



DATA MINING- FINAL ASSIGNMENT

Submitted to: DR. MD. MAHBUB CHOWDHURY MISHU



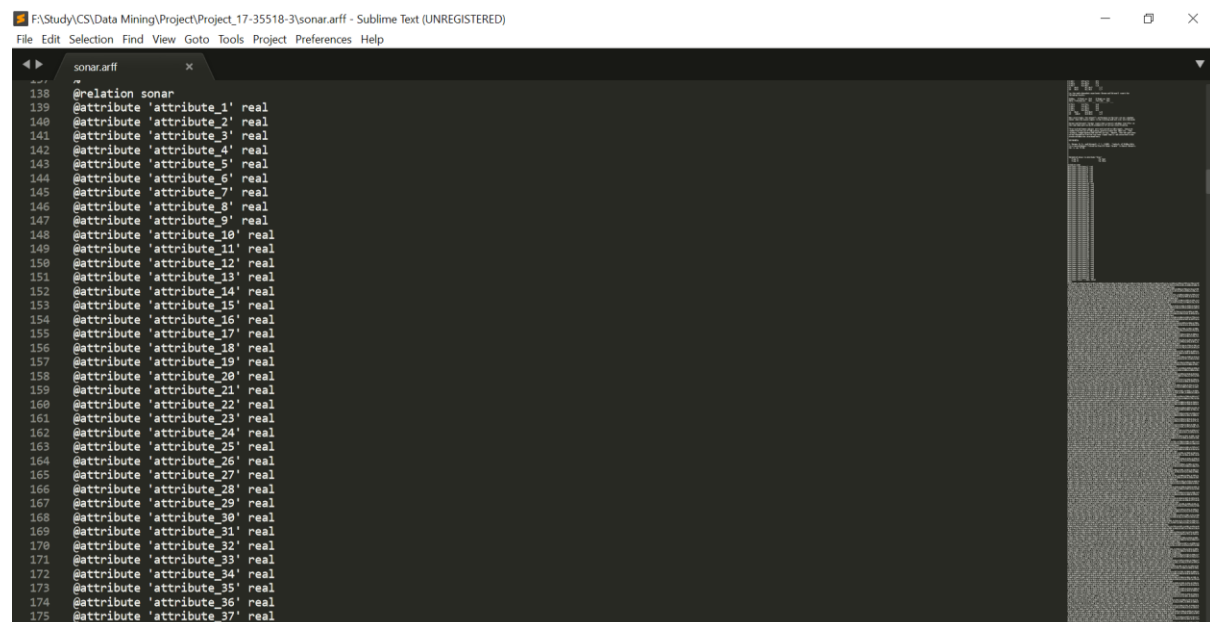
DECEMBER 7, 2020

Section: C

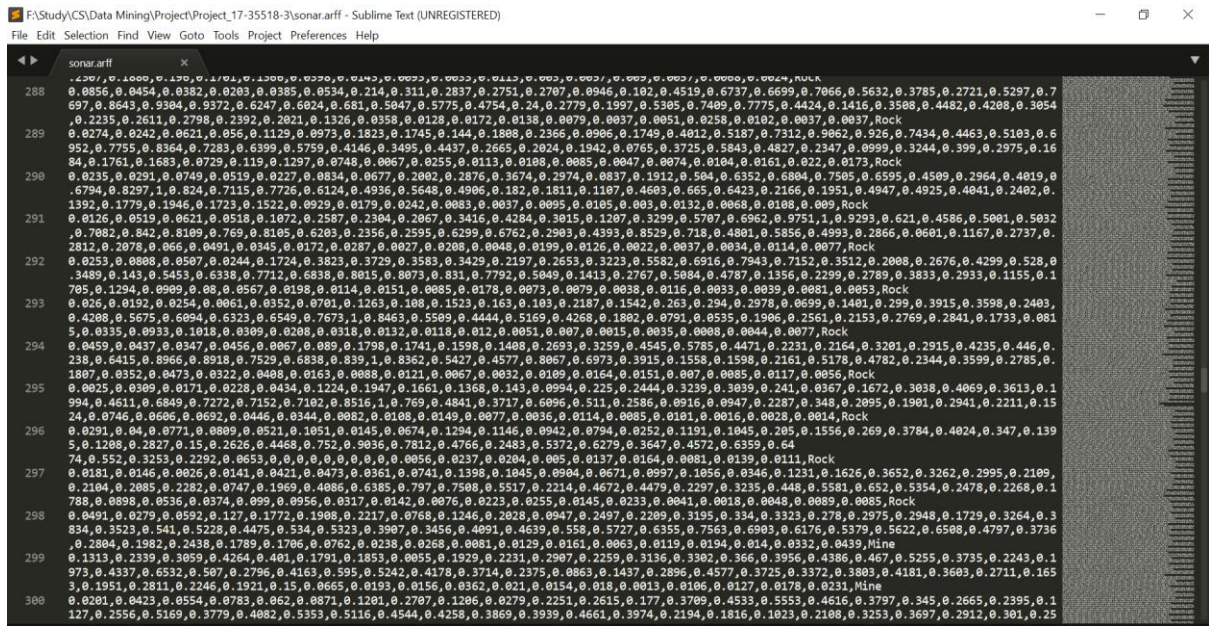
Introduction: Classification is one of the most important parts in data mining. By utilizing this we can solve many real world problems with greater accuracy & efficiency that was unprecedented. There exists a number of ways to classify objects into groups such as Naive Bayes, KNN, Association Rule Mining & Clustering. All of these techniques are used to classify objects. The method chosen for this particular experiment was **KNN** or **K- Nearest Neighbors** algorithm. It used Euclidean distance to classify objects. The dataset is about **“SONAR”** signals that was collected in order to differentiate between rocks & metals so that we can take the decision whether to mine for it or not. It has **208 instances & 60 attributes** in total. The primary task is to use KNN on the dataset to create a model & apply the model on test set to measure the accuracy. Based on the success ratio of the model we can use this on future mining projects to identify the materials that we can mine.

Dataset: The dataset named “Sonar.arff” was collected from UCI repository. The dataset has **208 instances & 60 attributes** containing signals from rocks & metals. It contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions & 97 patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

Each pattern is a **set of 60 numbers in the range 0.0 to 1.0**. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp.



```
sonar.arff
%
@relation sonar
@attribute 'attribute_1' real
@attribute 'attribute_2' real
@attribute 'attribute_3' real
@attribute 'attribute_4' real
@attribute 'attribute_5' real
@attribute 'attribute_6' real
@attribute 'attribute_7' real
@attribute 'attribute_8' real
@attribute 'attribute_9' real
@attribute 'attribute_10' real
@attribute 'attribute_11' real
@attribute 'attribute_12' real
@attribute 'attribute_13' real
@attribute 'attribute_14' real
@attribute 'attribute_15' real
@attribute 'attribute_16' real
@attribute 'attribute_17' real
@attribute 'attribute_18' real
@attribute 'attribute_19' real
@attribute 'attribute_20' real
@attribute 'attribute_21' real
@attribute 'attribute_22' real
@attribute 'attribute_23' real
@attribute 'attribute_24' real
@attribute 'attribute_25' real
@attribute 'attribute_26' real
@attribute 'attribute_27' real
@attribute 'attribute_28' real
@attribute 'attribute_29' real
@attribute 'attribute_30' real
@attribute 'attribute_31' real
@attribute 'attribute_32' real
@attribute 'attribute_33' real
@attribute 'attribute_34' real
@attribute 'attribute_35' real
@attribute 'attribute_36' real
@attribute 'attribute_37' real
```



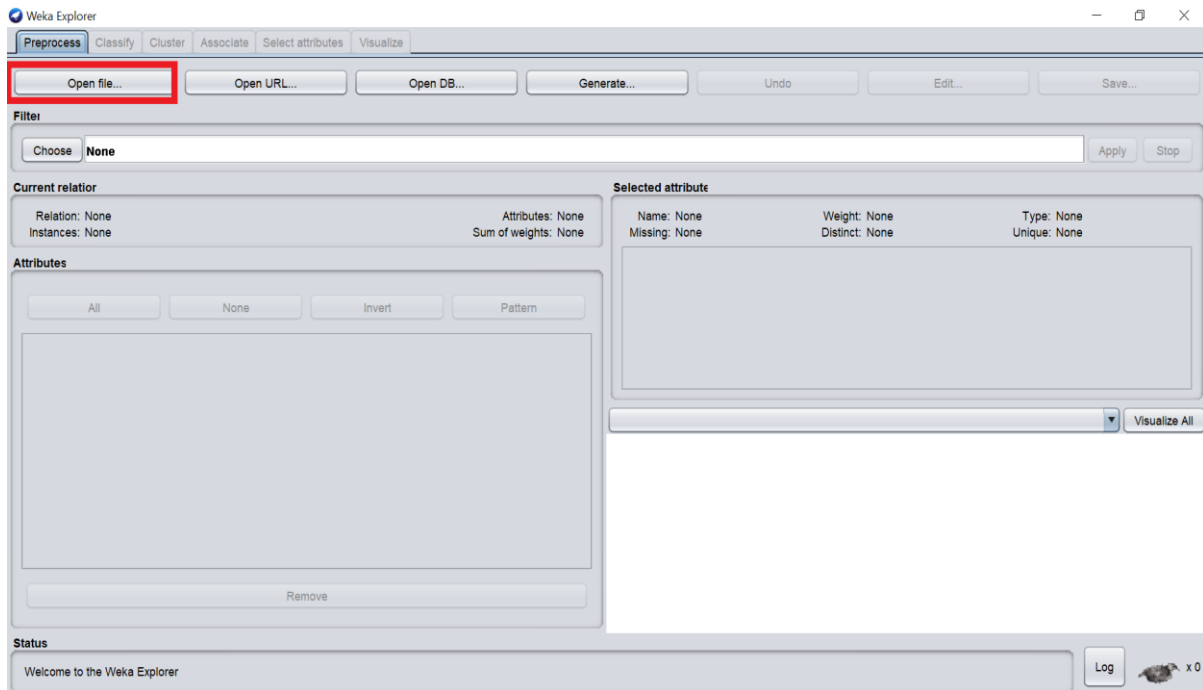
Method:

- Selecting a Dataset
- By using Euclidean distance, we need to measure the distances
- Taking the smallest distance value
- Classify the data to its closest neighbor
- Repeat the process for all the data points

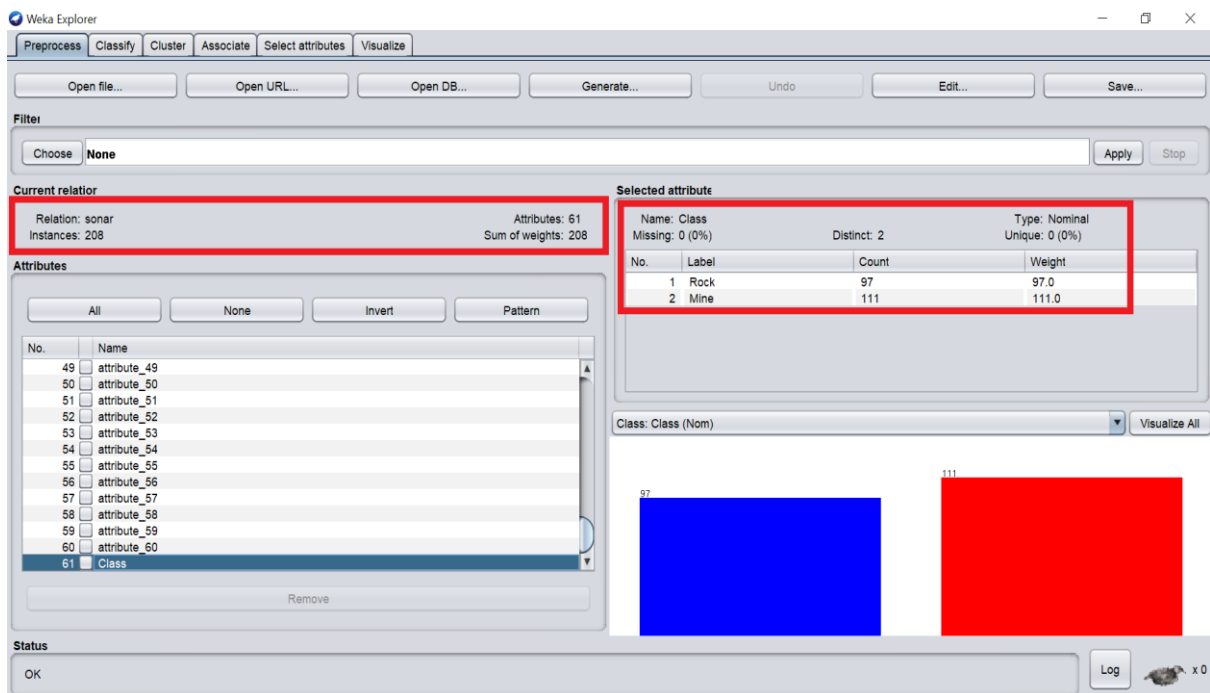
“WEKA” was used to perform this task.



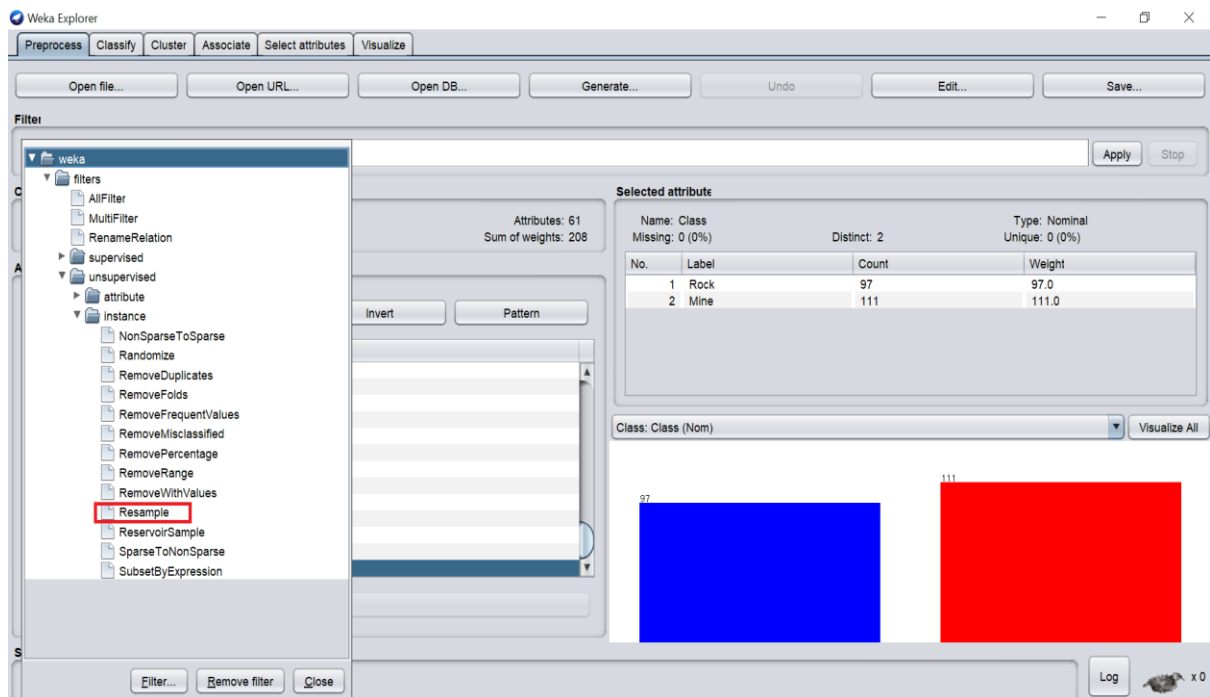
At First WEKA was launched & “Explorer” tab was clicked.



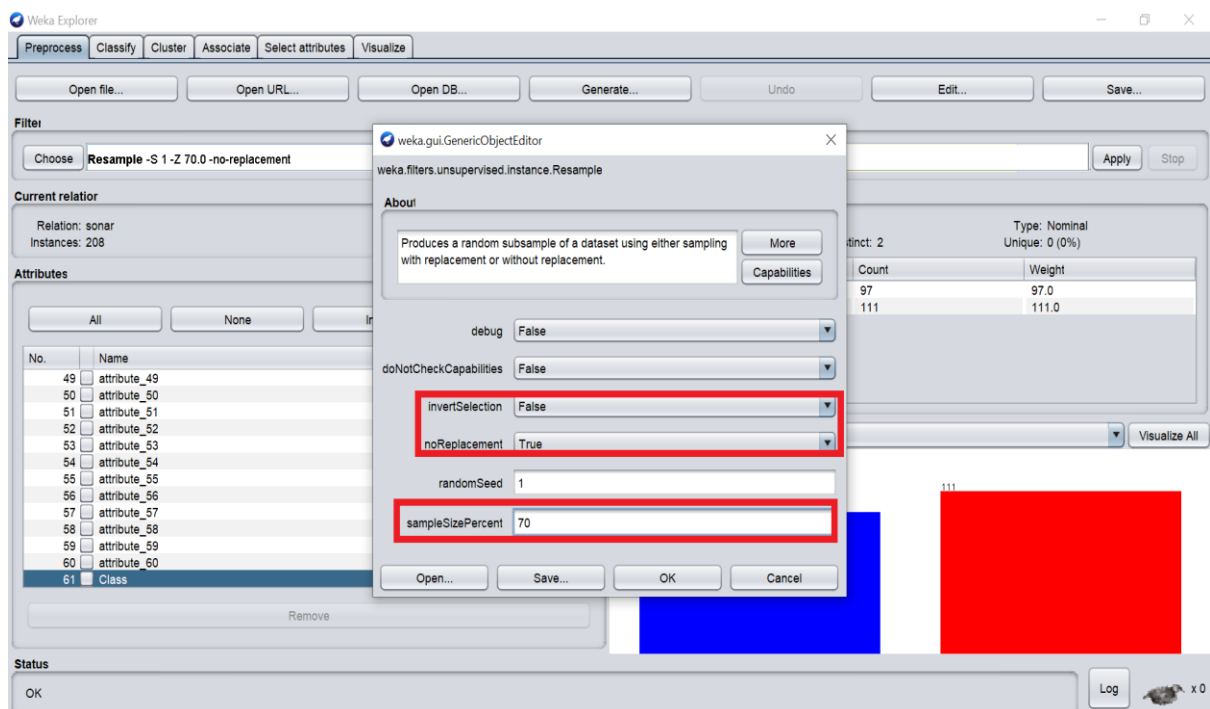
After clicking on Explorer tab this window was opened & we clicked on Open File to load the dataset.



We can see that the dataset contains **208 instances & 60 attributes** that are **numerical** and there aren't any missing values. The class attribute is **binary type** (Containing only two options) & each attribute is distributed among the 208 instances almost equally so we did not have to synthetically add any values to the dataset.



To examine our model, we needed to divide our dataset into two parts so that we could test our model & it's accuracy. After loading the dataset, we clicked on Filter> Unsupervised> Instance> Resample to divide the dataset into training set & test set.



We kept the invert selection false for the first time & no replacement true. For training set, we decided to go with 70% of the whole dataset. After Clicking Ok & Apply we had

The screenshot shows the Weka Explorer interface with the 'Resample' filter selected in the Filter list. The 'Current relation' is 'sonar-weka.filters.unsupervised.instance.Resample-S1-Z70.0-no-replac...' with 61 attributes and 145 instances. The 'Selected attribute' table shows the distribution of the 'Class' attribute:

No.	Label	Count	Weight
1	Rock	66	66.0
2	Mine	79	79.0

The 'Attributes' list shows 61 attributes, including 'Class'. The 'Status' bar indicates 'OK'.

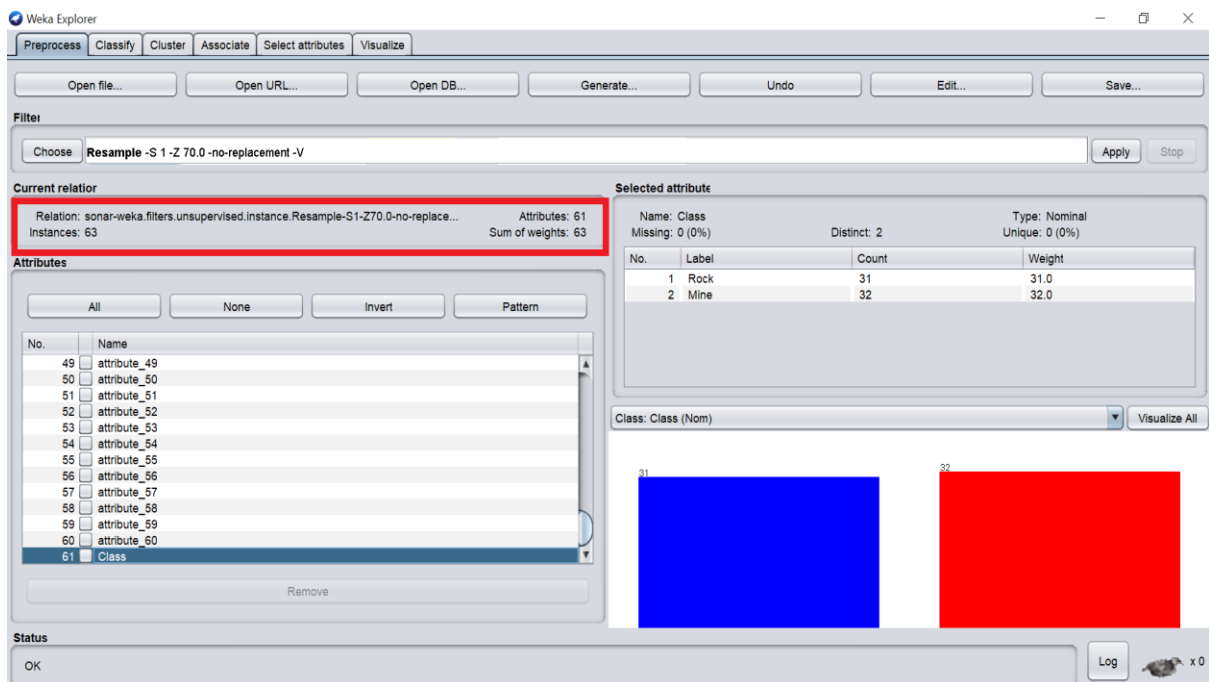
145 instances out of 208 instances & we saved this file as the training set.

The screenshot shows the Weka Explorer interface with the 'Resample' filter selected in the Filter list. The 'Current relation' is 'sonar' with 208 instances. A dialog box titled 'weka.gui.GenericObjectEditor' is open, showing the configuration for the 'Resample' filter:

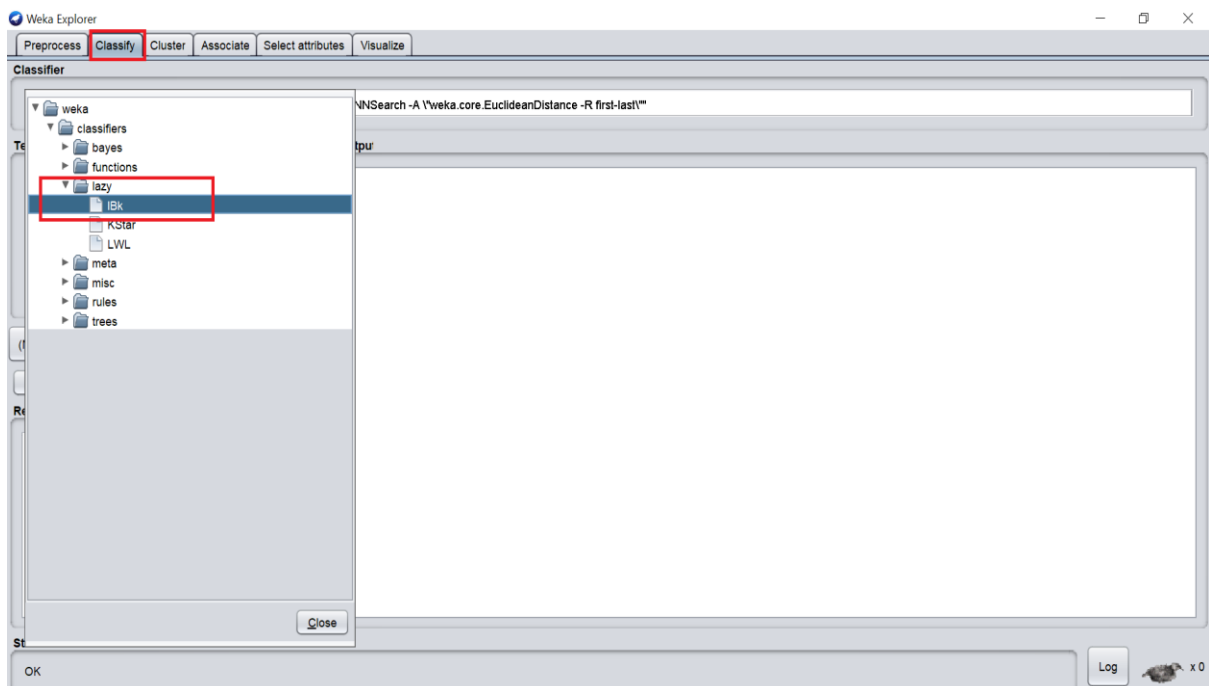
- debug: False
- doNotCheckCapabilities: False
- invertSelection: True
- noReplacement: True
- randomSeed: 1
- sampleSizePercent: 70.0

The 'Status' bar indicates 'OK'.

After saving the training set we clicked on Undo & the instances returned to 208 & then for the test set we turned on the invert selection to “TRUE” so that we didn’t have any overlapping values in both sets. After that we clicked ok & Apply.



We can see that the number of instances are **63 which about 30% of 208** & this was saved as the test set.



Then we closed WEKA and restarted the application in Explorer mode & opened the Training set. After that we clicked on Classify> Lazy> IBK to train the dataset by using KNN.

Classifier
Choose: KStar-B 20-M a

Test options
☐ Use training set
☐ Supplied test set
☒ Cross-validation Folds: 10
☐ Percentage split % 66
 More options...
 (Nom) Class
 Start Stop

Result list (right-click for options)
 21:57:38 - lazy.KStar

Classifier output

```

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      120      82.7586 %
Incorrectly Classified Instances    25      17.2414 %
Kappa statistic                    0.6475
Mean absolute error                0.171
Root mean squared error            0.4076
Relative absolute error            34.4722 %
Root relative squared error        81.8387 %
Total Number of Instances         145

=== Detailed Accuracy By Class ===
      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
      0.727    0.089    0.873    0.727    0.793    0.655  0.930    0.925    Rock
      0.911    0.273    0.800    0.911    0.852    0.655  0.923    0.921    Mine
Weighted Avg.   0.828    0.189    0.833    0.828    0.825    0.655  0.927    0.923

=== Confusion Matrix ===
  a  b  <-- classified as
48 18 | a = Rock
 7 72 | b = Mine
  
```

Status
OK Log x 0

We used 10-fold cross validation on the test set. The accuracy of the set was **82.7586%**. Out of 145 instances 25 were incorrectly classified. 18 values were incorrectly classified as “Rock” & 7 values were incorrectly classified as “Mine” & we saved the model.

Classifier
Choose: KStar-B 20-M a

Test options
☐ Use training set
☒ Supplied test set Set...
☐ Cross-validation Folds: 10
☐ Percentage split % 66
 More options...
 (Nom) Class
 Start Stop

Result list (right-click for options)
 21:57:38 - lazy.KStar

Classifier output

```

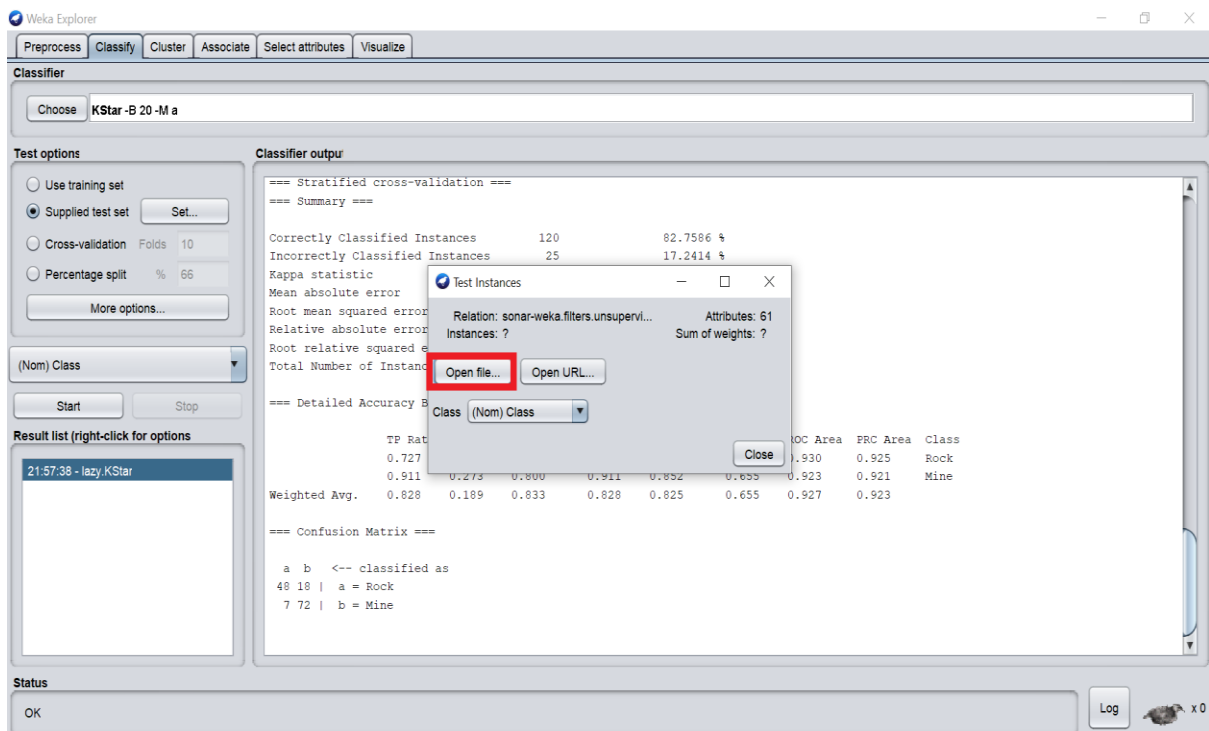
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      120      82.7586 %
Incorrectly Classified Instances    25      17.2414 %
Kappa statistic                    0.6475
Mean absolute error                0.171
Root mean squared error            0.4076
Relative absolute error            34.4722 %
Root relative squared error        81.8387 %
Total Number of Instances         145

=== Detailed Accuracy By Class ===
      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
      0.727    0.089    0.873    0.727    0.793    0.655  0.930    0.925    Rock
      0.911    0.273    0.800    0.911    0.852    0.655  0.923    0.921    Mine
Weighted Avg.   0.828    0.189    0.833    0.828    0.825    0.655  0.927    0.923

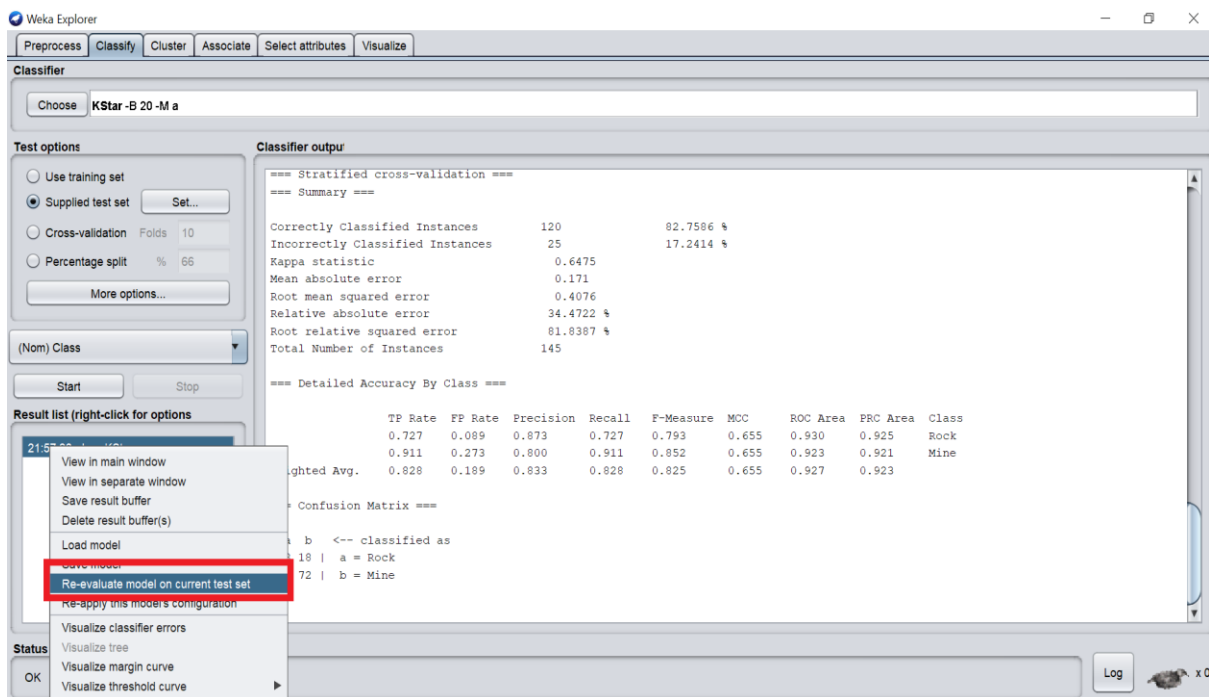
=== Confusion Matrix ===
  a  b  <-- classified as
48 18 | a = Rock
 7 72 | b = Mine
  
```

Status
OK Log x 0

Then we clicked on supplied test set to load the test set.



We clicked on Open file & after selecting the file clicked on close.



After that we used our previously saved model to evaluate accuracy on test set.

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose KStar-B 20-M a

Test options

☐ Use training set

☒ Supplied test set

☐ Cross-validation Folds 10

☐ Percentage split % 66

(Nom) Class

Result list (right-click for options)

21:57:38 - lazy.KStar

Classifier output

Instances: unknown (yet). Reading incrementally
Attributes: 61

=== Summary ===

Correctly Classified Instances	55	87.3016 %
Incorrectly Classified Instances	8	12.6984 %
Kappa statistic	0.7457	
Mean absolute error	0.1346	
Root mean squared error	0.3596	
Total Number of Instances	63	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.839	0.094	0.897	0.839	0.867	0.747	0.924	0.927	Rock
	0.906	0.161	0.853	0.906	0.879	0.747	0.916	0.902	Mine
Weighted Avg.	0.873	0.128	0.874	0.873	0.873	0.747	0.920	0.914	

=== Confusion Matrix ===

a	b	<-- classified as
26	5	a = Rock
3	29	b = Mine

Status

OK

x 0

Then we were able to see the accuracy improved from the initial training set (From **82.7586%** to **87.3016%**). Also the in total **8** instances were wrongly classified out of **63**.

Comparing with Naive Bayes:

In order to test our against other classifier algorithms, we chose “Naive Bayes” & created a model using the same training set.

The screenshot shows the Weka Explorer interface with the Naive Bayes classifier selected. The 'Classifier' tab is active, and the 'NaiveBayes' model is chosen. The 'Test options' section shows 'Cross-validation' with 'Folds' set to 10. The 'Result list' on the left shows the model name '14:49:57 - bayes.NaiveBayes'. The 'Classifier output' pane displays the following results:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      98      67.5862 %
Incorrectly Classified Instances    47      32.4138 %
Kappa statistic                    0.3616
Mean absolute error                 0.3089
Root mean squared error             0.5201
Relative absolute error             62.2518 %
Root relative squared error         104.4076 %
Total Number of Instances          145

=== Detailed Accuracy By Class ===

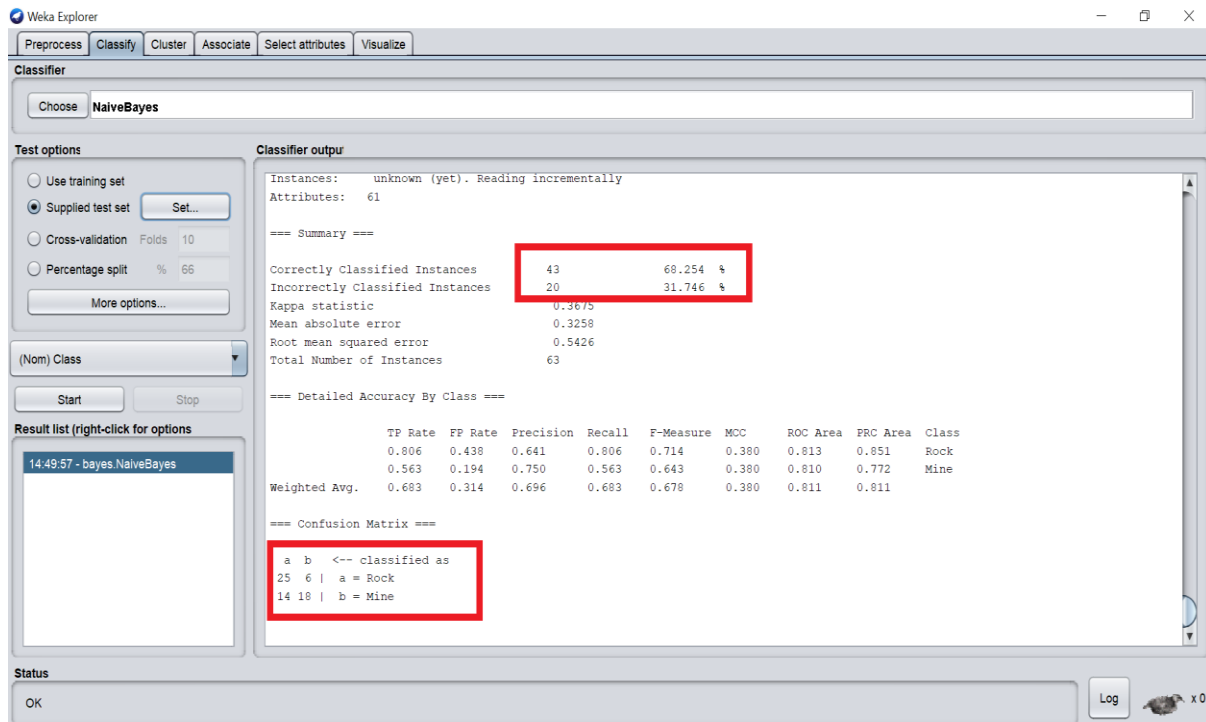
      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
      -----  -
      0.788    0.418    0.612     0.788    0.689     0.374    0.787    0.757    Rock
      0.582    0.212    0.767     0.582    0.662     0.374    0.788    0.785    Mine
Weighted Avg.   0.676    0.306    0.696     0.676    0.674     0.374    0.788    0.772

=== Confusion Matrix ===

 a  b  <-- classified as
52 14 | a = Rock
33 46 | b = Mine
```

The 'Status' bar at the bottom shows 'OK' and a 'Log' button.

Here, we can see that the accuracy is much lower than KNN model (From **82.7586%** to **67.5862%**) & the number of incorrectly classified is also higher (From **25** to **47**).



After evaluating the Naive Bayes model against the test set a drop in accuracy was seen from the training set which is much lower than the KNN model (From **87.3016% to 68.254%**) & number of incorrectly classified instances are also much higher than the KNN model (From **8 to 20**).

So, we clearly observed that the **KNN model** is far **superior** than the **Naive Bayes** model for this particular dataset. It can be stated that using KNN for this dataset was the right thing to do.

Conclusion:

The model developed by using KNN is producing an accuracy of **87.3016%** on test data sets that is likely to affect the future projects significantly if someone uses this model. It can dramatically reduce the cost by predicting the unseen instances.

Our modern world depends heavily on the minerals found in earth's crust. From metal, aluminum to silicon we need these materials to move forward & the demand for mining is more than ever. By reducing the overhead cost, we can benefit considerably & use the saved resources to deploy in other fields.

The findings found in the study noticeably states that the method used is superior in terms of accuracy thus able to predict the unseen instances more correctly. **87.3016% of accuracy** can significantly improve the future findings on test sets that will help interpreting the data greatly.

