

## FUNDAMENTALS OF DATA MINING

### Definition of Data Mining:

Data mining refers to extracting or mining knowledge from large amounts of data. Data mining can also be referred as knowledge mining from data, knowledge extraction, data archeology and data dredging.

### Applications of Data Mining:

- Business Intelligence applications
- Insurance
- Banking
- Medicine
- Retail/Marketing etc.

### Functionalities of Data Mining:

These functionalities are used to specify the kind of patterns to be found in data mining tasks:

- Descriptive
- Predictive

### The following are the functionalities of data mining:

1. Concept/Class description: Characterization and Discrimination:

Generalize, summarize and contrast data characteristics.

2. Mining frequent patterns, Associations and Correlations:

Frequent patterns are patterns that appear in a data set frequently.

3. Classification and Prediction:

Construct models that describe and distinguish classes or concepts for future prediction. Predicts some unknown or missing numerical values.

4. Cluster analysis:

Class label is unknown. Group data to form new classes. Maximizing intra-class similarity and minimizing inter-class similarity.

5. Outlier analysis:

Outlier: a data object that does not comply with the general behavior of data.

Noise or exception but is quite useful in fraud detection, rare events analysis.

## INTRODUCTION TO WEKA

WEKA (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand.

- A collection of open source of many data mining and machine learning algorithms, including:
  - pre-processing on data
  - Classification
  - clustering
  - association rule extraction
- Created by researchers at the University of Waikato in New Zealand
- Java based (also open source).

### Weka Main Features

- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 15 attribute/subset evaluators +
- 10 search algorithms for feature selection.
- 3 algorithms for finding association rules
- 3 graphical user interfaces:
  - “The Explorer” (exploratory data analysis)
  - “The Experimenter” (experimental environment)
  - “The KnowledgeFlow” (new process model inspired interface)

### Weka Download and Installation

- Download Weka (the stable version) from <http://www.cs.waikato.ac.nz/ml/weka/>
- Choose a self-extracting executable (including Java VM)
- After download is completed, run the self- extracting file to install Weka, and use the default set-ups.

## Start the Weka

- From windows desktop,
  - click “Start”, choose “All programs”,
  - Choose “Weka 3.6” to start Weka
  - Then the first interface window appears:

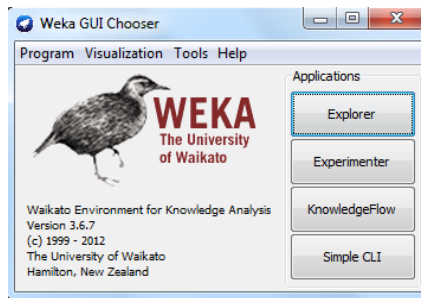


Fig 1. Weka GUI Chooser

## Weka Application Interfaces

- Explorer
  - preprocessing, attribute selection, learning, visualization
- Experimenter
  - testing and evaluating machine learning algorithms
- Knowledge Flow
  - visual design of KDD process
  - Explorer
- Simple Command-line
  - A simple interface for typing commands

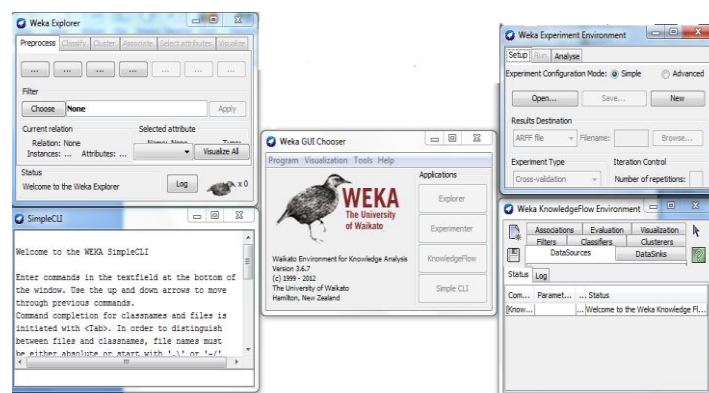


Fig 2. Weka Application Interfaces

## WEKA FUNCTIONS AND TOOLS

### Section Tabs

At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are grayed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.

The tabs are as follows:

1. **Preprocess:** Choose and modify the data being acted on.
2. **Classify:** Train and test learning schemes that classify or perform regression.
3. **Cluster:** Learn clusters for the data.
4. **Associate:** Learn association rules for the data.
5. **Select attributes:** Select the most relevant attributes in the data.
6. **Visualize:** View an interactive 2D plot of the data.

### Load data file

- Load data file in formats: ARFF, CSV, C4.5, binary
- Import from URL or SQL database (using JDBC)

### WEKA data formats

Data can be imported from a file in various formats:

- ARFF : (Attribute Relation File Format) has two sections:
  - i. The Header information defines attribute name, type and relations.
  - ii. The Data section lists the data records.
- CSV: Comma Separated Values (text file)
- C4.5: A format used by a decision induction algorithm. C4.5, requires two separated files
  - Name file: defines the names of the attributes
  - Date file: lists the records (samples)
- Binary: Data can also be read from a URL or from an SQL database (using JDBC)

## DECISION TREE INDUCTION

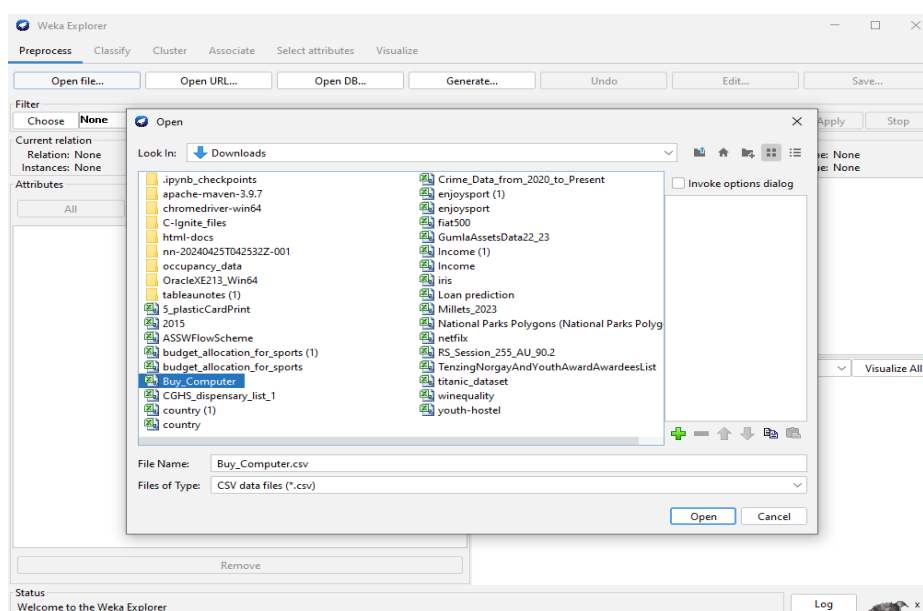
**AIM:** Implementation of Decision Tree Induction

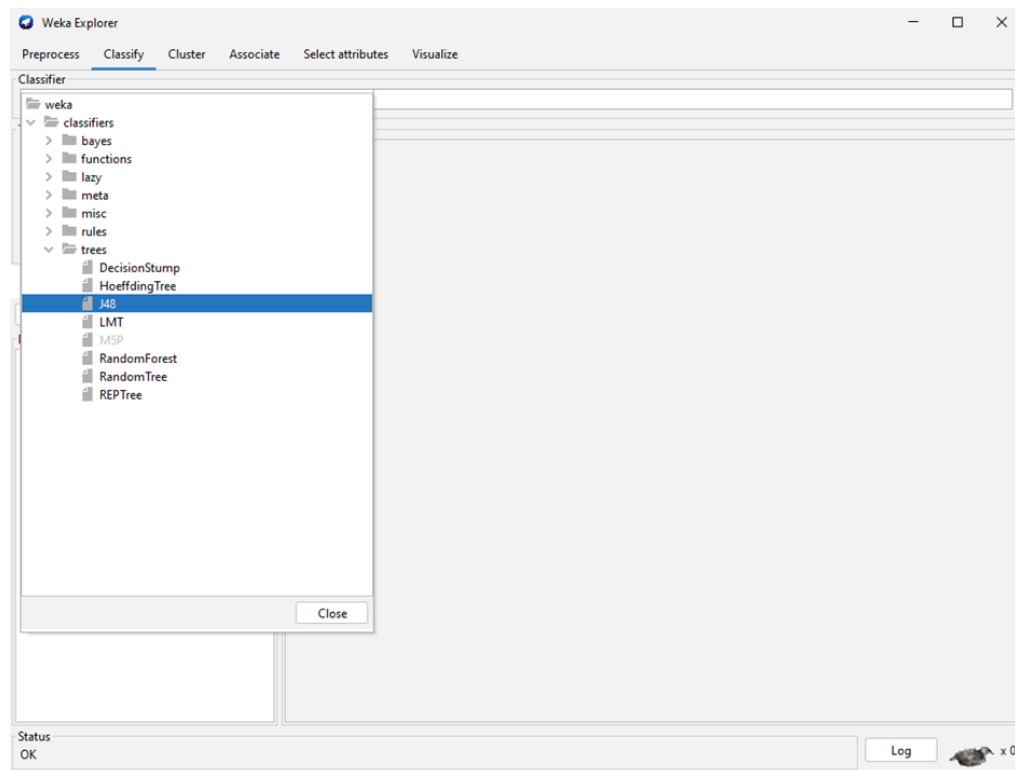
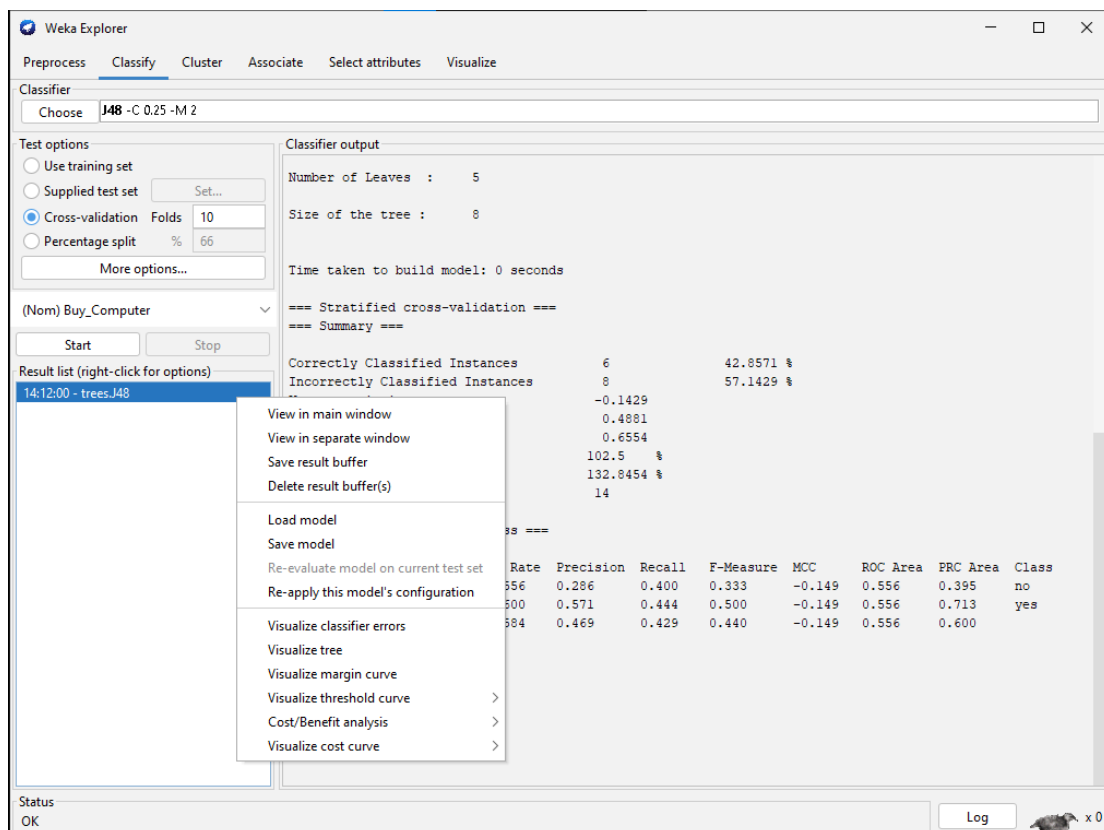
**WEKA TOOL:**

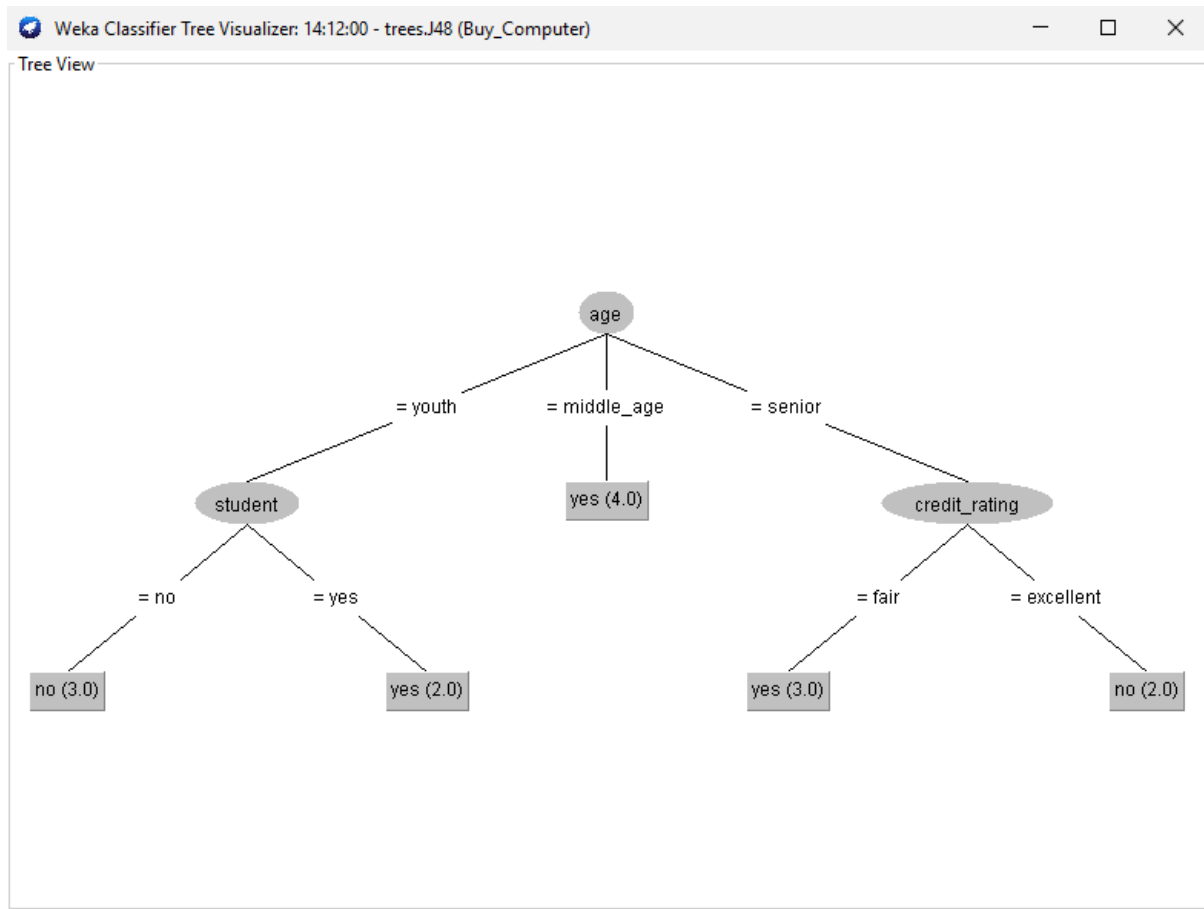
**STEP 1** – Open the Weka Tool.



**STEP 2** – Open the 'Buy\_Computer' Data set.



STEP 3 – Classify -> J48STEP 4 – Click on ‘Start’.

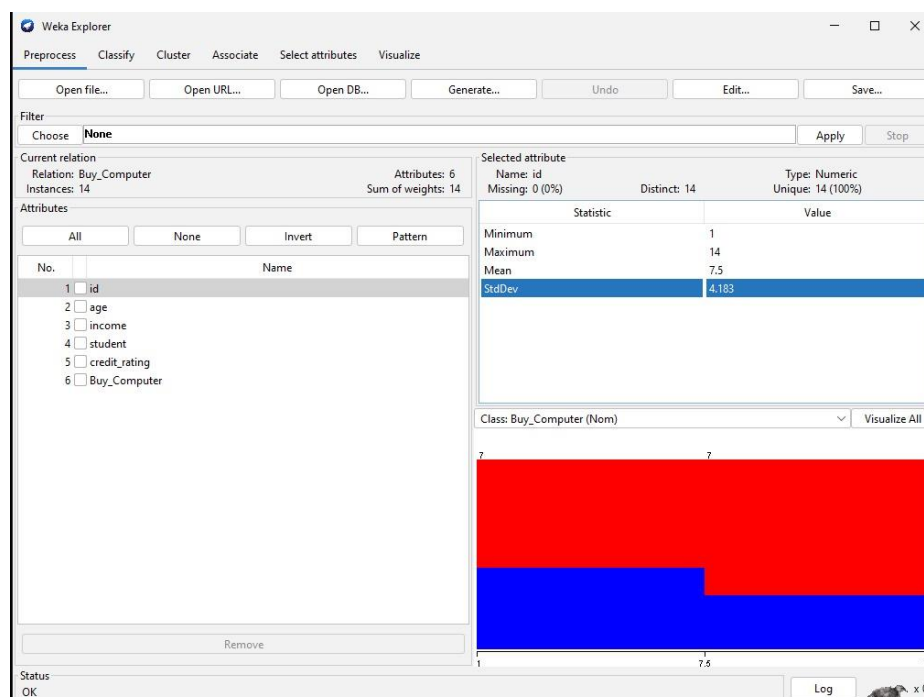
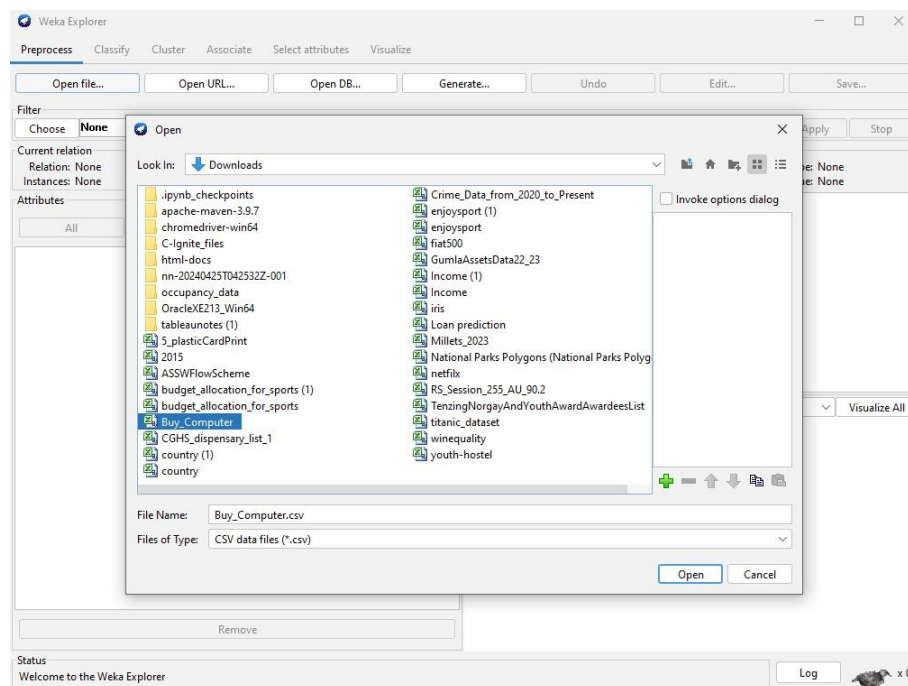
**OUTPUT:**

## INFORMATION GAIN

**AIM:** Calculating Information gain measures

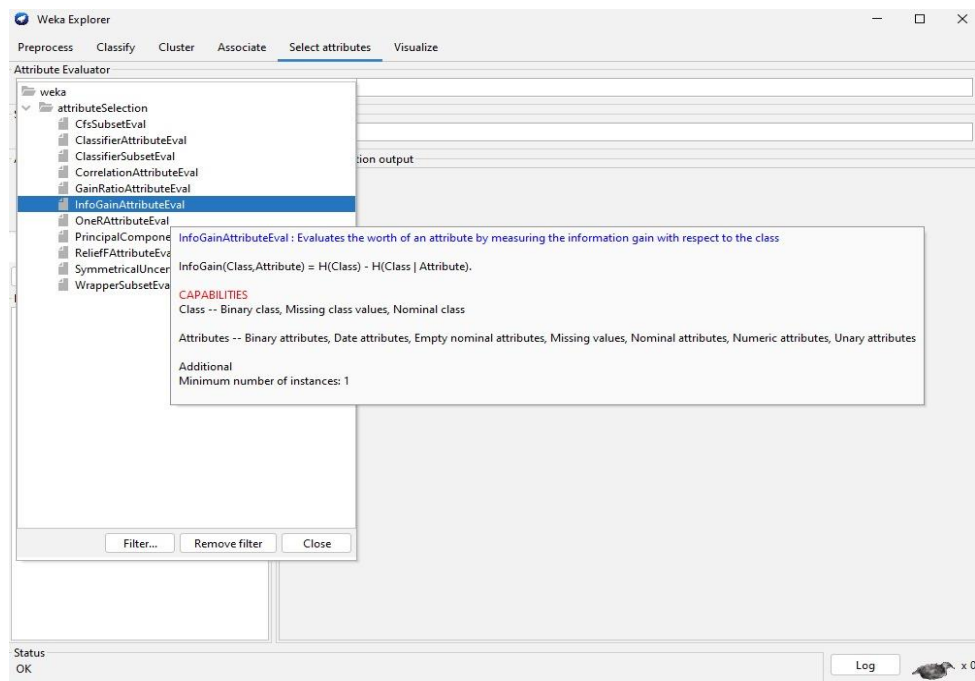
**WEKA TOOL:**

**STEP 1 – Loading the data.**

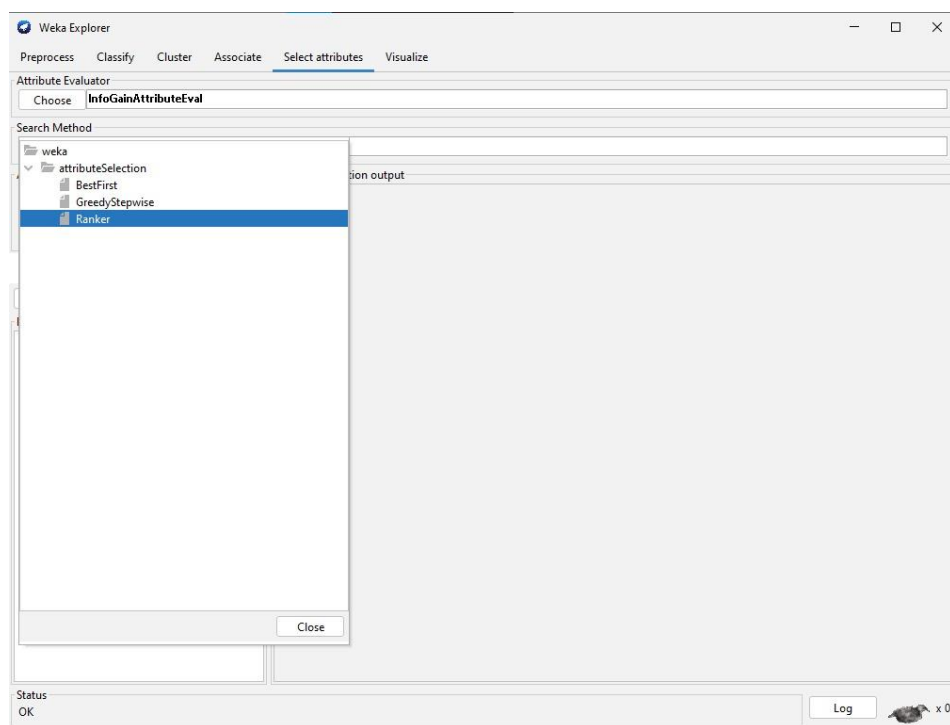




STEP 2 – Go to ‘select attributes’ tab and select ‘InfoGainAttributeEval’.

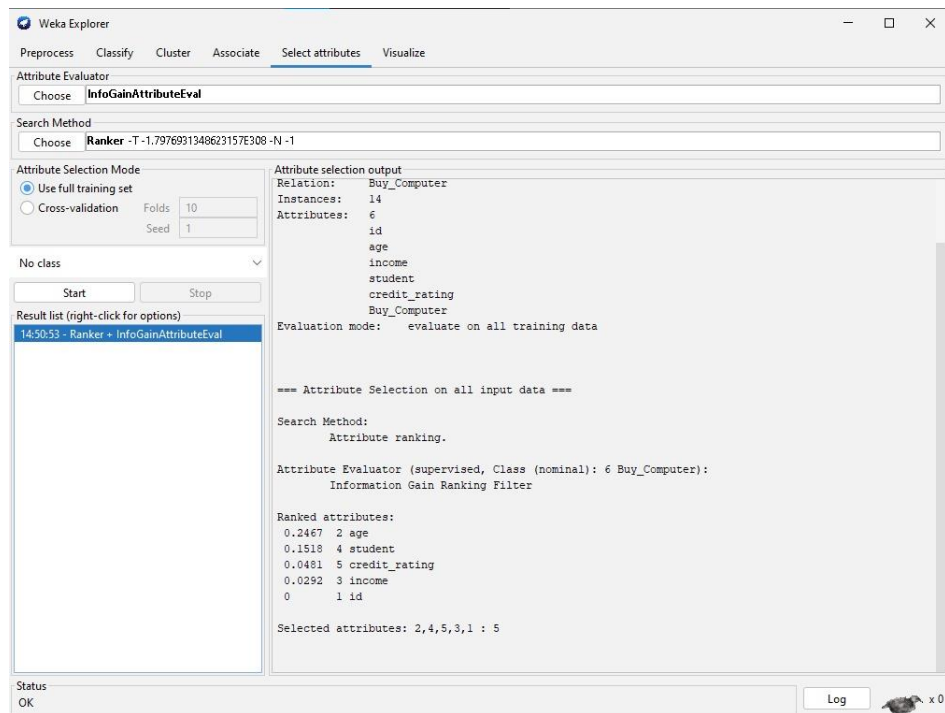


STEP 3 – In ‘search method’ select ‘Ranker’.



STEP 4 – Click on ‘Start’ and check for output.

## OUTPUT:



# NAÏVE BAYESIAN CLASSIFICATION

**AIM:** Classification of data using Bayesian approach

## PROGRAM:

```
In [1]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score
```

```
In [2]: data = pd.read_csv('Buy_Computer.csv')
data['Buy_Computer'] = data.Buy_Computer.map({'yes': 1, 'no': 0})
data['credit_rating'] = data.credit_rating.map({'fair': 0, 'excellent': 1})
data['student'] = data.student.map({'yes': 1, 'no': 0})
data['income'] = data.income.map({'low': 0, 'medium': 1, 'high': 2})
data['age'] = data.age.map({'youth': 0, 'middle_age': 1, 'senior': 2})
X = data.drop(['Buy_Computer'], axis=1)
y = data['Buy_Computer']
X
```

```
Out[2]:
```

	id	age	income	student	credit_rating
0	1	0	2	0	0
1	2	0	2	0	1
2	3	1	2	0	0
3	4	2	1	0	0
4	5	2	0	1	0
5	6	2	0	1	1
6	7	1	0	1	1
7	8	0	1	0	0
8	9	0	0	1	0
9	10	2	1	1	0
10	11	0	1	1	1
11	12	1	1	0	1
12	13	1	2	1	0
13	14	2	1	0	1

```
In [3]: Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)
clf = MLPClassifier()
clf.fit(Xtrain, ytrain)

C:\Users\abhin\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
warnings.warn(
```

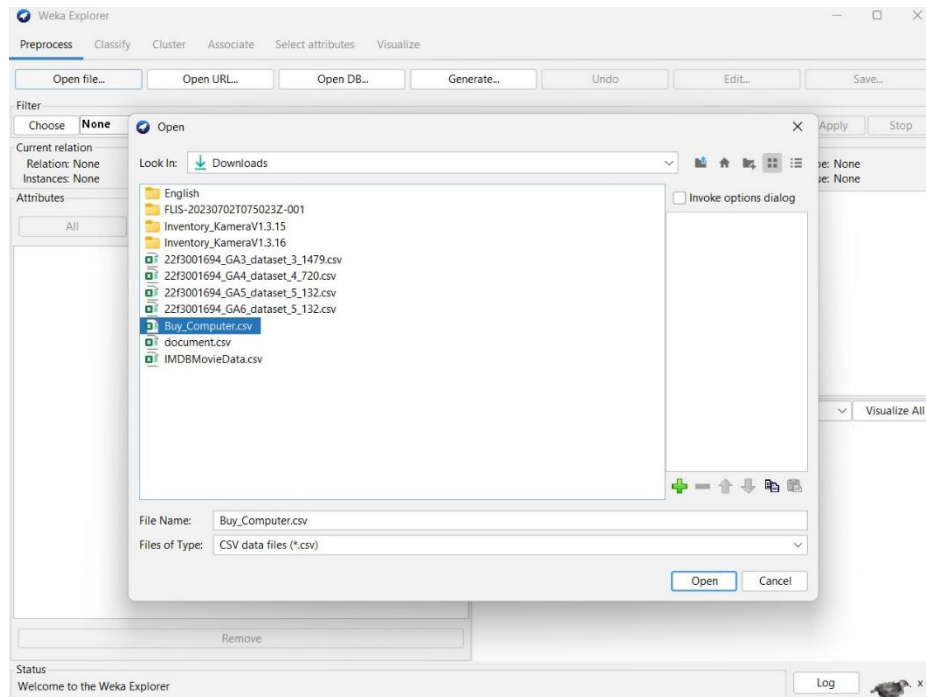
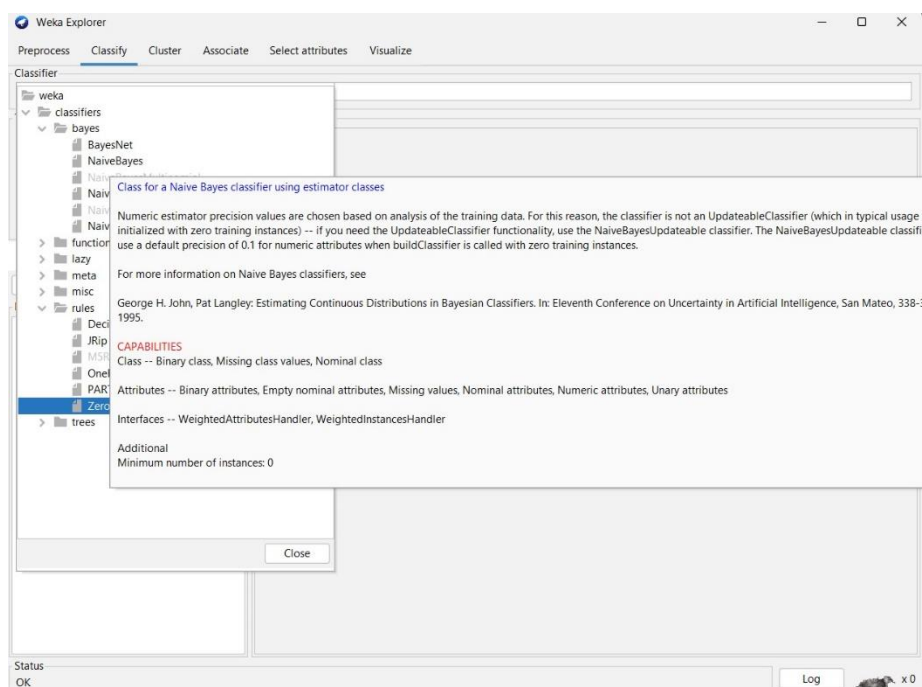
```
Out[3]: MLPClassifier()
```

```
In [4]: pred = clf.predict(Xtest)
print(pred)

[0 1 1 0]
```

```
In [5]: print('Accuracy: ', accuracy_score(ytest, pred))
print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))

Accuracy: 0.25
Recall: 0.3333333333333333
Precision: 0.5
Confusion Matrix:
[[0 1]
 [2 1]]
```

**WEKA TOOL:****STEP 1 – Load the Data.****STEP 2 – Classify tab -> Classifiers -> bayes -> Naïve Bayes**

## OUTPUT:

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 66

More options...

(Nom) Buy\_Computer

Start Stop

Result list (right-click for options)

09:33:55 - bayes.NaiveBayes

Classifier output

```

fair      3.0    7.0
excellent 4.0    4.0
[total]   7.0   11.0

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      8           57.1429 %
Incorrectly Classified Instances    6           42.8571 %
Kappa statistic                    0.0667
Mean absolute error                 0.4381
Root mean squared error             0.5418
Relative absolute error             91.9971 %
Root relative squared error        109.8188 %
Total Number of Instances          14

=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
0.400      0.333   0.400    0.400    0.400    0.067   0.511   0.529    no
0.667      0.600   0.667    0.667    0.667    0.067   0.511   0.690    yes
Weighted Avg.   0.571   0.505   0.571    0.571    0.571    0.067   0.511   0.632

=== Confusion Matrix ===
a b  <-- classified as
2 3 | a = no
3 6 | b = yes

```

Status OK

Log x0



**Predict on dataset which model has not seen before**

```
In [6]: y_pred=knn.predict(X_test)
print("Predicted values by the model : ")
print(y_pred)
print("Actual Target vslues in Test Dataset :")
print(y_test)
```

```
Predicted values by the model :
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
Actual Target vslues in Test Dataset :
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

C:\Users\abhin\anaconda3\lib\site-packages\sklearn\neighbors\\_classification.py:228: FutureWarning: Unlike other reduction functions (e.g. 'skew', 'kurtosis'), the default behavior of 'mode' typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of 'keepdims' will become False, the 'axis' over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set 'keepdims' to True or False to avoid this warning.

```
mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
In [7]: print("Confusion Matrix of above y_pred and y_test : \n",confusion_matrix(y_test,y_pred))
```

```
Confusion Matrix of above y_pred and y_test :
[[10  0  0]
 [ 0  8  1]
 [ 0  0 11]]
```

```
In [8]: print("Classification report : \n",classification_report(y_test,y_pred))
```

```
Classification report :
              precision    recall  f1-score   support

     0               1.00      1.00      1.00         10
     1               1.00      0.89      0.94          9
     2               0.92      1.00      0.96         11

 accuracy               0.97
 macro avg              0.97
 weighted avg           0.97
```

## K – MEANS

**AIM:** Implementation of K – means algorithm

### PROGRAM:

#### Importing necessary modules and Loading data

```
In [1]: from sklearn.cluster import KMeans
import numpy as np

X = np.array([[1.713,1.586], [0.180,1.786], [0.353,1.240],[0.940,1.566],[1.486,0.759],[1.266,1.106],[1.540,0.419],[0.459,1.799],
```

#### Training the kmeans model by using 3 centroids

```
In [2]: kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
```

```
In [3]: # Number of iterations taken by the model
kmeans.n_iter_
```

```
Out[3]: 3
```

```
In [4]: # The Final Centroid values
print(kmeans.cluster_centers_)

[[1.26633333 0.45466667]
 [0.33066667 1.60833333]
 [1.30633333 1.41933333]]
```

#### Prediction

```
In [5]: # Cluster of each Data point
print(kmeans.labels_)
```

```
[2 1 1 2 0 2 0 1 0]
```

```
In [6]: print('Cluster number of Data Point (0.906,0.606) is : ',kmeans.predict([[0.906, 0.606]]))

Cluster number of Data Point (0.906,0.606) is : [0]
```

### WEKA TOOL:

**STEP 1** - Create Excel Document of the above data and save it as (.CSV delimited) file type.

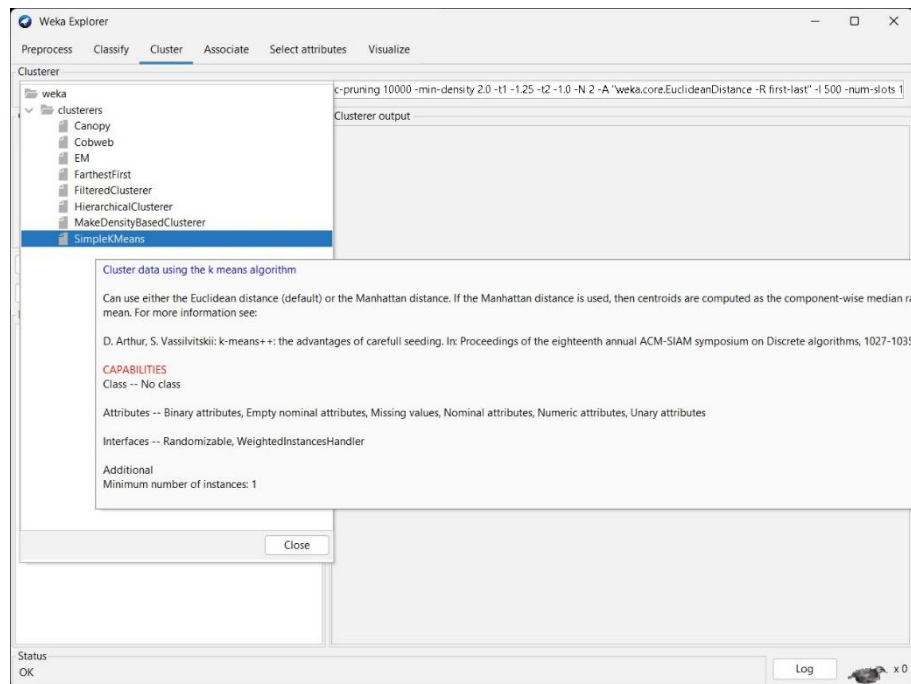
	A	B	C	D
1	X	Y		
2	185	72		
3	170	56		
4	168	60		
5	179	68		
6	182	72		
7	188	77		
8				



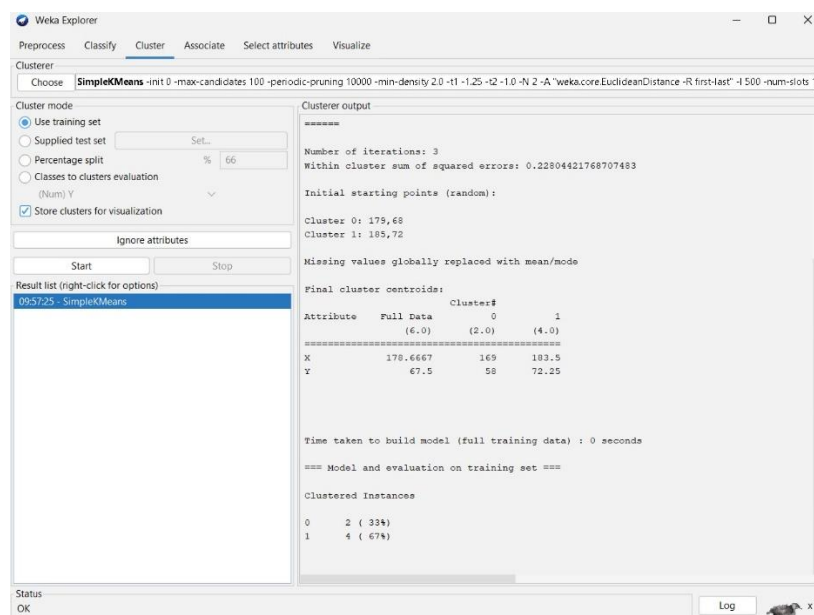
**STEP 2** – Open WEKA tool and click on WEKA tool and then click on explorer.

**STEP 3** - Click on open file tab and take the file from the desired location, the file should be of (.csv)

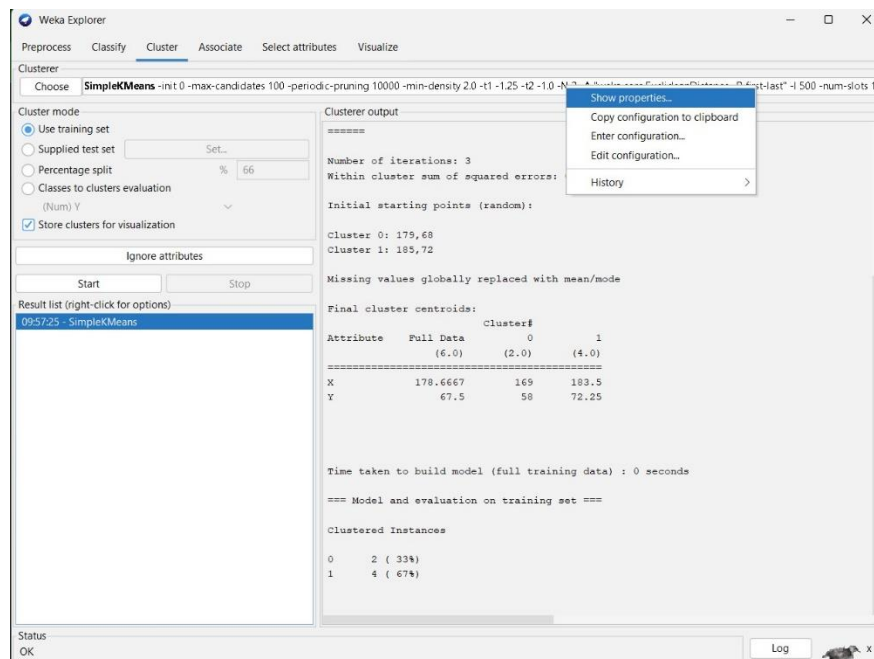
**STEP 4** - Select all attributes and click on cluster tab, In the 'Cluster' box click on 'Choose' button. In pull-down menu select WEKA Clusters and select the cluster scheme 'SimpleKMeans'. and select the simple KMeans option.



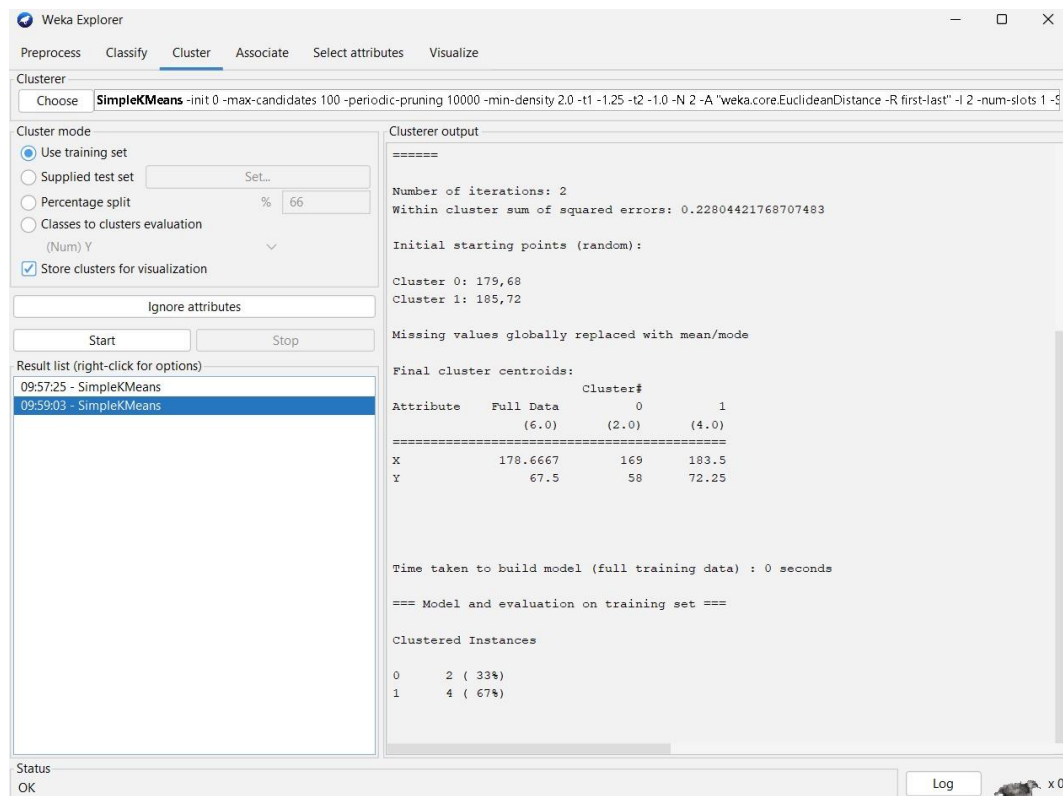
**STEP 5** - Click on Start. The results window showing the clusters is shown below:



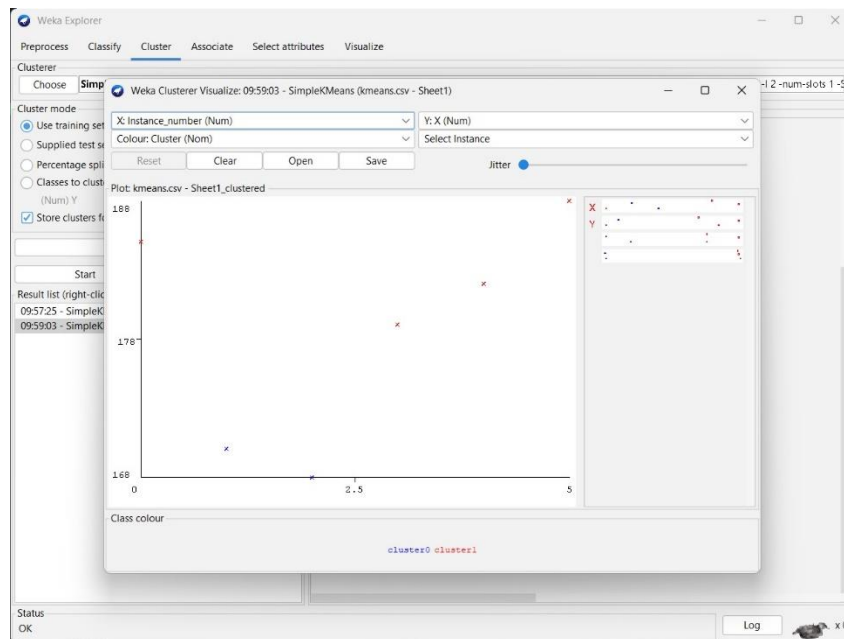
**STEP 6** - Now Change the number of iterations. Right click on the simplekmeans and click on show properties.



**STEP 7** - Change maxIterations to 2. Click on start button and see the results.



**STEP 8** - To see the data tuples in each cluster, right click on the result list and click on visualize cluster assignments.



## BIRCH ALGORITHM

**AIM:** Implementation of BIRCH algorithm.

**PROGRAM:**

```
import numpy as np
from sklearn.cluster import Birch
X = np.array([[9, 6], [10, 4], [4, 4],
              [5, 8], [3, 8], [2, 5], [8,5],[4,6],[8,4],[9,3]])
model= Birch(n_clusters=2)
clustering = model.fit(X)
print("Labels :",clustering.labels_)
```

Labels : [1 1 0 0 0 0 1 0 1 1]

## PAM ALGORITHM

**AIM:** Implementation of PAM algorithm.

**PROGRAM:**

```
9s pip install scikit-learn-extra
```

```
0s import numpy as np
from sklearn_extra.cluster import KMedoids
X = np.array([[9, 6], [10, 4], [4, 4],
              [5, 8], [3, 8], [2, 5], [8,5],[4,6],[8,4],[9,3]])
model_km = KMedoids(n_clusters=2)
km = model_km.fit(X)
print("Labels :",km.labels_)
print("Cluster centers :",km.cluster_centers_)
```

Labels : [1 1 0 0 0 0 1 0 1 1]  
Cluster centers : [[4. 6.]  
[8. 4.]]

## DBSCAN

**AIM:** Implementation of DB Scan algorithm.

### PROGRAM:

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

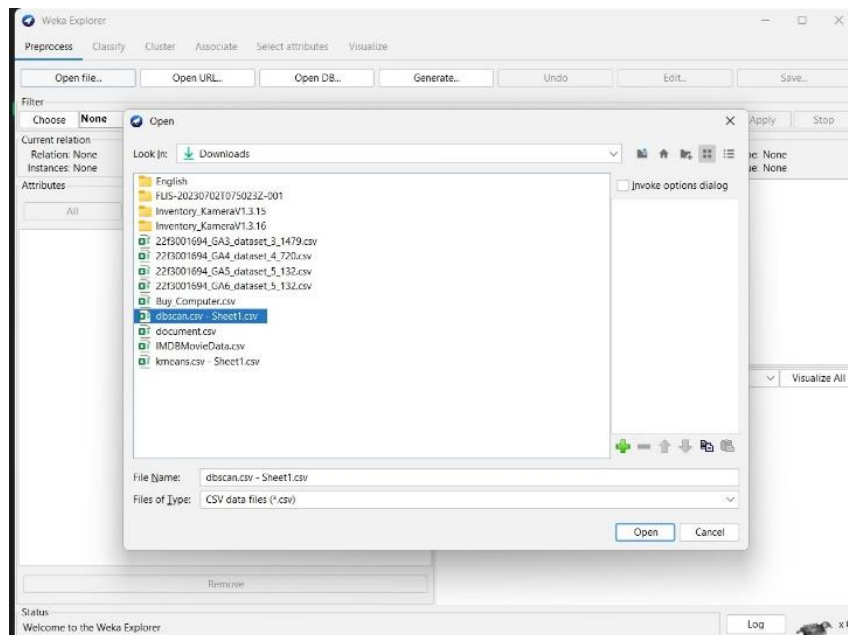
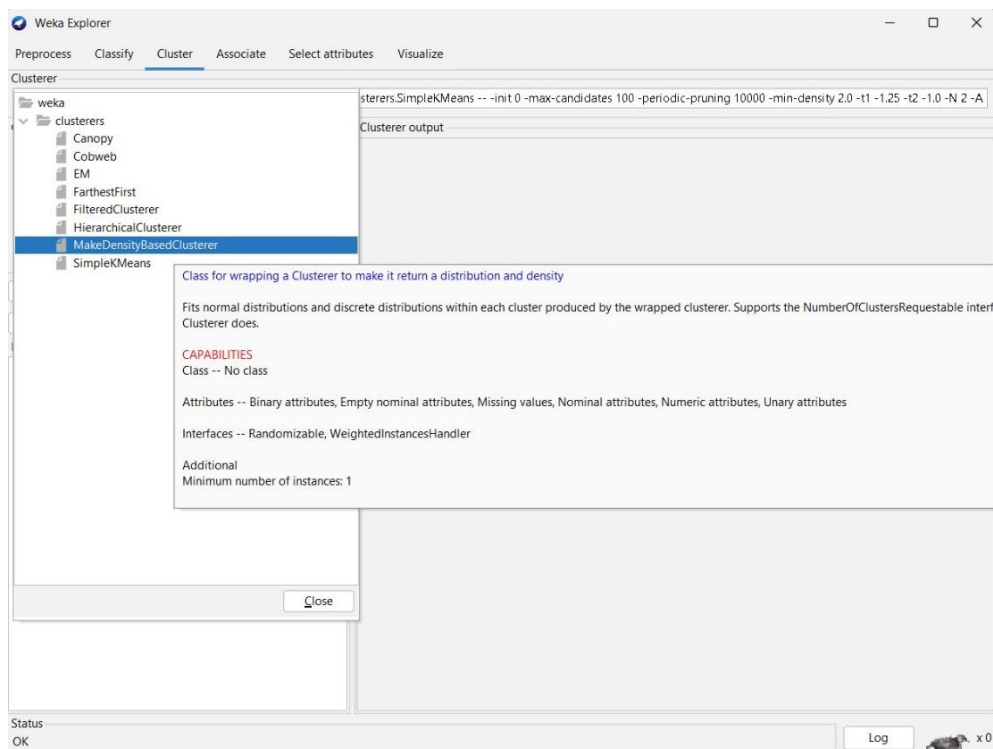
In [2]: X = np.array([[3, 7], [4, 6], [5, 5],
[7, 2], [7, 3], [6, 2], [7,2],[3,4],[3,3],[2,6],[3,4],[2,4]])
clustering = DBSCAN(eps=1.9, min_samples=4).fit(X)
clustering.labels_

Out[2]: array([-1, -1, -1,  0,  0,  0,  0,  1,  1, -1,  1,  1], dtype=int64)
```

### WEKA TOOL:

STEP 1 – Create the Data set.

A14	A	B	C
1	X	Y	
2	3	7	
3	4	6	
4	5	5	
5	7	2	
6	7	3	
7	6	2	
8	7	2	
9	3	4	
10	3	3	
11	2	6	
12	3	4	
13	2	4	

STEP 2 – Upload the Data set.STEP 3 – Cluster -> MakeDensityBasedCluster

**STEP 4 - Click 'Start'.**