Score" = "red")) +
  theme_minimal()
```

Output



Interpreting the results:

- Small K (K=1 to K=3): high variance due to overfitting, sensitive to noise.
- Moderate K (K=5 to K=10): usually a good balance between bias and variance and peak accuracy and F1-score often occur in this range
- Large K (e.g., K>15): Predictions become less flexible, can underfit if they become too generalised