EAS509_Project1

2023-04-23

Libraries used are

```
library(readxl)
library(ggplot2)
library(e1071)
library(class)
library(caret)
```

```
## Loading required package: lattice
```

```
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(MASS)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
library(reshape2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following object is masked from 'package:party':
##
##     where
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Read the Excel file, specifying the sheet number
mines_data <- read_excel("/Users/vinithavudhayagiri/Documents/Spring2023/SDM2/Project1/Mine_Dataset.xls", sheet = 2)

# Check the dimensions of the data frame
dim(mines_data)
```

```
## [1] 338   4
```

```
# View the entire data frame
mines_data
```

```
## # A tibble: 338 × 4
##        V     H     S     M
##    <dbl> <dbl> <dbl> <dbl>
##  1 0.338 0       0       1
##  2 0.320 0.182   0       1
##  3 0.287 0.273   0       1
##  4 0.256 0.455   0       1
##  5 0.263 0.545   0       1
##  6 0.241 0.727   0       1
##  7 0.254 0.818   0       1
##  8 0.235 1       0       1
##  9 0.353 0       0.6     1
## 10 0.335 0.182   0.6     1
## # … with 328 more rows
```

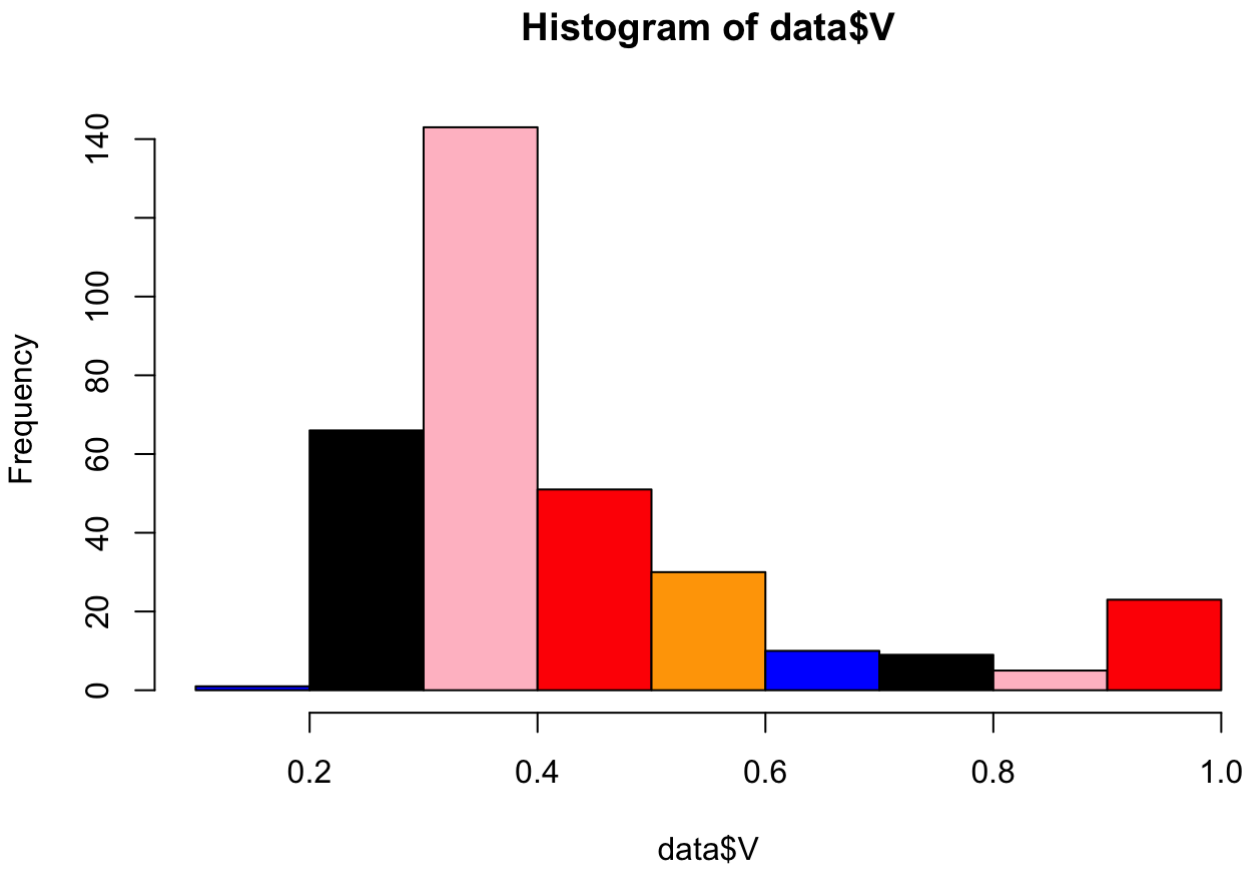
```
# Remove rows with missing values and store the result in a new data frame
data <- na.omit(mines_data)

# Check the dimensions of the cleaned data frame
dim(data)
```

```
## [1] 338   4
```

```
# Get column names of mines_data
names(data)
```

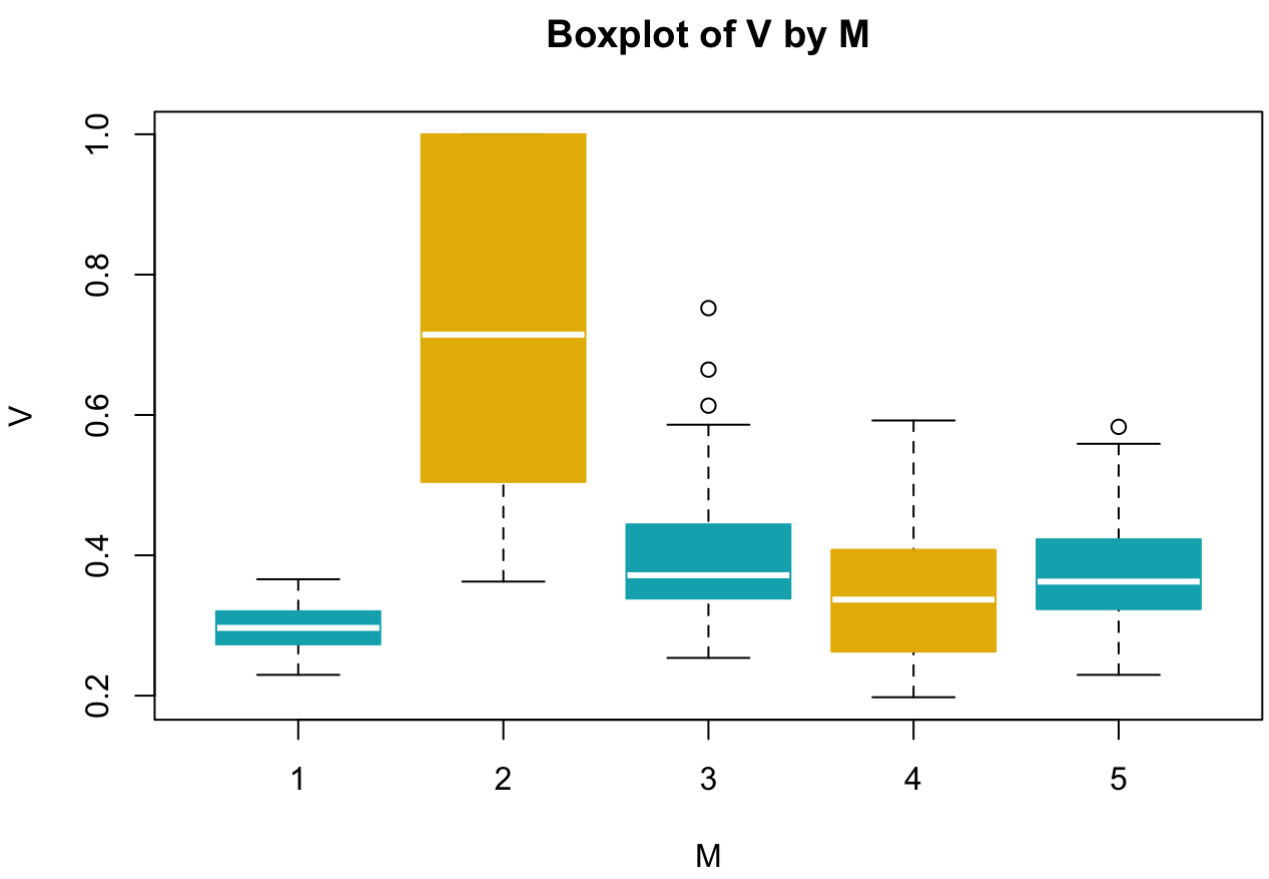
```
## [1] "V" "H" "S" "M"
```

```
# create a vector of colors
my_colors <- c("blue", "black", "pink", "red", "orange")

# plot histogram with voltage V
hist(data$V, col = my_colors)
```

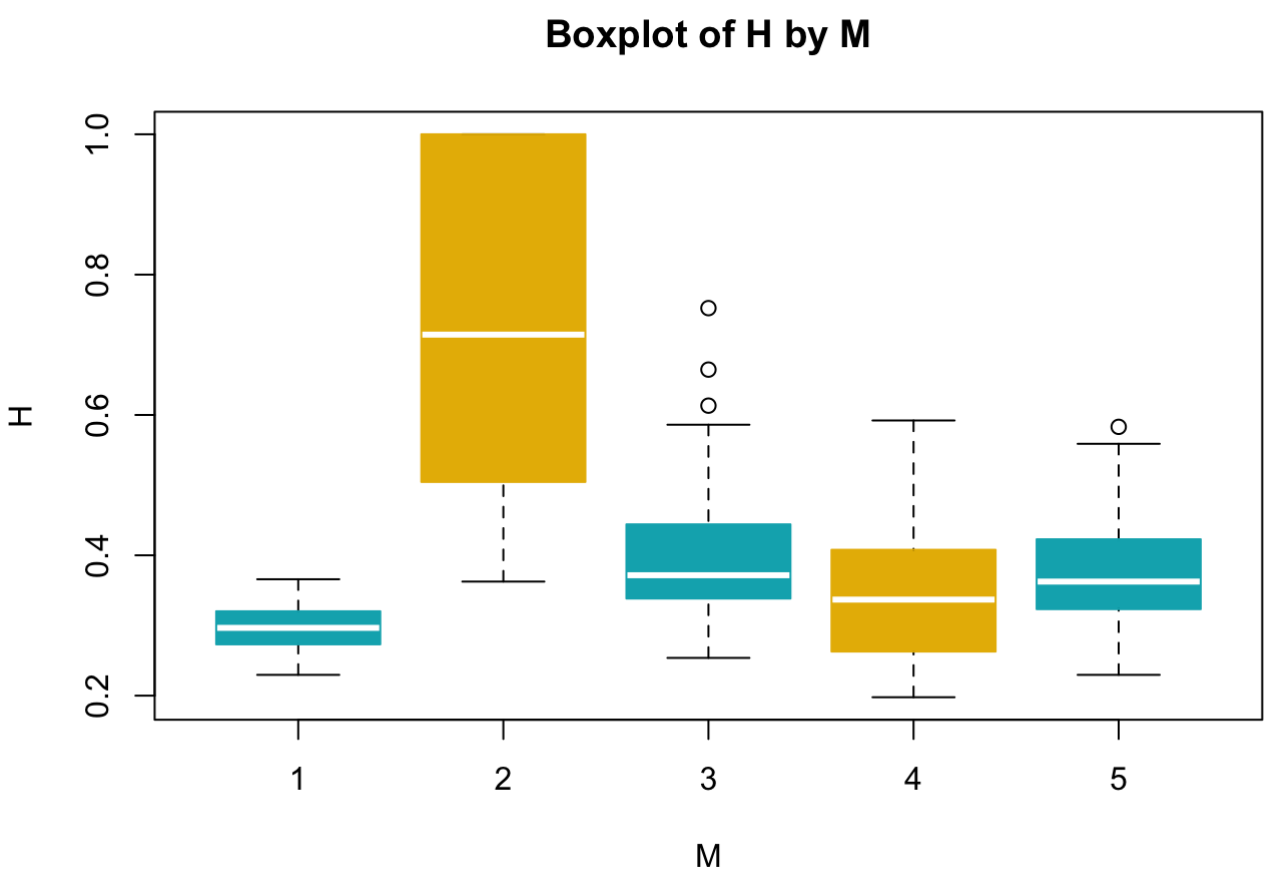


Box plot for different combinations of attributes with the target variable Mine Type M

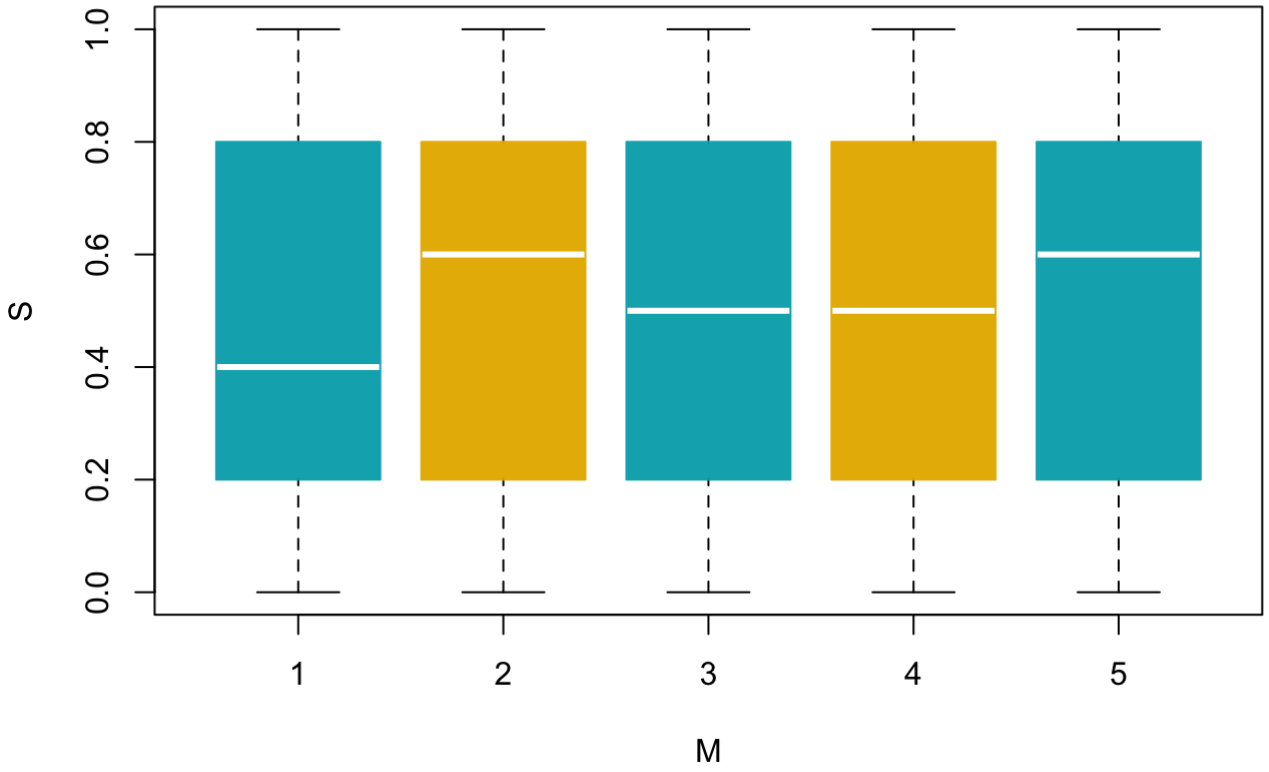
```
colors <- c("#00AFBB", "#E7B800")

# plot the boxplot
boxplot(V ~ M, data = data,
        main = "Boxplot of V by M",
        xlab = "M", ylab = "V",
        col = colors,
        boxcol = colors,
        whiskercol = colors,
        outliercol = colors,
        medcol = "white")
```




```
colors <- c("#00AFBB", "#E7B800")
# plot the boxplot
boxplot(V ~ M, data = data,
        main = "Boxplot of H by M",
        xlab = "M", ylab = "H",
        col = colors,
        boxcol = colors,
        whiskercol = colors,
        outliercol = colors,
        medcol = "white")
```




```
colors <- c("#00AFBB", "#E7B800")
# plot the boxplot
boxplot(S ~ M, data = data,
        main = "Boxplot of S by M",
        xlab = "M", ylab = "S",
        col = colors,
        boxcol = colors,
        whiskercol = colors,
        outliercol = colors,
        medcol = "white")
```




```
# randomly split the data into train and test sets
train <- sample(c(TRUE, FALSE), nrow(data), replace = TRUE, prob=c(0.80, 0.20)) # random
ly sample 80% of rows to be in the train set
test <- (!train) # assign the remaining 20% of rows to the test set

# subset the data based on train and test indices
train_data <- data[train, ] # subset the rows in the train set
test_data <- data[test, ] # subset the rows in the test set

# print the dimensions of the train and test sets
cat("Train set dimensions:", dim(train_data), "\n")
```

```
## Train set dimensions: 274 4
```

```
cat("Test set dimensions:", dim(test_data), "\n")
```

```
## Test set dimensions: 64 4
```

Logistic Regression

```
# Logistic Regression
# fit logistic regression model
log_model <- glm(M ~ V + H + S, data = data)
# summarize the model
summary(log_model)
```

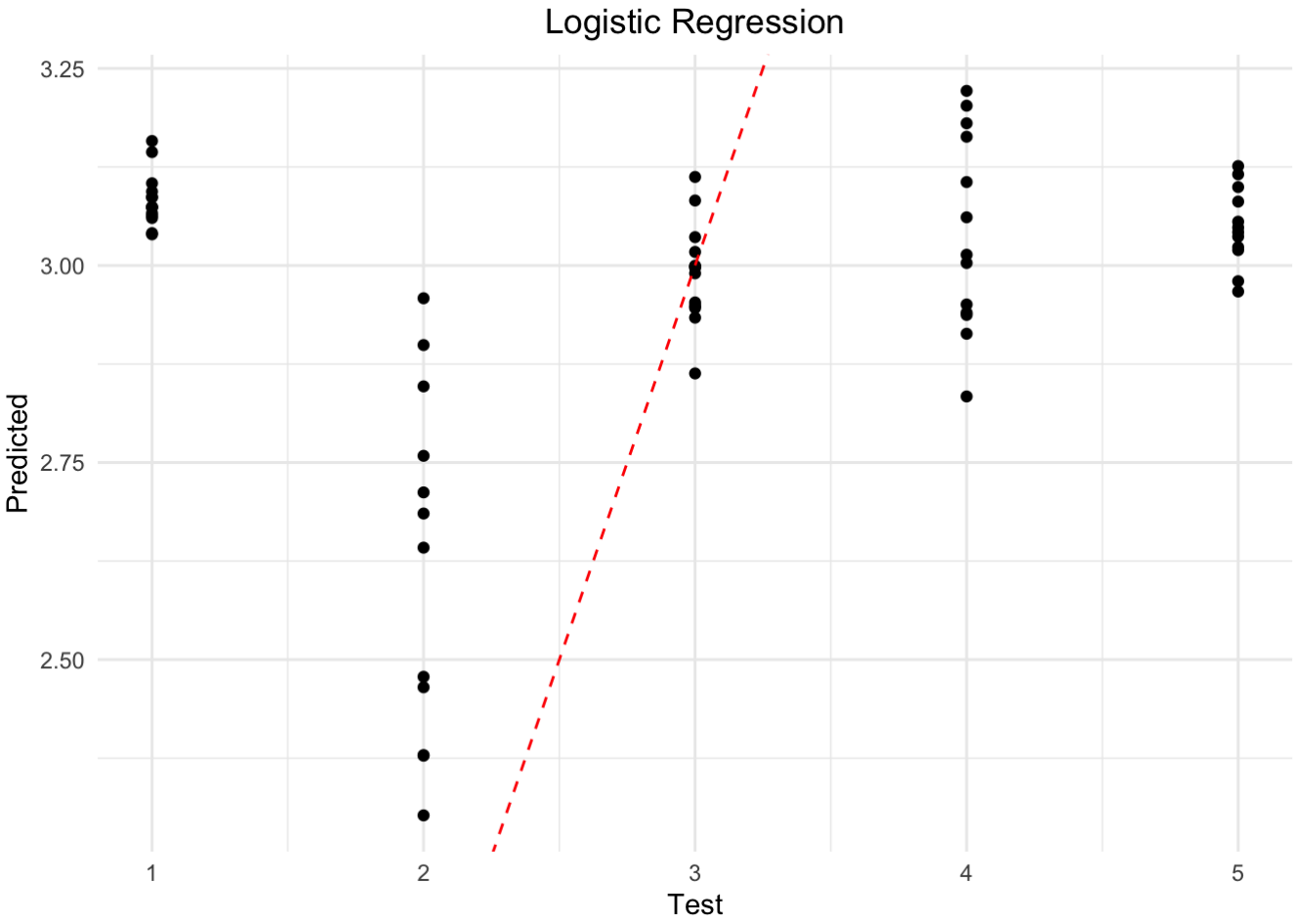
```
##
## Call:
## glm(formula = M ~ V + H + S, data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.21070  -0.86470  -0.00332   1.01415   2.23621
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.40900    0.29408  11.592  < 2e-16 ***
## V           -1.10656    0.42484  -2.605  0.00961 **
## H           -0.07468    0.27116  -0.275  0.78316
## S            0.11556    0.22379   0.516  0.60594
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.989152)
##
##     Null deviance: 679.24  on 337  degrees of freedom
## Residual deviance: 664.38  on 334  degrees of freedom
## AIC: 1197.6
##
## Number of Fisher Scoring iterations: 2
```

```
# predict outcome for test data
log_predictions <- predict(log_model, newdata = test_data, type = "response")
# make binary predictions based on threshold 0.5
log_predictions_binary <- ifelse(log_predictions > 0.5, 1, 0)
# calculate accuracy
log_accuracy <- mean(log_predictions_binary == test_data$M)
cat("The Accuracy is ", log_accuracy, "\n")
```

```
## The Accuracy is  0.21875
```

Visualizing the data using logistic regression

```
# predict outcome for new data
predictions <- predict(log_model, newdata = test_data, type = "response")
log_df <- data.frame(Test = data$M[test], Predicted = predictions)
# plot the graph
ggplot(log_df, aes(x = Test, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(x = "Test", y = "Predicted", title = "Logistic Regression") +
  ggtitle("Logistic Regression") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



Naive Bayes

```
# train Naive Bayes classifier
nb_model <- naiveBayes(M ~ V + H + S, data = data)
# summarize the model
summary(nb_model)
```

```
##           Length Class  Mode
## apriori   5      table  numeric
## tables    3      -none- list
## levels    5      -none- character
## isnumeric 3      -none- logical
## call      4      -none- call
```

```
# predict outcome for test data
nb_predictions <- predict(nb_model, newdata = test_data)
# view the predictions
nb_predictions
```

```
##  [1] 1 4 1 1 1 1 1 1 2 2 2 2 3 2 3 5 1 1 5 3 5 3 3 5 5 3 5 3 5 5 5 5 1 3 4 1 1 1
## [39] 5 1 5 1 1 1 1 1 1 1 1 2 2 2 2 5 1 1 4 1 5 5 5 5 5 5
## Levels: 1 2 3 4 5
```

```
# calculate accuracy
nb_accuracy <- mean(nb_predictions == test_data$M)
cat("The Accuracy is ", nb_accuracy, "\n")
```

```
## The Accuracy is  0.5625
```

Visualizing the data using Naive Bayes

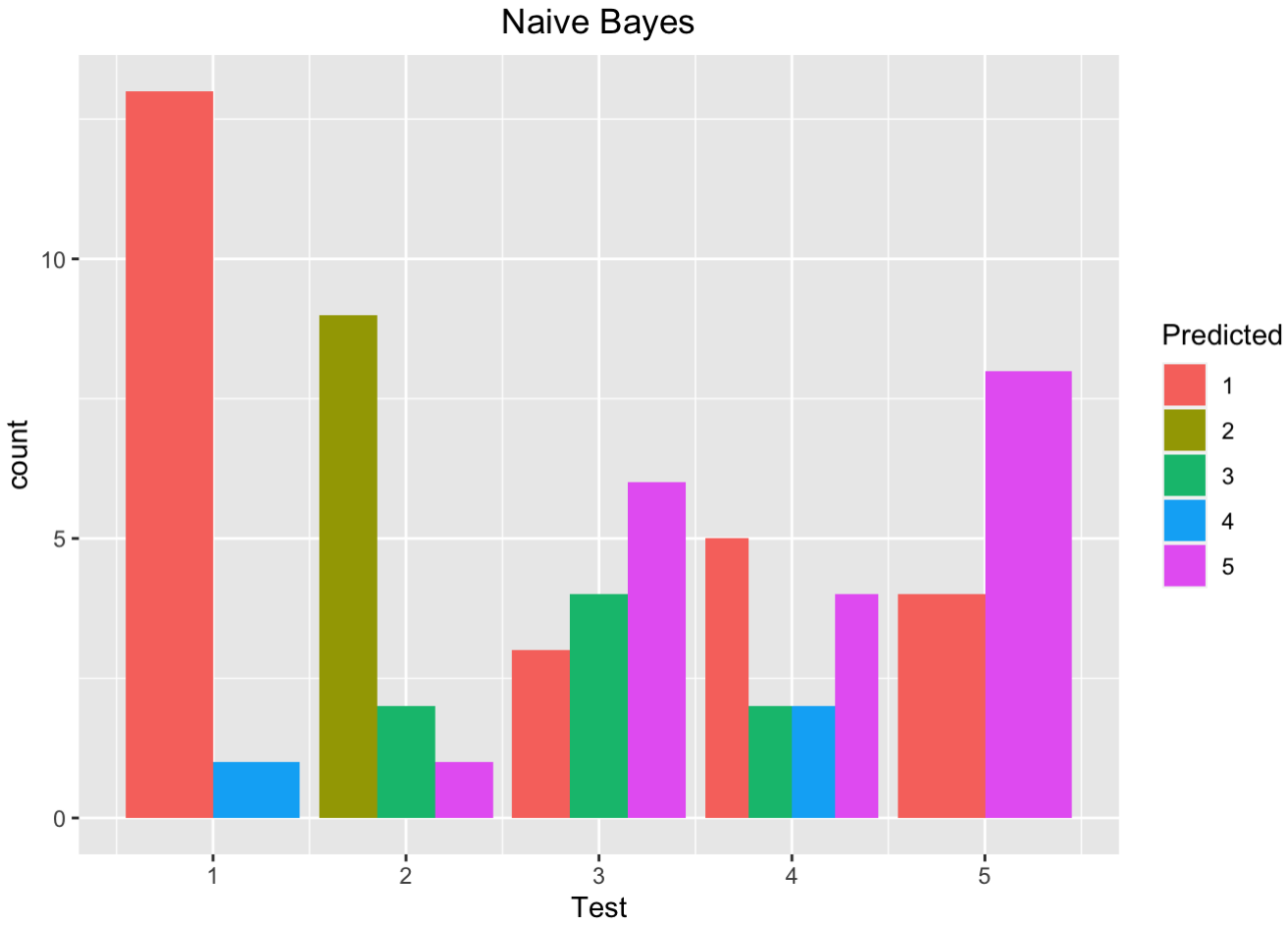
```
# train Naive Bayes classifier
nb_model <- naiveBayes(M ~ V + H + S, data = data)
# summarize the model
summary(nb_model)
```

```
##           Length Class  Mode
## apriori   5      table  numeric
## tables    3      -none- list
## levels    5      -none- character
## isnumeric 3      -none- logical
## call      4      -none- call
```

```
# predict outcome for new data
predictions <- predict(nb_model, newdata = test_data)
# view the predictions
predictions
```

```
##  [1] 1 4 1 1 1 1 1 1 2 2 2 2 3 2 3 5 1 1 5 3 5 3 3 5 5 3 5 3 5 5 5 5 1 3 4 1 1 1
## [39] 5 1 5 1 1 1 1 1 1 1 1 2 2 2 2 5 1 1 4 1 5 5 5 5 5 5
## Levels: 1 2 3 4 5
```

```
# plot bar graph between test and predicted values
# create a data frame with the test values and predicted values
nb_df <- data.frame(Test = data$M[test], Predicted = predictions)

# plot a bar graph
ggplot(nb_df, aes(x = Test, fill = Predicted)) +
  geom_bar(position = "dodge") +
  ggtitle("Naive Bayes") +
  theme(plot.title = element_text(hjust = 0.5))
```

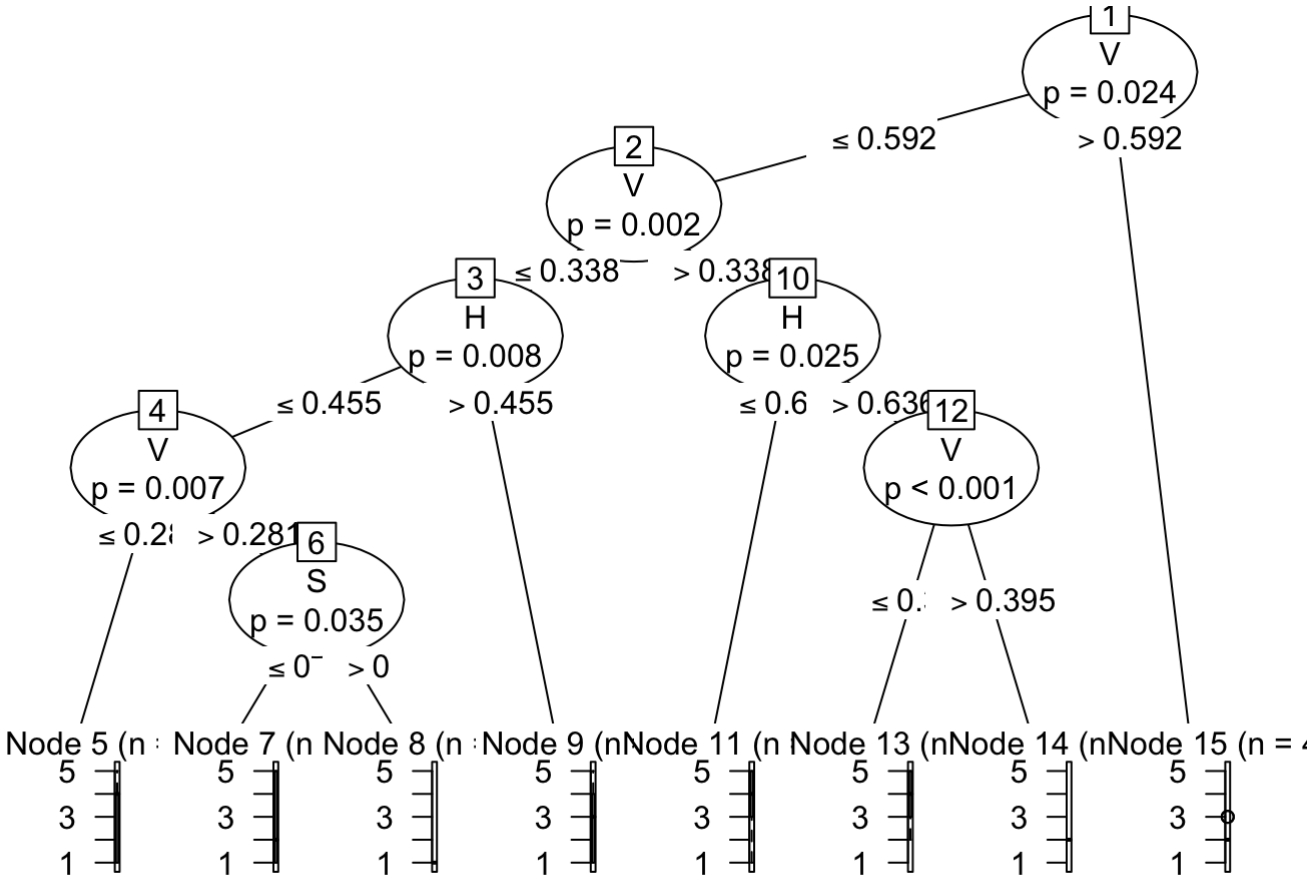


Decision Trees

```
# Set the output file for the decision tree plot
png(file = "decision_tree.png")

# Create a conditional inference tree using the ctree function
output.tree <- ctree(M ~ V + H + S, data = mines_data)

# Plot the decision tree
plot(output.tree)
```

Visualizing the data using Decision Trees

```
# plot the decision tree
plot(output.tree)
```



Support Vector Machine

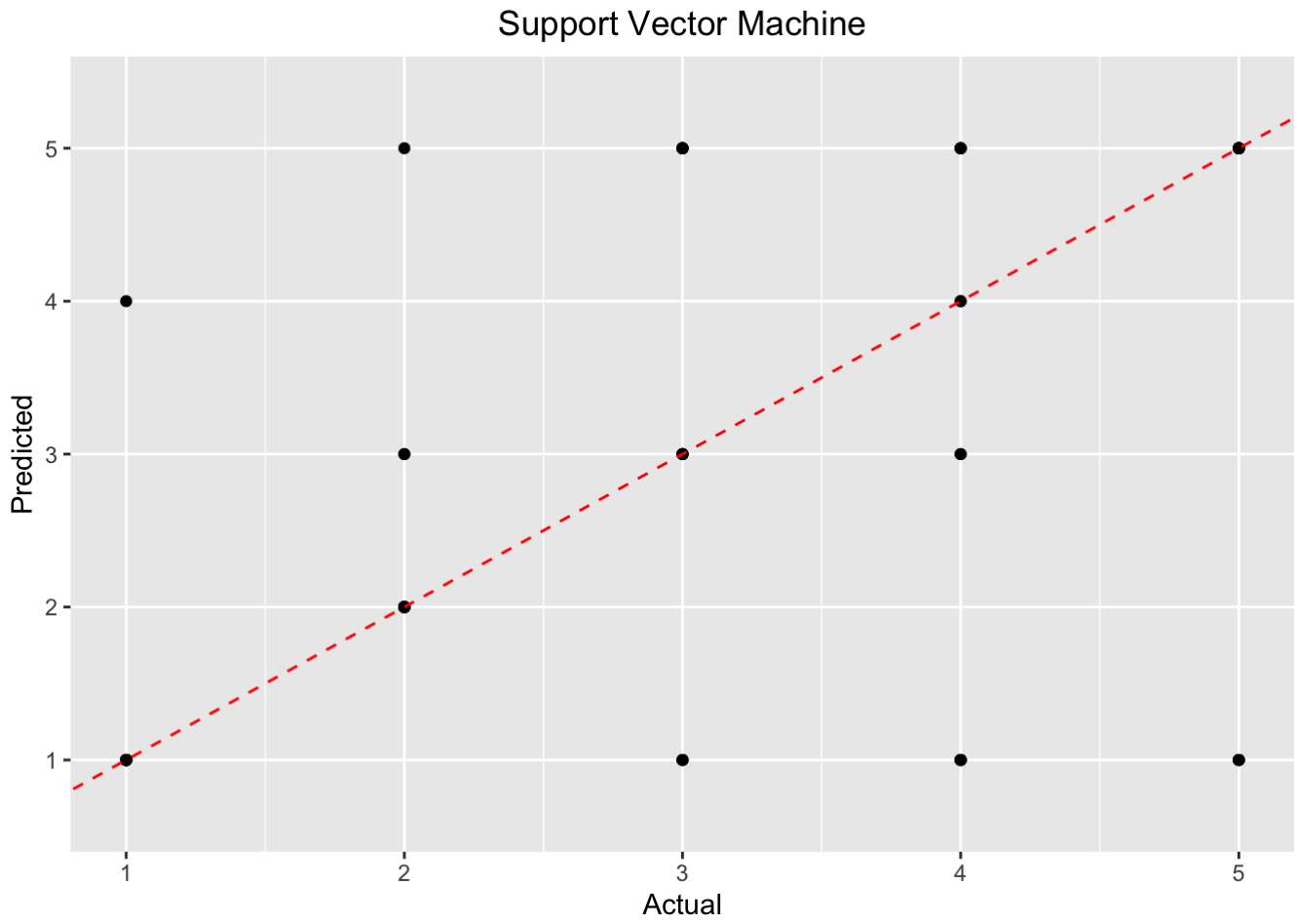
```
# fit SVM model
svm_model <- svm(M ~ V + H + S, data = data)
# summarize the model
summary(svm_model)
```

```
##
## Call:
## svm(formula = M ~ V + H + S, data = data)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.3333333
##     epsilon:  0.1
##
##
## Number of Support Vectors:  287
```

```
# predict outcome for test data
svm_predictions <- predict(svm_model, newdata = test_data)
```

Visualizing the data using Support Vector Machine

```
# Create data frame with actual and predicted values
svm_df <- data.frame(actual = data$M[test], predicted = predictions)
# Create scatter plot
ggplot(svm_df, aes(x = actual, y = predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(x = "Actual", y = "Predicted", title = "Support Vector Machine") +
  theme(plot.title = element_text(hjust = 0.5))
```



Linear Discriminant Analysis (LDA)

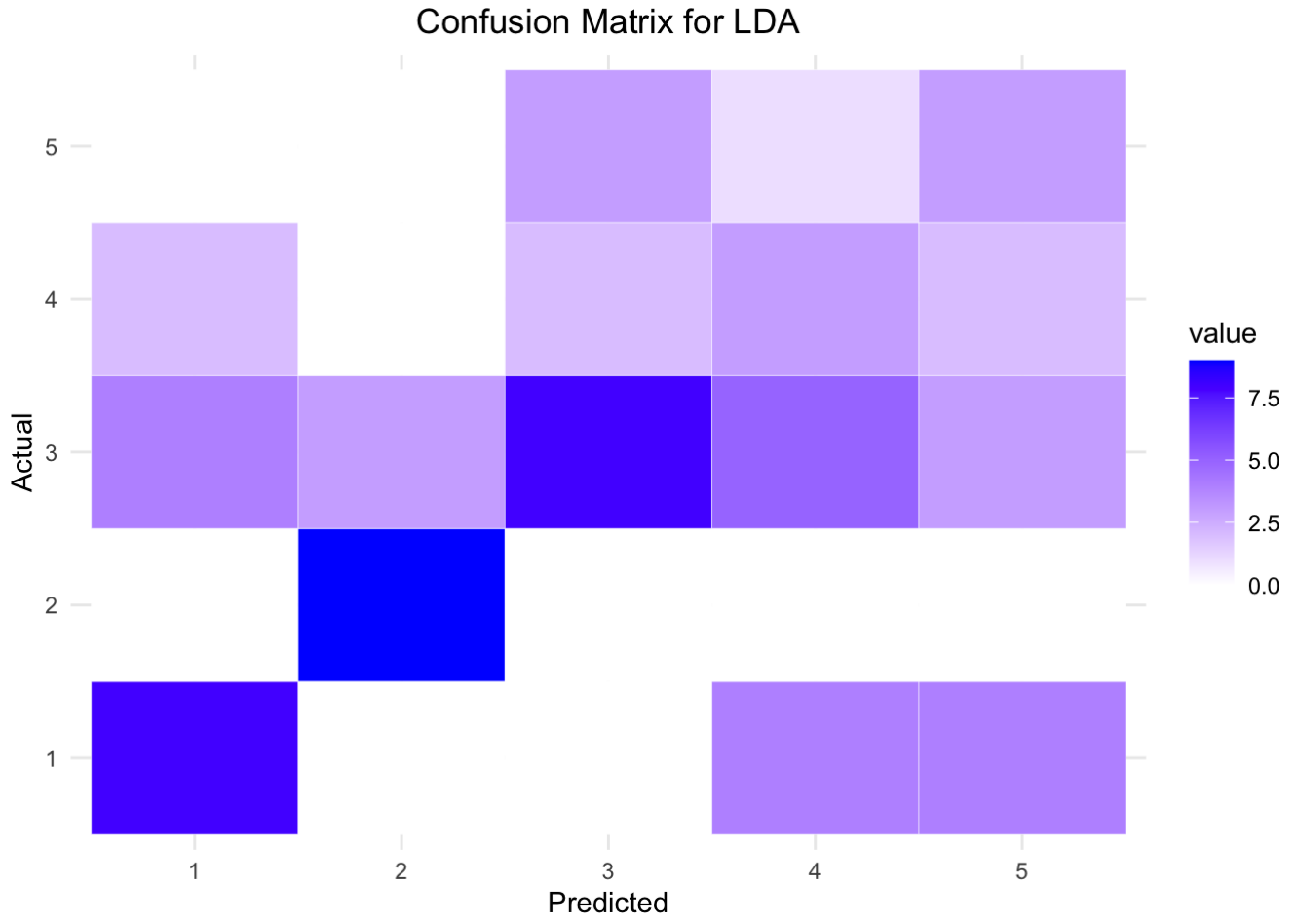
```
# fit LDA model
model_lda <- lda(M ~ V + H + S, data = data)
# summarize the model
summary(model_lda)
```

```
##         Length Class  Mode
## prior    5     -none- numeric
## counts   5     -none- numeric
## means   15     -none- numeric
## scaling  9     -none- numeric
## lev      5     -none- character
## svd      3     -none- numeric
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
# predict outcome for test data
predictions_lda <- predict(model_lda, newdata = test_data)
```

Visualizing the data using Linear Discriminant Analysis (LDA)

```
# Create confusion matrix
confusion_matrix <- table(data$M[test], predictions_lda$class)

# Visualize confusion matrix
confusion_df <- as.data.frame.matrix(confusion_matrix)
confusion_df$Actual <- rownames(confusion_df)
confusion_df_melted <- melt(confusion_df, id.vars = "Actual")
ggplot(confusion_df_melted, aes(x = Actual, y = variable, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(x = "Predicted", y = "Actual", title = "Confusion Matrix for LDA") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



Quadratic Discriminant Analysis (QDA)

```
# Fit QDA model
model_qda <- qda(M ~ V + H + S, data = data)
# Summarize the model
summary(model_qda)
```

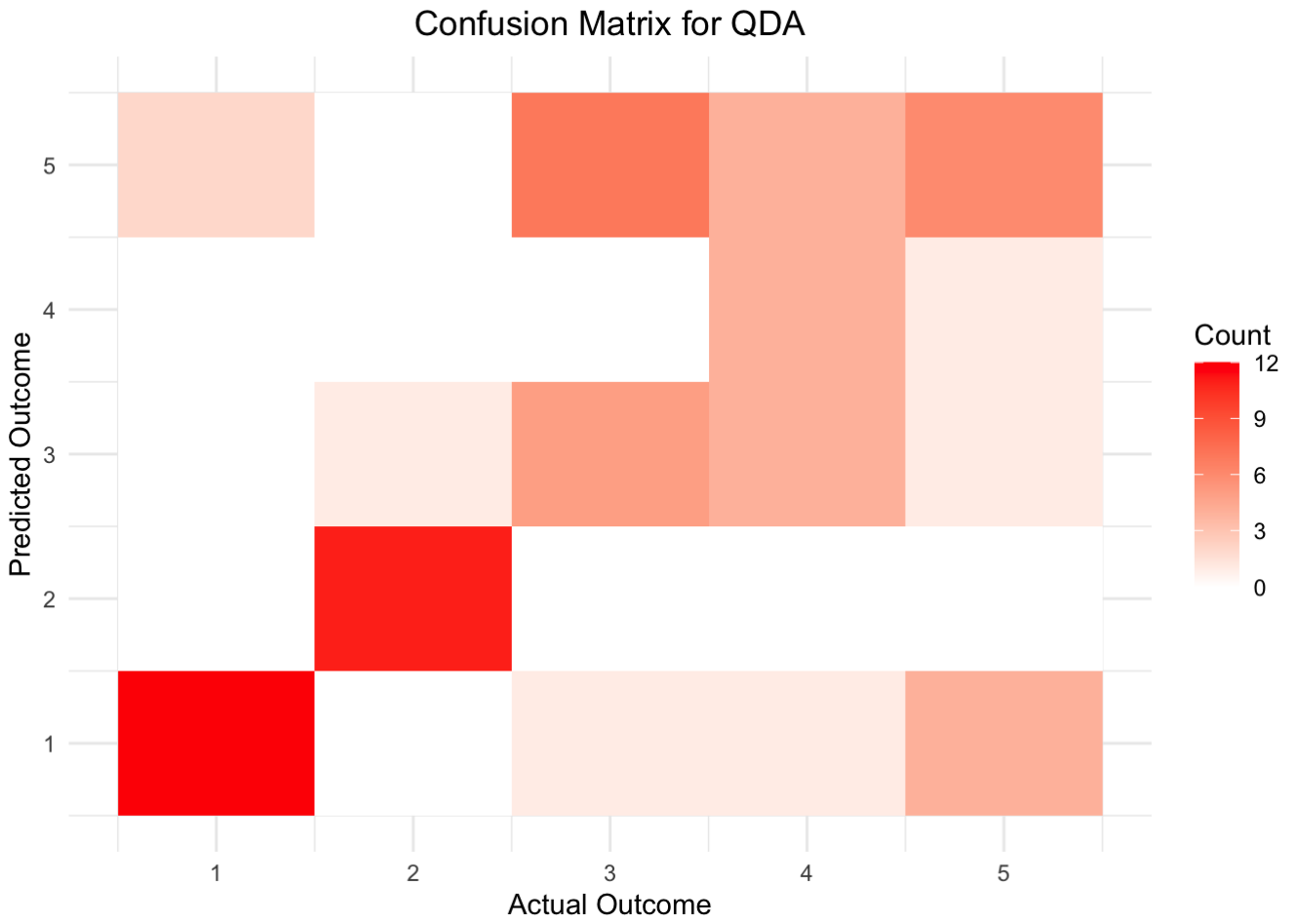
```
##         Length Class  Mode
## prior    5     -none- numeric
## counts   5     -none- numeric
## means   15     -none- numeric
## scaling 45     -none- numeric
## ldet     5     -none- numeric
## lev      5     -none- character
## N        1     -none- numeric
## call     3     -none- call
## terms    3     terms  call
## xlevels  0     -none- list
```

```
# Predict outcome for test data
predictions_qda <- predict(model_qda, newdata = test_data)
# Find the accuracy of predictions
qda_accuracy <- mean(predictions_qda$class == test_data$M)
cat("The accuracy is ", qda_accuracy, "\n")
```

```
## The accuracy is  0.59375
```

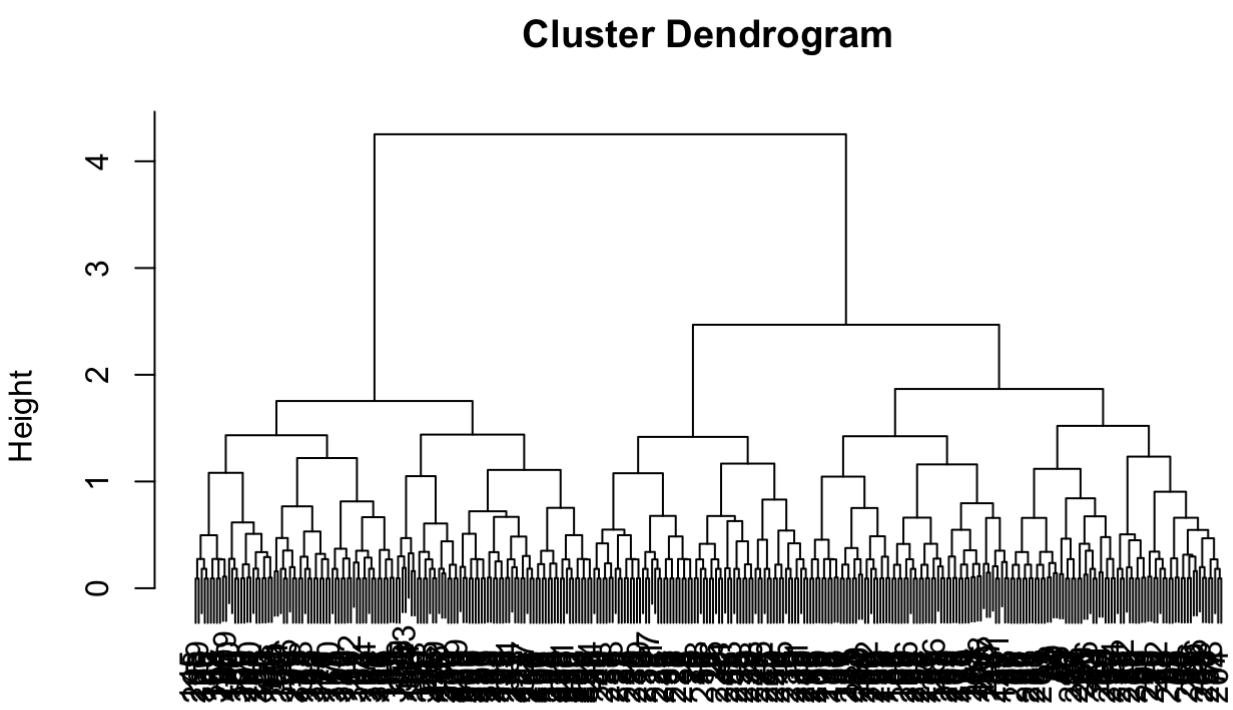
Visualizing the data using Quadratic Discriminant Analysis (QDA)

```
# Create confusion matrix
confusion_matrix <- table(data$M[test], predictions_qda$class)

# Melt the confusion matrix into a data frame
melted_matrix <- melt(confusion_matrix)

# Create a heat map of the confusion matrix
ggplot(data = melted_matrix, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Confusion Matrix for QDA",
       x = "Actual Outcome",
       y = "Predicted Outcome",
       fill = "Count") +
  theme(plot.title = element_text(hjust = 0.5))
```


Confusion Matrix for QDA
Predicted Outcome
Actual Outcome
1
2
3
4
5
Count
12
9
6
3
0

Hierarchical Clustering

```
# Set the seed for reproducibility
set.seed(123)

# Specify the proportion of data to be used for training
train_pct <- 0.7

# Get the number of rows in the data set
n <- nrow(data)

# Sample random indices from the data set without replacement
train_indices <- sample.int(n, train_pct*n, replace=FALSE)

# Subset the data into training and testing sets
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

# calculate the distance matrix
dist_mat <- dist(data)

# perform hierarchical clustering with complete linkage
hclust_result <- hclust(dist_mat, method = "complete")

# plot the dendrogram
plot(hclust_result)
```




```
# Define the number of clusters
num_clusters <- 3

# Train a hierarchical clustering model on the training data
hclust_result <- hclust(dist(train_data[, 1:3]))

# Assign cluster labels to the training data using k-means clustering
train_clusters <- cutree(hclust_result, k = num_clusters)

# Use the cluster labels to predict the clusters of the test data
test_clusters <- apply(test_data[, 1:3], 1, function(x) {
  # Calculate the distance from each point to each cluster center
  cluster_centers <- aggregate(train_data[, 1:3], list(train_clusters), mean)
  distances <- apply(cluster_centers[, -1], 1, function(c) sqrt(sum((x - c)^2)))

  # Assign the closest cluster label to the test point
  closest_cluster <- which.min(distances)
  return(closest_cluster)
})

# Print the predicted cluster labels for the test data
print(test_clusters)
```

```
##   [1] 3 3 1 1 3 3 2 1 2 2 1 3 3 2 2 1 3 3 2 2 3 3 2 2 3 3 3 1 2 3 2 2 1 1 1 3 1
##  [38] 1 2 2 1 3 1 2 2 1 2 2 3 2 1 1 3 3 1 2 2 2 1 3 1 2 1 2 1 3 3 2 3 1 3 2 1 3
##  [75] 3 2 3 1 2 1 1 3 3 1 1 2 1 1 1 3 1 2 1 2 2 3 3 3 1 1 2 1
```

Visualizing the data using Hierarchical Clustering

```
# Create a data frame with test data and predicted clusters
test_clusters_df <- data.frame(test_data, cluster = as.factor(test_clusters))

# Plot the test data with different colors representing the predicted clusters
color_list <- c("red", "blue", "green")
ggplot(test_clusters_df, aes(x = test_data$M, y = test_clusters, color = cluster)) +
  geom_point(size = 3) +
  labs(x = "Variable 1", y = "Variable 2", title = "Scatterplot of Test Data by Cluste
r") +
  scale_color_manual(values = color_list) +
  theme(plot.title = element_text(hjust = 0.5))
```