# harinris_project1

Harin Rishabh

2024-04-11

# Ingesting dataset

```
library(readxl)
excel_file = "Mine_dataset.xls"
df = read_excel(excel_file, sheet=2)
```

# Data pre-processing

```
names(df) = c("Voltage", "Height", "Soil_Type", "Landmine_Type")
df$Landmine_Type = factor(df$Landmine_Type)
```

Looks like Soil Type feature was accidentally normalized. Converting it back to a categorical feature with 6 values.

```
df$Soil_Type = cut(as.numeric(df$Soil_Type), 6, labels = FALSE)
df$Soil_Type = factor(df$Soil_Type)
df
```

```
## # A tibble: 338 × 4
##    Voltage Height Soil_Type Landmine_Type
##      <dbl>  <dbl> <fct>     <fct>
##  1   0.338  0     1         1
##  2   0.320  0.182 1         1
##  3   0.287  0.273 1         1
##  4   0.256  0.455 1         1
##  5   0.263  0.545 1         1
##  6   0.241  0.727 1         1
##  7   0.254  0.818 1         1
##  8   0.235  1     1         1
##  9   0.353  0     4         1
## 10   0.335  0.182 4         1
## # ℹ 328 more rows
```

# Data Exploration and Analysis

```
head(df)
```

```
## # A tibble: 6 × 4
##   Voltage Height Soil_Type Landmine_Type
##     <dbl>  <dbl> <fct>     <fct>
## 1   0.338  0     1         1
## 2   0.320  0.182 1         1
## 3   0.287  0.273 1         1
## 4   0.256  0.455 1         1
## 5   0.263  0.545 1         1
## 6   0.241  0.727 1         1
```

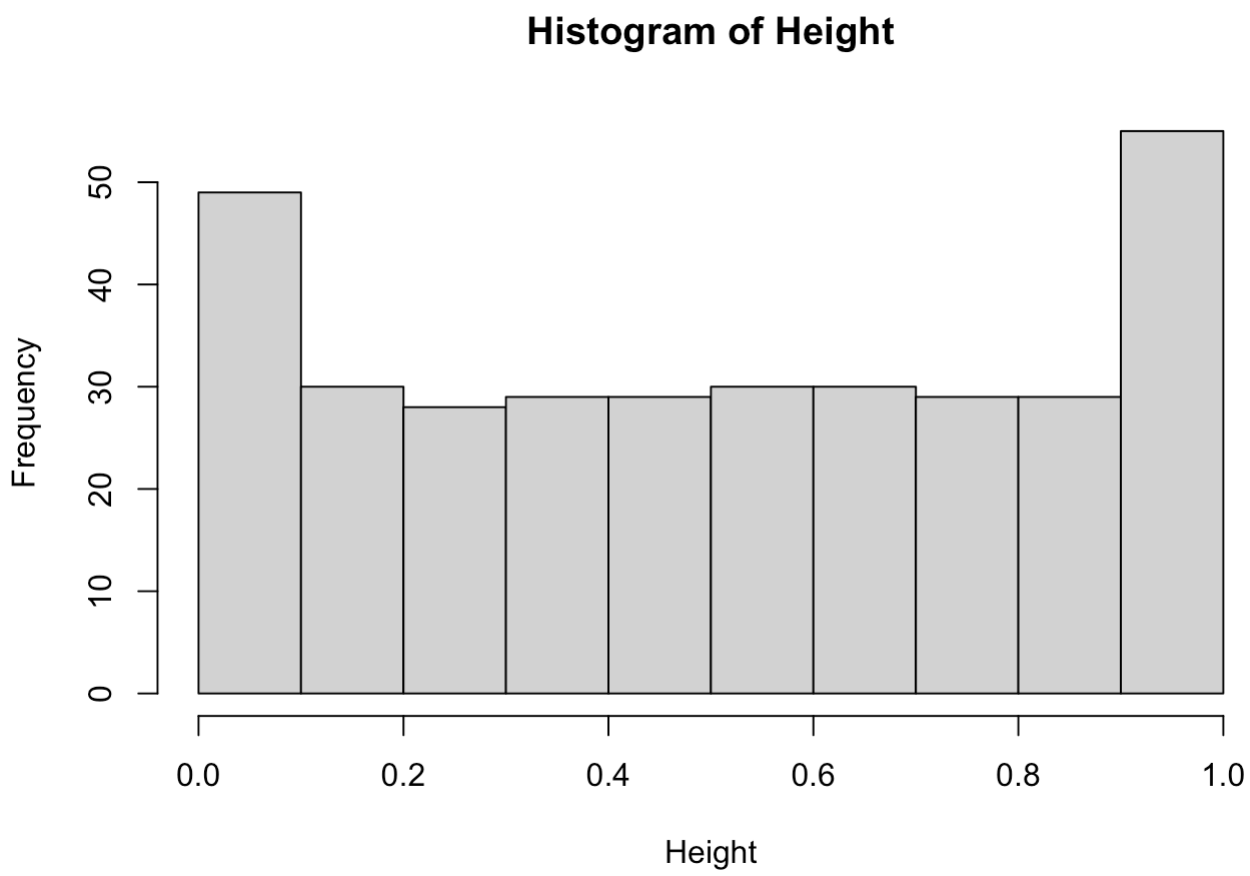
```
summary(df)
```

```
##     Voltage           Height       Soil_Type Landmine_Type
##  Min.   :0.1977   Min.   :0.0000   1:59      1:71
##  1st Qu.:0.3097   1st Qu.:0.2727   2:51      2:70
##  Median :0.3595   Median :0.5455   3:56      3:66
##  Mean   :0.4306   Mean   :0.5089   4:57      4:66
##  3rd Qu.:0.4826   3rd Qu.:0.7273   5:58      5:65
##  Max.   :1.0000   Max.   :1.0000   6:57
```

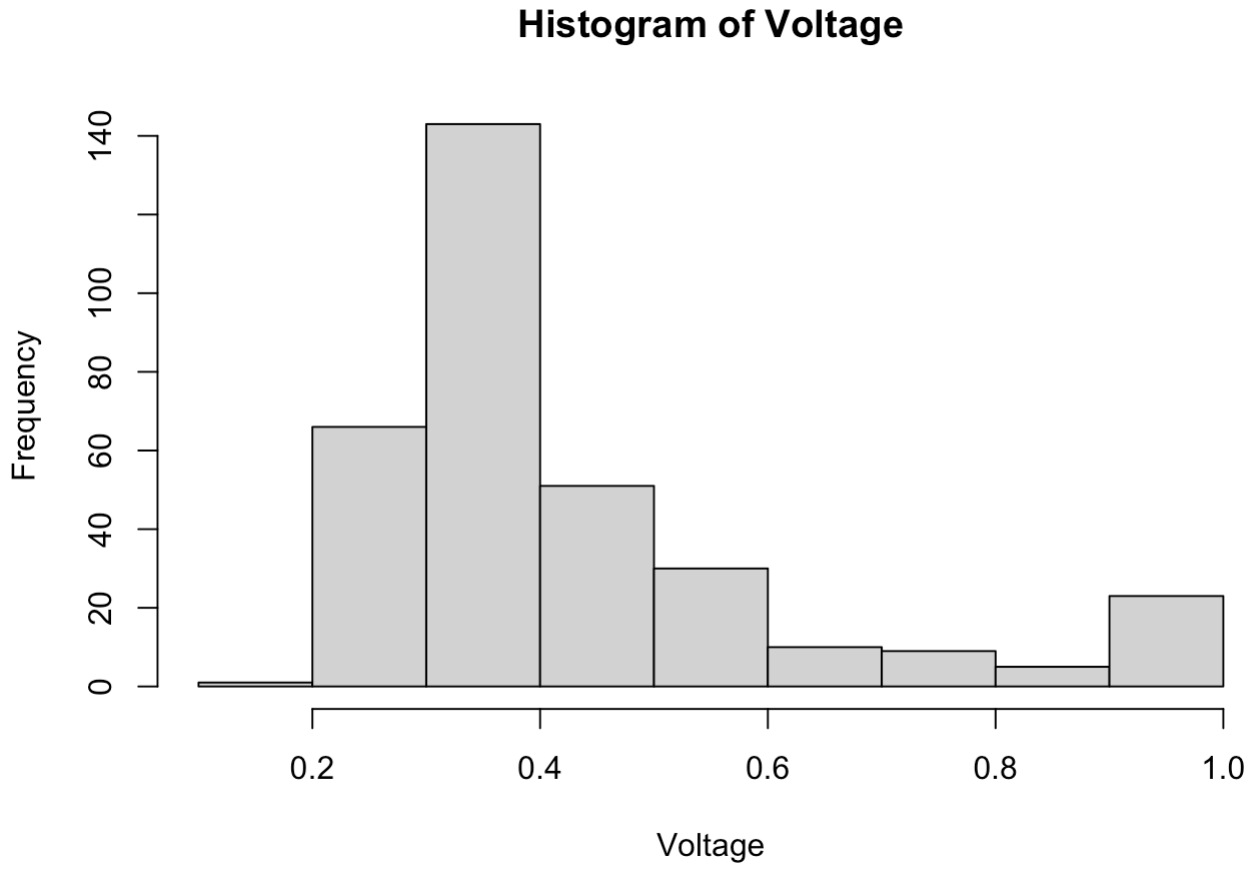
```
# Checking for columns with any missing values
colSums(is.na(df))
```

```
##       Voltage        Height     Soil_Type Landmine_Type
##             0             0             0             0
```

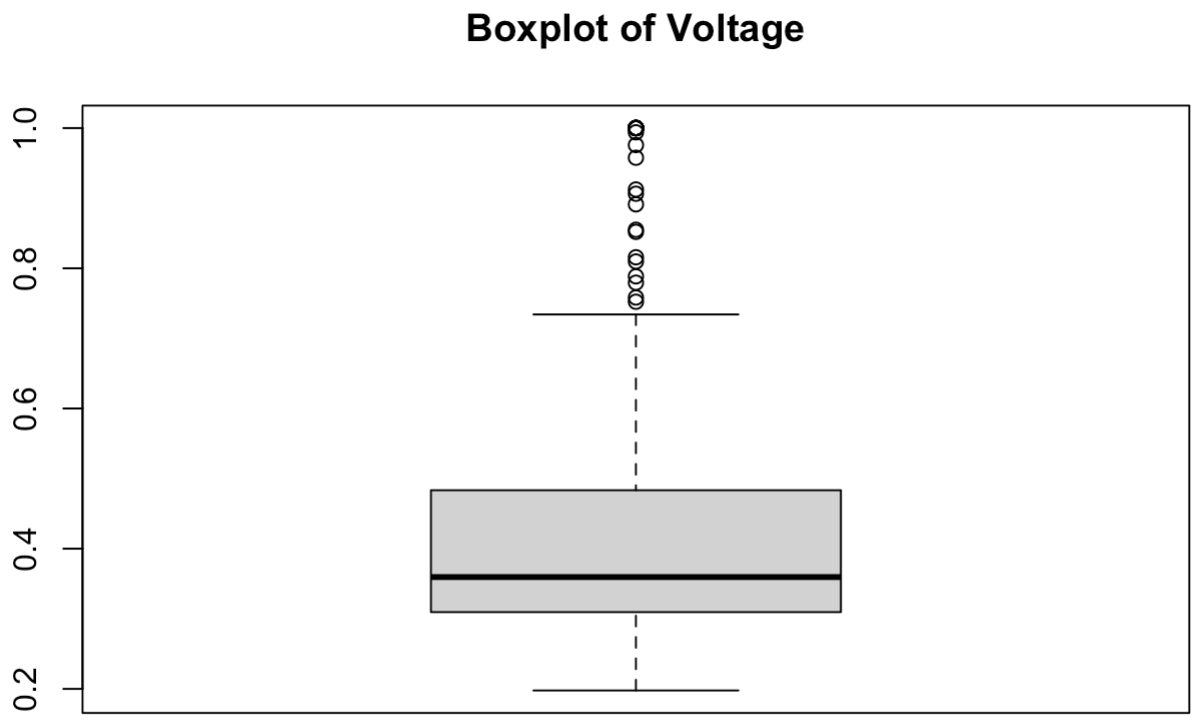
```
# Histogram for Height
hist(df$Height, main="Histogram of Height", xlab="Height")
```




```
# Histogram for Voltage
hist(df$Voltage, main="Histogram of Voltage", xlab="Voltage")
```



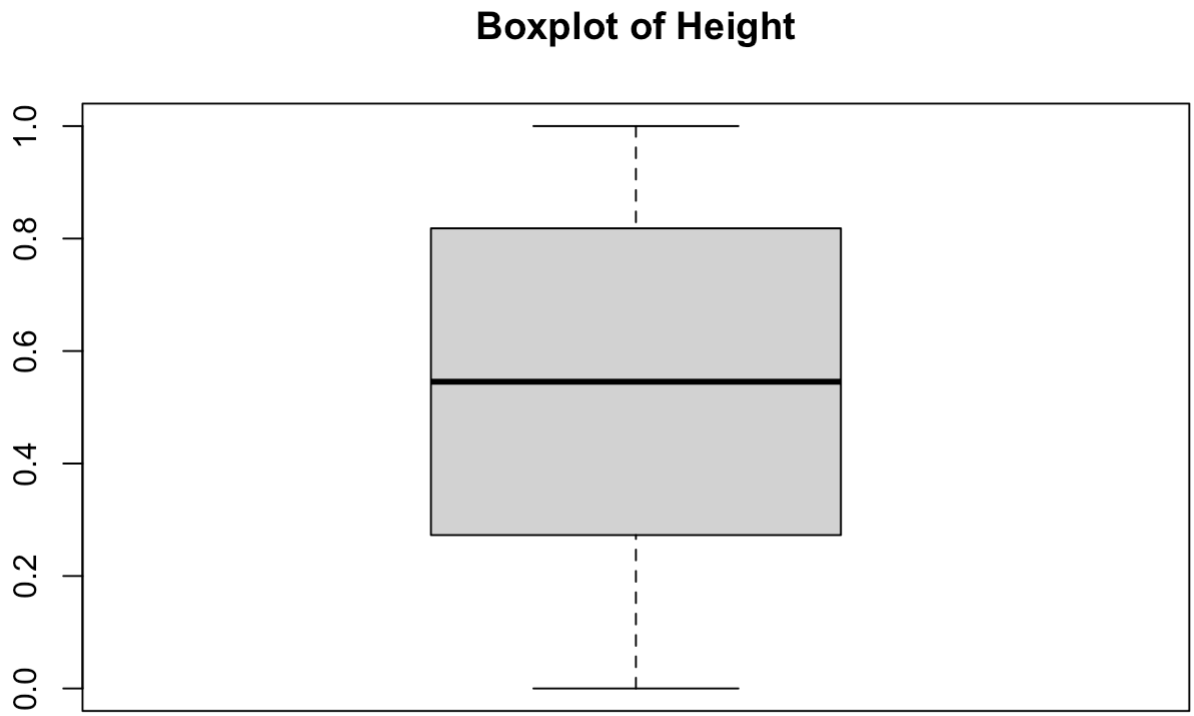
```
# Boxplot for Voltage
boxplot(df$Voltage, main="Boxplot of Voltage")
```



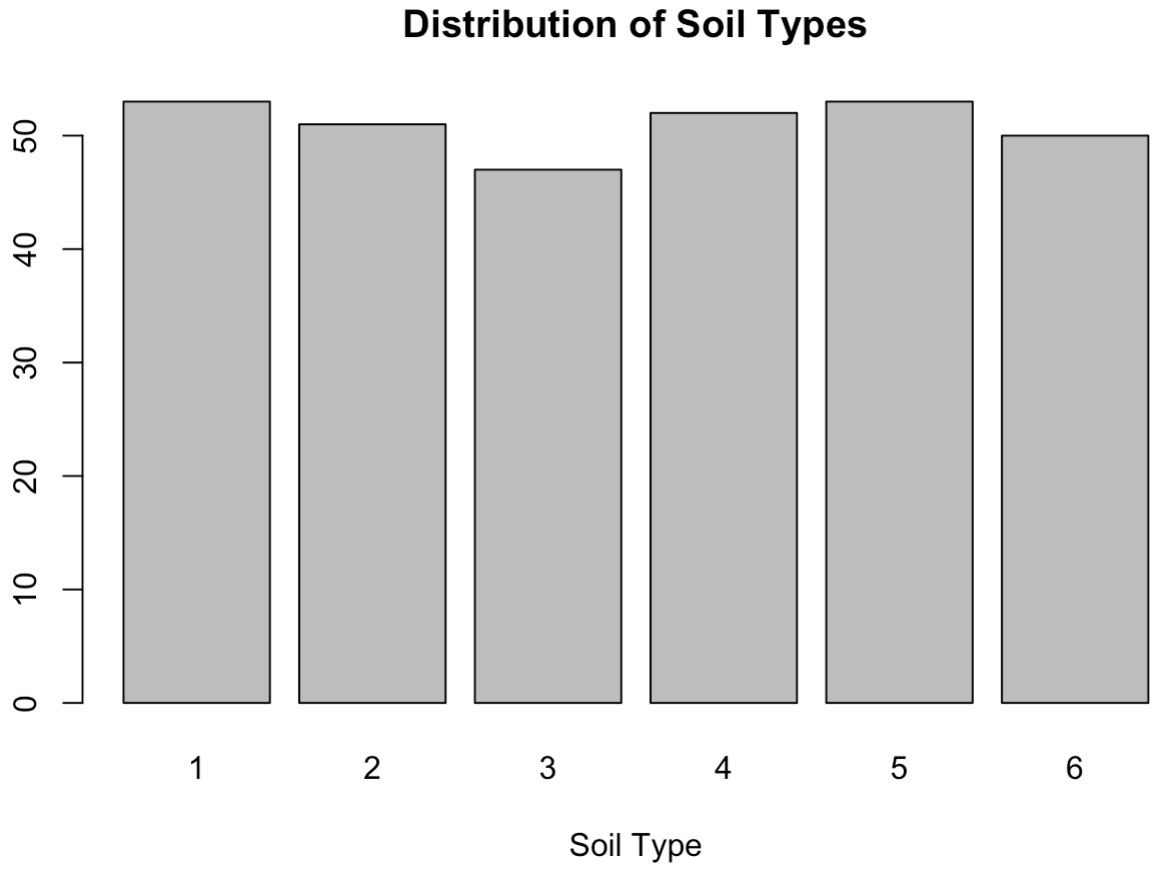

Voltage column has a few outliers. We need to remove these. For this, we use IQR technique.

```
# Calculate the interquartile range (IQR)
Q1 <- quantile(df$Voltage, 0.25)
Q3 <- quantile(df$Voltage, 0.75)
IQR <- Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Remove outliers
df <- df[df$Voltage >= lower_bound & df$Voltage <= upper_bound, ]
```

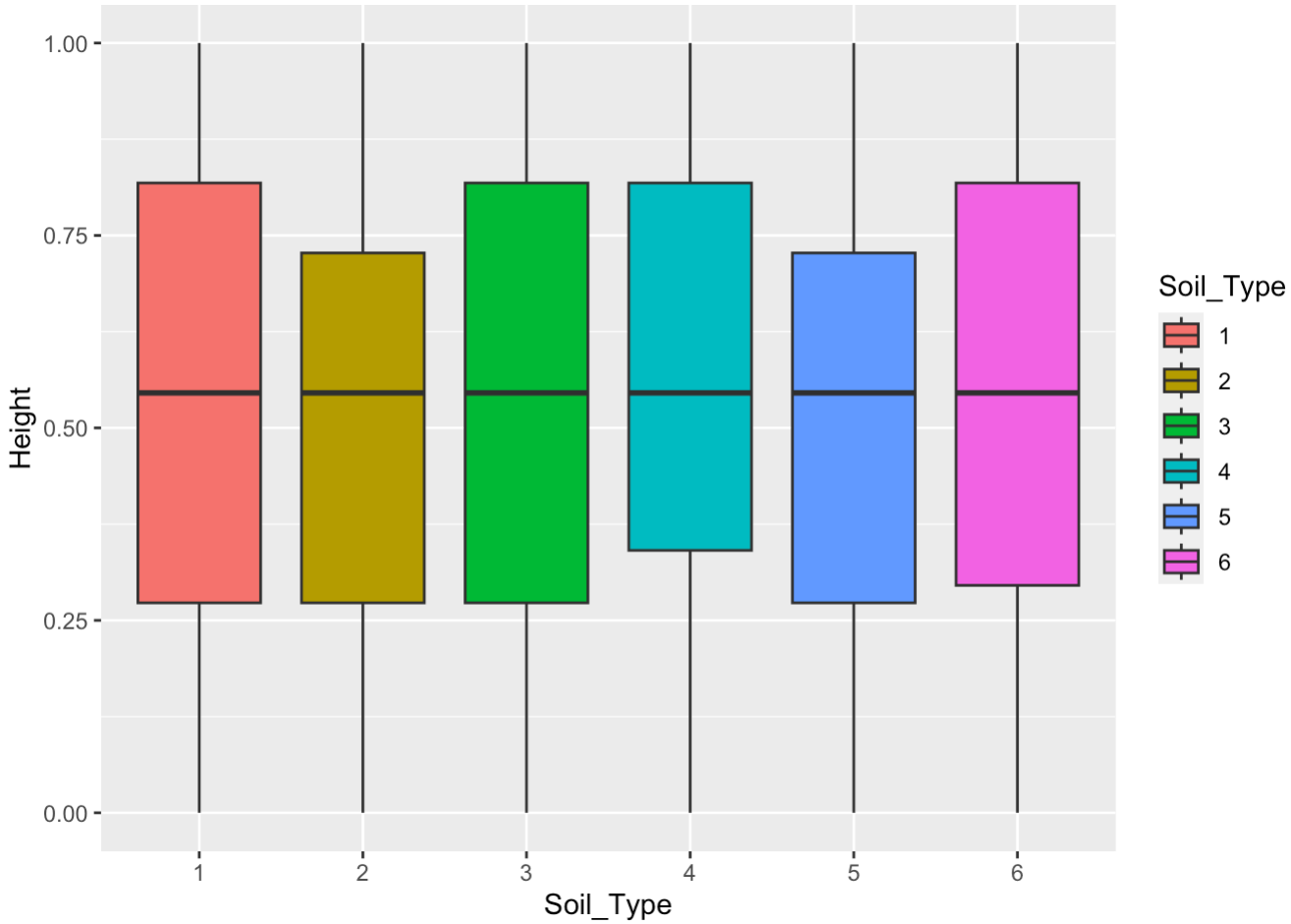
```
# Boxplot for Height
boxplot(df$Height, main="Boxplot of Height")
```



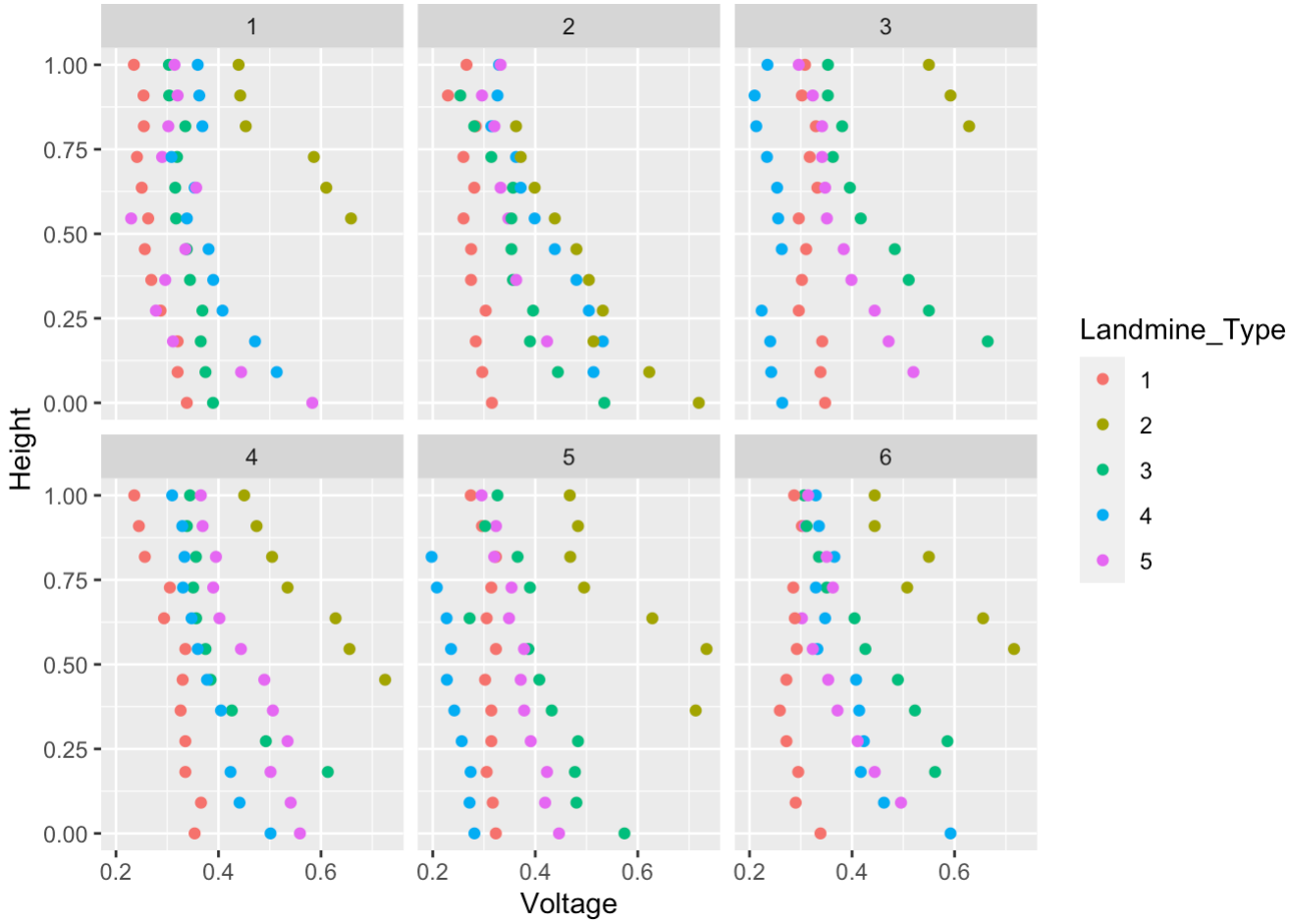

```
# Bar plot and Box plot for Soil Type
barplot(table(df$Soil_Type), main="Distribution of Soil Types", xlab="Soil Type")
```



```
# Box plot for examining distributions
library(ggplot2)
ggplot(df, aes(x = Soil_Type, y = Height, fill = Soil_Type)) + geom_boxplot()
```




```
# Using ggplot2 for a faceted scatter plot
ggplot(df, aes(x = Voltage, y = Height, color = Landmine_Type)) +
    geom_point() +
    facet_wrap(~Soil_Type)
```




Creating test and train splits

```
sample = sample(c(TRUE, FALSE), nrow(df), replace=TRUE, prob=c(0.75,0.25))
train_data = df[sample, ]
test_data = df[!sample, ]
dim(train_data)
```

```
## [1] 243   4
```

```
dim(test_data)
```

```
## [1] 63  4
```

# Running classification models

## Naive Bayes Classifier

```
library(naivebayes)
```

```
## Warning: package 'naivebayes' was built under R version 4.3.2
```

```
## naivebayes 1.0.0 loaded
```

```
## For more information please visit:
```

```
## https://majkamichal.github.io/naivebayes/
```

```
# Train the Naive Bayes classifier
nb_model <- naive_bayes(Landmine_Type ~ ., data = train_data)

# Make predictions on the test data
nb_predictions <- predict(nb_model, newdata = test_data)
```

```
## Warning: predict.naive_bayes(): more features in the newdata are provided as
## there are probability tables in the object. Calculation is performed based on
## features to be found in the tables.
```

```
# View the predictions
head(nb_predictions)
```

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

Printing evaluation metrics

```
library(caret)
```

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

```
# Define a confusion matrix
conf_matrix_nb = confusionMatrix(data = nb_predictions, reference = test_data$Landmin
e_Type)
conf_matrix_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5
##          1 13  0  2  4  3
##          2  0  7  0  0  0
##          3  0  6  3  6  5
##          4  1  0  0  3  0
##          5  0  0  2  3  5
##
## Overall Statistics
##
##                Accuracy : 0.4921
##                  95% CI : (0.3638, 0.6211)
##     No Information Rate : 0.254
##     P-Value [Acc > NIR] : 4.101e-05
##
##                   Kappa : 0.377
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9286   0.5385  0.42857  0.18750  0.38462
## Specificity            0.8163   1.0000  0.69643  0.97872  0.90000
## Pos Pred Value         0.5909   1.0000  0.15000  0.75000  0.50000
## Neg Pred Value         0.9756   0.8929  0.90698  0.77966  0.84906
## Prevalence             0.2222   0.2063  0.11111  0.25397  0.20635
## Detection Rate         0.2063   0.1111  0.04762  0.04762  0.07937
## Detection Prevalence   0.3492   0.1111  0.31746  0.06349  0.15873
## Balanced Accuracy      0.8724   0.7692  0.56250  0.58311  0.64231
```

```
# Extract accuracy
accuracy = conf_matrix_nb$overall["Accuracy"]

print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.492063492063492"
```

# Support Vector Classsification

```
library(e1071)

# Train the SVM classifier
svm_model <- svm(Landmine_Type ~ ., data = train_data)

# Make predictions on the test data
svm_predictions <- predict(svm_model, newdata = test_data)

# View the predictions
head(svm_predictions)
```

```
## 1 2 3 4 5 6
## 1 1 1 1 1 1
## Levels: 1 2 3 4 5
```

Printing evaluation metrics

```
# Define a confusion matrix
conf_matrix_svm = confusionMatrix(data = svm_predictions, reference = test_data$Landm
ine_Type)
conf_matrix_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5
##          1 11  0  0  5  3
##          2  0  7  0  0  0
##          3  3  6  6 11 10
##          4  0  0  0  0  0
##          5  0  0  1  0  0
##
## Overall Statistics
##
##                Accuracy : 0.381
##                  95% CI : (0.2615, 0.512)
##     No Information Rate : 0.254
##     P-Value [Acc > NIR] : 0.01778
##
##                   Kappa : 0.2659
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.7857   0.5385  0.85714    0.000  0.00000
## Specificity            0.8367   1.0000  0.46429    1.000  0.98000
## Pos Pred Value         0.5789   1.0000  0.16667      NaN  0.00000
## Neg Pred Value         0.9318   0.8929  0.96296    0.746  0.79032
## Prevalence             0.2222   0.2063  0.11111    0.254  0.20635
## Detection Rate         0.1746   0.1111  0.09524    0.000  0.00000
## Detection Prevalence   0.3016   0.1111  0.57143    0.000  0.01587
## Balanced Accuracy      0.8112   0.7692  0.66071    0.500  0.49000
```

```
# Extract accuracy
accuracy = conf_matrix_svm$overall["Accuracy"]

print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.380952380952381"
```

# K-nearest neighbours

```
library(class)
# Train the KNN classifier
knn_model <- knn(train = train_data[, -4], test = test_data[, -4], cl = train_data$La
ndmine_Type, k = 5)

# View the predictions
head(knn_model)
```

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

Printing evaluation metrics

```
# Define a confusion matrix
conf_matrix_knn = confusionMatrix(data = knn_model, reference = test_data$Landmine_Ty
pe)
conf_matrix_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3 4 5
##          1 3 0 0 4 3
##          2 0 3 0 0 0
##          3 6 3 3 4 5
##          4 2 7 0 5 2
##          5 3 0 4 3 3
##
## Overall Statistics
##
##                Accuracy : 0.2698
##                  95% CI : (0.1657, 0.3965)
##     No Information Rate : 0.254
##     P-Value [Acc > NIR] : 0.4333
##
##                   Kappa : 0.0994
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.21429  0.23077  0.42857  0.31250  0.23077
## Specificity           0.85714  1.00000  0.67857  0.76596  0.80000
## Pos Pred Value        0.30000  1.00000  0.14286  0.31250  0.23077
## Neg Pred Value        0.79245  0.83333  0.90476  0.76596  0.80000
## Prevalence            0.22222  0.20635  0.11111  0.25397  0.20635
## Detection Rate        0.04762  0.04762  0.04762  0.07937  0.04762
## Detection Prevalence  0.15873  0.04762  0.33333  0.25397  0.20635
## Balanced Accuracy     0.53571  0.61538  0.55357  0.53923  0.51538
```

```
# Extract accuracy
accuracy = conf_matrix_knn$overall["Accuracy"]

print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.26984126984127"
```

# Logistic Regression

```
library(nnet)

# Train the multinomial logistic regression model
logit_model <- multinom(Landmine_Type ~ ., data = train_data)
```

```
## # weights:  45 (32 variable)
## initial  value 391.093413
## iter  10 value 320.973653
## iter  20 value 271.477855
## iter  30 value 256.826459
## iter  40 value 256.030827
## iter  50 value 254.796263
## iter  60 value 254.683017
## iter  70 value 254.664855
## iter  80 value 254.659519
## iter  90 value 254.651637
## iter 100 value 254.636447
## final  value 254.636447
## stopped after 100 iterations
```

```
# Make predictions on the test data
logit_predictions <- predict(logit_model, newdata = test_data, type = "class")

# View the predictions
head(logit_predictions)
```

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

Printing evaluation metrics

```
# Define a confusion matrix
conf_matrix_logit = confusionMatrix(data = logit_predictions, reference = test_data$L
andmine_Type)
conf_matrix_logit
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5
##          1 11  0  0  7  2
##          2  0  9  0  0  0
##          3  0  4  4  6  7
##          4  1  0  1  3  2
##          5  2  0  2  0  2
##
## Overall Statistics
##
##                Accuracy : 0.4603
##                  95% CI : (0.3339, 0.5906)
##     No Information Rate : 0.254
##     P-Value [Acc > NIR] : 0.000313
##
##                   Kappa : 0.3379
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.7857   0.6923  0.57143  0.18750  0.15385
## Specificity            0.8163   1.0000  0.69643  0.91489  0.92000
## Pos Pred Value         0.5500   1.0000  0.19048  0.42857  0.33333
## Neg Pred Value         0.9302   0.9259  0.92857  0.76786  0.80702
## Prevalence             0.2222   0.2063  0.11111  0.25397  0.20635
## Detection Rate         0.1746   0.1429  0.06349  0.04762  0.03175
## Detection Prevalence   0.3175   0.1429  0.33333  0.11111  0.09524
## Balanced Accuracy      0.8010   0.8462  0.63393  0.55120  0.53692
```

```
# Extract accuracy
accuracy = conf_matrix_logit$overall["Accuracy"]

print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.46031746031746"
```

# Decision Trees

```
library(tree)

# Train the classification tree
tree_model <- tree(Landmine_Type ~ ., data = train_data)

# Make predictions on the test data
tree_predictions <- predict(tree_model, newdata = test_data, type = "class")

# View the predictions
head(tree_predictions)
```

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

Printing evaluation metrics

```
# Define a confusion matrix
conf_matrix_tree = confusionMatrix(data = tree_predictions, reference = test_data$Lan
dmine_Type)
conf_matrix_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5
##          1 12  0  1  2  3
##          2  0  7  0  0  0
##          3  0  6  4 11  9
##          4  1  0  0  3  0
##          5  1  0  2  0  1
##
## Overall Statistics
##
##                Accuracy : 0.4286
##                  95% CI : (0.3046, 0.5595)
##     No Information Rate : 0.254
##     P-Value [Acc > NIR] : 0.001882
##
##                   Kappa : 0.3127
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.8571   0.5385  0.57143  0.18750  0.07692
## Specificity            0.8776   1.0000  0.53571  0.97872  0.94000
## Pos Pred Value         0.6667   1.0000  0.13333  0.75000  0.25000
## Neg Pred Value         0.9556   0.8929  0.90909  0.77966  0.79661
## Prevalence             0.2222   0.2063  0.11111  0.25397  0.20635
## Detection Rate         0.1905   0.1111  0.06349  0.04762  0.01587
## Detection Prevalence   0.2857   0.1111  0.47619  0.06349  0.06349
## Balanced Accuracy      0.8673   0.7692  0.55357  0.58311  0.50846
```

```
# Extract accuracy
accuracy = conf_matrix_tree$overall["Accuracy"]

print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.428571428571429"
```

# Random Forests

```
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
```

```
# Train the Random Forest classifier
rf_model <- randomForest(Landmine_Type ~ ., data = train_data)

# Make predictions on the test data
rf_predictions <- predict(rf_model, newdata = test_data)

# View the predictions
head(rf_predictions)
```

```
## 1 2 3 4 5 6
## 1 1 1 1 1 1
## Levels: 1 2 3 4 5
```

Printing evaluation metrics

```
# Define a confusion matrix
conf_matrix_rf = confusionMatrix(data = rf_predictions, reference = test_data$Landmin
e_Type)
conf_matrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5
##          1 11  0  1  1  2
##          2  0  7  1  1  0
##          3  0  0  4  4  8
##          4  0  6  0  6  1
##          5  3  0  1  4  2
##
## Overall Statistics
##
##                Accuracy : 0.4762
##                  95% CI : (0.3488, 0.6059)
##     No Information Rate : 0.254
##     P-Value [Acc > NIR] : 0.0001167
##
##                   Kappa : 0.3487
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.7857   0.5385  0.57143  0.37500  0.15385
## Specificity            0.9184   0.9600  0.78571  0.85106  0.84000
## Pos Pred Value         0.7333   0.7778  0.25000  0.46154  0.20000
## Neg Pred Value         0.9375   0.8889  0.93617  0.80000  0.79245
## Prevalence             0.2222   0.2063  0.11111  0.25397  0.20635
## Detection Rate         0.1746   0.1111  0.06349  0.09524  0.03175
## Detection Prevalence   0.2381   0.1429  0.25397  0.20635  0.15873
## Balanced Accuracy      0.8520   0.7492  0.67857  0.61303  0.49692
```

```
# Extract accuracy
accuracy = conf_matrix_rf$overall["Accuracy"]

print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.476190476190476"
```

# Results

The dataset consisted of 4 columns. We had 2 continuous value columns, height and voltage measurement. Soil type was a categorical variable that consisted of 6 types. The target variable was Mine type. We ran several classification algorithms to classify the mines into the predetermined labels.

The data had no missing values. The height column had some outliers which were removed to improve accuracy.The data was split into test and train samples.

Overall, the data was clean but too limited to achieve reasonable accuracy. After removing outliers we had around 300 observations which is too less for 3 features of which one was categorical. All our models give a max accuracy of around 45%.