CSE509
Project-1

Land Mines

Amulya Reddy Datla
UB Person number: 50560100
amulyare@buffalo.edu

Jagruthi Reddy Ghanapati
UB Person number: 50560478
jghanapa@buffalo.edu

Arun Karthik Periyaswamy
UB Person number: 50557719
aperiyas@buffalo.edu

1 Dataset Selection

We have selected the **Landmines** dataset after reviewing the provided datasets exasens and Landmines

2 Data Exploration and Understanding

We obtained the Landmines dataset, along with its associated metadata, from the UCI Machine Learning Repository (ucimlrepo). We separated the features and target variable into two distinct datasets X and y. The dataset comprises 338 observations, each consisting of three feature variables and one target variable.

```
> print(land_mines$metadata)
$uci_id
[1] 763

$name
[1] "Land Mines"

$repository_url
[1] "https://archive.ics.uci.edu/dataset/763/land+mines-1"

$data_url
[1] "https://archive.ics.uci.edu/static/public/763/data.csv"

$abstract
[1] "Detection of mines buried in the ground is very important in terms of safety of life and property. Many dif
ferent methods have been used in this regard; however, it has not yet been possible to achieve 100% success. Min
e detection process consists of sensor design, data analysis and decision algorithm phases. The magnetic anomaly
method works according to the principle of measuring the anomalies resulting from the object in the magnetic fie
ld that disturbs the structure of it, the magnetic field, and the data obtained at this point are used to determ
ine the conditions such as motion and position. The determination of parameters such as position, depth or direc
tion of motion using magnetic anomaly has been carried out since 1970."
```

Figure 1: figure 1

The above image is obtained by printing the metadata of the Landmines dataset. Repo link, source, citations, abstract and many other details about the dataset are printed as a part of metadata.

```
# Variable information
print(land_mines$variables)
```

```
##   name    role       type demographic
## 1    V Feature Continuous          NA
## 2    H Feature Continuous          NA
## 3    S Feature Continuous          NA
## 4    M  Target    Integer          NA
##
description
## 1                                                                        voltage: output voltage v
alue of FLC sensor due to magnetic distortion
## 2                                                                                                 h
igh: the height of the sensor from the ground
## 3 soil type: 6 different soil types depending on the moisture condition [dry and sandy, dry and humus, dry and limy, humi
d and sandy, humid and humus, humid and limy]
## 4                                                                   mine type: mine types commonly e
ncountered on land (5 different mine classes)
##   units missing_values
## 1     V             no
## 2    cm             no
## 3  <NA>             no
## 4  <NA>             no
```


Figure 2: figure 2

The above image obtained by running the R script gives the detailed information about the variables in the dataset those includes names, datatypes and demographics.

```
str(X)
```

```
## 'data.frame':    338 obs. of  3 variables:
##  $ V: num  0.338 0.32 0.287 0.256 0.263 ...
##  $ H: num  0 0.182 0.273 0.455 0.545 ...
##  $ S: num  0 0 0 0 0 0 0 0 0.6 0.6 ...
```

```
str(y)
```

```
## 'data.frame':    338 obs. of  1 variable:
##  $ M: int  1 1 1 1 1 1 1 1 1 1 ...
```

Figure 3: figure 3

The above image obtained by running the R script gives the details regarding internal structure of features and target. The str() function is used to show the internal structure of an object. X object contains the features of the landmines dataset y contains the target value of the landmines dataset.

```
#Converting target variable to factor
y<-factor(y$M)
str(y)
```

```
##  Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Figure 4: figure 4

The above image shows that target variable y is converted into a factor to represent in categorical values, to ease the classification task. As shown in the output it is classified into 5 levels. Each of the values of target will be categorized into any of those 5 factors.

3 Data Quality Assessment

The below image shows quality of the data by finding the duplicates from the features and target values. It is shown from the below image that the target values is having 333 duplicate values. It is obvious that the target values are factored so there will be duplicates occurrence of the target data.

```
# Finding duplicates in the features and target
duplicate_X <- sum(duplicated(land_mines$data$features))
duplicate_y <- sum(duplicated(land_mines$data$targets))
cat("Number of duplicate rows in X: ", duplicate_X, "\n")
```

```
## Number of duplicate rows in X:  0
```

```
cat("Number of duplicate entries in y: ", duplicate_y, "\n")
```

```
## Number of duplicate entries in y:  333
```

Figure 5: figure 5

4 Data Summary and Distributions

4.1 Summary

```
#Summary of the features
summary(X)
```

```
##        V                H                S         
##  Min.   :0.1977   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.3097   1st Qu.:0.2727   1st Qu.:0.2000  
##  Median :0.3595   Median :0.5455   Median :0.6000  
##  Mean   :0.4306   Mean   :0.5089   Mean   :0.5036  
##  3rd Qu.:0.4826   3rd Qu.:0.7273   3rd Qu.:0.8000  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

Figure 6: figure 6

The summary of the Landmines dataset shows that V ranges from 0.1977 to 1.0000, with a mean of 0.4306 and a median of 0.3595, indicating a slight right skew. H ranges from 0.0000 to 1.0000, with a mean of 0.5089 and median of 0.5455, indicating a balanced distribution. S ranges from 0.0000 to 1.0000, with a mean of 0.5036 and median of 0.6000, indicating a higher concentration towards the upper range.

4.2 Distributions

The first histogram in the below image shows the distribution of voltage measurements, with a prominent peak around 0.3V and a right-skewed pattern that gradually decreases until about 0.75V. It has a few outliers to the right most edge. The other plot in the image shows the distribution of height (H). There is a slight downward trend in the higher values. The counts are evenly distributed across all the values.

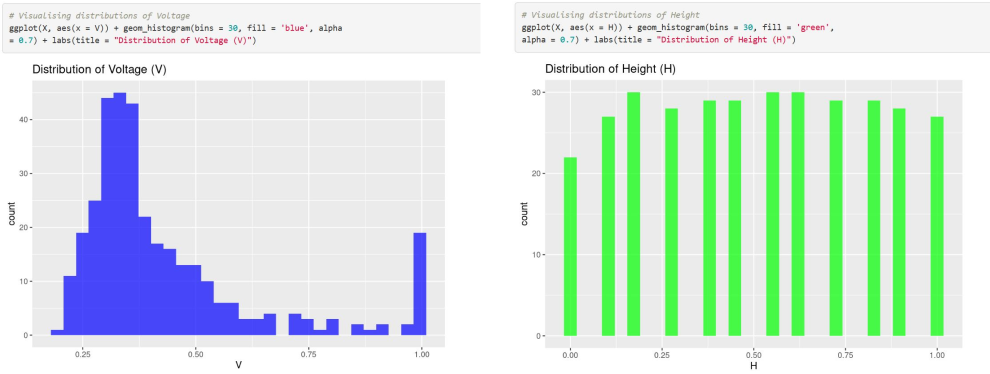

Figure 7: figure 7

The first histogram in the below image shows distribution of Soil (S). The plot explains that distribution of soil is evenly with very slight ups and downs. The other plot in the image shows the distribution of mine type. From the plot we can see the distribution of 5 different mine types. Their frequencies are almost evenly distributed in the dataset.

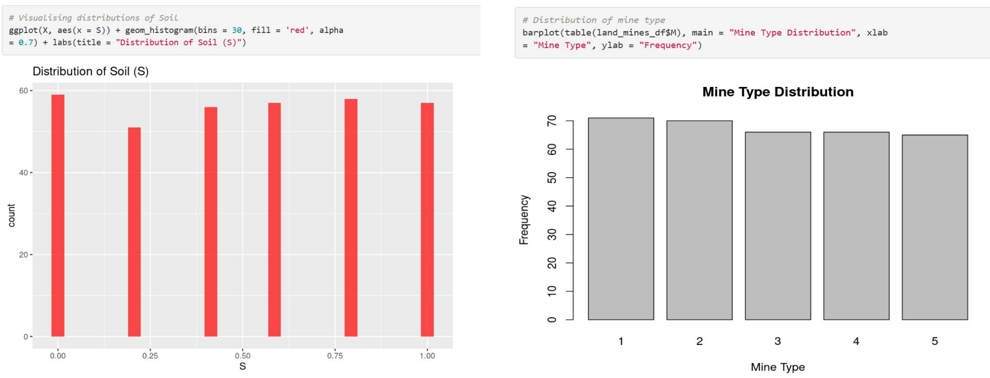

Figure 8: figure 8

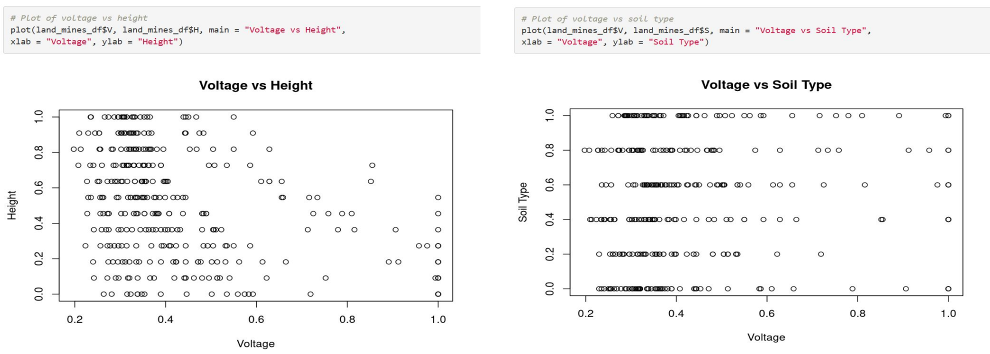

Figure 9: figure 9

The above figure shows the scatterplot of voltage vs height and voltage vs soil type. From the plots it is clearly seen that there is no linear relationship between th evariables in two plots. In the second plot points seem to cluster along horizontal lines. This suggests that there are distinct soil types, each characterized by a specific range of voltage values.

```r
# Plot of voltage by minetype
boxplot(V ~ M, data = land_mines_df, main = "Voltage by Mine Type",
xlab = "Mine Type", ylab = "Voltage")
```

```r
# Plot of height by minetype
boxplot(H ~ M, data = land_mines_df, main = "Height by Mine Type",
xlab = "Mine Type", ylab = "Height")
```

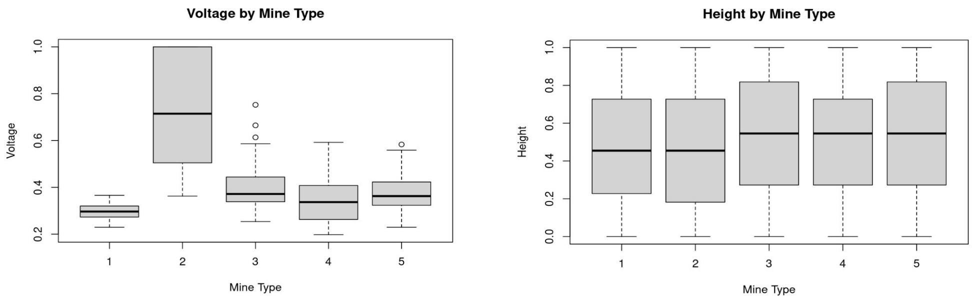

Figure 10: figure 10

The above image consisting of two box plots above, titled "Voltage by Mine Type" and "Height by Mine Type," illustrate the distribution of voltage and height values for different mine types. The "Voltage by Mine Type" plot shows variation in voltage levels, with Mine Type 2 having the highest median voltage and Mine Type 1 having the lowest. Some mine types exhibit outliers, indicating anomaly voltage readings. In contrast, the "Height by Mine Type" plot reveals that the distributions of height values across mine types are relatively similar, with only slight differences in median and spread, and no apparent outliers in any category.

5 Principal Component Analysis (PCA)

```r
# Apply PCA
X_scaled <- scale(X)
pca <- prcomp(X_scaled, center = TRUE, scale. = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3
## Standard deviation     1.1770 0.9987 0.7856
## Proportion of Variance 0.4618 0.3325 0.2057
## Cumulative Proportion  0.4618 0.7943 1.0000
```

Figure 11: figure 11

The scale(X) function standardizes the features, ensuring a mean of 0 and standard deviation of 1 for easier comparison in ML algorithms. The prcomp() function performs PCA on the standardized data, reducing its dimensionality. The PCA results reveal three principal components: PC1 with the highest standard deviation of 1.1770, explaining 46.18% of the variance, PC2 with a standard deviation of 0.9987, accounting for 33.25% and PC3 with 0.7856, explaining 20.57%, resulting in a cumulative variance of 100%.

6 K-means Clustering

```r
# K-means clustering model
set.seed(123)
kmeans_result <- kmeans(X_scaled, centers = 3, nstart = 25)
print(kmeans_result)
```

```
## K-means clustering with 3 clusters of sizes 144, 144, 50
##
## Cluster means:
##            V          H          S
## 1 -0.3014557  0.1710922  0.8692317
## 2 -0.3948505  0.1484011 -0.9140660
## 3  2.0053616 -0.9201408  0.1291228
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 2
##  [38] 2 2 2 1 1 1 1 1 1 1 3 3 3 3 2 2 2 2 3 3 3 3 3 1 1 1 3 3 2 2 2 2 2 3 3 3 3
##  [75] 3 1 1 1 3 3 3 3 3 3 2 2 3 3 3 3 3 1 1 1 2 2 2 2 2 2 2 2 3 1 1 1 1 1 1 2 2
## [112] 2 2 2 2 2 3 1 1 1 1 1 1 1 3 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 1 1
## [149] 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 1 1 1 1 1 1 1 3 2 2
## [186] 2 2 2 2 2 3 1 1 1 1 1 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1
## [223] 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1 1 3 3 2 2 3 3 1 1 2 2
## [260] 2 3 3 1 1 3 3 3 2 3 3 1 1 2 2 2 2 1 1 1 3 2 2 2 1 1 1 1 3 2 2 3 1 1 1 2 2
## [297] 2 2 1 1 1 2 2 2 2 1 1 1 2 2 2 2 1 1 1 1 2 2 2 2 3 1 1 1 2 2 1 1 1 1 3 2 2
## [334] 2 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 194.8586 195.5825 104.2797
##  (between_SS / total_SS =  51.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Figure 12: figure 12

The above image shows a K-means clustering analysis where data points are grouped into 3 clusters of sizes 144, 144, 50 using 25 random starts. The cluster centroids represent the average values of the features for each cluster. The clustering achieved 51.1% between cluster to total sum of squares ratio, suggesting moderate cluster separation.

7 Logistic Regression

```r
# logistic regression model
data <- data.frame(X, y = as.factor(y))
model_multinom <- multinom(y ~ ., data = data)
```

```
## # weights:  25 (16 variable)
## initial  value 543.990014
## iter  10 value 411.174973
## iter  20 value 354.088472
## iter  30 value 353.431094
## final  value 353.427302
## converged
```

```r
summary(model_multinom)
```

```
## Call:
## multinom(formula = y ~ ., data = data)
##
## Coefficients:
##   (Intercept)        V         H          S
## 2  -32.771305 64.62394 13.504539 -3.7998918
## 3  -12.413425 31.03676  3.807539 -0.7349769
## 4   -5.547538 15.61813  1.148904 -0.1752905
## 5   -9.708252 25.07504  2.700693 -0.3493845
##
## Std. Errors:
##   (Intercept)        V         H         S
## 2    4.109183 7.610529 2.1051675 1.2949626
## 3    1.677159 4.096268 0.8511527 0.5984940
## 4    1.253255 3.453616 0.6502918 0.5223790
## 5    1.491917 3.824198 0.7613781 0.5681517
##
## Residual Deviance: 706.8546
## AIC: 738.8546
```

Figure 13: figure 13

The above multinomial logistic regression model converged after 30 iterations, reducing the deviance from 543.99 to 353.42. The coefficients represent log-odds for categories 2-5 relative to the baseline, with both positive and negative values indicating varying relationships with predictors. The fit for the model is summarized by a Residual Deviance of 706.85 and an AIC of 738.85.

8 K-Nearest Neighbor

```
> print(knn_model)
k-Nearest Neighbors

271 samples
  3 predictor
  5 classes: '1', '2', '3', '4', '5'

Pre-processing: centered (3), scaled (3)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 246, 242, 245, 243, 244, 244, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.4329659  0.2898084
   2  0.4354798  0.2938798
   3  0.4750041  0.3427135
   4  0.4384792  0.2976271
   5  0.3920050  0.2396782
   6  0.3750254  0.2168534
   7  0.3599965  0.1976595
   8  0.3748624  0.2171614
   9  0.3809518  0.2251727
  10  0.4004710  0.2496163
  11  0.4007866  0.2497683
  12  0.4223951  0.2766792
  13  0.3936976  0.2412349
  14  0.4009404  0.2515474
  15  0.3948396  0.2431880
  16  0.4259780  0.2825400
  17  0.3941772  0.2418686
  18  0.4017476  0.2501618
  19  0.3975335  0.2460078
  20  0.3783414  0.2216260

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 3.
> print(paste("The Best k value for k-NN is", knn_model$bestTune$k))
[1] "The Best k value for k-NN is 3"
```

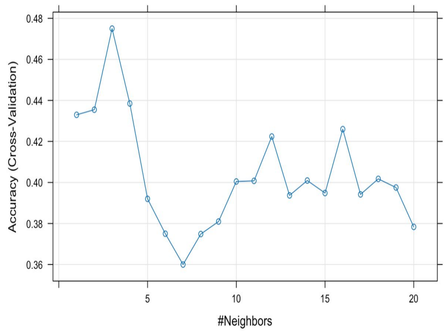

Figure 14: figure 14

The above image shows the output obtained by K-Nearest Neighbors model and a plot to determine the optimal k value. From the above obtained output and the plot it is clear that the tuning parameters ran for 20 values of k and from that it gets the highest accuracy and kappa values at k = 3. So this explains that the k=3 is the best k value for this knn model.

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

          Reference
Prediction  1  2  3  4  5
         1  7  0  1  1  1
         2  0 11  1  0  0
         3  1  0  5  3  2
         4  1  3  2  6  5
         5  5  0  4  3  5

Overall Statistics

               Accuracy : 0.5075
                 95% CI : (0.3824, 0.6318)
    No Information Rate : 0.209
    P-Value [Acc > NIR] : 6.272e-08

                  Kappa : 0.3852

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.5000   0.7857  0.38462  0.46154  0.38462
Specificity            0.9434   0.9811  0.88889  0.79630  0.77778
Pos Pred Value         0.7000   0.9167  0.45455  0.35294  0.29412
Neg Pred Value         0.8772   0.9455  0.85714  0.86000  0.84000
Prevalence             0.2090   0.2090  0.19403  0.19403  0.19403
Detection Rate         0.1045   0.1642  0.07463  0.08955  0.07463
Detection Prevalence   0.1493   0.1791  0.16418  0.25373  0.25373
Balanced Accuracy      0.7217   0.8834  0.63675  0.62892  0.58120
> cat("The Accuracy of k-NN model:", round(confusion_matrix_knn$overall["Accuracy"] * 100, 2), "%\n")
The Accuracy of k-NN model: 50.75 %
```

Figure 15: figure 15

The above image conssists of confusion matrix that shows the performance of a k-Nearest Neighbors (k-NN) model for a multi-class classification for 5 different classes of mines. The overall accuracy of the model is 50.75%, which indicates that the model correctly predicts the class label for 50.75% of the instances.

9 Random Forest Model

```
> print(model_rf)

Call:
 randomForest(formula = y ~ V + H + S, data = data, ntree = 100)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 1

        OOB estimate of  error rate: 52.37%
```

Figure 16: figure 16

In the above image obtained by running the R script Out of Bag (OOB) error for the random forest model with 100 trees and 1 variable is 52.37%. The OOB error estimates the performance of the model on unseen data by averaging predictions from trees that exclude specific data points during training.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3  4  5
         1 71  0  3  0  4
         2  0 70  1  0  0
         3  0  0 61  1  4
         4  0  0  0 63  1
         5  0  0  1  2 56

Overall Statistics

               Accuracy : 0.9497
                 95% CI : (0.9207, 0.9704)
    No Information Rate : 0.2101
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9371

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            1.0000   1.0000   0.9242   0.9545   0.8615
Specificity            0.9738   0.9963   0.9816   0.9963   0.9890
Pos Pred Value         0.9103   0.9859   0.9242   0.9844   0.9492
Neg Pred Value         1.0000   1.0000   0.9816   0.9891   0.9677
Prevalence             0.2101   0.2071   0.1953   0.1953   0.1923
Detection Rate         0.2101   0.2071   0.1805   0.1864   0.1657
Detection Prevalence   0.2308   0.2101   0.1953   0.1893   0.1746
Balanced Accuracy      0.9869   0.9981   0.9529   0.9754   0.9253
```

Figure 17: figure 17

The above image consists of the confusion matrix and statistics that shows the performance of the random forest model on the test data. The model achieves a high accuracy of 94.97%, correctly classifying most instances. It excels in predicting classes 1, 2, and 5, with high sensitivity and specificity. The model struggles slightly with classes 3 and 4, but still maintains a reasonable level of accuracy. Overall, the model demonstrates good performance in classifying the different classes.

```
Random Forest

338 samples
  3 predictor
  5 classes: '1', '2', '3', '4', '5'

No pre-processing
Resampling: Cross-Validated (20 fold)
Summary of sample sizes: 320, 321, 322, 320, 322, 319, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.5532652  0.4400585
  3     0.5650127  0.4551959

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 3.
```

Figure 18: figure 18

The above image shows that random forest model is using 20-fold cross-validation to assess its performance. The model is trained on a dataset with 338 samples and 3 predictors, aiming to classify instances into 5 classes. The hyperparameter controls the number of variables considered at each split. The results indicate that setting to 3 yields the best accuracy of 56.50% and a Kappa value of 0.4551959.

10 Conclusion

The analysis reveals that the Random Forest model significantly outperforms other techniques, achieving an accuracy of 95.27%, which strongly suggests its suitability for this specific dataset and problem. In contrast, K-Nearest Neighbors struggled achieved 50.75% accuracy, indicating its ineffectiveness, while K-means clustering explained approximately half the data variance. The logistic regression model showed some improvement over the null model, but its comparative performance remains unclear. In conclusion we can select Random forest model for processing this dataset.