

## PRACTICAL-9

**AIM: Perform classification with WEKA tool.**

**a. using Decision Tree Classifier**

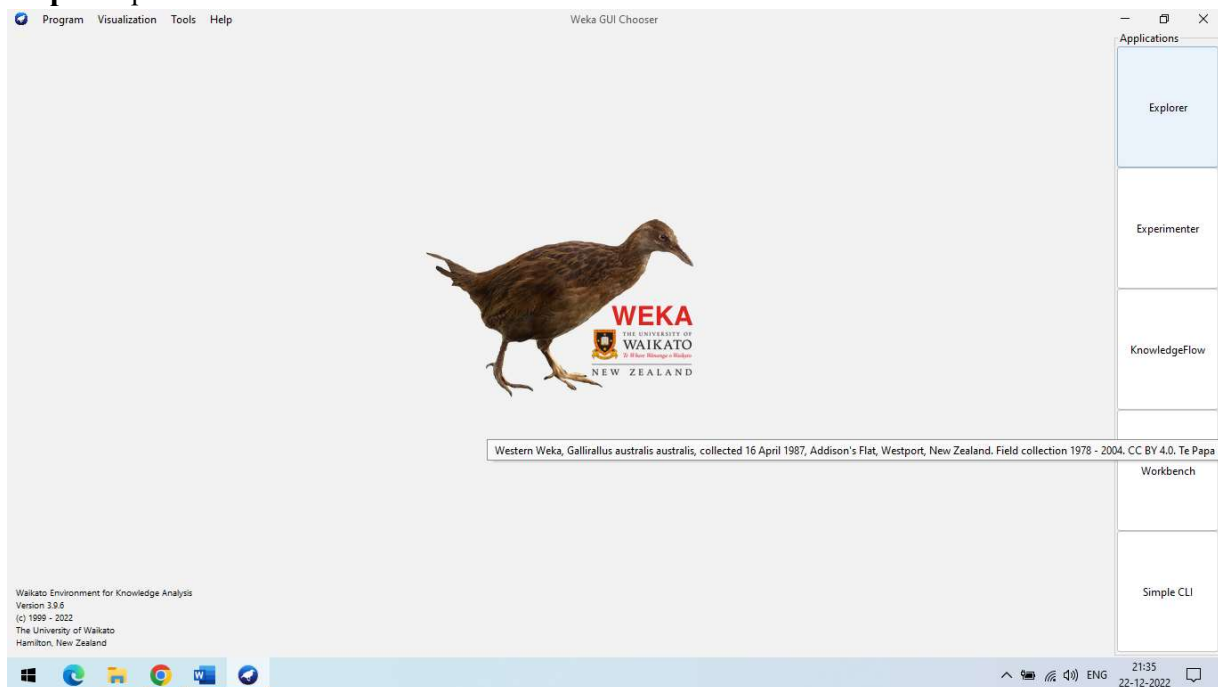
**b. using Naïve Bayes Classifier**

**c. using Multilayer Perceptron.**

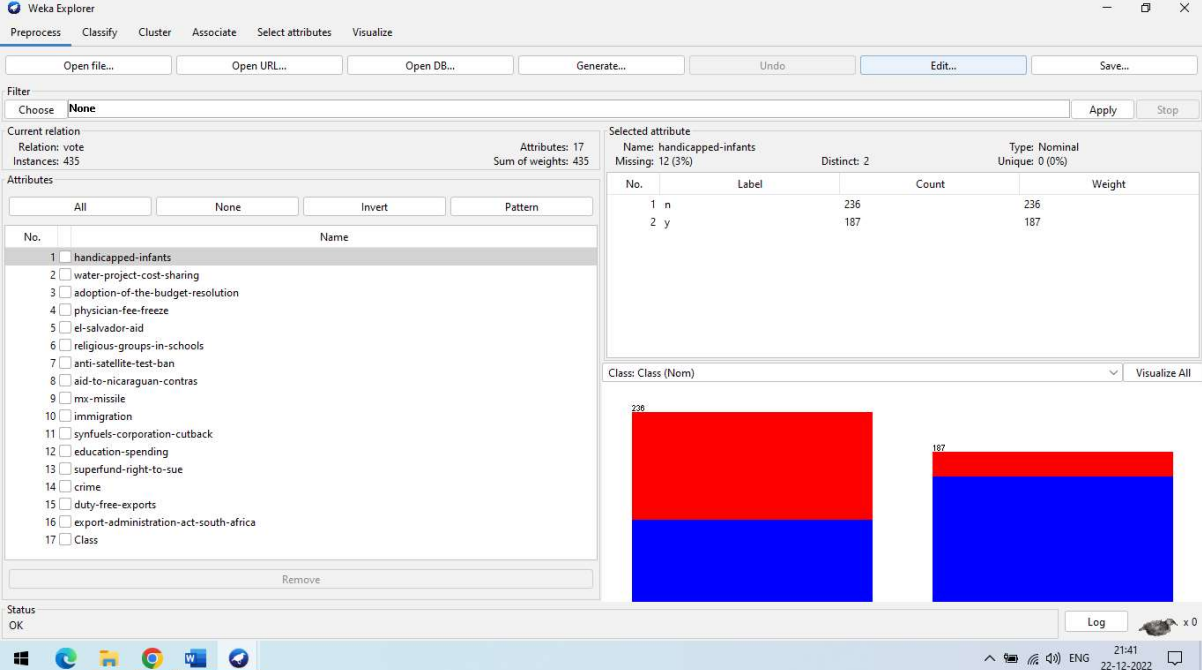
**Steps:**

**(A) Using Decision Tree Classifier:-** A decision tree is a class discriminator that recursively partitions the training set until each partition consists entirely or dominantly of examples from one class. Each non-leaf node of the tree contains a split point that is a test on one or more attributes and determines how the data is partitioned. A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node. Decision Tree Mining is a type of data mining technique that is used to build Classification Models. It builds classification models in the form of a tree-like structure, just like its name. This type of mining belongs to supervised class learning.

**Step 1: Open Weka Tool**

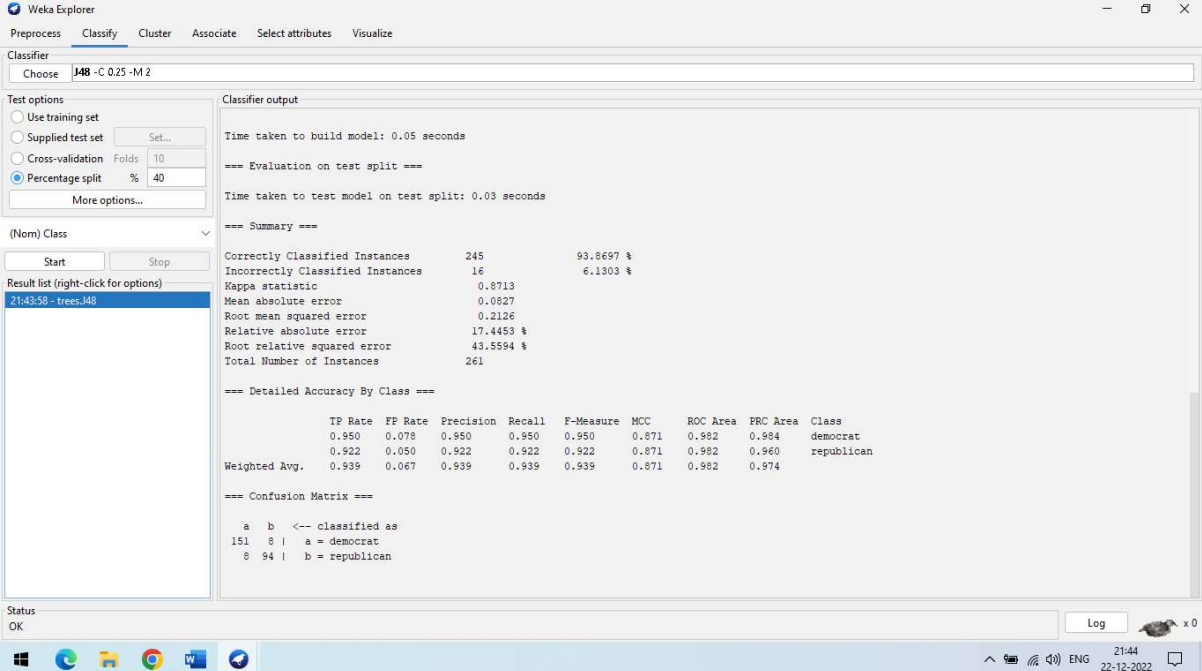


## Step 2: weka→explorer→open file→ vote. Arff



The screenshot shows the Weka Explorer interface. The 'Preprocess' tab is selected. The 'Open file...' button is highlighted. The 'Filter' section shows 'None' selected. The 'Current relation' section shows 'Relation: vote' and 'Instances: 435'. The 'Attributes' list on the left includes 'handicapped-infants', 'water-project-cost-sharing', 'adoption-of-the-budget-resolution', 'physician-fee-freeze', 'el-salvador-aid', 'religious-groups-in-schools', 'anti-satellite-test-ban', 'aid-to-nicaraguan-contras', 'mx-missile', 'immigration', 'synfuels-corporation-cutback', 'education-spending', 'superfund-right-to-sue', 'crime', 'duty-free-exports', 'export-administration-act-south-africa', and 'Class'. The 'Selected attribute' section shows 'Name: handicapped-infants' and 'Type: Nominal'. The 'Class: Class (Nom)' dropdown is set to 'Class (Nom)'. The 'Visualize All' button is visible.

## Step 3: Go to the classify >> choose filter j48 >> give percentage split(40%) >> click start



The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'J48 - C 0.25 - M 2'. The 'Test options' section shows 'Percentage split' selected with a value of 40%. The 'Result list (right-click for options)' shows '21:43:58 - trees.J48' selected. The 'Classifier output' section displays the following information:

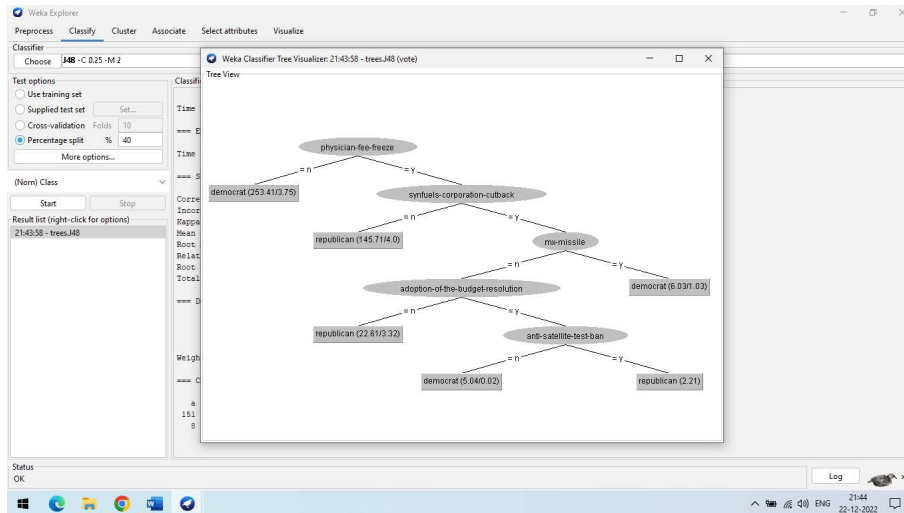
```

Time taken to build model: 0.05 seconds
=== Evaluation on test split ===
Time taken to test model on test split: 0.03 seconds
=== Summary ===
Correctly Classified Instances      245      93.8697 %
Incorrectly Classified Instances    16       6.1303 %
Kappa statistic                    0.8713
Mean absolute error                 0.0627
Root mean squared error             0.2126
Relative absolute error             17.4453 %
Root relative squared error         43.5594 %
Total Number of Instances          261

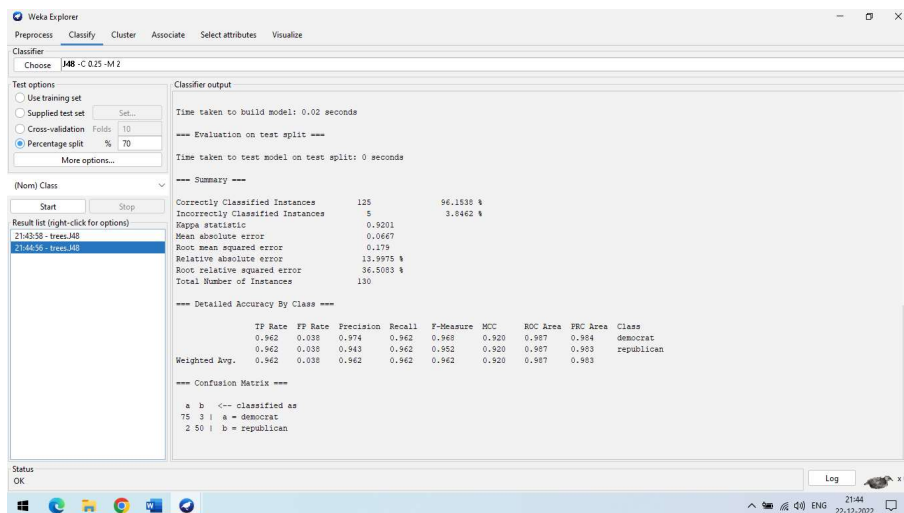
=== Detailed Accuracy By Class ===
      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.950    0.078    0.950     0.950    0.950     0.871    0.982    0.984    democrat
      0.922    0.050    0.922     0.922    0.922     0.871    0.982    0.960    republican
Weighted Avg.   0.939    0.067    0.939     0.939    0.939     0.871    0.982    0.974

=== Confusion Matrix ===
  a  b  <-- classified as
151  8  a = democrat
  8  94 b = republican
  
```

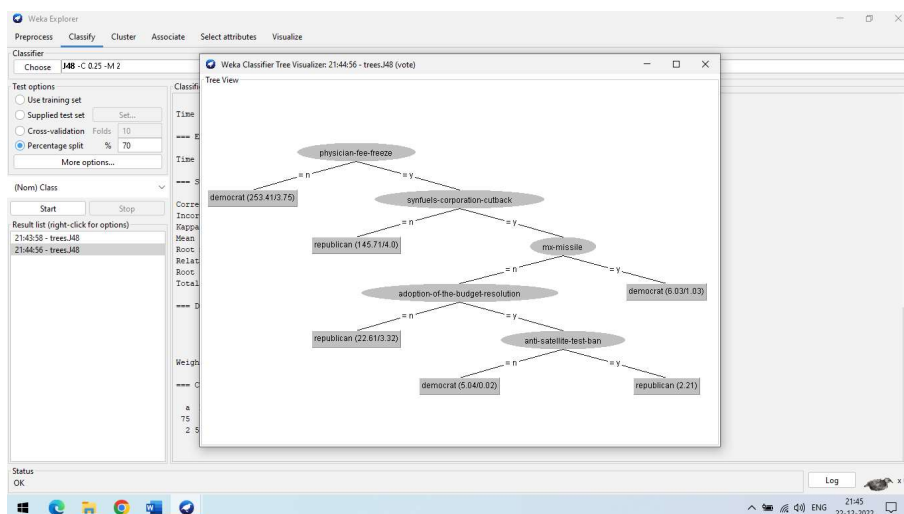
## Step 4: right click on result list >> visualize tree



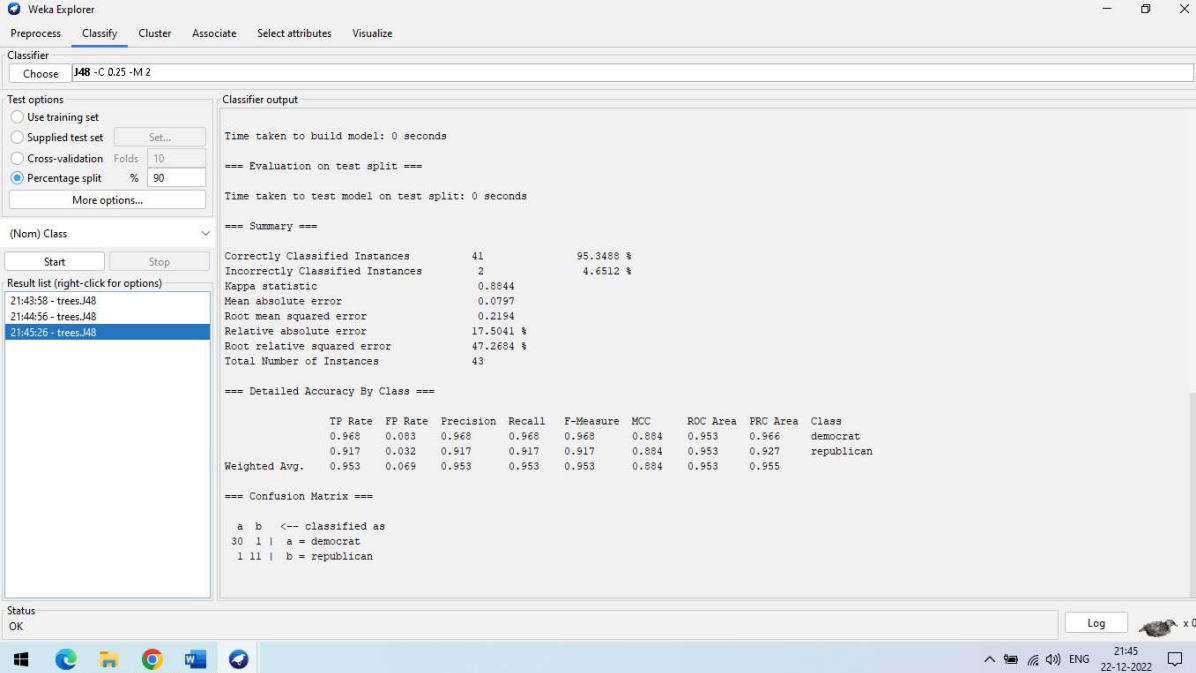
## Step 5: Go to the classify >> choose filter j48 >> give percentage split(70%) >> click start



## Step 6: right click on result list >> visualize tree



**Step 7:** Go to the classify >> choose filter j48 >> give percentage split(90%) >> click start



**Classifier**  
 Choose: J48 -C 0.25 -M 2

**Test options**  
☐ Use training set  
☐ Supplied test set  
☐ Cross-validation Folds: 10  
☒ Percentage split % 90

**(Nom) Class**  
 Start Stop

**Result list (right-click for options)**  
 21:43:58 - trees.J48  
 21:44:56 - trees.J48  
 21:45:26 - trees.J48

**Classifier output**

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	41	95.3488 %
Incorrectly Classified Instances	2	4.6512 %
Kappa statistic	0.8844	
Mean absolute error	0.0797	
Root mean squared error	0.2194	
Relative absolute error	17.5041 %	
Root relative squared error	47.2694 %	
Total Number of Instances	43	

=== Detailed Accuracy By Class ===

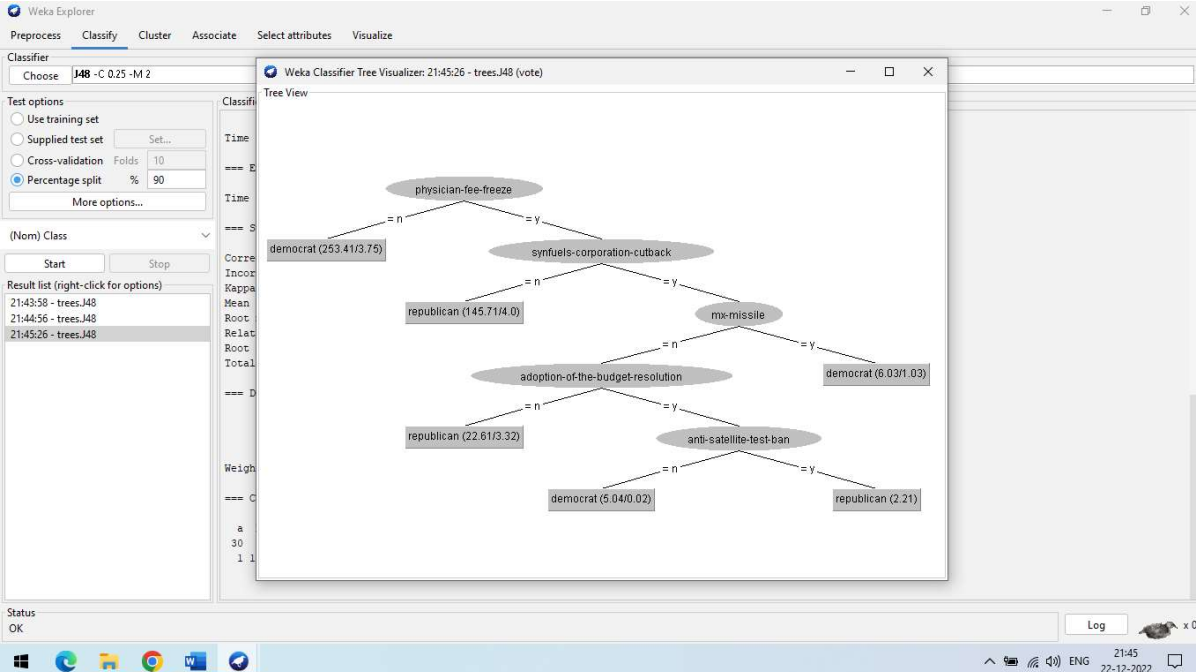
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0.968	0.083	0.968	0.968	0.968	0.884	0.953	0.966	democrat
	0.917	0.032	0.917	0.917	0.917	0.884	0.953	0.927	republican

=== Confusion Matrix ===

```

a b <-- classified as
30 1 | a = democrat
1 11 | b = republican
  
```

**Step 8:** right click on result list >> visualize tree



**Weka Classifier Tree Visualizer: 21:45:26 - trees.J48 (vote)**

**Tree View**

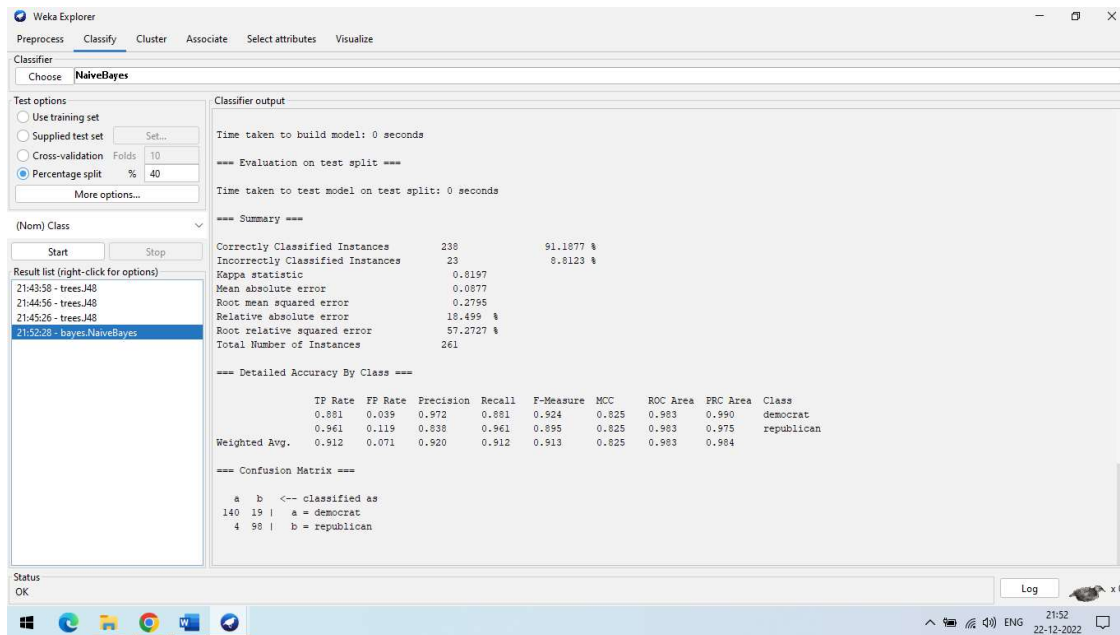
```

graph TD
    Root[physician-fee-freeze] -- n --> L1[democrat 253.41/3.75]
    Root -- y --> L2[synfuels-corporation-cutback]
    L2 -- n --> L3[republican 145.71/4.0]
    L2 -- y --> L4[mx-missile]
    L4 -- n --> L5[adoption-of-the-budget-resolution]
    L4 -- y --> L6[democrat 6.03/1.03]
    L5 -- n --> L7[republican 22.61/3.32]
    L5 -- y --> L8[anti-satellite-test-ban]
    L8 -- n --> L9[democrat 5.04/0.02]
    L8 -- y --> L10[republican 2.21]
  
```

## (B) Using Naïve Bayes Classifier: -

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts based on the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

**Step 1:** Go to the classify >> choose filter NaiveBayes >> give percentage split(40%) >> click start.



The screenshot shows the Weka Explorer interface with the NaiveBayes classifier selected. The 'Test options' section shows 'Percentage split' set to 40%. The 'Classifier output' section displays the following results:

```

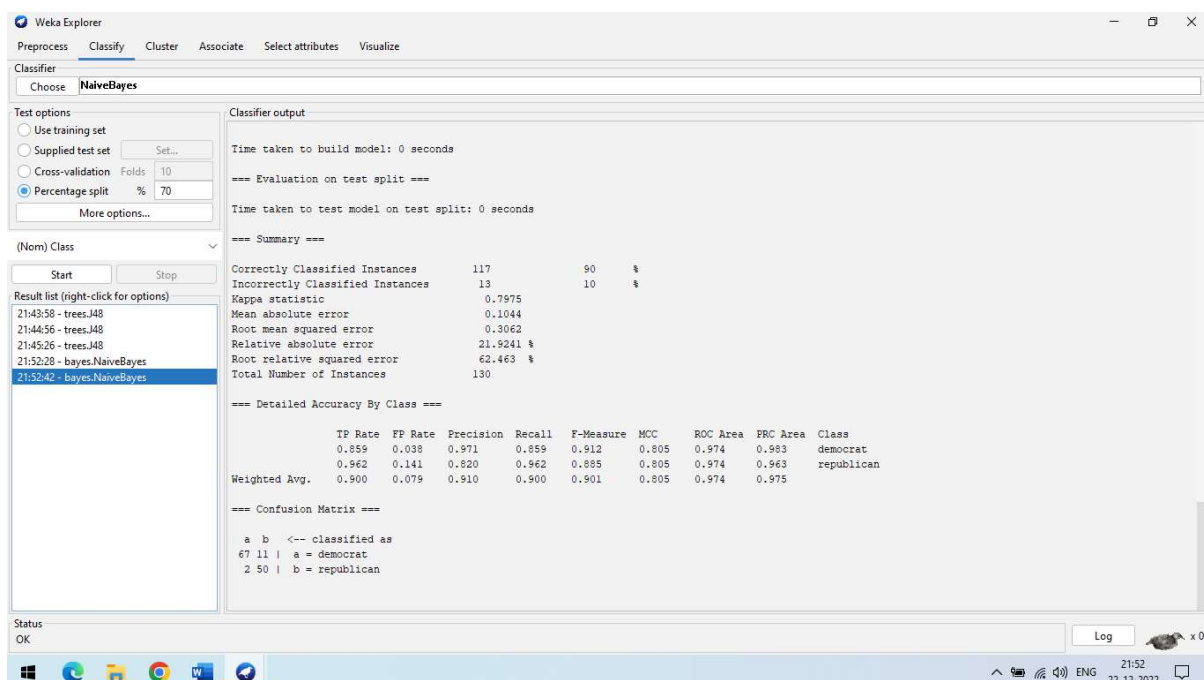
Time taken to build model: 0 seconds
=== Evaluation on test split ===
Time taken to test model on test split: 0 seconds

=== Summary ===
Correctly Classified Instances      238      91.1877 %
Incorrectly Classified Instances    23       8.8123 %
Kappa statistic                    0.8197
Mean absolute error                 0.0877
Root mean squared error            0.2795
Relative absolute error             18.499 %
Root relative squared error        97.2727 %
Total Number of Instances         261

=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
               -----  -----  -
0.881   0.039   0.972   0.881   0.924   0.825   0.983   0.990   democrat
0.961   0.119   0.838   0.961   0.895   0.825   0.983   0.975   republican
Weighted Avg.   0.912   0.071   0.920   0.912   0.913   0.825   0.983   0.984

=== Confusion Matrix ===
  a  b  <-- classified as
140 19 | a = democrat
  4 98 | b = republican
  
```

**Step 2:** Go to the classify >> choose filter NaiveBayes >> give percentage split(70%) >> click start.



The screenshot shows the Weka Explorer interface with the NaiveBayes classifier selected. The 'Test options' section shows 'Percentage split' set to 70%. The 'Classifier output' section displays the following results:

```

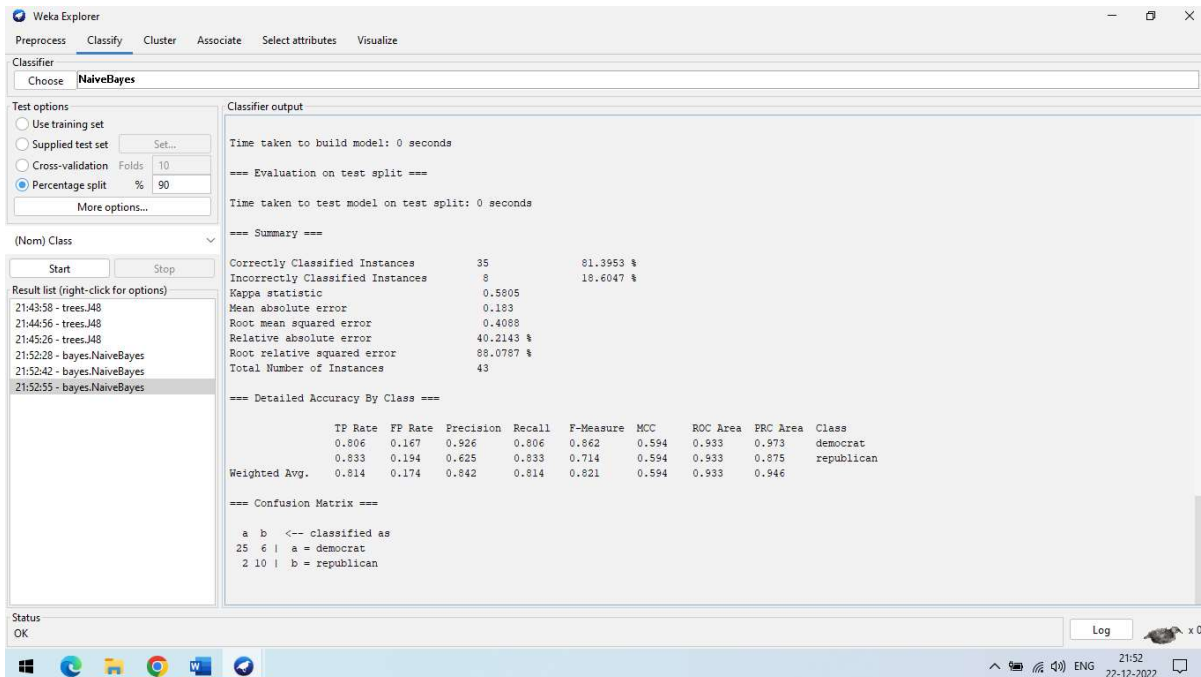
Time taken to build model: 0 seconds
=== Evaluation on test split ===
Time taken to test model on test split: 0 seconds

=== Summary ===
Correctly Classified Instances      117      90 %
Incorrectly Classified Instances    13      10 %
Kappa statistic                    0.7975
Mean absolute error                 0.1044
Root mean squared error            0.3062
Relative absolute error             21.9241 %
Root relative squared error        62.463 %
Total Number of Instances         130

=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
               -----  -----  -
0.859   0.038   0.971   0.859   0.912   0.805   0.974   0.983   democrat
0.962   0.141   0.820   0.962   0.885   0.805   0.974   0.963   republican
Weighted Avg.   0.900   0.079   0.910   0.900   0.901   0.805   0.974   0.975

=== Confusion Matrix ===
  a  b  <-- classified as
 67 11 | a = democrat
   2 50 | b = republican
  
```

**Step 3:** Go to the classify >> choose filter NaiveBayes >> give percentage split(90%) >> click start.



The screenshot shows the Weka Explorer interface with the NaiveBayes classifier selected. The 'Test options' section shows 'Percentage split' set to 90%. The 'Classifier output' section displays the following results:

```

Time taken to build model: 0 seconds
=== Evaluation on test split ===
Time taken to test model on test split: 0 seconds
=== Summary ===
Correctly Classified Instances      35      81.3953 %
Incorrectly Classified Instances    8      18.6047 %
Kappa statistic                    0.5605
Mean absolute error                 0.183
Root mean squared error             0.4088
Relative absolute error             40.2143 %
Root relative squared error         88.0787 %
Total Number of Instances          43

=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
0.806   0.167   0.926   0.806   0.862   0.594   0.933   0.973   democrat
0.833   0.194   0.625   0.833   0.714   0.594   0.933   0.875   republican
Weighted Avg.   0.814   0.174   0.842   0.814   0.821   0.594   0.933   0.946

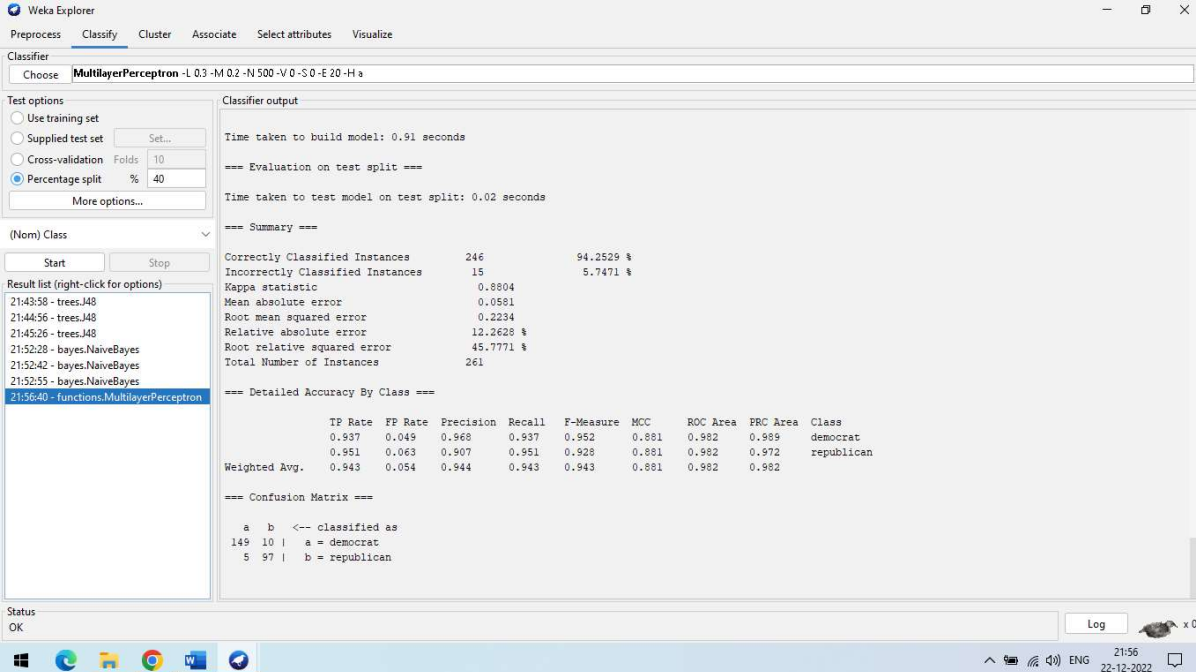
=== Confusion Matrix ===
  a  b  <-- classified as
25  6  | a = democrat
 2 10 | b = republican
  
```

### (c) Using Multilayer Perceptron:-

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons. In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer. Every node in the multi-layer perceptron uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula. Now that we are done with the theory part of multi-layer perception, let's go ahead and implement some code in python using the TensorFlow library.



**Step 1:** Go to the classify >> choose filter MultilayerPerceptron >> give percentage split(40%) >> click start.



**Weka Explorer**  
 Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose **MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a**

**Test options**  
☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds: 10  
☒ Percentage split %: 40  
 More options...

**Classifier output**

Time taken to build model: 0.91 seconds  
 Time taken to test model on test split: 0.02 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	246	94.2529 %
Incorrectly Classified Instances	15	5.7471 %
Kappa statistic	0.8804	
Mean absolute error	0.0581	
Root mean squared error	0.2234	
Relative absolute error	12.2628 %	
Root relative squared error	45.7771 %	
Total Number of Instances	261	

=== Detailed Accuracy By Class ===

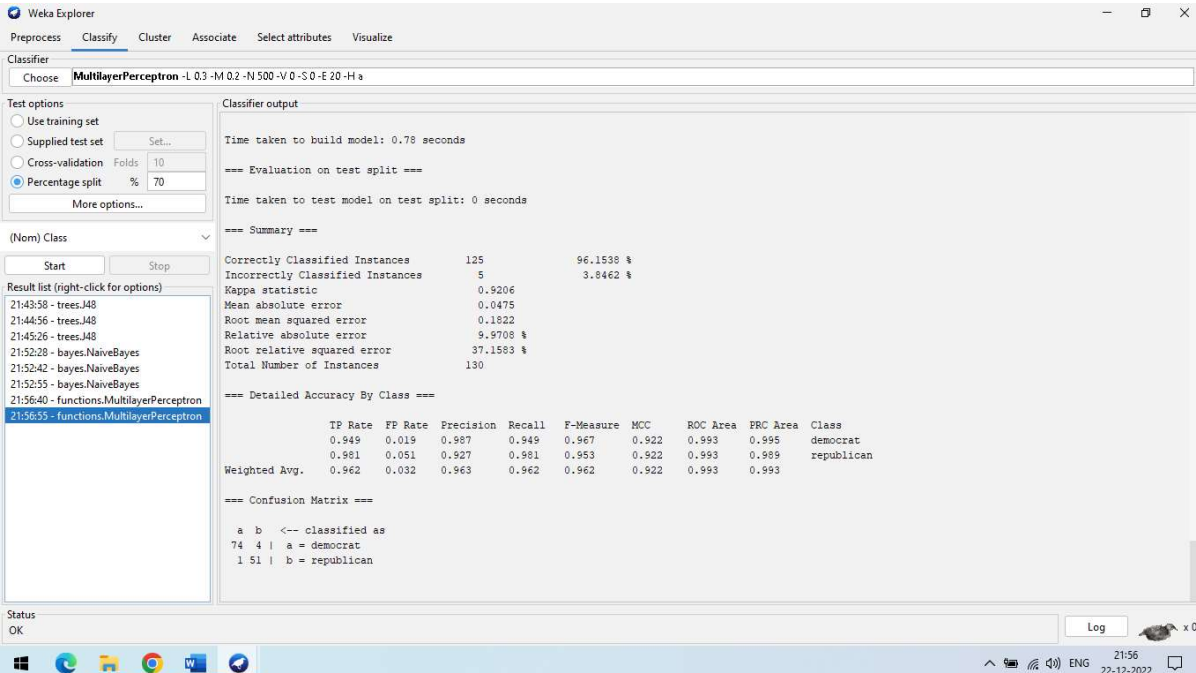
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0.937	0.049	0.968	0.937	0.952	0.881	0.982	0.939	democrat
	0.951	0.063	0.907	0.951	0.928	0.881	0.982	0.972	republican

=== Confusion Matrix ===

	a	b	<-- classified as
149 10	a = democrat		
5 97	b = republican		

Status: OK

**Step 2:** Go to the classify >> choose filter MultilayerPerceptron >> give percentage split(70%) >> click start.



**Weka Explorer**  
 Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose **MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a**

**Test options**  
☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds: 10  
☒ Percentage split %: 70  
 More options...

**Classifier output**

Time taken to build model: 0.78 seconds  
 Time taken to test model on test split: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	125	96.1538 %
Incorrectly Classified Instances	5	3.8462 %
Kappa statistic	0.9206	
Mean absolute error	0.0475	
Root mean squared error	0.1822	
Relative absolute error	9.9708 %	
Root relative squared error	37.1583 %	
Total Number of Instances	130	

=== Detailed Accuracy By Class ===

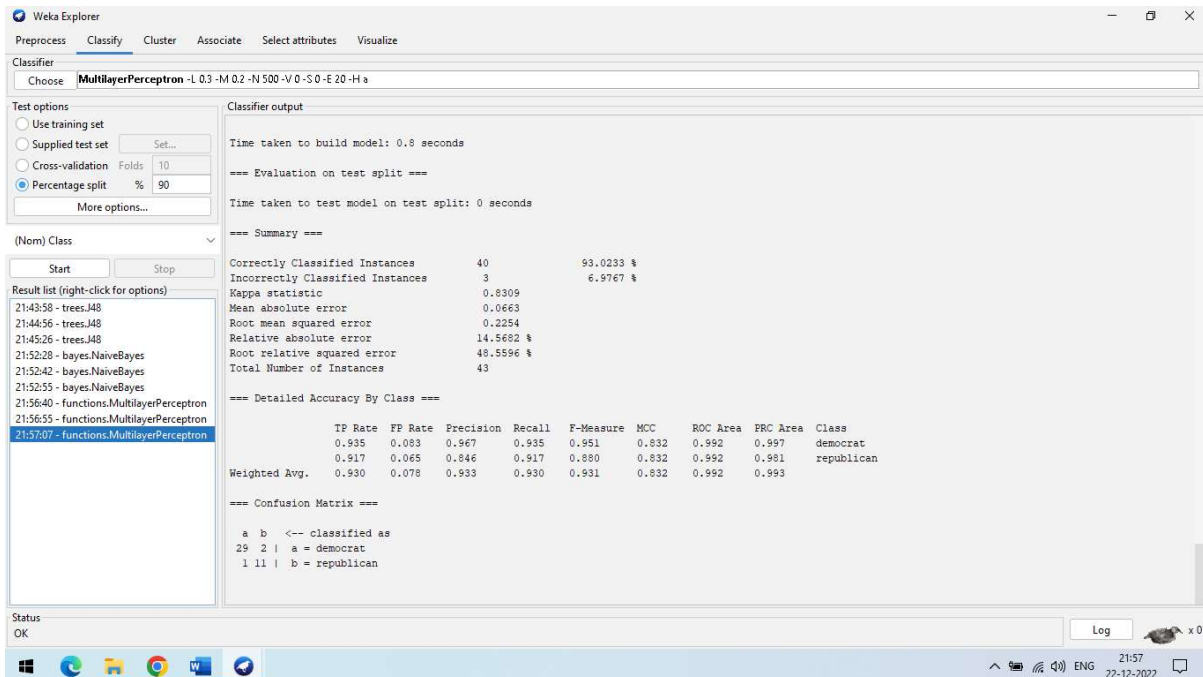
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
Weighted Avg.	0.949	0.019	0.987	0.949	0.967	0.922	0.993	0.995	democrat
	0.961	0.051	0.927	0.961	0.953	0.922	0.993	0.909	republican

=== Confusion Matrix ===

	a	b	<-- classified as
74 4	a = democrat		
1 51	b = republican		

Status: OK

**Step 3:** Go to the classify >> choose filter MultilayerPerceptron >> give percentage split(90%) >> click start.



The screenshot shows the Weka Explorer interface with the MultilayerPerceptron classifier selected. The 'Percentage split' is set to 90%. The 'Classifier output' pane displays the following results:

```

Time taken to build model: 0.8 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances      40      93.0233 %
Incorrectly Classified Instances    3       6.9767 %
Kappa statistic                    0.8309
Mean absolute error                 0.0663
Root mean squared error             0.2254
Relative absolute error             14.5682 %
Root relative squared error         48.5596 %
Total Number of Instances          43

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
Weighted Avg.   0.935   0.083   0.967     0.935   0.951     0.832   0.992   0.997   democrat
                0.917   0.065   0.846     0.917   0.880     0.832   0.992   0.981   republican

=== Confusion Matrix ===

  a  b  <-- classified as
29  2  | a = democrat
 1 11 | b = republican
  
```

### ❖ Accuracy Table For Classification:

Split Percentage (%)	Accuracy (%)			Total instances					
	J48	Naïve Bayes	Multilayer Perceptron	J48		Naïve Bayes		Multilayer Perceptron	
				Correctly identified	Incorrectly identified	Correctly identified	Incorrectly identified	Correctly identified	Incorrectly identified
70	96.1538	90	96.1538	125	5	117	13	125	5
40	93.8697	91.1877	94.2529	245	16	238	23	246	15
90	95.3488	81.3953	93.0233	41	2	35	8	40	3



## PRACTICAL-10

### AIM: Perform Clustering using WEKA tool.

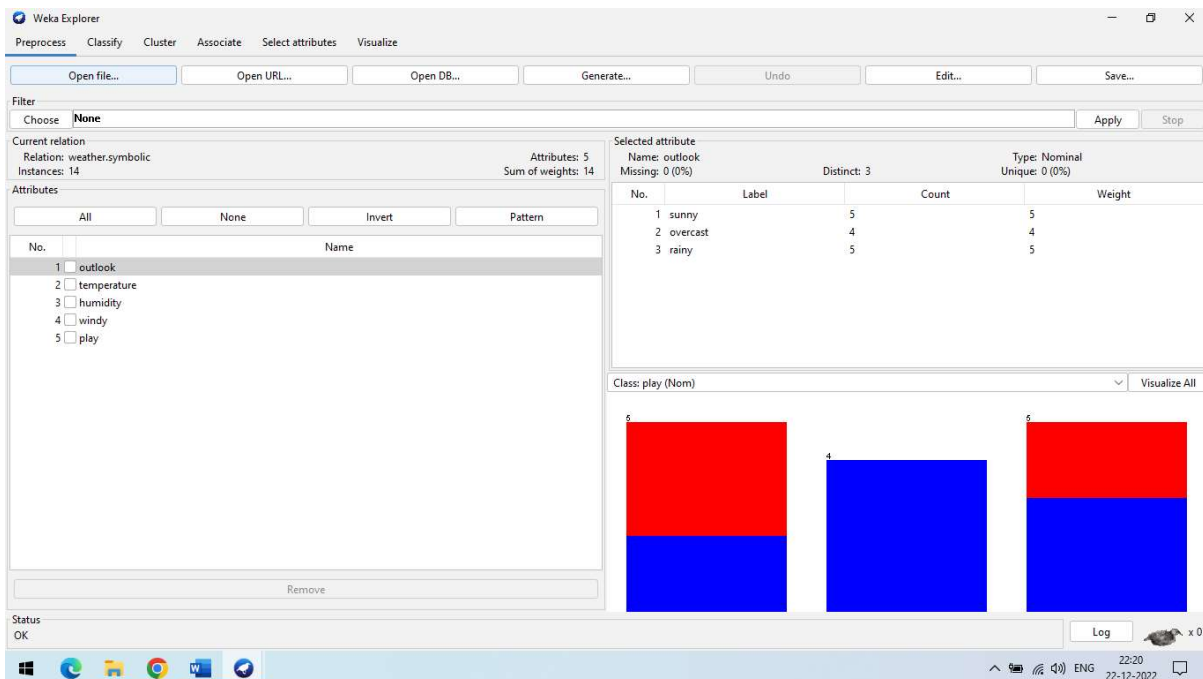
#### Steps:

A clustering algorithm finds groups of similar instances in the entire dataset. Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. WEKA supports several clustering algorithms such as EM, FilteredClusterer, HierarchicalClusterer, SimpleKMeans and so on.

#### 1. Using Simplekmeans

The WEKA SimpleKMeans algorithm uses Euclidean distance measure to compute distances between instances and clusters.

**Step 1:** open the WEKA tool and select the nominal data in the dataset



WEKA Explorer - Filter tab

Current relation: weather.symbolic, Instances: 14, Attributes: 5, Sum of weights: 14

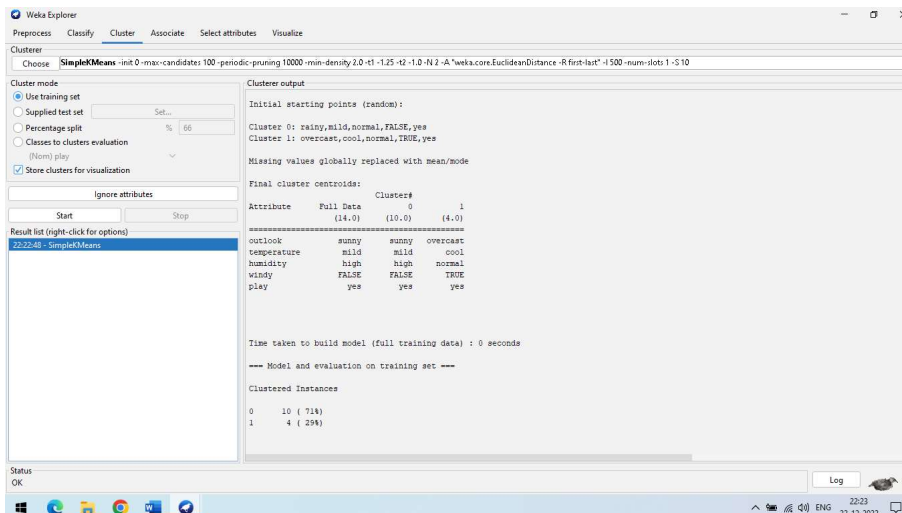
Selected attribute: Name: outlook, Missing: 0 (0%), Distinct: 3, Type: Nominal, Unique: 0 (0%)

No.	Label	Count	Weight
1	sunny	5	5
2	overcast	4	4
3	rainy	5	5

Class: play (Nom)

Visualize All

**Step 2:** To perform clustering, select the "Cluster" tab in the Explorer and click on the "Choose" button and choose a simpleKMeans algorithm. and then click start



WEKA Explorer - Cluster tab

Clusterer: SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 1.25 -t2 1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode: ☒ Use training set, ☐ Supplied test set, ☐ Percentage split, ☐ Classes to clusters evaluation, ☒ Store clusters for visualization

Clusterer output:

Initial starting points (random):

Cluster 0: rainy,mild,normal,FALSE,yes

Cluster 1: overcast,cool,normal,TRUE,yes

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Full Data	Cluster# 0	Cluster# 1
outlook	(14.0)	(10.0)	(4.0)
temperature		mild	cool
humidity		high	normal
windy		FALSE	TRUE
play		yes	yes

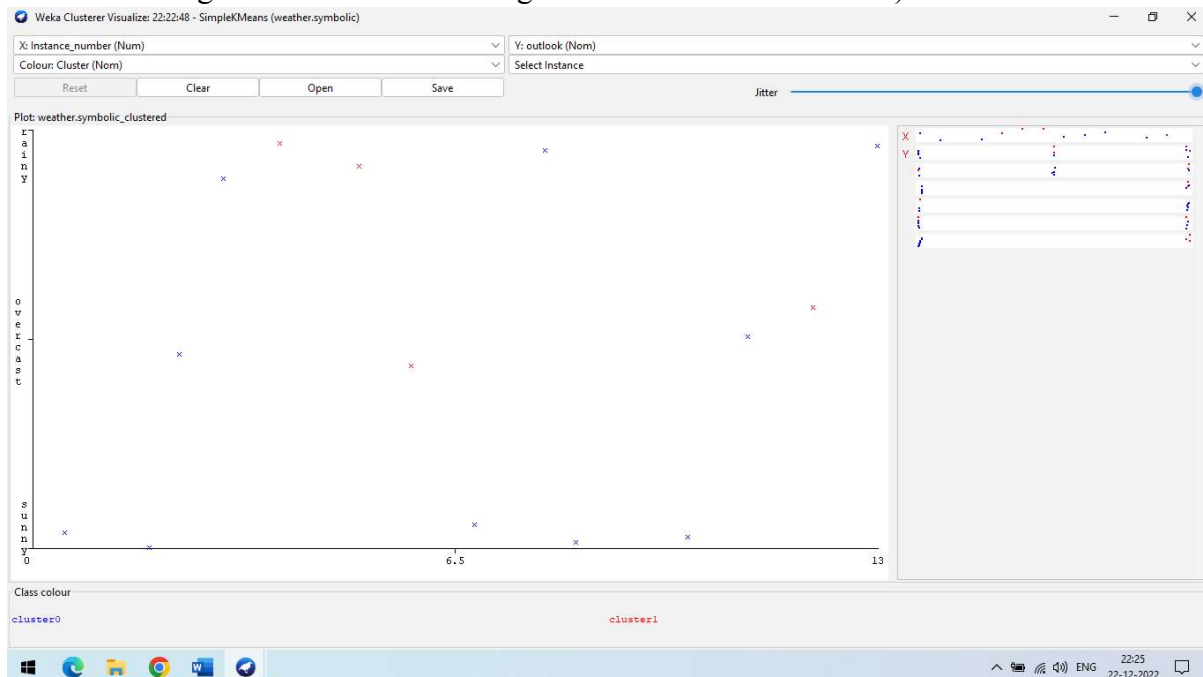
Time taken to build model (full training data) : 0 seconds

Model and evaluation on training set

Clustered Instances

Cluster	Count	Percentage
0	10	(71%)
1	4	(29%)

**Step 3:** Right click on simpleKMeans -> visualize cluster assignment (Another way to grasp the characteristics of each cluster is to visualize them. To do so, right-click the result set on the result. Selecting to visualize cluster assignments from the list column)

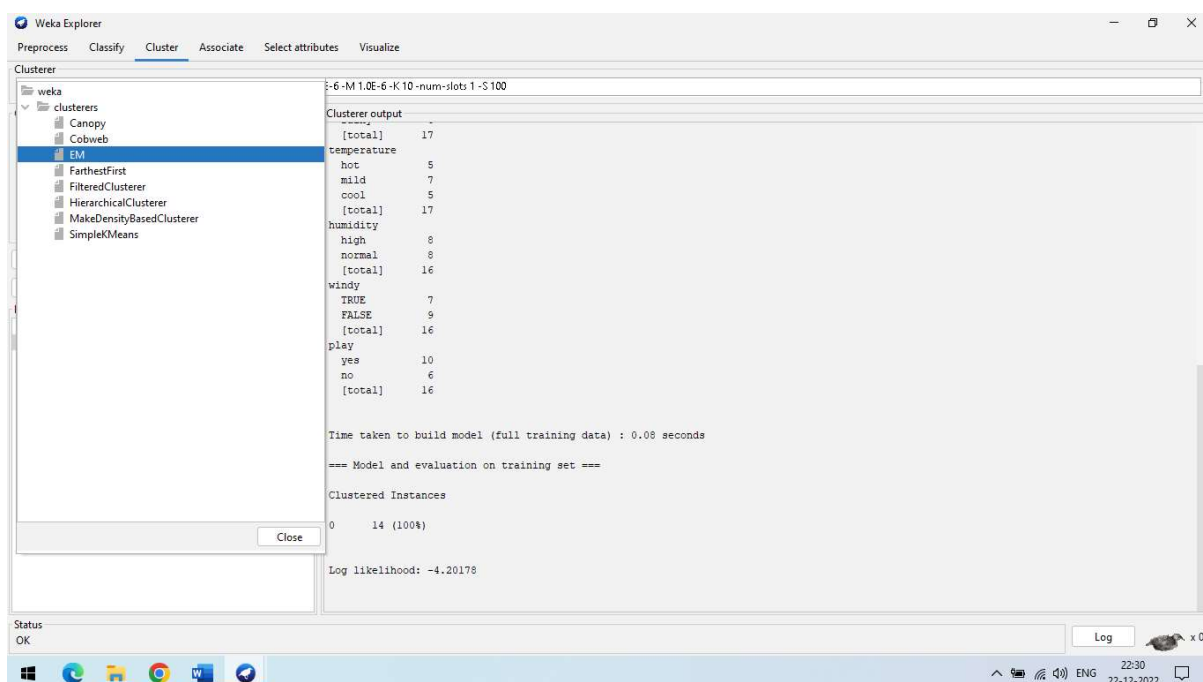


## 2. Using EM Clustering

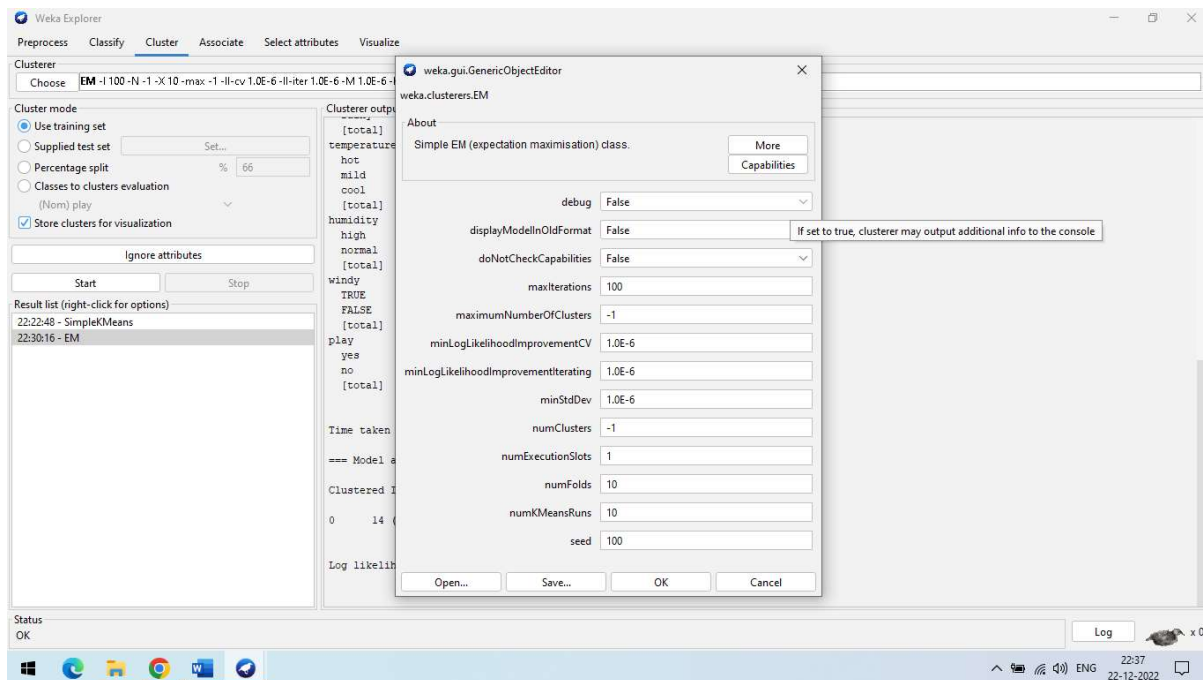
In Weka EM selects the number of clusters automatically by maximizing the logarithm of the likelihood of future data, estimated using cross-validation. Beginning with one cluster, it continues to add clusters until the estimated log-likelihood decreases.

**Step 1:** weka→explorer→open file→weather.nominal.arff→cluster-> EM algorithm

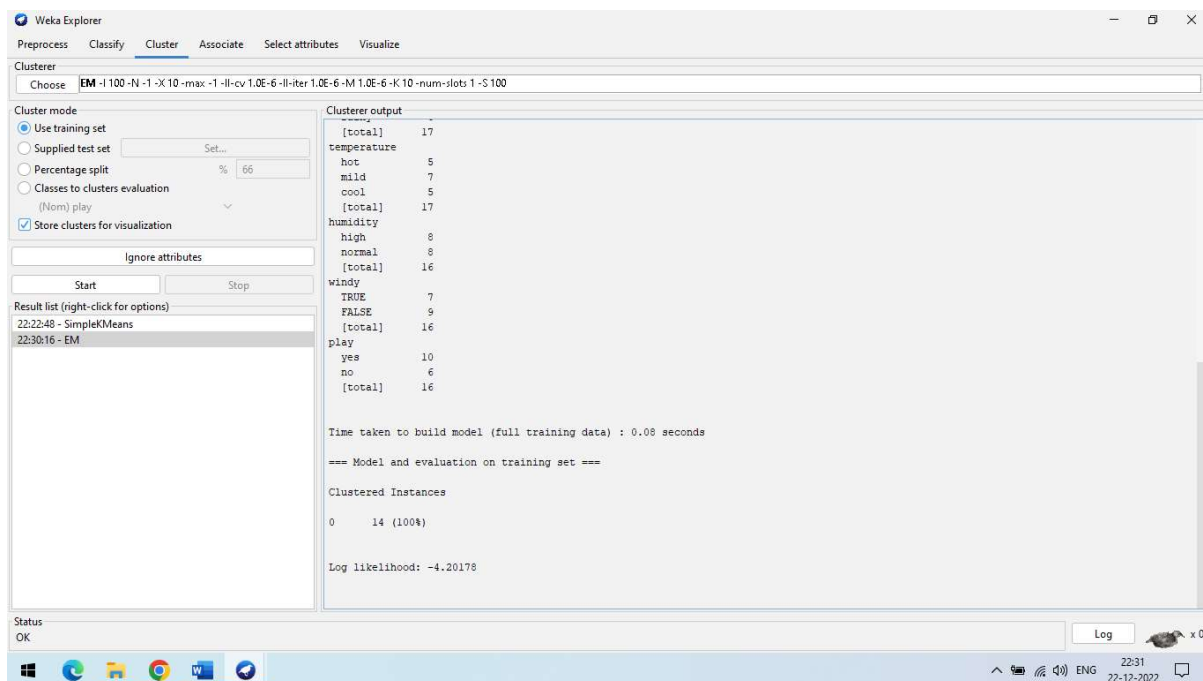
(To perform clustering, select the "Cluster" tab in the Explorer and click on the "Choose" button and choose a EM algorithm.)



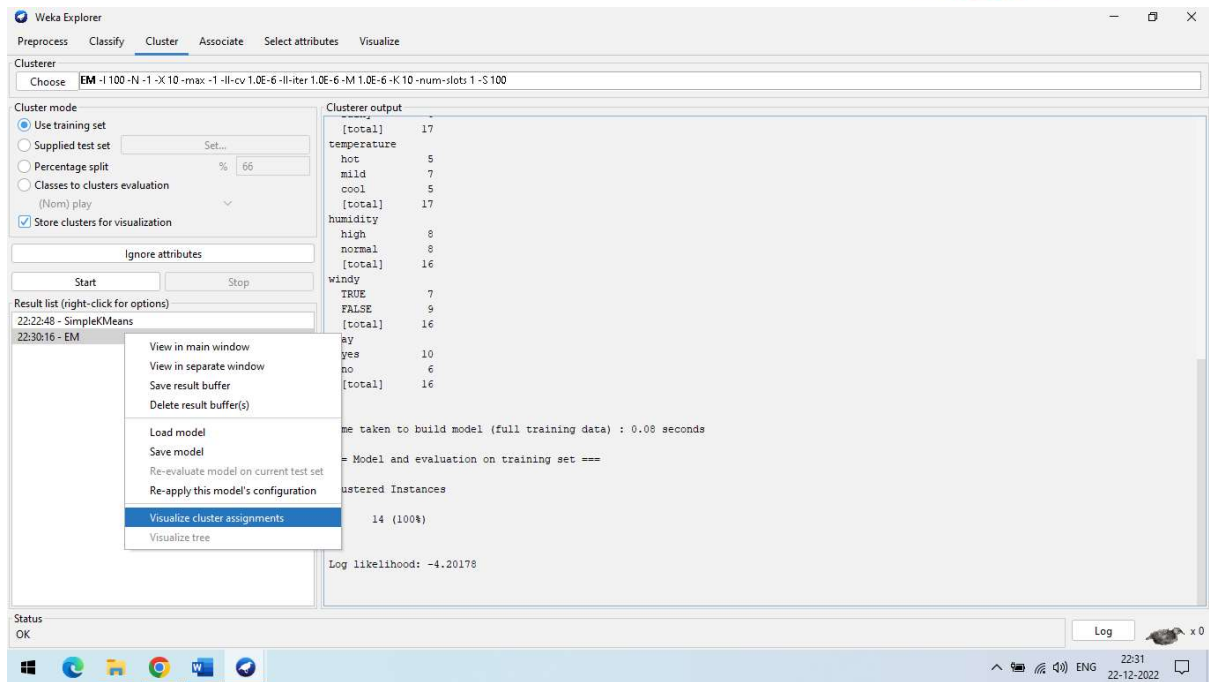
**Step 2:** Click Cluster to apply the clustering algorithm to our loaded data click on the choose button and change the value if you want.



**Step 3:** After that, Select EM to Start the Clustering.



**Step 4** Right-click on EM in result section (Another way to grasp the characteristics of each cluster is to visualise them. To do so, right-click the result set on the result. Selecting to visualise cluster assignments from the list column.)



**Step 5** Click on visualize cluster assignment .(Another way to grasp the characteristics of each cluster is to visualise them. To do so, right-click the result set on the result. Selecting to visualise cluster assignments from the list column.)

