PROJECT 1

Dataset: Mines dataset

https://archive-beta.ics.uci.edu/dataset/763/land+mines-1

The mines dataset is a multivariate dataset, which consists of 4 columns V, H, S, M. We have total of 338 samples in the dataset.

V-Voltage

Output voltage of FLC sensor due to magnetic distortion. The values of voltage range from 0 to 10.6

H-High

The height of the sensor from the ground, the value is in centimeter scale and ranges from 0 to 20.

S-Soil type

S denotes the 6 different soil types depending on the moisture condition of the soil.

1- Dry and Sandy
2- Dry and Humus
3- Dry and Limy
4- Humid and Sandy
5- Humid and Humus
6- Humid and Limy

M- Mine type

Here we are taking M as the dependent variable which is the output we are going to predict. Mine type which is commonly encountered on the land, we have 5 different mine types.

1- Null
2- Anti-Tank
3- Anti-Personnel
4- Booby Trapped Anti-Personnel
5- M14 Anti-personnel

As we have 5 different types in the output variable M it is a multi variate classification.

Firstly we need to import ‘readxl’ library to use ‘read_excel ( )’ function, it basically used to import or read an excel file from the local desktop and assigning them into a variable ‘mines_data’.

We have installed all the libraries used in the project.

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

Finding the dimension of the dataset

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

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

Printing the sample of the mines dataset.

```
# 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
```

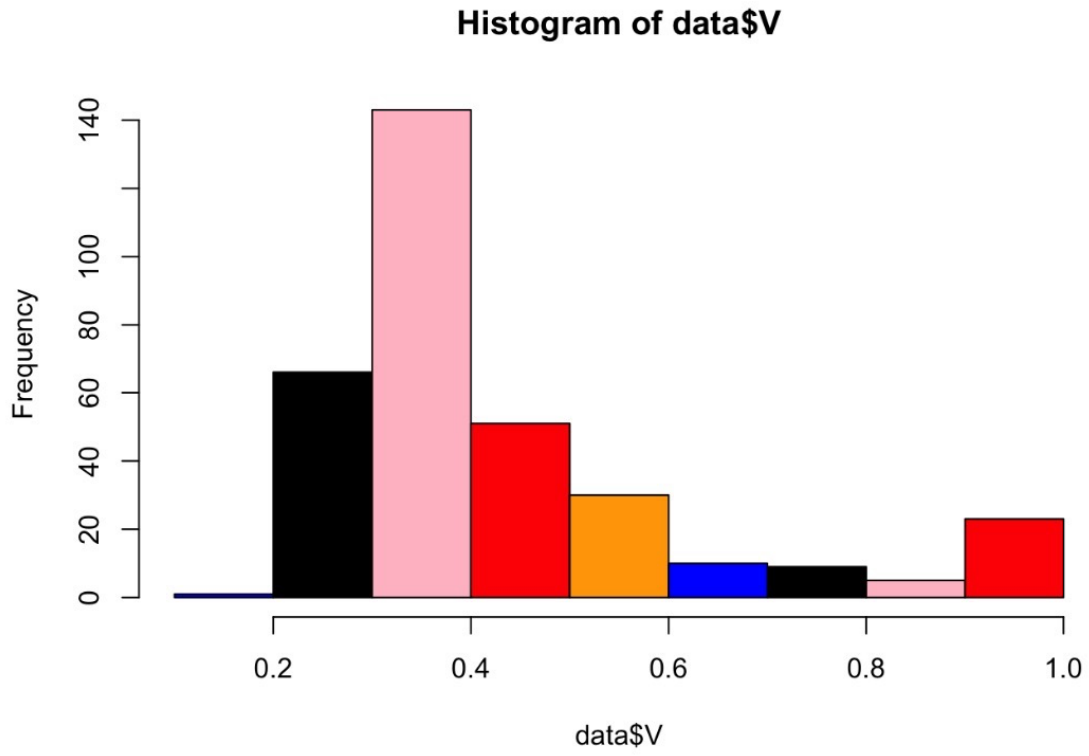
Now we are going to check if any null data is there in the dataset and printing the dimension after omitting the null dataset.

```r
# 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
```

Histogram Plot

Plotting the histogram plot for the output response variable to check how many values belong to each class.



From the above plot we can describe those values 1 and 2 are most in equal number while 3,4,5 is equal which is little less than 1 and 2 class.

Names function gives all the columns in the dataset.

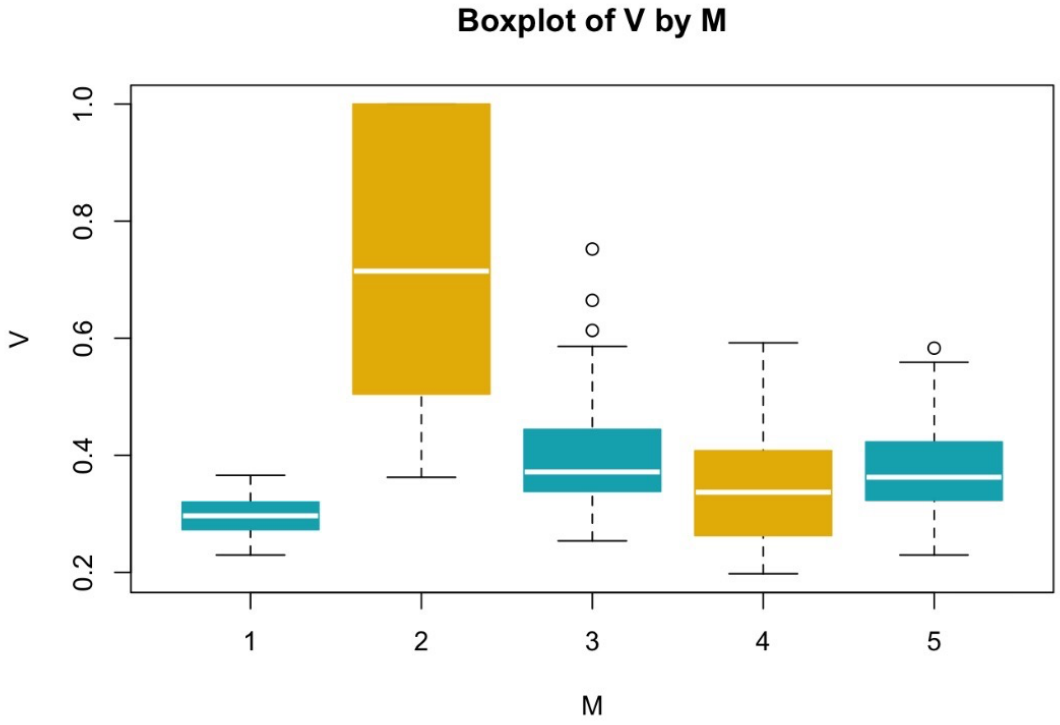
```
names(mines_data)
```

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

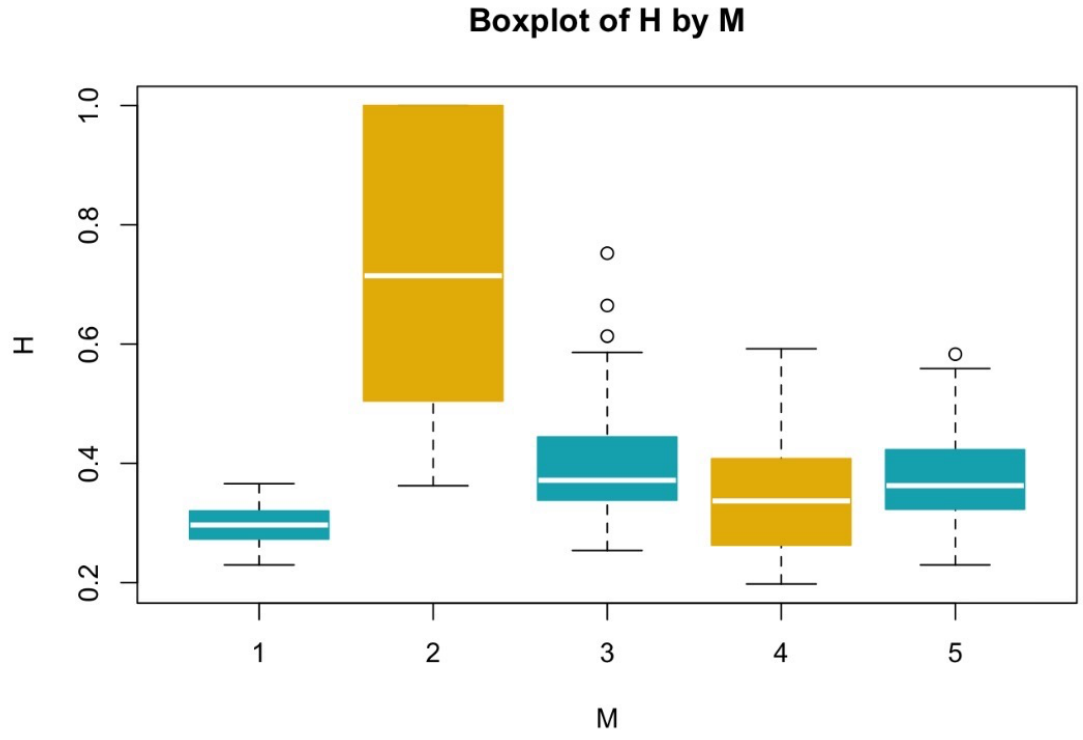
Box Plot

Now we are going to plot the box plot for each input variable with respect to the output variable M.

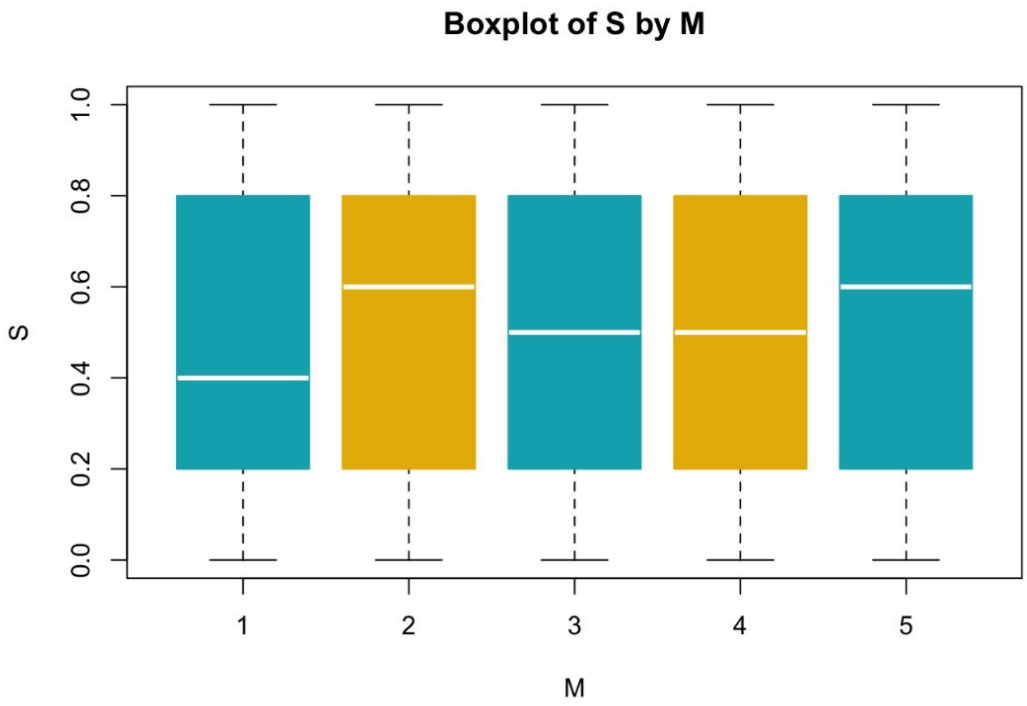
```
colors <- c("#00AFBB", "#E7B800")

# 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")
```



Now we are dividing our data set into training data and test data with respective 80% and 20% and checking the dimension of the training data.

```
## Train set dimensions: 277 4
```

Checking the dimension of the testing dataset.

```
## Test set dimensions: 61 4
```

After splitting the data, we are performing different models to our dataset and checks the accuracy for each model and predicts which model works the best to our dataset.

The models which we are using are-

- Logistic Regression
- Naïve Bayes
- Decision Tree
- Support Vector Machine
- LDA
- QDA
- Hierarchical clustering

Logistic Regression: -

Logistic regression is a statistical technique used to analyze and model the relationship between a binary dependent variable and one or more independent variables.

Using **glm** function we can fit the model. log_model is the logistic regression model that is being fitted to our data. "M" is the dependent variable and "V", "H", and "S" are independent variables.

summary function displays the statistics of our log_model.

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
```

Using predict function we are going to predict the values on the test data.

Converting the predicted probabilities ie, log_predictions generated from log_model into binary predictions

Calculating accuracy using the binary predictions ie, log_predictions_binary.

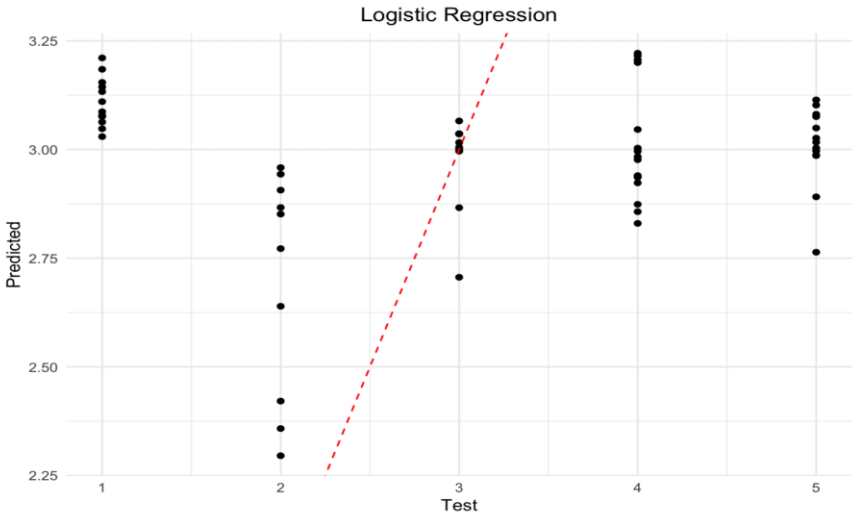
```
# predict outcome for test data
log_predictions <- predict(log_model, newdata = test_data, type = "response")
# make binary predictions based on threshold 0.5
log_predictions_binary <- ifelse(log_predictions > 0.5, 1, 0)
# calculate accuracy
log_accuracy <- mean(log_predictions_binary == test_data$M)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

```
## The Accuracy is  0.1967213
```

Visualization:

Plotting the graph between the actual and predicted values.

```
# 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))
```



From the above visualization, it can be observed that the values predicted are far from the actual values and therefore the accuracy obtained is less which is approximately 19.6%

Naïve Bayes:

Naive Bayes is a probabilistic machine learning algorithm used for classification tasks. It is based on Bayes' theorem, which describes the probability of an event occurring based on prior knowledge of conditions that might be related to the event.

naiveBayes is the function we used to fit the model on the train data

The nb_model is being trained to predict the value of a binary target variable ‘M’ based on the values of three predictor variables ‘V’, ‘H’, and ‘S’ using our data.

summary function displays the statistics of our nb_model.

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
```

Using predict function we are going to predict the values on the test data.

Predicting the values using trained nb_model on test_data.

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

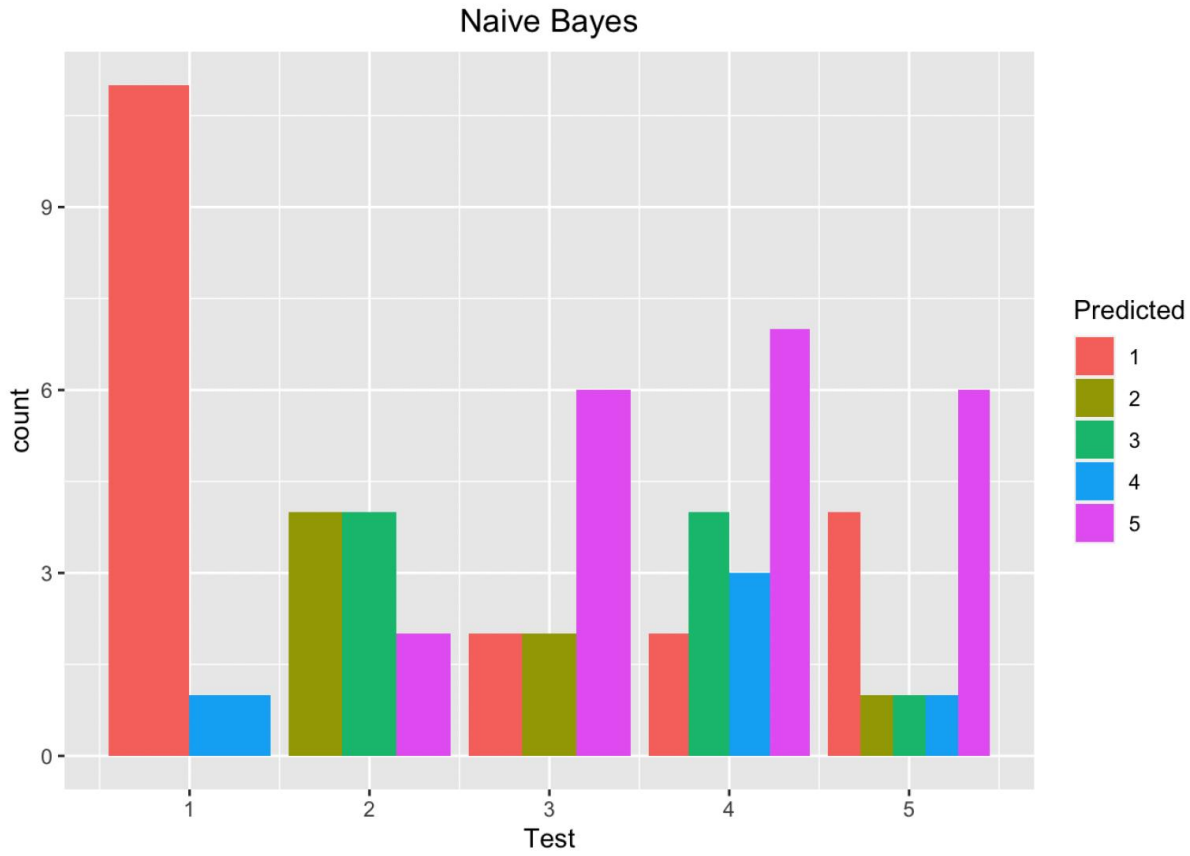
Calculating accuaracy using the predictions ie, nb_predictions.

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

```
## The Accuracy is  0.3934426
```

Visualization:

Plotting the graph between the actual and predicted values.



From the above graph we can describe that when actual value is 1 we have high 1 predicted values and some of the values are 4 when 1 is the actual value, when 2 is the actual value we got the predicted value as 2,3 and 5, coming to 3 when 3 is the actual value we got higher count 5 value. For 3,4,5 are the actual values we got the predicted value as 5 in all the 3 cases which means our model does not fit best on the dataset.

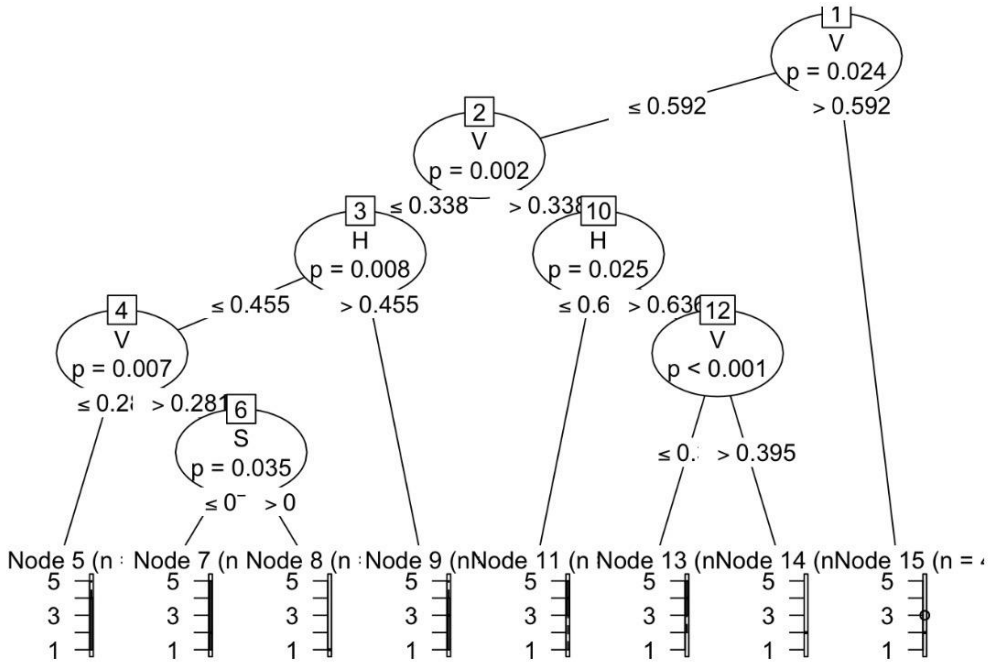
We got the accuracy of 39% which is better compared to logistic regression.

Decision Tree: -

Decision Trees are hierarchal structure which consists of branches and nodes where nodes represent the probabilities and branch indicates the output of the node. First we set the output file to get decision tree image in that file. And created a decision tree for our model using ctree function and plot the decision tree into our decision_tree png file.

We have imported the party package for decision tree usage. We have used the function called **ctree** which is there in the party package for fitting the model on the tarin data.

```
# 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)
```



Coming to above plot the data is divided into different nodes and each level with different independent variable we are using we are going to predict the output and fitting the model.

While coming to the top with the independent variable V and with the p=0.024 we got the accuracy of more than 59%.

Support Vector Machine: -

Support Vector Machine is a model which uses hyperplane and separates the classes in the plane.

We trained the SVM model using **svm** function and summarized the model to get the details about it.

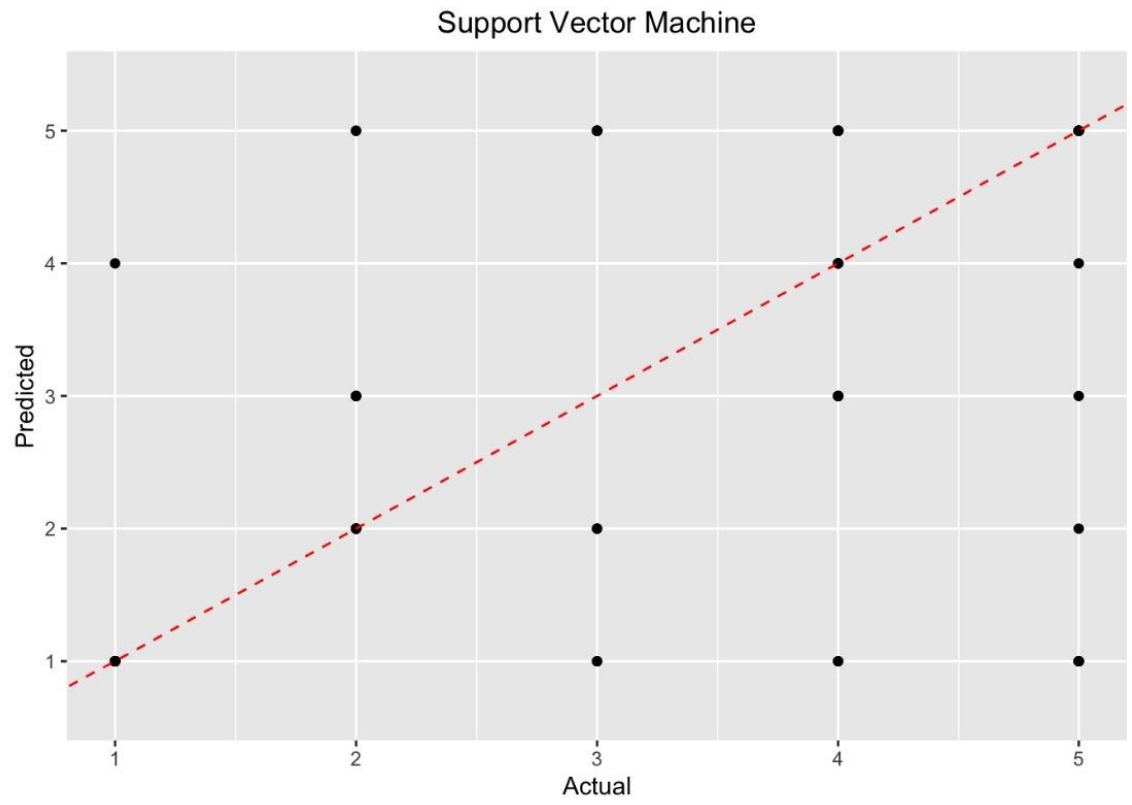
```
# fit SVM model
svm_model <- svm(M ~ V + H + S, data = data)
# summarize the model
summary(svm_model)
```

After training we predict on unseen data which is the test data, and the prediction value is

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

After predicting the values, the scatter plot is drawn between the actual and predicted values, here we the plot is divided into different classes.

Visualization:



For some classes, the model performs without the much classification error, but for most other classes the misclassification rate is very high. For example, class 1 has only one misclassification. On the other hand, for other classes the predicted results have high entropy like class 3, in which not even one sample was correctly classified.

Linear Discriminant Analysis: -

LDA is dimensionality reduction process where we reduce the number of dimensions, and it also helps in linear combination.

We find the patterns in the training data using fit **lda** function and view the summary of the model where we get the detailed view of the model.

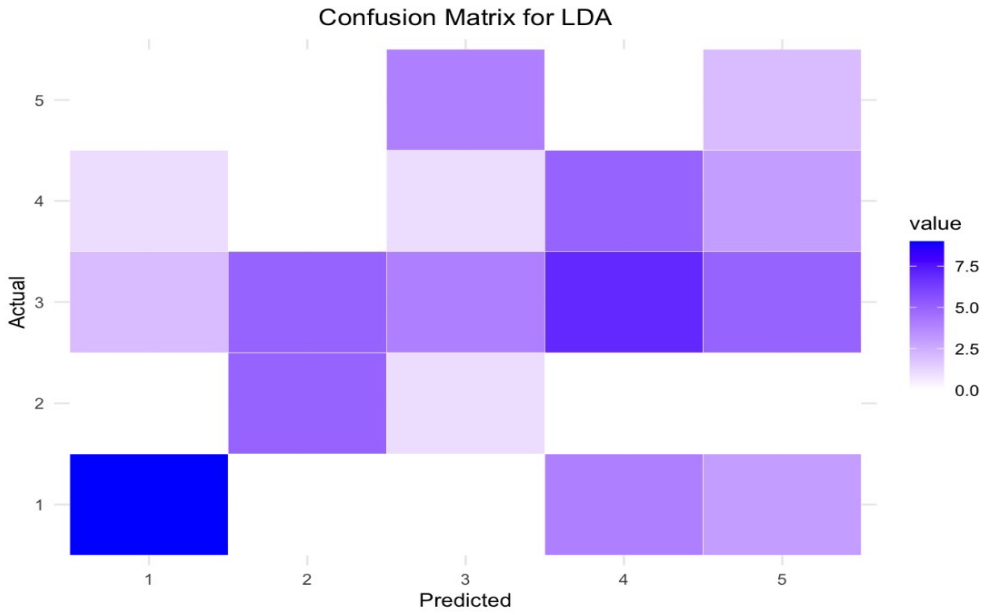
```
# fit LDA model
model_lda <- lda(M ~ V + H + S, data = data)
# summarize the model
summary(model_lda)
```

And we used that patterns to predict on the test data. Where the prediction value is

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

Visualization:

We are going to visualize the predict value vs actual value with confusion matrix.



There are 5 classes we predicted them for each class and if we get the darker color for the prediction means we got the best prediction and we got that in between actual and predicted class which is 1 and class 2 compared to class 3 ,4 and 5. In class 3 the more predicted as class 4 compared to class 3 as we got the dark color for 3 to 4.

Quadratic Discriminant Analysis: -

QDA is also same as LDA which is used to find the linear combination of features that separate the classes in the space. QDA is used when there is different covariance in the each of the class. So, then QDA performs best compared to LDA.

With the training data we developed the QDA model using the **qda** function and to view the details of the model and how the model trained we used the summary function.

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

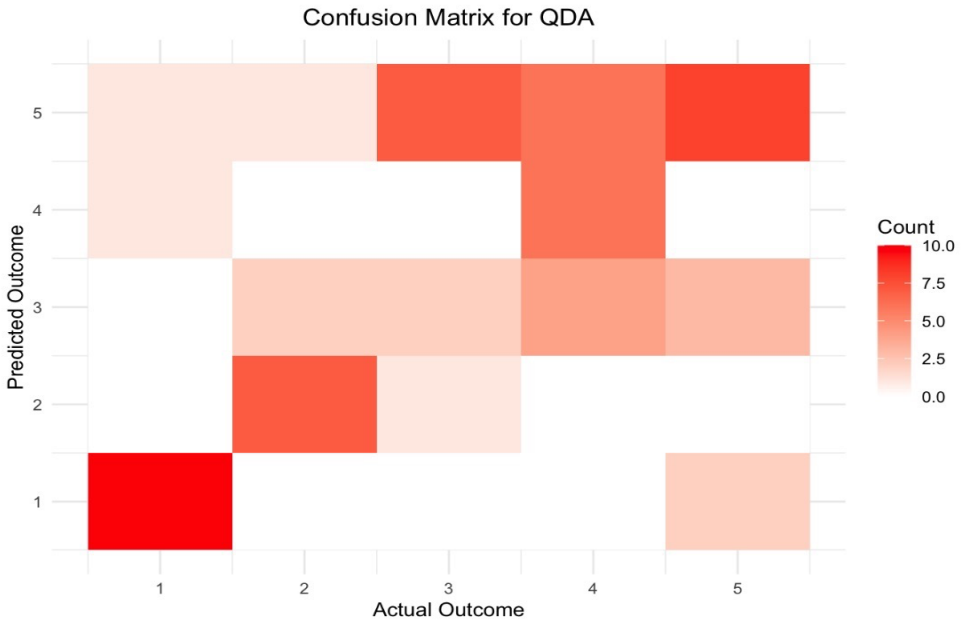
After the model training we predicted the outcome using the test data and found the accuracy for the model.The accuracy is 0.5409.

```
# 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  0.5409836
```

Visualization:

Again, for the qda also we plot the confusion matrix same as lda.



Here we observe that for the predicted class 1,5 are near to the actual values compared to other classes 2,3and 4 which are very low compared to actual values.

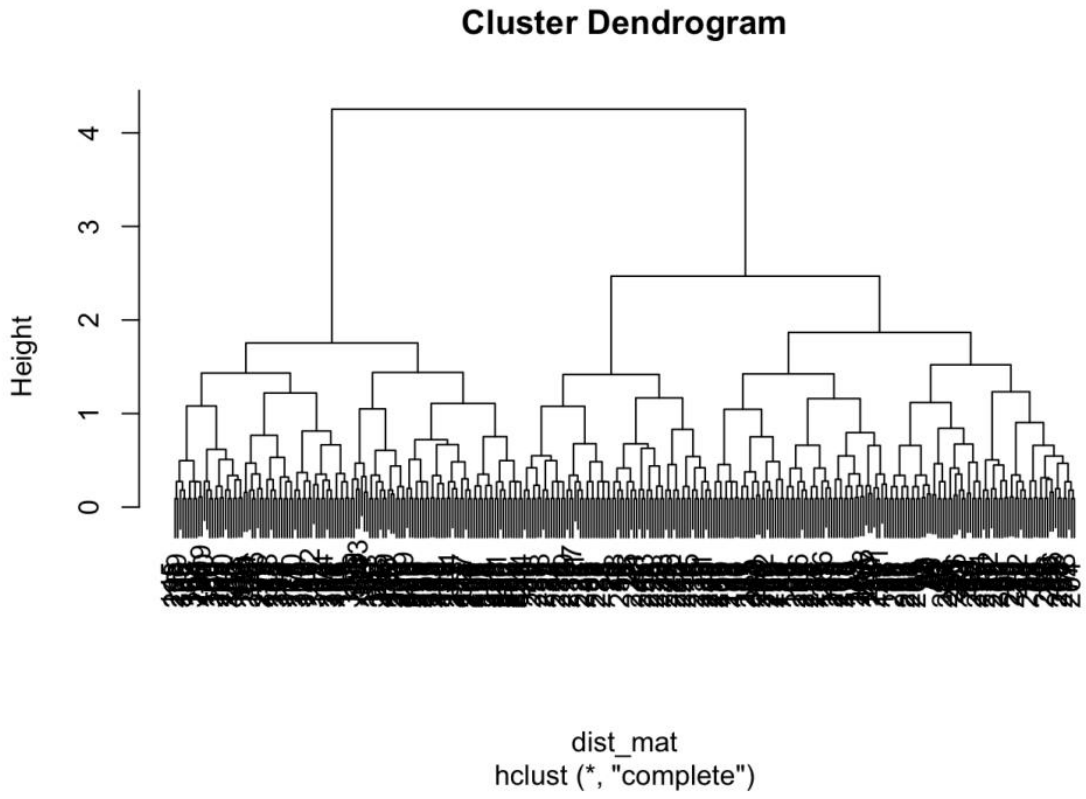
Hierarchical Clustering: -

Clustering refers to a very broad set of techniques for finding subgroups, or clusters, in a data set. We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other. We have different types of clustering like K-means and hierarchical. In K-means we have to define the k value based on that groups will be formed while for hierarchical clustering is an alternative approach which does not require a particular choice of k.

We have used the **hclust** function for fitting the train data and train the model.

```
set.seed(123)
train_pct <- 0.7
n <- nrow(data)
train_indices <- sample.int(n, train_pct*n, replace=FALSE)
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)
```

In hierarchical clustering a dendrogram is built starting from the leaves and combining clusters up to the trunk.



Based on the independent variables we have defined the number of clusters, as we have 3 we have defined the clusters as 3 and based on that we have predicted values.

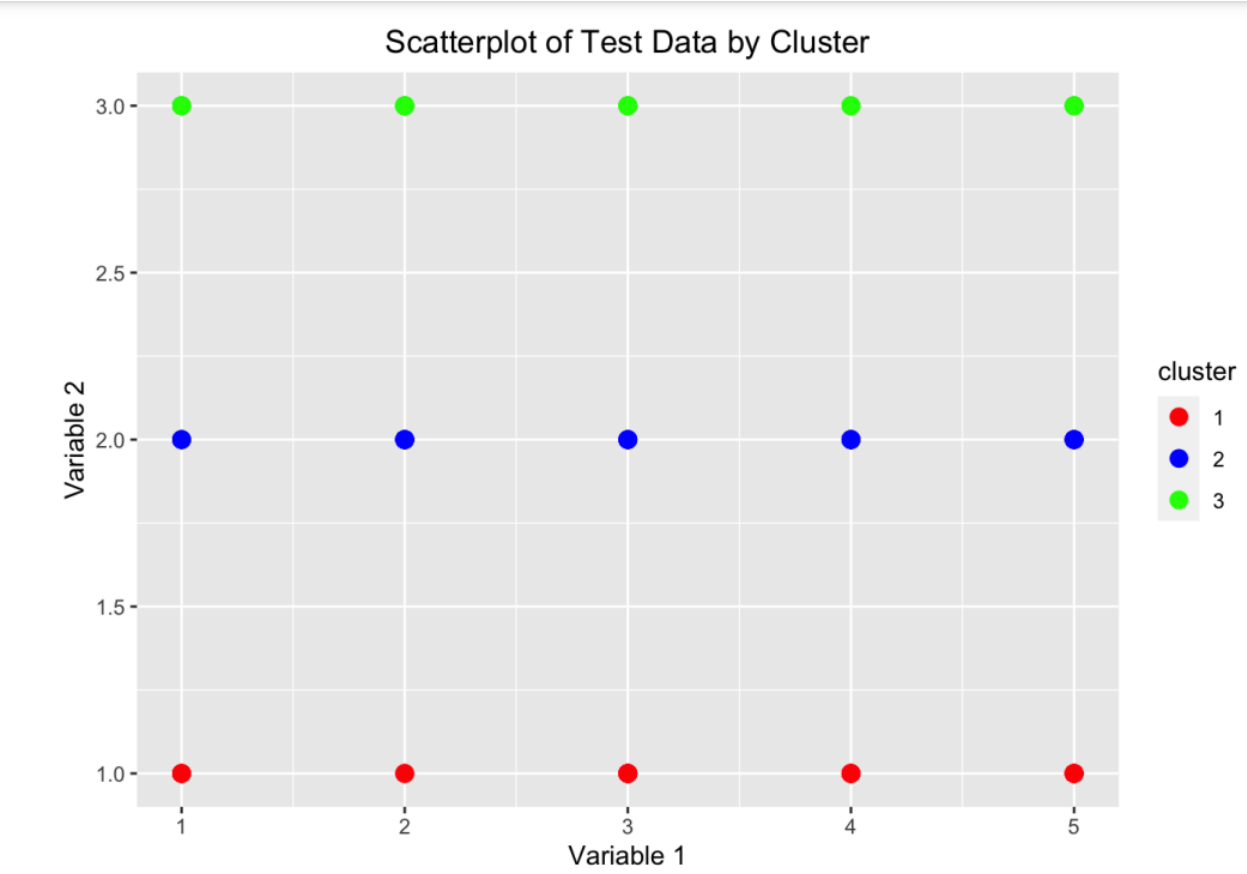
```r
# Define the number of clusters
num_clusters <- 3

# 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
```

Visualization:

```r
# 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)
```

Using the scatterplot we have visualized the predicted data using the hierarchical clustering.



The above graph depicts that as we have taken 3 clusters and all the output variables were used to predict the future values.

Conclusion:

For our dataset with decision tree, QDA and Hierarchical clustering our prediction is best compared to all other models performed.

Team Members:

Alekhya Monavarthi -50469035

Jayasree Keesari - 50477962

Preethi Abhilasha Vaddi-50483865

Vinitha Vudhayagiri-50478854