EXPERIMENT NO: 5

AIM: To implement Linear Discriminant Analysis, Quadratic Discriminant Analysis and Naive Bayes algorithms on Stock market Dataset. `

DESCRIPTION:

- **Linear Discriminant Analysis (LDA)** is a supervised learning algorithm used for classification tasks in machine learning. It is a technique used to find a linear combination of features that best separates the classes in a dataset. Quadratic Discriminant Analysis (QDA) is a supervised learning algorithm and extension of linear discriminant analysis. QDA models are designed to be used for classification problems .i.e. when the response variable can be placed into classes or categories. Naive Bayes classifiers (NB) are a collection of classification algorithms based on Bayes Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle ,i.e. every pair of features being classified is independent of each other.

  Formula for LDA :-

  For a two-class problem, LDA calculates the discriminant function as follows:

  For Class 1:

  $$D_1(x) = x \cdot \frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + \log(\pi_1)$$

  For Class 2:

  $$D_2(x) = x \cdot \frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + \log(\pi_2)$$

  Where:

  - $D_1(x)$ and $D_2(x)$ are the discriminant functions for Class 1 and Class 2.
  - x is the feature vector.
  - μ_1 and μ_2 are the means of the features for Class 1 and Class 2.
  - σ^2 is the common variance of the features.
  - π_1 and π_2 are the prior probabilities of Class 1 and Class 2.

- **Quadratic Discriminant Analysis (QDA)** is a supervised machine learning algorithm used for classification tasks. It is a generative model, meaning that it assumes that the data is generated from a known distribution, in this case a multivariate Gaussian distribution. QDA estimates the mean and covariance matrix for each class, and then uses this information to calculate the probability of a new data point belonging to each class. The data point is then assigned to the class with the highest probability.

  Formula for QDA :-

  $$D_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{1}{2}\log|\Sigma_k| + \log(\pi_k)$$

  Where:

  - $D_k(x)$ is the discriminant function for Class k.
  - x is the feature vector.
  - μ_k is the mean vector of the features for Class k.
  - Σ_k is the covariance matrix for Class k.
  - π_k is the prior probability of Class k.

- **Naive Bayes** is a supervised machine learning algorithm used for classification tasks. It is a probabilistic classifier, which means that it predicts on the basis of the probability of an object. Naive Bayes is based on Bayes' theorem, which is a mathematical formula for calculating the probability of an event occurring, given that another event has already occurred. In the context of Naive Bayes, the event we are trying to predict is the class label of a data point, and the events we have already observed are the values of the data point's features.

Steps for Navie Bayes :-

1. Calculate the prior probabilities $P(C_k)$ for each class.
2. Estimate the likelihood $P(x|C_k)$ for each feature given each class. This is often done using probability density functions.
3. For a new observation with features x, calculate the posterior probabilities $P(C_k|x)$ for each class using Bayes' theorem.
4. Assign the observation to the class with the highest posterior probability.

Formula for Navie Bayes :-

$$P(C_k|x) = \frac{P(x|C_k) \cdot P(C_k)}{P(x)}$$

Where:

- $P(C_k|x)$ is the posterior probability of class C_k given the features x.
- $P(x|C_k)$ is the likelihood of observing features x given class C_k.
- $P(C_k)$ is the prior probability of class C_k.
- $P(x)$ is the probability of observing features x.

CODE:

```
# Load required libraries
library(quantmod)
library(MASS)
library(e1071)
library(caret)
library(ggplot2)
library(pROC)
#Fetch the stock market data
symbols <- c("AAPL","MSFT")
#calculate  daily returns
start_date <- "2020-01-01"
end_date <- "2023-01-01"
getSymbols(symbols, from = start_date, to = end_date)
```

OUTPUT:

```
> getSymbols(symbols, from = start_date, to = end_date)
[1] "AAPL" "MSFT"
```

CODE:

```
stock_data <- merge(Ad(AAPL),Ad(MSFT))
colnames(stock_data) <- symbols
stock_returns <- diff(log(stock_data))
#creating labels based on return  threshold
return_threshold <- 0.01
stock_labels <- ifelse(apply(abs(stock_returns), 1, max) > return_threshold, "High", "Low")
# Combine returns and labels into a data frame
stock_df <- data.frame(stock_returns, stock_labels)
# Split data into training and testing sets
set.seed(123)
train_indices <- sample(1:nrow(stock_df),0.75* nrow(stock_df))
train_data <- stock_df[train_indices, ]
test_data <- stock_df[-train_indices, ]
# When the input data contains missing values, make sure our 'test_data' does not have any missing or NA values. We can use the is.na() function to identify them
any_na <-apply(test_data,2,function(column)
  any(is.na(column)))
```

CODE:

```
print(any_na)
```

OUTPUT:

```
> print(any_na)
        AAPL         MSFT stock_labels
        TRUE         TRUE         TRUE
```

CODE:

 # If we find missing values, we can clean the data by removing or imputing them before using it for prediction.
 # Remove rows with missing values

```
test_data <- test_data[complete.cases(test_data), ]
```

#LDA

```
lda_model <- lda(stock_labels ~ ., data = train_data)
lda_predictions <- predict(lda_model, newdata = test_data)$class
```

#QDA

```
qda_model <- qda(stock_labels ~ ., data = train_data)
qda_predictions <- predict(qda_model, newdata = test_data)$class
```

#NAIVEBAYES Classification

```
nb_model <- naiveBayes(stock_labels ~ ., data = train_data)
nb_predictions <- predict(nb_model, newdata = test_data)
```

#calculate accuracies

```
lda_accuracy <- mean(lda_predictions == test_data$stock_labels)
qda_accuracy <- mean(qda_predictions == test_data$stock_labels)
nb_accuracy <- mean(nb_predictions == test_data$stock_labels)
```

 # Print the accuracy results

```
cat("LDA ACCURACY:" , lda_accuracy,"\n")
cat("QDA ACCURACY:" , qda_accuracy, "\n")
cat("navive bayes accuracy:",nb_accuracy, "\n")
```

OUTPUT:

```
> cat("LDA ACCURACY:" , lda_accuracy,"\n")
LDA ACCURACY: 0.75
> cat("QDA ACCURACY:" , qda_accuracy, "\n")
QDA ACCURACY: 0.9787234
> cat("navive bayes accuracy:",nb_accuracy, "\n")
navive bayes accuracy: 0.9734043
```

CODE:

 # Inorder to avoid the error: `data` and `reference` should be factors with the same levels.

```
test_data$stock_labels=as.factor(test_data$stock_labels)
```

#Calculate and print confusion matrix

```
lda_cm <- confusionMatrix(lda_predictions, test_data$stock_labels)
qda_cm <- confusionMatrix(qda_predictions, test_data$stock_labels)
nb_cm <- confusionMatrix(nb_predictions, test_data$stock_labels)
print(lda_cm)
print(qda_cm)
print(nb_cm)
```

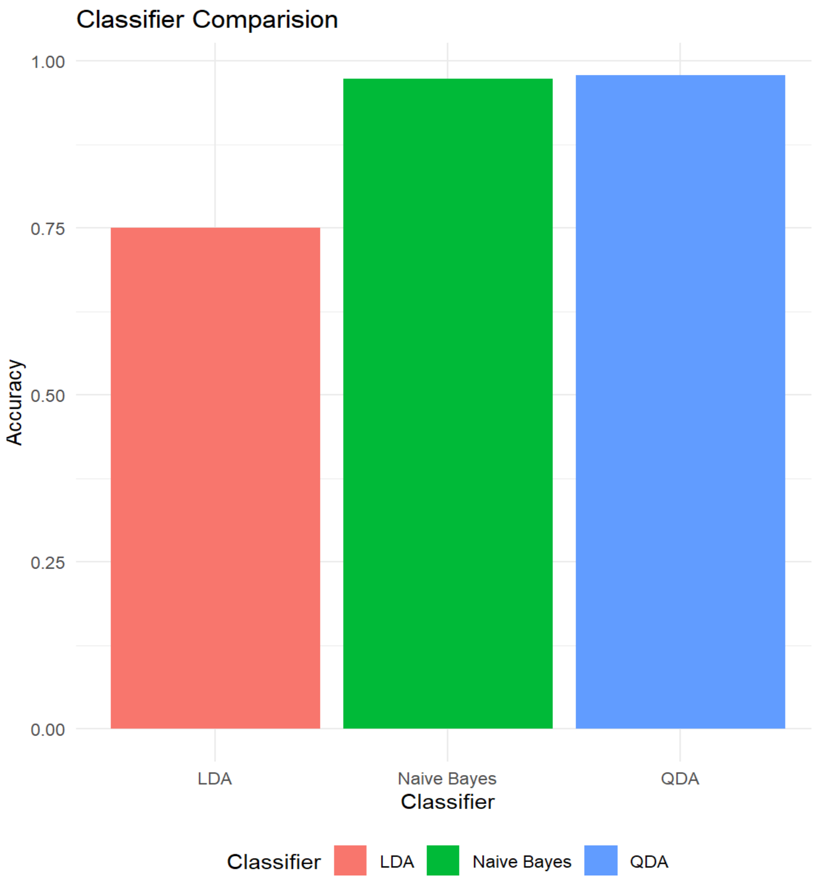
OUTPUT:

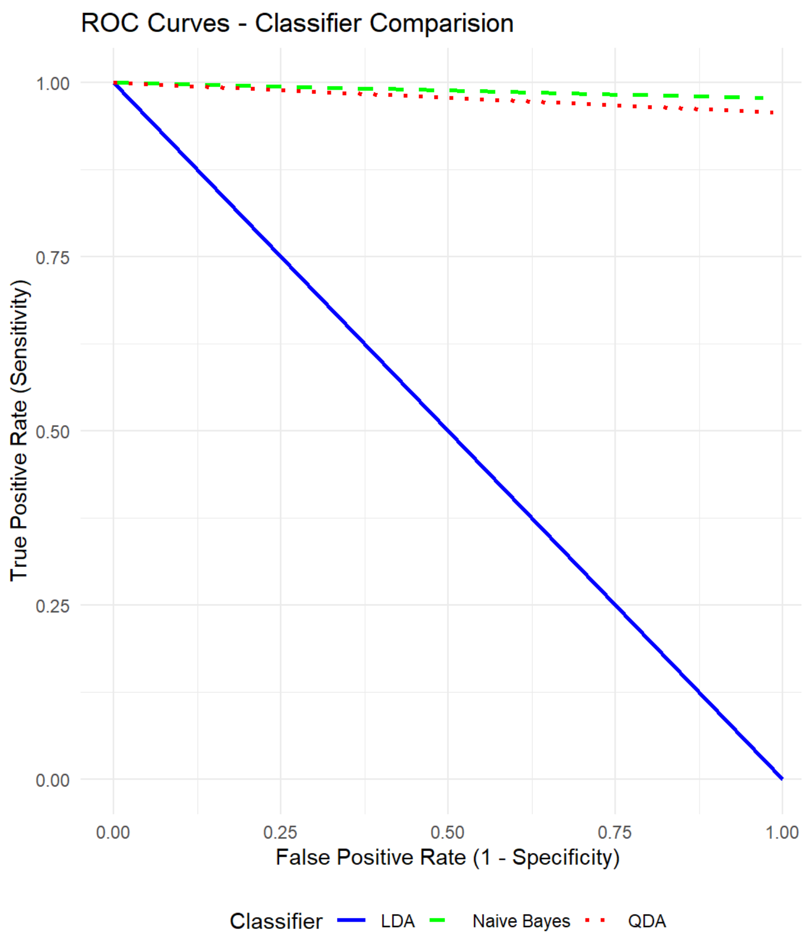
```
> print(lda_cm)
Confusion Matrix and Statistics

          Reference
Prediction High Low
      High  141  47
      Low     0   0

               Accuracy : 0.75
                 95% CI : (0.6818, 0.8102)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : 0.5391

                  Kappa : 0

 Mcnemar's Test P-Value : 1.949e-11

            Sensitivity : 1.00
            Specificity : 0.00
         Pos Pred Value : 0.75
         Neg Pred Value :  NaN
             Prevalence : 0.75
         Detection Rate : 0.75
   Detection Prevalence : 1.00
      Balanced Accuracy : 0.50

       'Positive' Class : High

> print(qda_cm)
Confusion Matrix and Statistics

          Reference
Prediction High Low
      High  139   2
      Low     2  45

               Accuracy : 0.9787
                 95% CI : (0.9464, 0.9942)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9433

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9858
            Specificity : 0.9574
         Pos Pred Value : 0.9858
         Neg Pred Value : 0.9574
             Prevalence : 0.7500
         Detection Rate : 0.7394
   Detection Prevalence : 0.7500
      Balanced Accuracy : 0.9716

       'Positive' Class : High

> print(nb_cm)
Confusion Matrix and Statistics

          Reference
Prediction High Low
      High  137   1
      Low     4  46

               Accuracy : 0.9734
                 95% CI : (0.939, 0.9913)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9306

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9716
            Specificity : 0.9787
         Pos Pred Value : 0.9928
         Neg Pred Value : 0.9200
             Prevalence : 0.7500
         Detection Rate : 0.7287
   Detection Prevalence : 0.7340
      Balanced Accuracy : 0.9752

       'Positive' Class : High
```

CODE:

```
# Create data frame for accuracy comparison
accuracy_df <- data.frame(Classifier = c("LDA", "QDA", "Naive Bayes"),Accuracy =
c(lda_accuracy,qda_accuracy,nb_accuracy))

#create accuracy plot
accuracy_plot <- ggplot(accuracy_df, aes(x= Classifier, y= Accuracy, fill = Classifier))+ geom_bar(stat = "identity",
position = "dodge")+labs(y = "Accuracy", title = "Classifier Comparision")+theme_minimal()+theme(legend.position =
"bottom")
print(accuracy_plot)

#create ROC  curves
roc_lda <- roc(test_data$stock_labels, as.numeric(lda_predictions == "high"))
roc_qda <- roc(test_data$stock_labels, as.numeric(qda_predictions == "High"))
roc_nb <- roc(test_data$stock_labels, as.numeric(nb_predictions == "High"))

# Determine the common length of sensitivities
min_length <- min(length(roc_lda$sensitivities), length(roc_qda$sensitivities), length(roc_nb$sensitivities))

# Combine ROC curve data into a single data frame
roc_data <- data.frame(
  FPR = c(roc_lda$specificities[1:min_length], roc_qda$specificities[1:min_length],
roc_nb$specificities[1:min_length]),
  TPR = c(roc_lda$sensitivities[1:min_length], roc_qda$sensitivities[1:min_length],
roc_nb$sensitivities[1:min_length]),
  Classifier = rep(c("LDA", "QDA", "Naive Bayes"), each = min_length)
)

# Create combined ROC curve plot
roc_combined_plot <- ggplot(roc_data, aes(x = FPR, y = TPR, color = Classifier, linetype = Classifier)) +
geom_line(linewidth = 1)+
  labs(x = "False Positive Rate (1 - Specificity)", y = "True Positive Rate (Sensitivity)", title = "ROC Curves - Classifier
Comparision") +
  scale_color_manual(values = c("blue", "green", "red"))+
  scale_linetype_manual(values = c("solid", "dashed", "dotted"))+
  theme_minimal()+
  theme(legend.position = "bottom")
print(roc_combined_plot)
```

OUTPUT:


Classifier Comparision
Accuracy
1.00
0.75
0.50
0.25
0.00
LDA
Naive Bayes
QDA
Classifier
Classifier LDA Naive Bayes QDA


ROC Curves - Classifier Comparision
True Positive Rate (Sensitivity)
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
False Positive Rate (1 - Specificity)
Classifier LDA Naive Bayes QDA

Findings:

Quadratic Discriminant Analysis (QDA) is the best algorithm with accuracy rate 97.8% among the Linear Discriminant Analysis(LDA) with accuracy rate 75% ,naive bayes classification with accuracy rate 97.3%.