Math574M_HW2

Saheed Adisa, Ganiyu

2023-09-16

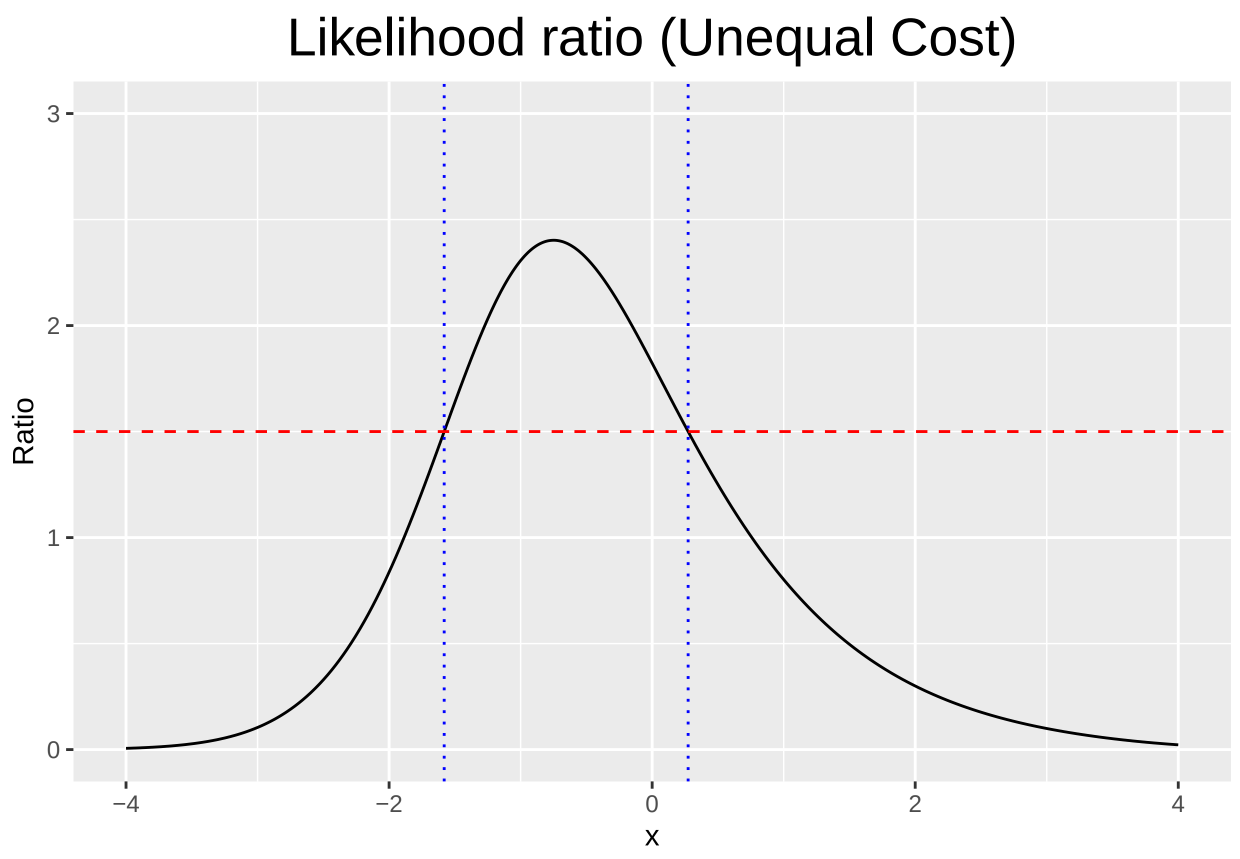
5. (Bayes Rule for Unequal Costs)

(c) Provide a numerical solution for the Bayes decision boundary.

```r
# Load required libraries
library(ggplot2)

# Define the probability density functions (PDFs)
g1 <- function(x) dnorm(x, mean = 0, sd = 1)
g0 <- function(x) 0.65 * dnorm(x, mean = 1, sd = 1) + 0.35 * dnorm(x, mean = -1, sd = 2)

# Define the equation to solve in order to get the boundary values
likelihood_ratio <- function(x) g1(x) / g0(x)- (3/2)
boundary <- c( (uniroot(likelihood_ratio, c(-4,0)))$root, (uniroot(likelihood_ratio, c(0,4)))$root)
cat("Boundary: ",boundary)
```

```
## Boundary:  -1.58113 0.2736474
```

```r
# Calculate the values of the likelihood ratio for the x values in [-4,4]
x = seq(-4, 4, by = 0.01)
likelihood_ratio2 <- function(x) g1(x) / g0(x)
ratio <- likelihood_ratio2(x)

# Create a data frame for the plot
data <- data.frame(x = x, y = ratio)

# Create the plot
ggplot(data, aes(x = x, y = y)) +
  geom_line() +
  geom_hline(yintercept = 3/2, linetype = "dashed", color = "red") +
  geom_vline(xintercept = boundary, linetype = "dotted", color = "blue") +
  xlim(-4, 4) +
  ylim(0, 3) +
  labs(
    title = "Likelihood ratio (Unequal Cost)",
    x = "x",
    y = "Ratio"
  )+
theme(plot.title = element_text(size = 20, hjust = 0.5))
```



Question 6. (Two-Class Classification Problem: Scenario 1)

Generate 100 observations from a bivariate Gaussian distribution $N(\mu_1, \Sigma_1)$ with $\mu_1 = (2,1)^T$ and $\Sigma_1 = \mathbf{I}$ (the identity matrix), and label them as Green. Generate 100 observations from a bivariate Gaussian distribution $N(\mu_2, \Sigma_2)$ with $\mu_2 = (1,2)^T$ and $\Sigma_2 = \mathbf{I}$, and label them as Red.

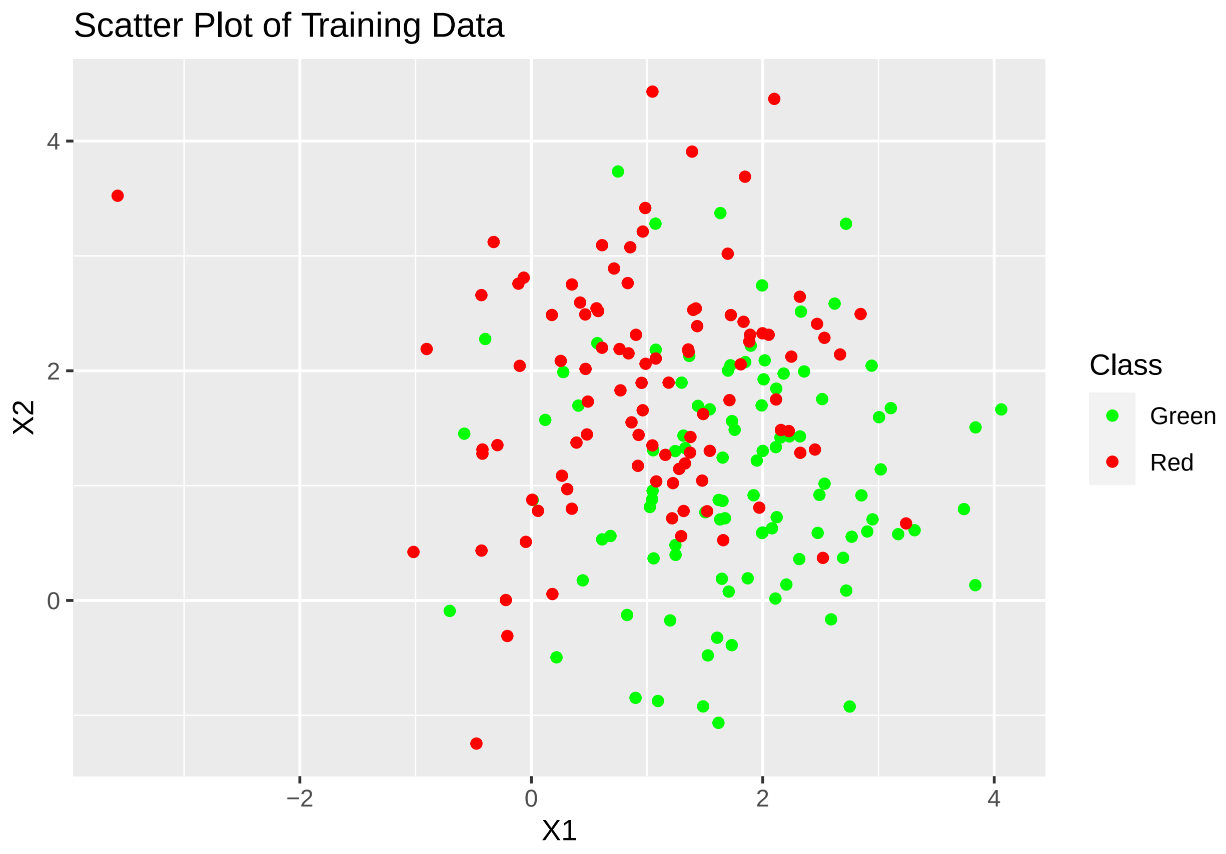
(a) Write R code to generate the training data. Set the seed with set.seed(2023) before calling the random number generation function.

```r
library(MASS)      # loading the MASS package
library(ggplot2)  # Load ggplot2 for plotting
set.seed(2023)

# setting mean and covariance
n1 = 100
mean1 <- c(2,1)
mean2 <- c(1,2)
cov1 = matrix(c(1,0,0,1),nrow=2)
cov2 = matrix(c(1,0,0,1),nrow=2)
green_train = mvrnorm(100,mean1,cov1)     # generating a random sample of 50 data points from a multiva
class1_labels <- rep("Green", n1)
red_train = mvrnorm(n1,mean2,cov2)
```

```r
class2_labels <- rep("Red", n1)

# Combining both green and red data, then add class labels
training <- rbind(data.frame(green_train, Class = class1_labels),
                  data.frame(red_train, Class = class2_labels))
```

(b) Draw the scatter plot of the training data, using different labels/colors for two classes.

```r
# Plot the scatter plot of training data
library(ggplot2)  # Load ggplot2 for plotting
ggplot(training, aes(x = X1, y = X2, color = Class)) +
  geom_point() +
  scale_color_manual(values = c("Green" = "green", "Red" = "red")) +
  labs(title = "Scatter Plot of Training Data", x = "X1", y = "X2")
```



(c) Generate a test set, with 500 observations from each class, using set.seed(2024). Save the data set for future use.

```r
set.seed(2024)

# Generate test data
n2 <- 500  # Number of observations per class
green_test <- mvrnorm(n2, mean1, cov1)
```

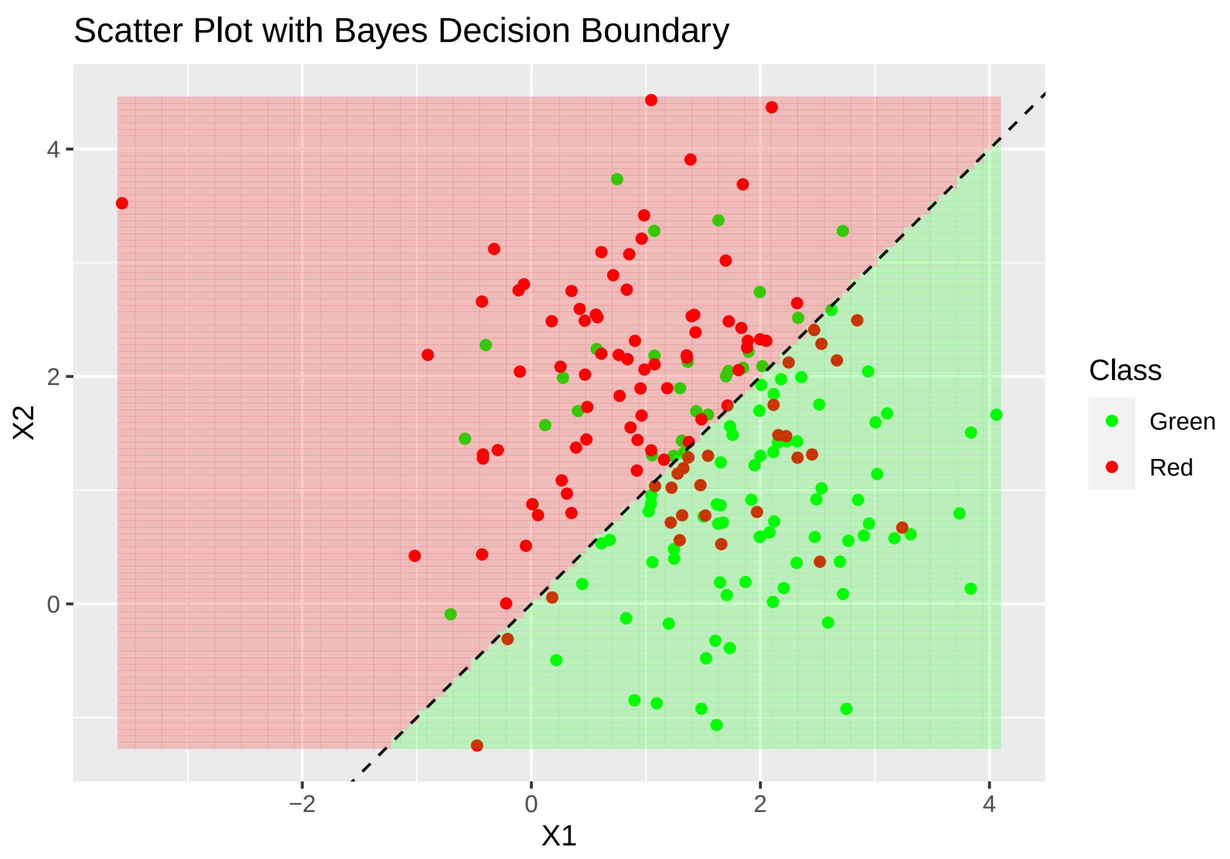
```r
class1_test_labels <- rep("Green", n2)
red_test <- mvrnorm(n2, mean2, cov2)
class2_test_labels <- rep("Red", n2)

# Combine data and labels for the test set
testing <- rbind(data.frame(green_test, Class = class1_test_labels),
                    data.frame(red_test, Class = class2_test_labels))
```

7. (Bayes Classification Rule: Scenario 1)

Assume two classes in Scenario 1 have the same prior probabilities, i.e., $\mathbb{P}(\text{ Green }) = \mathbb{P}(\text{ Red }) = \frac{1}{2}$.
#### (a) Using the 0-1 loss, derive the Bayes classifier. Simplify the solution as much as you can.
#### (b) Add the Bayes decision boundary to the scatter plot drawn in Question 6.

```r
# Function to calculate the conditional desity of X given Y based on Cov = identity
gi_of_X <- function(x1, x2, mean_i) {
  exponent <- -1/2*(((x1 - mean_i[1])^2) + ((x2 - mean_i[2])^2))
  return(1/(2*pi)*exp(exponent))
}

# Creating boundary points to add the Bayes decision boundary to the scatter plot
boundary_x1 <- seq(min(training$X1), max(training$X1), length.out = 100)
boundary_x2 <- seq(min(training$X2), max(training$X2), length.out = 100)

# Create a grid of points for the decision boundary
boundary_points <- expand.grid(X1 = boundary_x1, X2 = boundary_x2)

# Classifying each point
boundary_points$decision <- apply(boundary_points, 1, function(point) {
  prob_class1 <- gi_of_X(point[1], point[2], mean1)
  prob_class2 <- gi_of_X(point[1], point[2], mean2)
  return(ifelse(prob_class1 >= prob_class2, "Green", "Red"))
})

# Plot the scatter plot with the decision boundary
ggplot() +
  geom_point(data = training, aes(x = X1, y = X2, color = Class)) +
  geom_tile(data = boundary_points, aes(x = X1, y = X2, fill = decision), alpha = 0.2) +
  geom_abline(intercept = 0, slope = 1, color = "black", linetype = "dashed") +    # The boundary x1-x2
  scale_color_manual(values = c("Green" = "green", "Red" = "red")) +
  scale_fill_manual(values = c("Green" = "green", "Red" = "red"), guide = FALSE) +
  labs(title = "Scatter Plot with Bayes Decision Boundary", x = "X1", y = "X2")
```

```
## Warning: The `guide` argument in `scale_*()` cannot be `FALSE`. This was deprecated in
## ggplot2 3.3.4.
## i Please use "none" instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```r
# Function to classify data points based on the Bayes rule
Bayes_rule <- function(data, mean_1, mean_2) {
  prob_class1 <- gi_of_X(data$X1, data$X2, mean_1)
  prob_class2 <- gi_of_X(data$X1, data$X2, mean_2)
  return(ifelse(prob_class1 >= prob_class2, "Green", "Red"))
}

# Training data classification
training_pred <- Bayes_rule(training, mean1, mean2)

# Test data classification (using the saved test data from Question 6)
testing_pred <- Bayes_rule(testing, mean1, mean2)

# Calculate training and test errors
training_error <- mean(training_pred != training$Class)
test_error <- mean(testing_pred != testing$Class)
cat("Training Error:", training_error, "\n")
```

(c) Compute the training and test errors for the Bayes classifier, using the data generated in Question 6. Report the training and test errors.

```
## Training Error: 0.275
```

```r
cat("Test Error:", test_error, "\n")
```

```
## Test Error: 0.235
```

Comment: The training and testing error for Baye's classifier are 27.5% and 23.5% respectively

8. (Linear Method for Classification: Scenario 1)

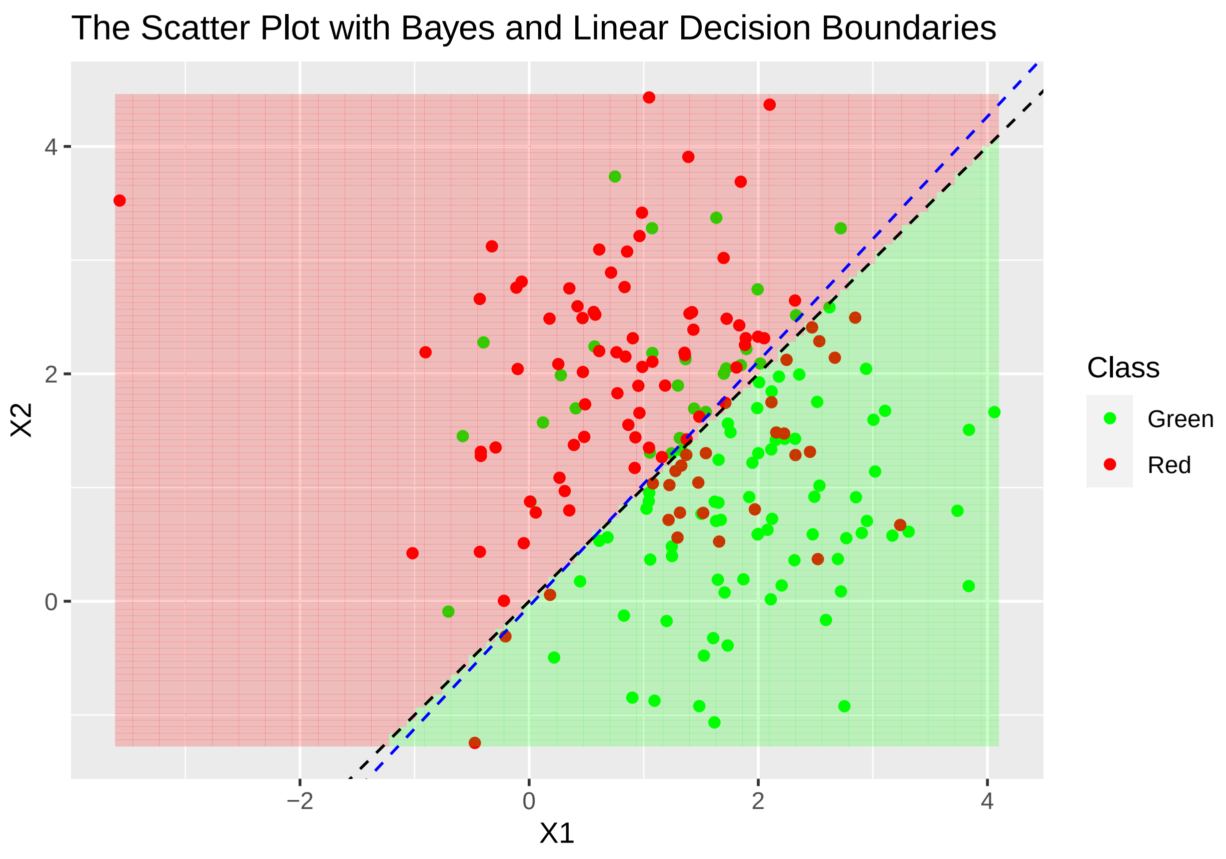
```r
# Fiiting the model for trainind data
# Create a new numeric column based on the "Class" column
training$ClassNumeric <- ifelse(training$Class == "Green", 1, 0)
testing$ClassNumeric <- ifelse(testing$Class == "Green", 1, 0)
linear_model <- lm(ClassNumeric ~ X1 + X2, data = training)
summary(linear_model)
```

(a) Train the linear regression model, using the function " lm(y ∼ x) ", with the training set generated in Question 6.

```
##
## Call:
## lm(formula = ClassNumeric ~ X1 + X2, data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9320 -0.3865  0.0126  0.3702  0.9632
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.49309    0.06900   7.147 1.69e-11 ***
## X1           0.16779    0.02951   5.687 4.63e-08 ***
## X2          -0.15581    0.02917  -5.341 2.53e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4349 on 197 degrees of freedom
## Multiple R-squared:  0.2548, Adjusted R-squared:  0.2472
## F-statistic: 33.68 on 2 and 197 DF,  p-value: 2.629e-13
```

```r
# Extract the coefficients of the linear model
coefficients <- coef(linear_model)

# Calculate the coefficients for the decision boundary line
slope <- -coefficients["X1"] / coefficients["X2"]
intercept <- (0.5-coefficients["(Intercept)"]) / coefficients["X2"]

# Adding the linear decision boundary to the scatter plot
ggplot() +
  geom_point(data = training, aes(x = X1, y = X2, color = Class)) +
  geom_tile(data = boundary_points, aes(x = X1, y = X2, fill = decision), alpha = 0.2) +
  geom_abline(intercept = 0, slope = 1, color = "black", linetype = "dashed") +     # The boundary x2 =
  geom_abline(intercept = intercept, slope = slope, color = "blue", linetype = "dashed") +     # The bo
  scale_color_manual(values = c("Green" = "green", "Red" = "red")) +
  scale_fill_manual(values = c("Green" = "green", "Red" = "red"), guide = FALSE) +
  labs(title = "The Scatter Plot with Bayes and Linear Decision Boundaries", x = "X1", y = "X2")
```

(b) Add the linear decision boundary line to the scatter plot (which already has Bayes boundary).



```
# Predict class labels for the training and testing data
training_pred_linear <- ifelse(predict(linear_model, newdata = training, type = "response") >= 0.5, "Gre
testing_pred_linear <- ifelse(predict(linear_model, newdata = testing, type = "response") >= 0.5, "Green

# computing training and test errors for the linear classifier
training_error_linear <- mean(training_pred_linear != training$Class)
test_error_linear <- mean(testing_pred_linear != testing$Class)

# training and test errors printing
cat("Training Error:", training_error_linear, "\n")
```

(c) Report the training and test errors for the linear classifier.

```
## Training Error: 0.28
```

```
cat("Test Error:", test_error_linear, "\n")
```

```
## Test Error: 0.238
```

Comment: The training and testing error for the linear model are 28% and 23.8% respectively.