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

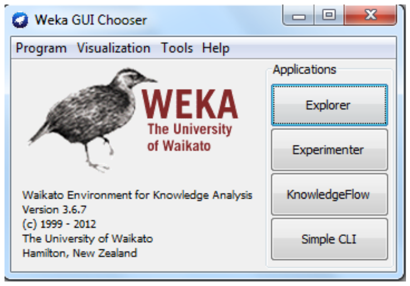
- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 15 attribute/subset evaluators +
- 10 search algorithms for feature selection.
- 3 algorithms for finding association rules
- 3 graphical user interfaces:
  - "The Explorer" (exploratory data analysis)
  - "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:

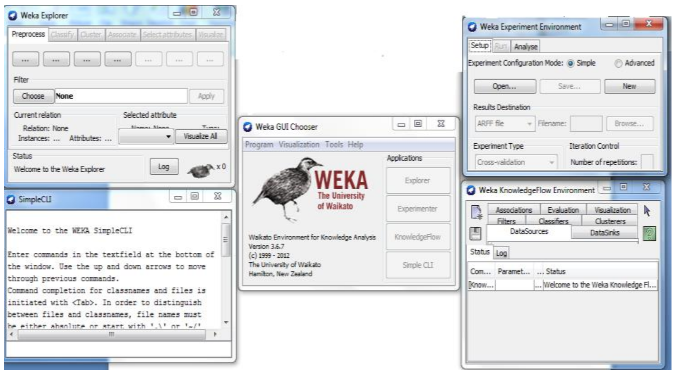


Waikato Environment for Knowledge Analysis
Version 3.6.7
(c) 1999 - 2012
The University of Waikato
Hamilton, New Zealand

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



Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of the window. Use the up and down arrows to move through previous commands.
Command completion for classnames and files is initiated with <Tab>. In order to distinguish between files and classnames, file names must be either absolute or start with '.\' or '~/'

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

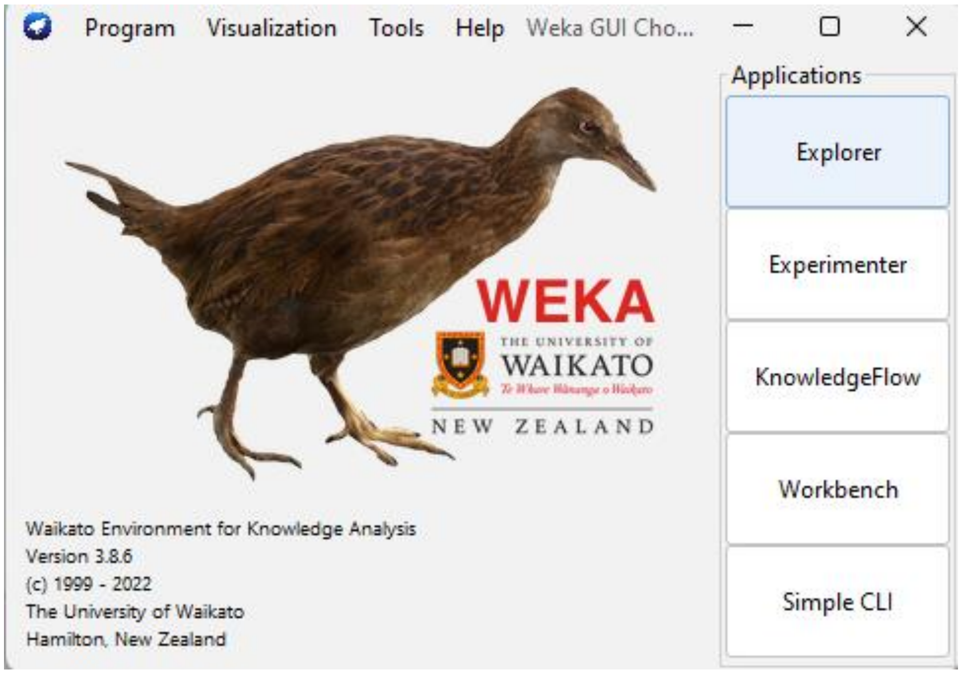
  Date file: lists the records (samples)
- Binary: Data can also be read from a URL or from an SQL database (using JDBC)

DECISION TREE INDUCTION

AIM: Implementation of Decision Tree Induction

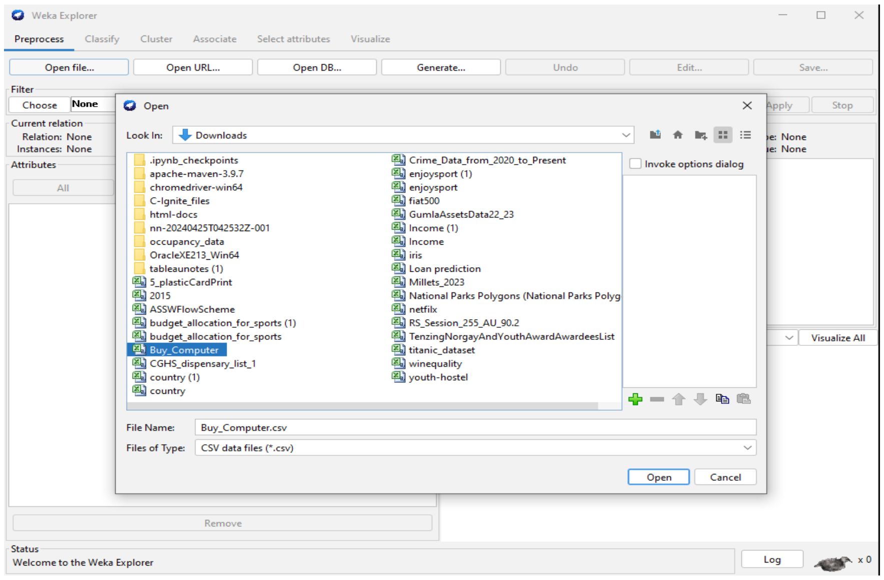
WEKA TOOL:

STEP 1 – Open the Weka Tool.



Waikato Environment for Knowledge Analysis
Version 3.8.6
(c) 1999 - 2022
The University of Waikato
Hamilton, New Zealand

STEP 2 – Open the 'Buy_Computer' Data set.



None

Current relation
Relation: None
Instances: None

Open

Look In: Downloads

```
.ipynb_checkpoints                     Crime_Data_from_2020_to_Present
apache-maven-3.9.7                     enjoysport (1)
chromedriver-win64                     enjoysport
C-Ignite_files                         fiat500
html-docs                              GumlaAssetsData22_23
nn-20240425T042532Z-001                Income (1)
occupancy_data                         Income
OracleXE213_Win64                      iris
tableaunotes (1)                       Loan prediction
5_plasticCardPrint                     Millets_2023
2015                                   National Parks Polygons (National Parks Polyg
ASSWFlowScheme                         netfilx
budget_allocation_for_sports (1)       RS_Session_255_AU_90.2
budget_allocation_for_sports           TenzingNorgayAndYouthAwardAwardeesList
Buy_Computer                           titanic_dataset
CGHS_dispensary_list_1                 winequality
country (1)                            youth-hostel
country
```

Invoke options dialog

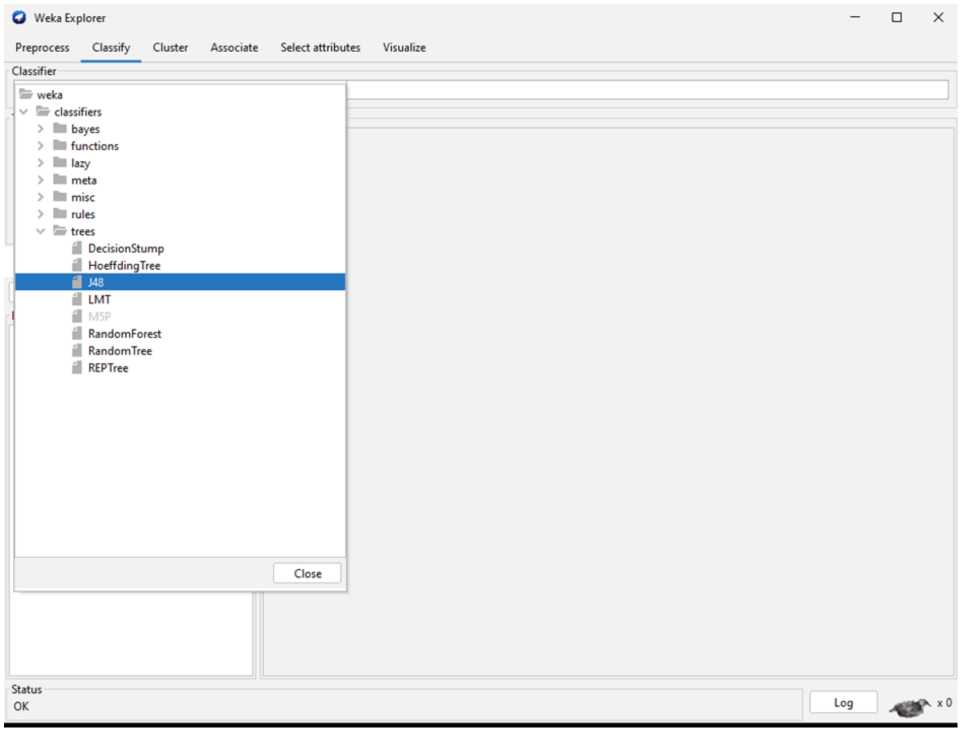
File Name: Buy_Computer.csv

Files of Type: CSV data files (*.csv)

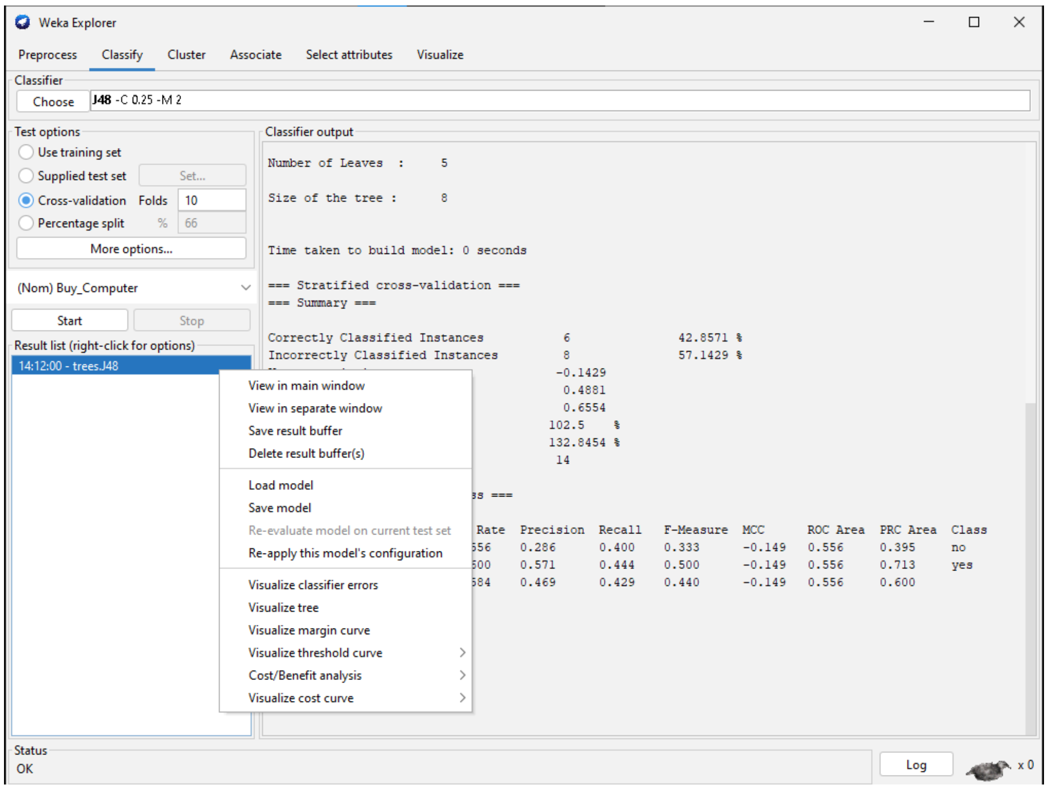
Status
Welcome to the Weka Explorer

x 0

STEP 3 – Classify -> J48



STEP 4 – Click on 'Start'.



Classifier: J48 -C 0.25 -M 2

Test options: Cross-validation Folds 10

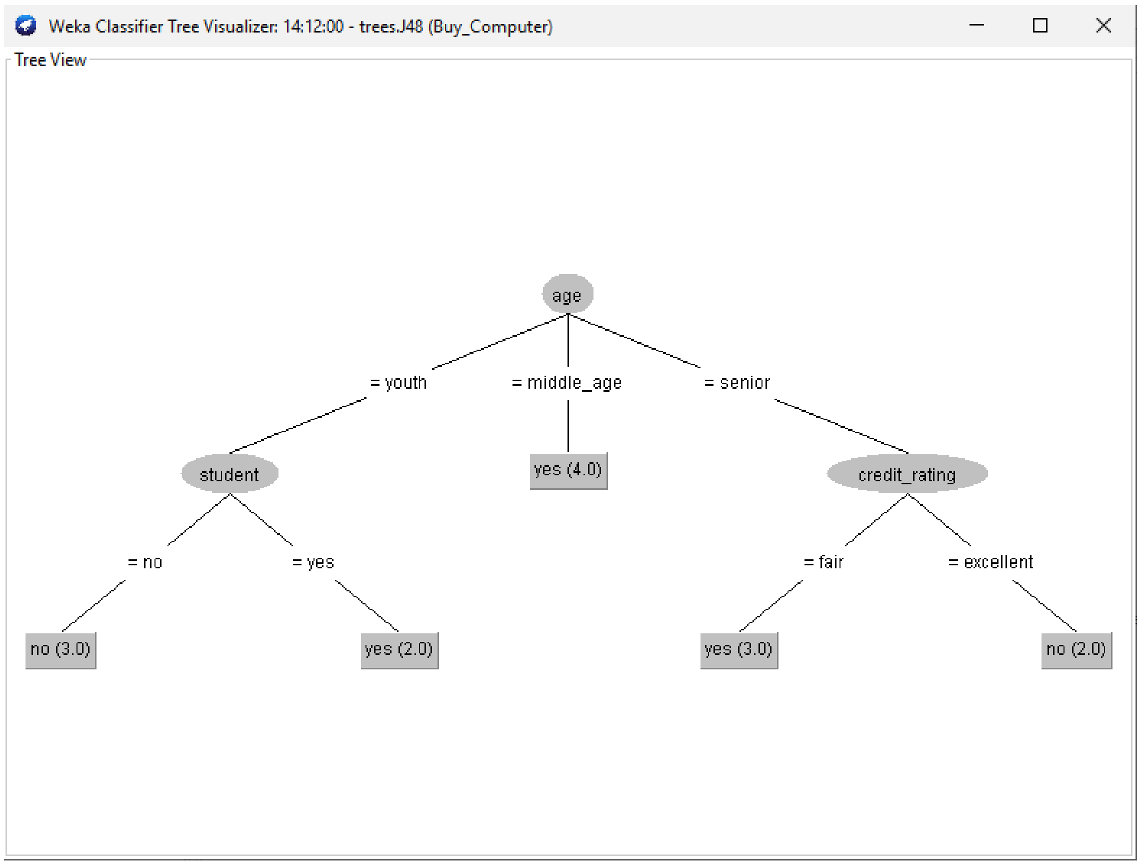
(Nom) Buy_Computer

Result list: 14:12:00 - trees.J48

Classifier output

```
Number of Leaves  :     5

Size of the tree :      8


Time taken to build model: 0 seconds

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

Correctly Classified Instances           6               42.8571 %
Incorrectly Classified Instances         8               57.1429 %
                                        -0.1429
                                         0.4881
                                         0.6554
                                       102.5    %
                                       132.8454 %
                                        14

                  ss ===

                  Rate   Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                  556    0.286      0.400    0.333      -0.149   0.556     0.395     no
                  500    0.571      0.444    0.500      -0.149   0.556     0.713     yes
                  584    0.469      0.429    0.440      -0.149   0.556     0.600
```

OUTPUT:

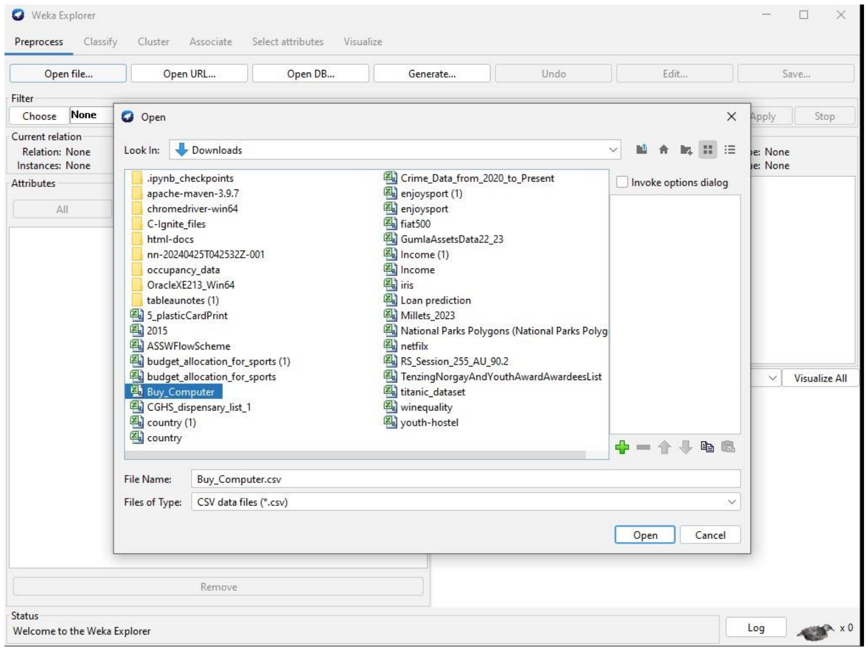

Weka Classifier Tree Visualizer: 14:12:00 - trees.J48 (Buy_Computer)
Tree View
age
= youth
= middle_age
= senior
student
yes (4.0)
credit_rating
= no
= yes
= fair
= excellent
no (3.0)
yes (2.0)
yes (3.0)
no (2.0)

INFORMATION GAIN

AIM: Calculating Information gain measures

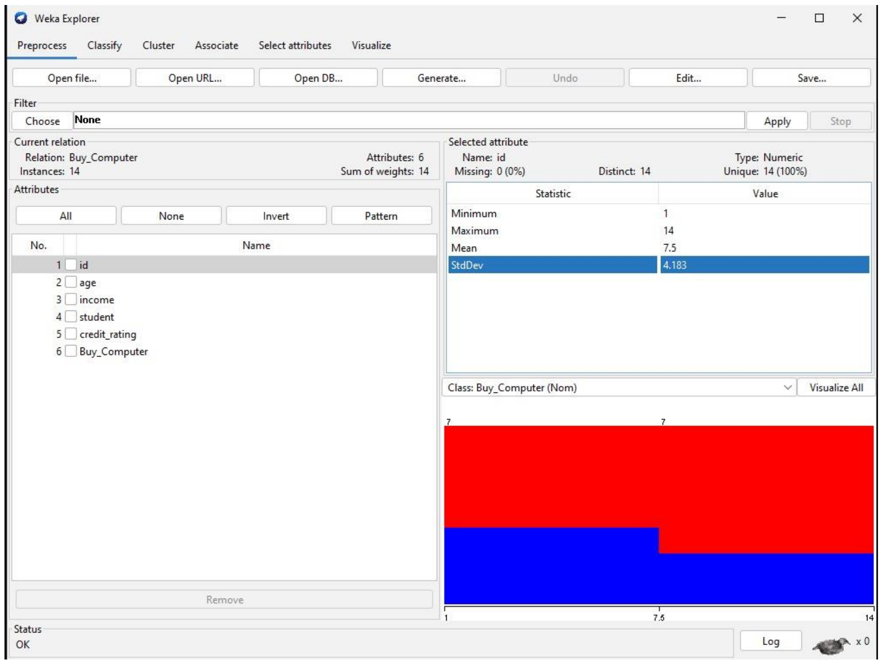
WEKA TOOL:

STEP 1 – Loading the data.



File Name: Buy_Computer.csv
Files of Type: CSV data files (*.csv)

Status: Welcome to the Weka Explorer



Filter: None

Current relation
Relation: Buy_Computer Attributes: 6
Instances: 14 Sum of weights: 14

Attributes

No.	Name
1	id
2	age
3	income
4	student
5	credit_rating
6	Buy_Computer

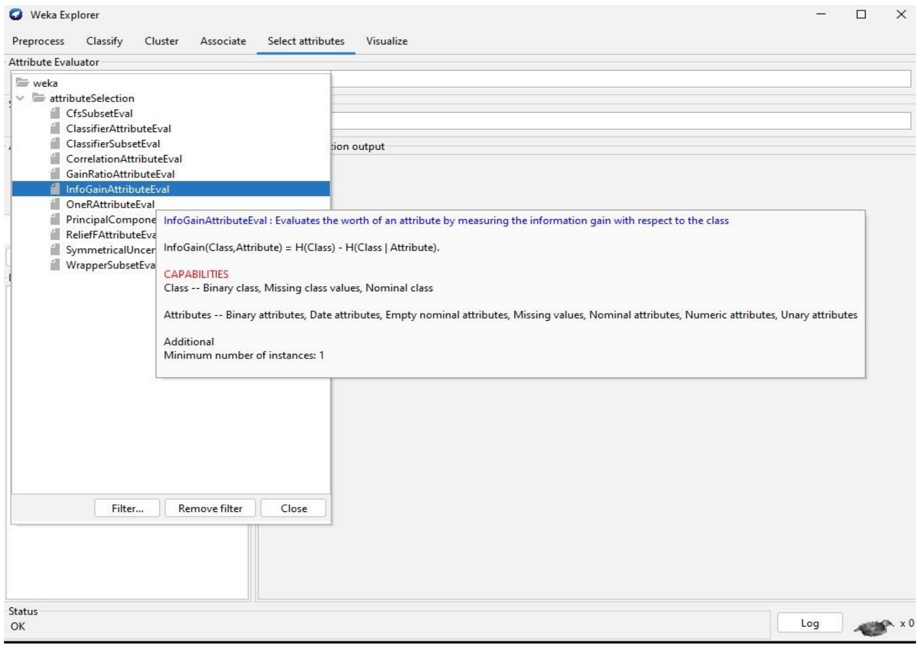
Selected attribute
Name: id Type: Numeric
Missing: 0 (0%) Distinct: 14 Unique: 14 (100%)

Statistic	Value
Minimum	1
Maximum	14
Mean	7.5
StdDev	4.183

Class: Buy_Computer (Nom)

Status: OK

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



InfoGainAttributeEval : Evaluates the worth of an attribute by measuring the information gain with respect to the class

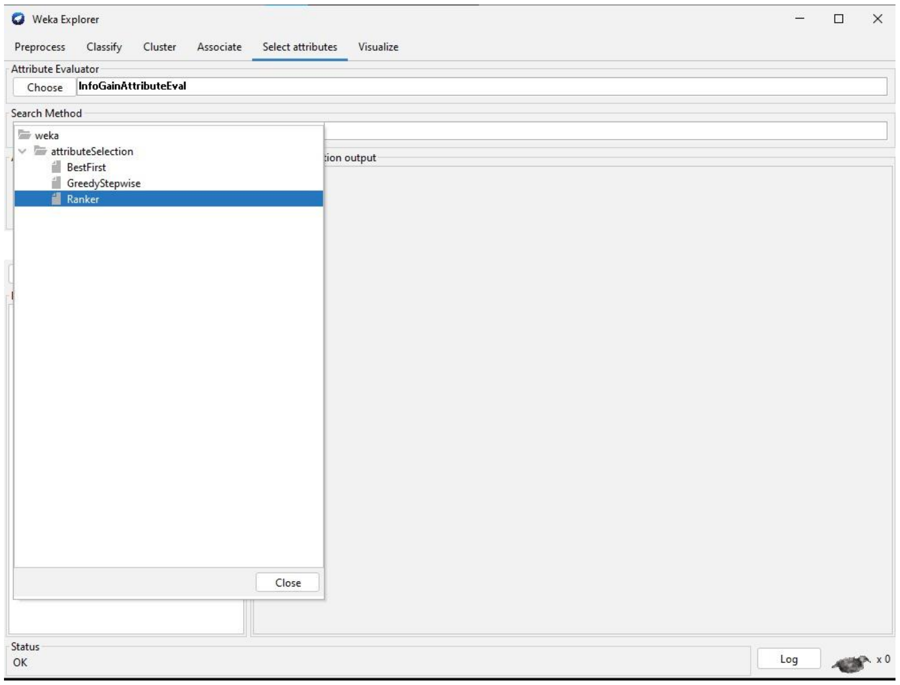
InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute).

CAPABILITIES
Class -- Binary class, Missing class values, Nominal class

Attributes -- Binary attributes, Date attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Additional
Minimum number of instances: 1

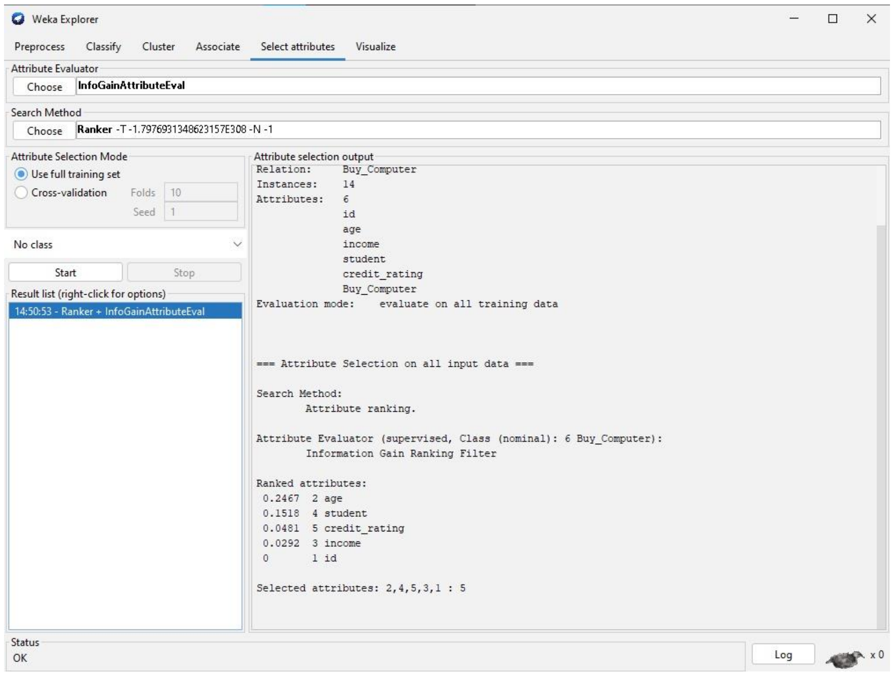
STEP 3 – In ‘search method’ select ‘Ranker’.



Attribute Evaluator: Choose InfoGainAttributeEval

STEP 4 – Click on 'Start' and check for output.

OUTPUT:



Attribute Evaluator: InfoGainAttributeEval

Search Method: Ranker -T -1.7976931348623157E308 -N -1

Attribute Selection Mode: Use full training set; Folds 10; Seed 1

Result list: 14:50:53 - Ranker + InfoGainAttributeEval

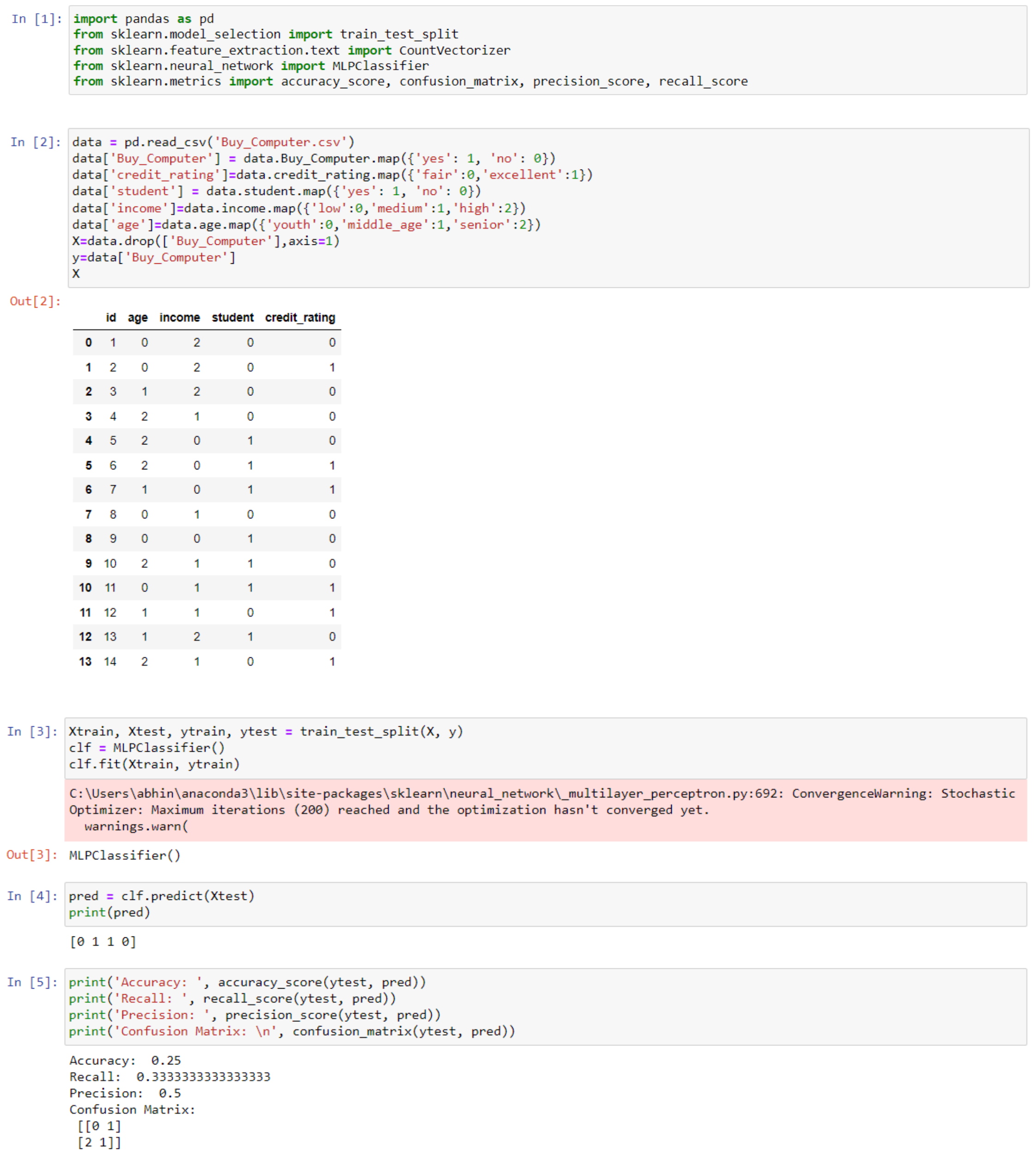
```
Relation:     Buy_Computer
Instances:    14
Attributes:   6
              id
              age
              income
              student
              credit_rating
              Buy_Computer
Evaluation mode:    evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 6 Buy_Computer):
        Information Gain Ranking Filter

Ranked attributes:
 0.2467  2 age
 0.1518  4 student
 0.0481  5 credit_rating
 0.0292  3 income
 0       1 id

Selected attributes: 2,4,5,3,1 : 5
```

Status: OK

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

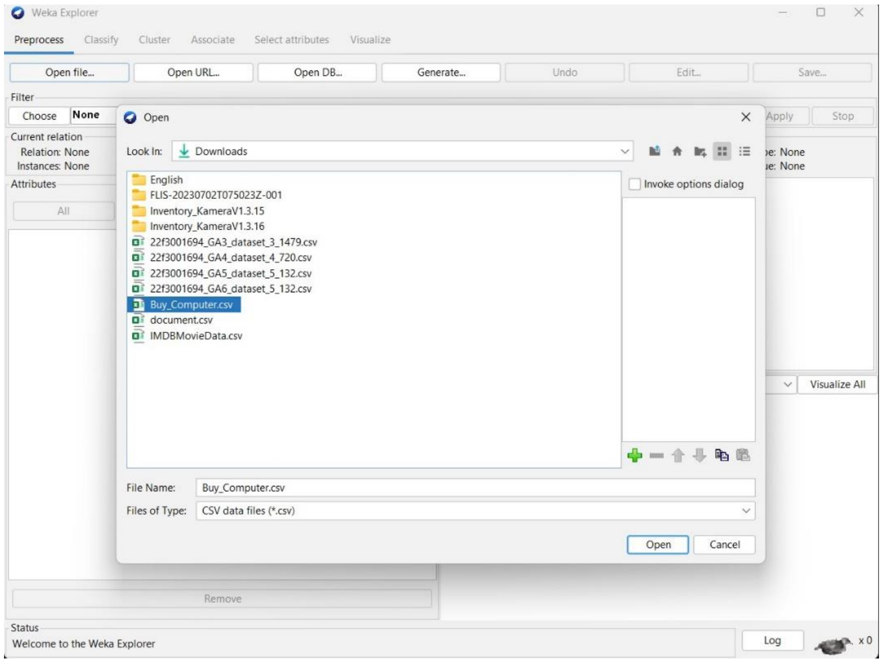
```
In [3]: Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)
        clf = MLPClassifier()
        clf.fit(Xtrain, ytrain)

        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.

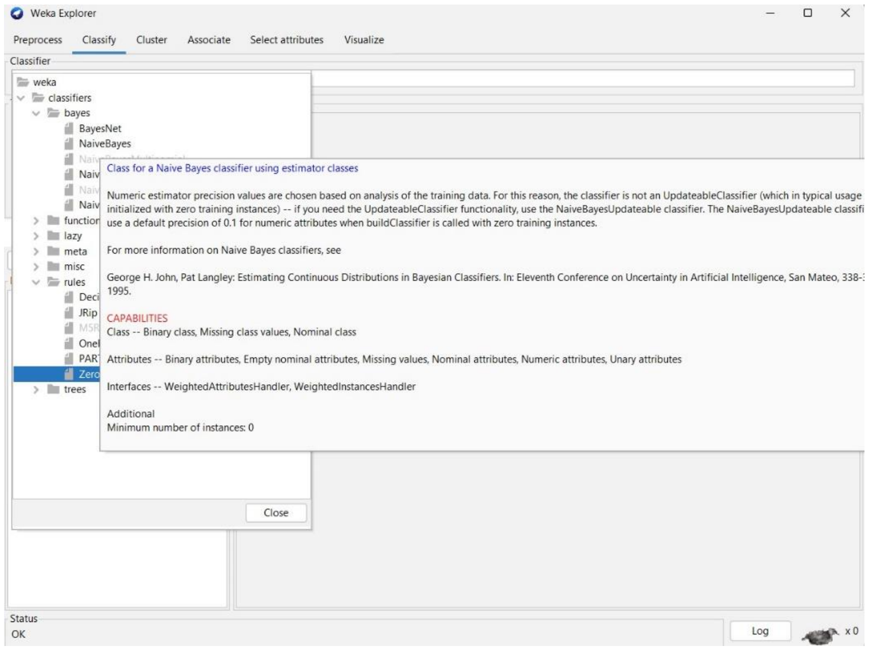


File Name: Buy_Computer.csv

Files of Type: CSV data files (*.csv)

Status
Welcome to the Weka Explorer

STEP 2 – Classify tab -> Classifiers -> bayes -> Naïve Bayes



Class for a Naive Bayes classifier using estimator classes

Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an UpdateableClassifier (which in typical usage initialized with zero training instances) -- if you need the UpdateableClassifier functionality, use the NaiveBayesUpdateable classifier. The NaiveBayesUpdateable classifi use a default precision of 0.1 for numeric attributes when buildClassifier is called with zero training instances.

For more information on Naive Bayes classifiers, see

George H. John, Pat Langley: Estimating Continuous Distributions in Bayesian Classifiers. In: Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, 338- 1995.

CAPABILITIES
Class -- Binary class, Missing class values, Nominal class

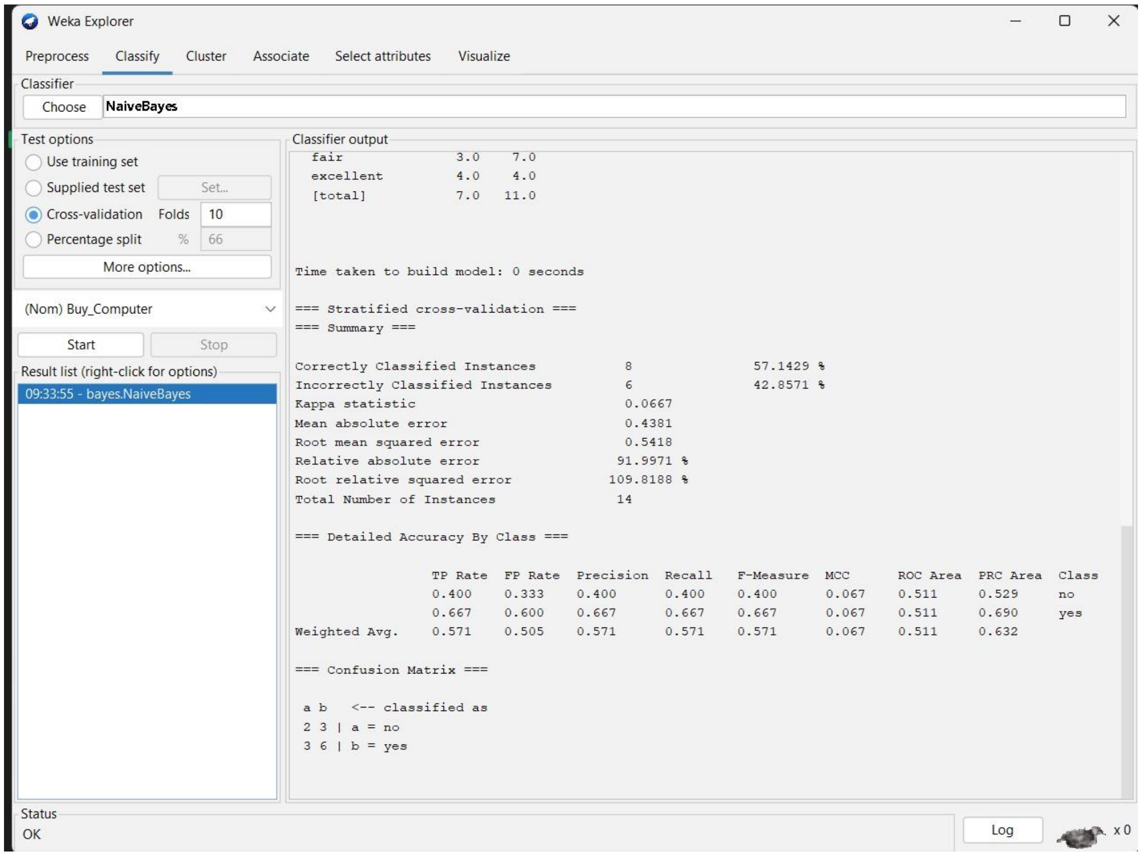
Attributes -- Binary attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Interfaces -- WeightedAttributesHandler, WeightedInstancesHandler

Additional
Minimum number of instances: 0

Status
OK

OUTPUT:



Classifier: NaiveBayes

Test options: Cross-validation [✓] Folds 10; Percentage split % 66

(Nom) Buy_Computer

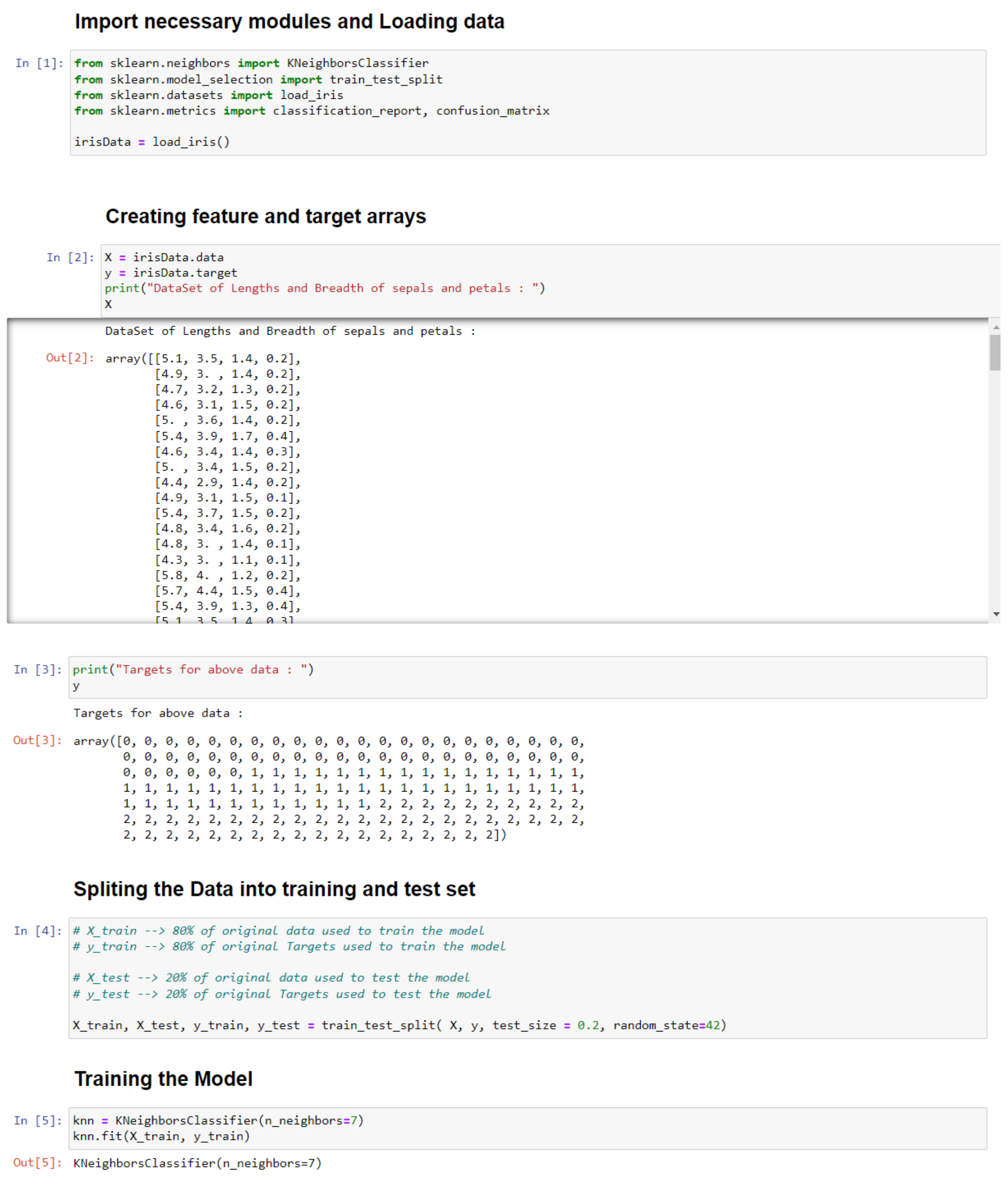
09:33:55 - bayes.NaiveBayes

```
  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
```

K – NEAREST NEIGHBOURS

AIM: Classification of data using K – nearest neighbour approach

PROGRAM:



Import necessary modules and Loading data

```
In [1]: from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.datasets import load_iris
        from sklearn.metrics import classification_report, confusion_matrix

        irisData = load_iris()
```

Creating feature and target arrays

```
In [2]: X = irisData.data
        y = irisData.target
        print("DataSet of Lengths and Breadth of sepals and petals : ")
        X

        DataSet of Lengths and Breadth of sepals and petals :

Out[2]: array([[5.1, 3.5, 1.4, 0.2],
               [4.9, 3. , 1.4, 0.2],
               [4.7, 3.2, 1.3, 0.2],
               [4.6, 3.1, 1.5, 0.2],
               [5. , 3.6, 1.4, 0.2],
               [5.4, 3.9, 1.7, 0.4],
               [4.6, 3.4, 1.4, 0.3],
               [5. , 3.4, 1.5, 0.2],
               [4.4, 2.9, 1.4, 0.2],
               [4.9, 3.1, 1.5, 0.1],
               [5.4, 3.7, 1.5, 0.2],
               [4.8, 3.4, 1.6, 0.2],
               [4.8, 3. , 1.4, 0.1],
               [4.3, 3. , 1.1, 0.1],
               [5.8, 4. , 1.2, 0.2],
               [5.7, 4.4, 1.5, 0.4],
               [5.4, 3.9, 1.3, 0.4],
               [5.1, 3.5, 1.4, 0.3]
```

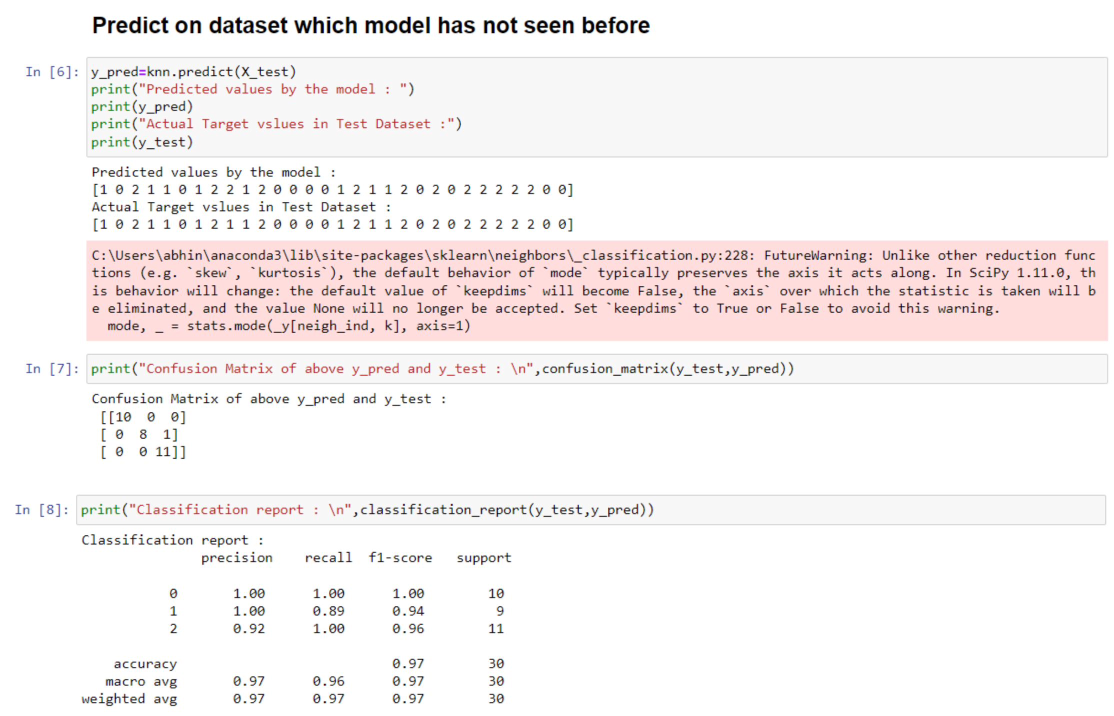
```
In [3]: print("Targets for above data : ")
        y

        Targets for above data :

Out[3]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Spliting the Data into training and test set

```
In [4]: # X_train --> 80% of original data used to train the model
        # y_train --> 80% of original Targets used to train the model

        # X_test --> 20% of original data used to test the model
        # y_test --> 20% of original Targets used to test the model

        X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.2, random_state=42)
```

Training the Model

```
In [5]: knn = KNeighborsClassifier(n_neighbors=7)
        knn.fit(X_train, y_train)

Out[5]: KNeighborsClassifier(n_neighbors=7)
```



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 func
tions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, th
is behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will b
e 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]]
```

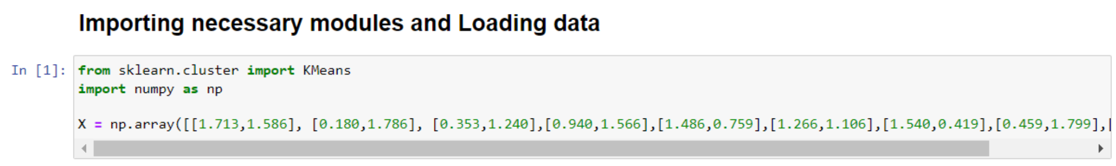
In [8]:
```
print("Classification report : \n",classification_report(y_test,y_pred))
```

```
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        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```

K – MEANS

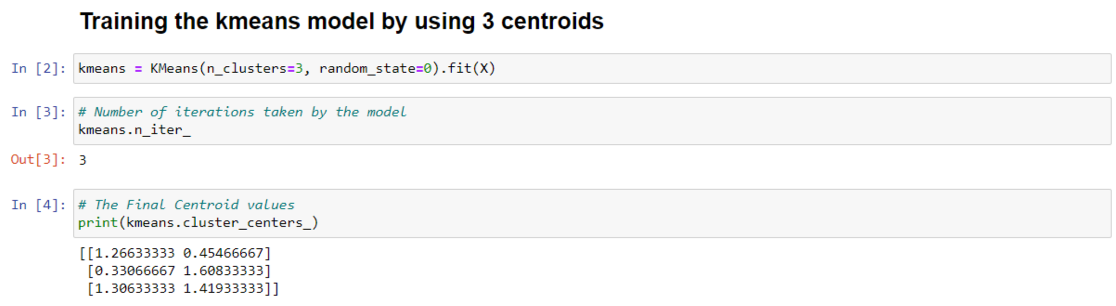
AIM: Implementation of K – means algorithm

PROGRAM:



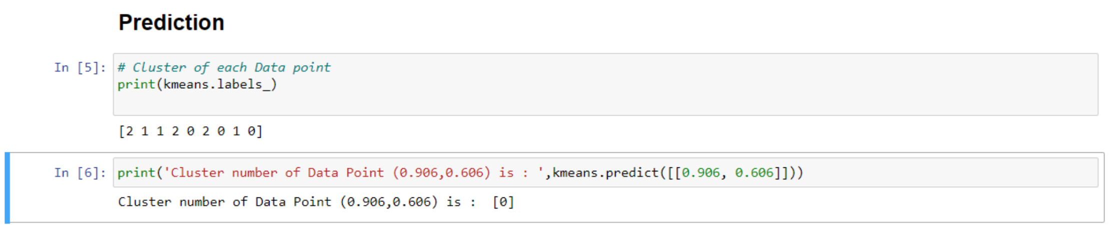
Importing necessary modules and Loading data

```
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]]
```

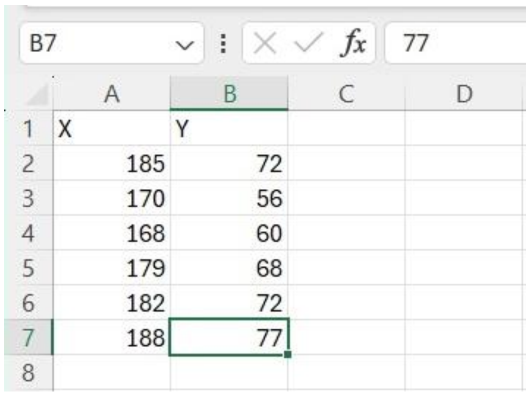


Prediction

```
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.



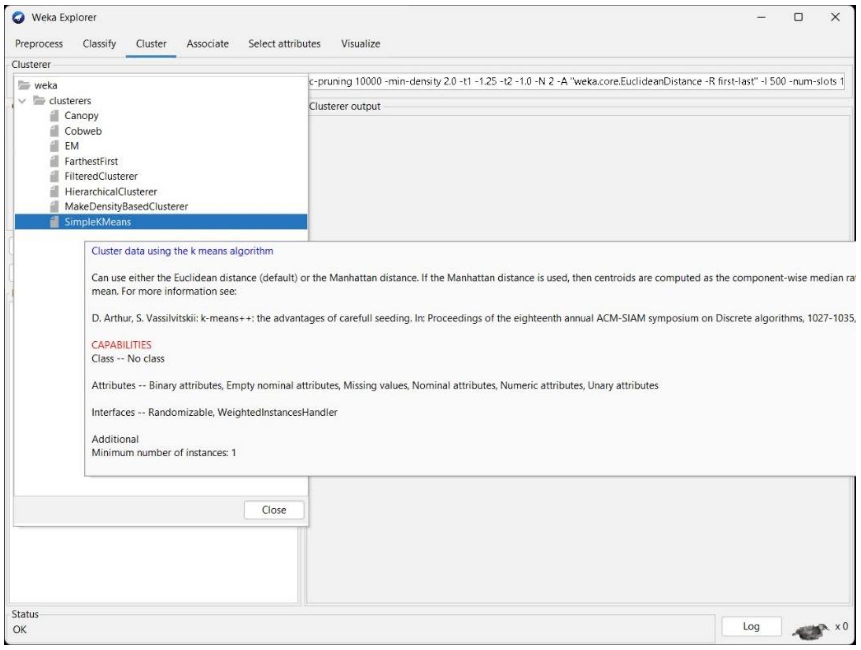
B7 fx 77

	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				

<u>STEP 2</u> – Open WEKA tool and click on WEKA tool and then click on explorer.

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

<u>STEP 4</u> - 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.



c-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

Cluster data using the k means algorithm

Can use either the Euclidean distance (default) or the Manhattan distance. If the Manhattan distance is used, then centroids are computed as the component-wise median ra
mean. For more information see:

D. Arthur, S. Vassilvitskii: k-means++: the advantages of carefull seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 1027-1035,

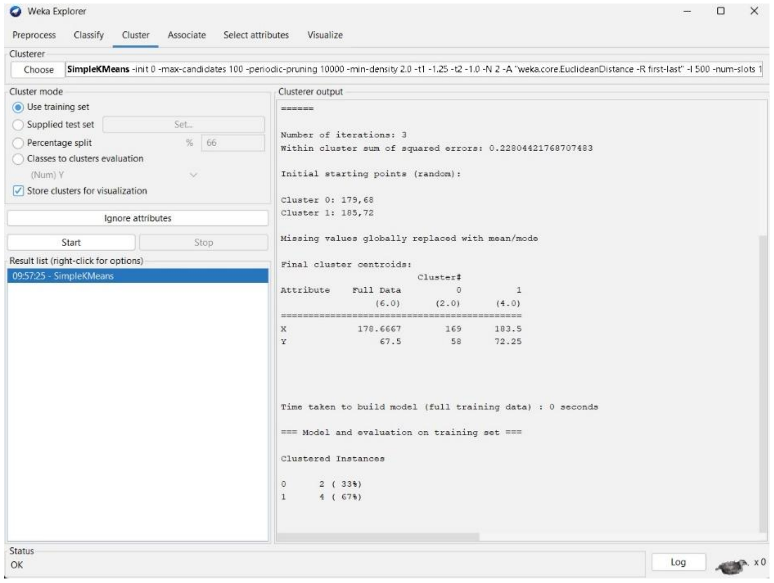
CAPABILITIES
Class -- No class

Attributes -- Binary attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Interfaces -- Randomizable, WeightedInstancesHandler

Additional
Minimum number of instances: 1

<u>STEP 5</u> - Click on Start.The results window showing the clusters is shown below:



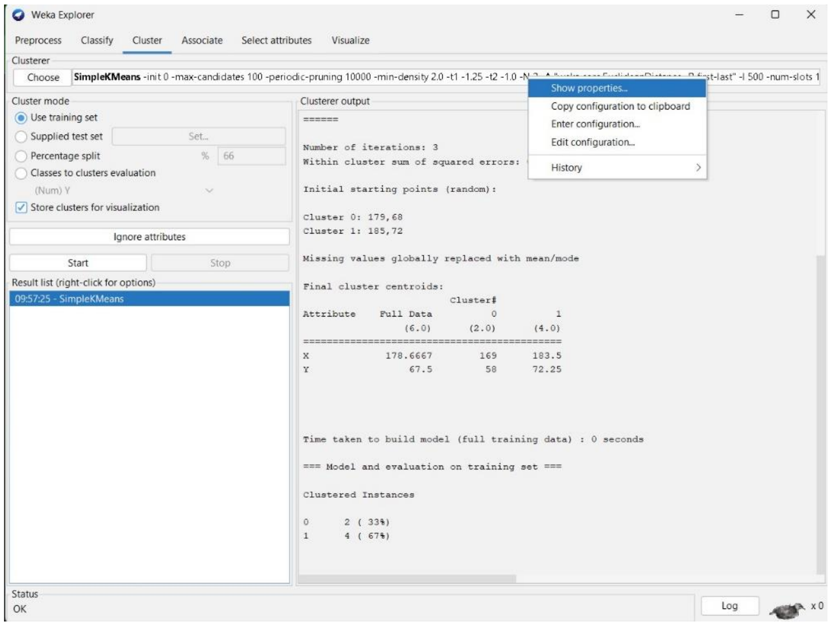
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1

Cluster mode: [✓ Use training set]; Percentage split: 66; (Num) Y; [✓ Store clusters for visualization]

Result list: 09:57:25 - SimpleKMeans

```
======

Number of iterations: 3
Within cluster sum of squared errors: 0.22804421768707483

Initial starting points (random):

Cluster 0: 179,68
Cluster 1: 185,72

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                 (6.0)      (2.0)      (4.0)
============================================
X             178.6667        169      183.5
Y                 67.5         58      72.25




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

=== Model and evaluation on training set ===

Clustered Instances

0       2 ( 33%)
1       4 ( 67%)
```

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



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

Percentage split % 66

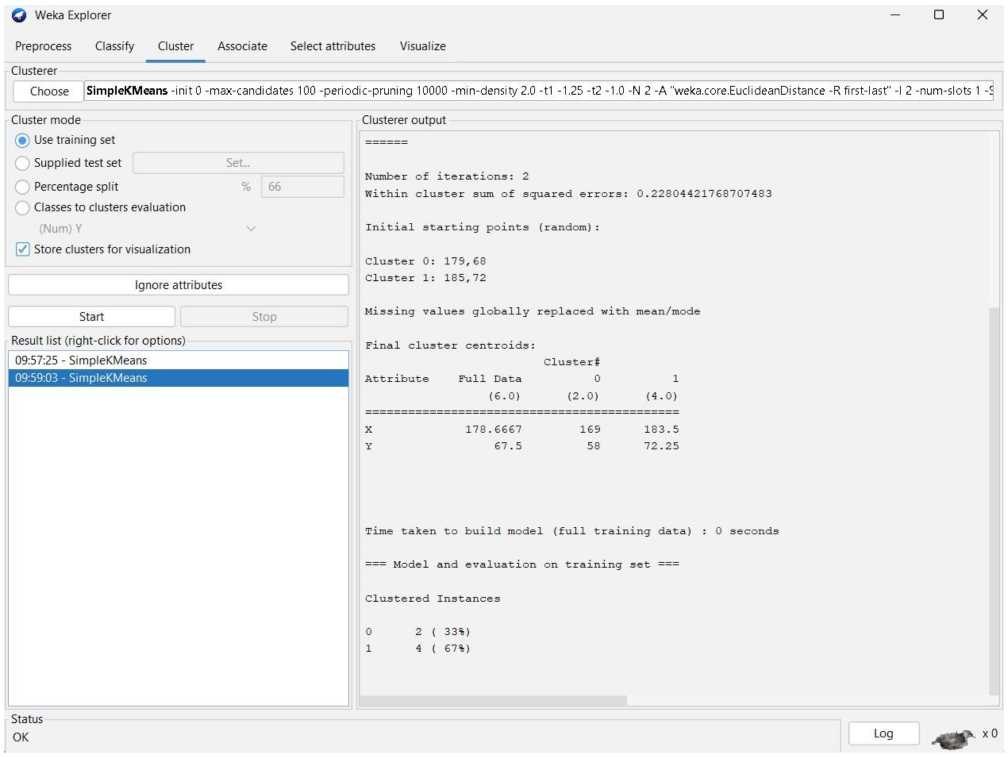
09:57:25 - SimpleKMeans

Clusterer output

```
======

Number of iterations: 3
Within cluster sum of squared errors:

Initial starting points (random):

Cluster 0: 179,68
Cluster 1: 185,72

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                 (6.0)      (2.0)      (4.0)
============================================
X             178.6667        169      183.5
Y                 67.5         58      72.25




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

=== Model and evaluation on training set ===

Clustered Instances

0      2 ( 33%)
1      4 ( 67%)
```

Status: OK

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



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 2 -num-slots 1 -S

Percentage split % 66

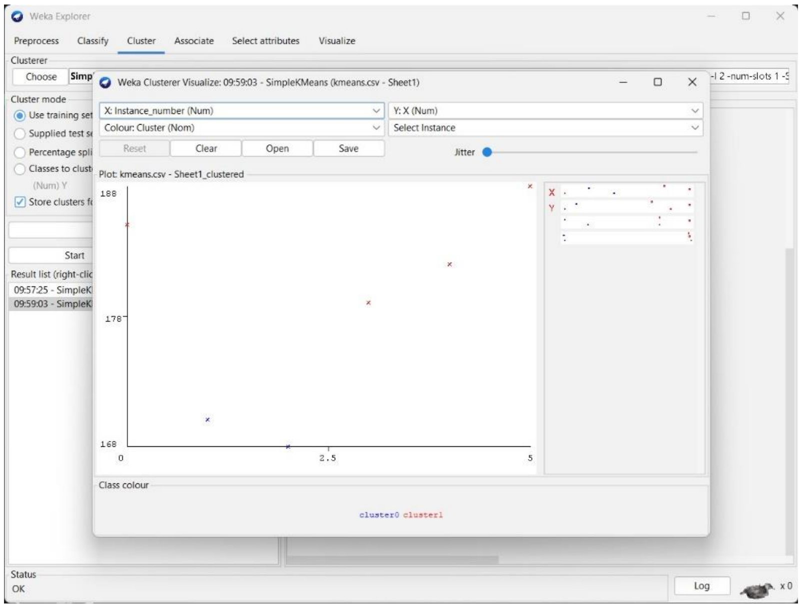
09:57:25 - SimpleKMeans
09:59:03 - SimpleKMeans

Clusterer output

```
======

Number of iterations: 2
Within cluster sum of squared errors: 0.22804421768707483

Initial starting points (random):

Cluster 0: 179,68
Cluster 1: 185,72

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                 (6.0)      (2.0)      (4.0)
============================================
X             178.6667        169      183.5
Y                 67.5         58      72.25




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

=== Model and evaluation on training set ===

Clustered Instances

0      2 ( 33%)
1      4 ( 67%)
```

Status: OK

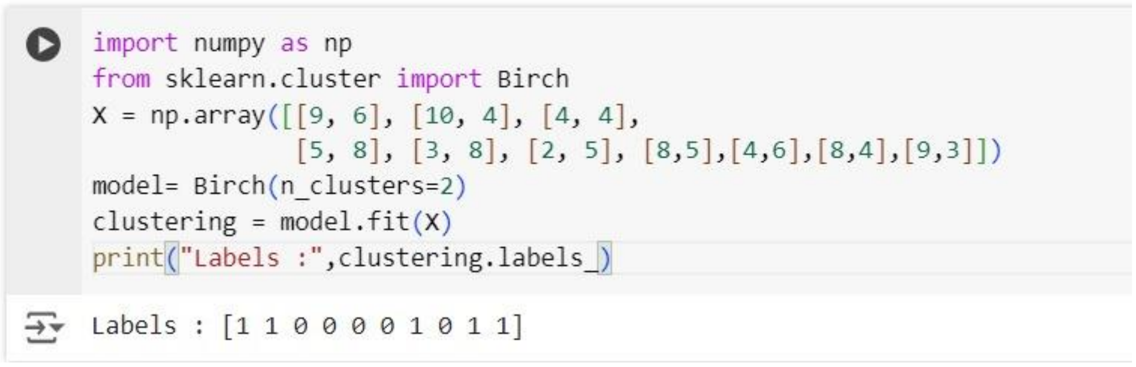
STEP 8 - To see the data tuples in each cluster,right click on the result list and click on visualize cluster assignments.



BIRCH ALGORITHM

AIM: Implementation of BIRCH algorithm.

PROGRAM:



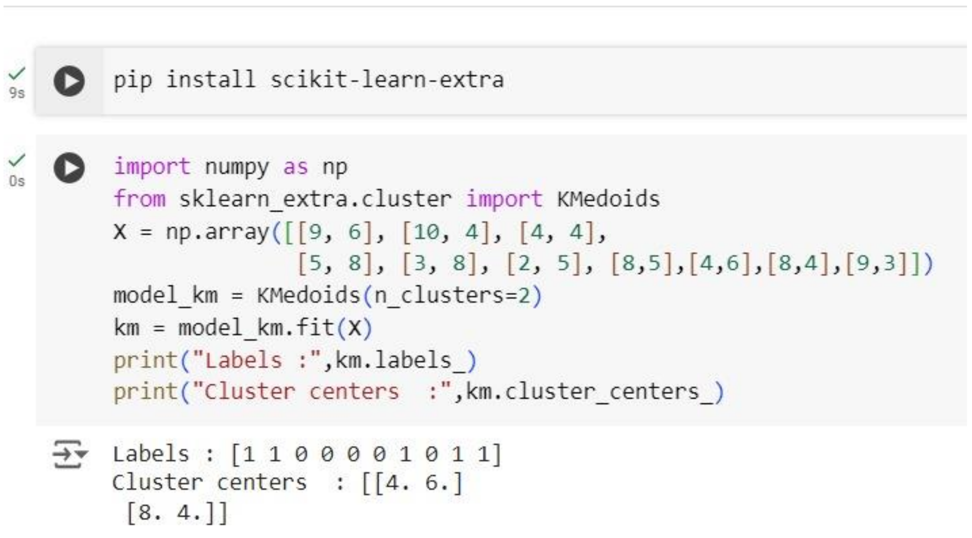
```
import numpy as np
from sklearn.cluster import Birch
X = np.array([[9, 6], [10, 4], [4, 4],
              [5, 8], [3, 8], [2, 5], [8,5],[4,6],[8,4],[9,3]])
model= Birch(n_clusters=2)
clustering = model.fit(X)
print("Labels :",clustering.labels_)
```

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

PAM ALGORITHM

AIM: Implementation of PAM algorithm.

PROGRAM:



```
pip install scikit-learn-extra
```

```
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

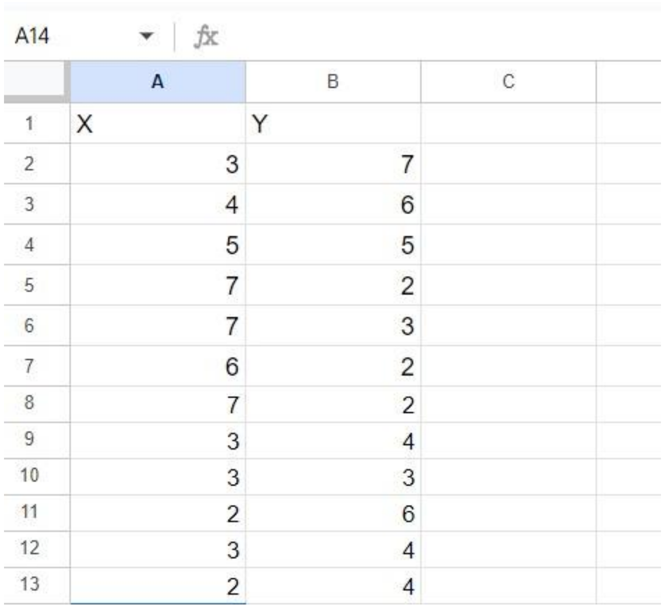
AIM: Implementation of DB Scan algorithm.

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:

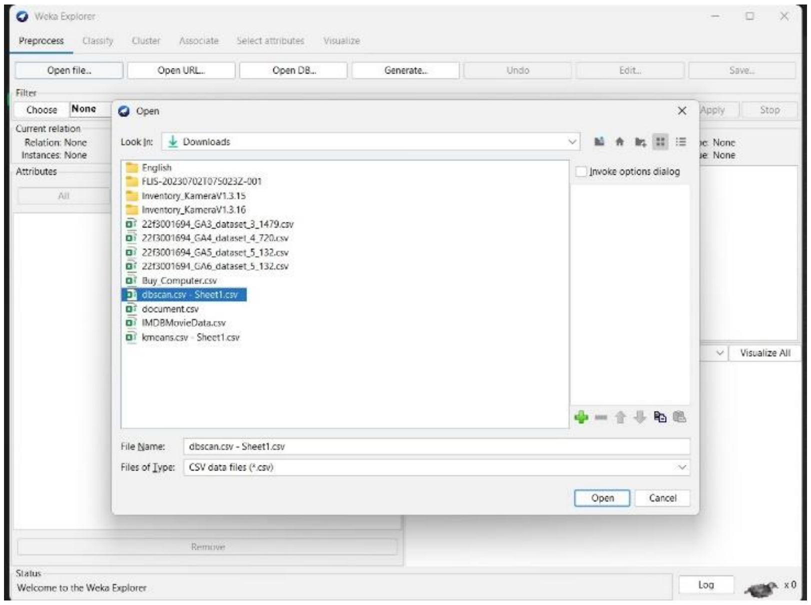
STEP 1 – Create the Data set.



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.

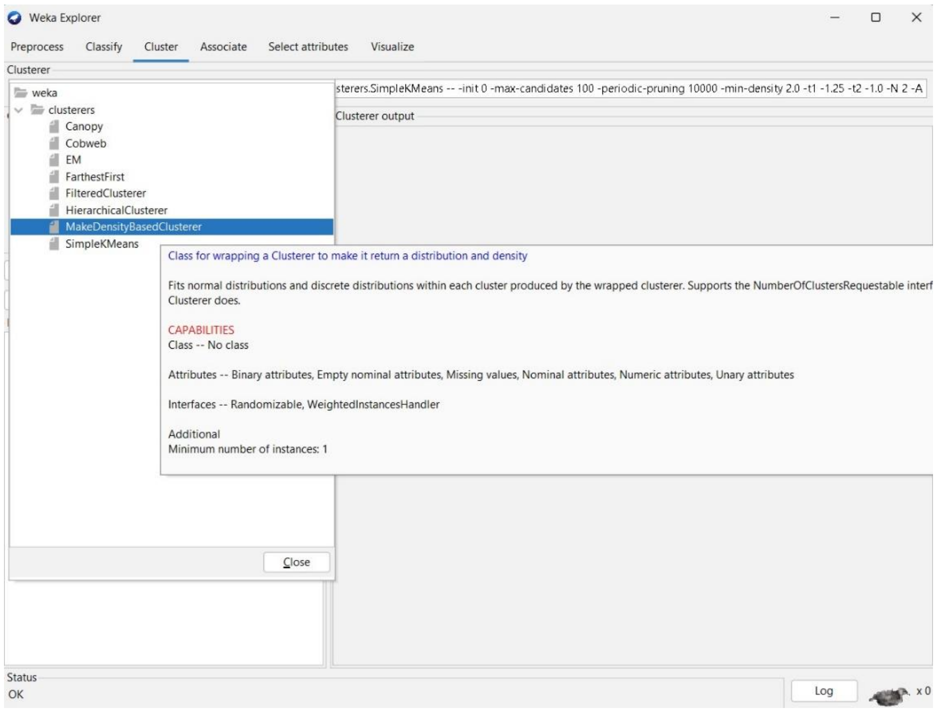


English
FLIS-20230702T075023Z-001
Inventory_KameraV1.3.15
Inventory_KameraV1.3.16
22f3001694_GA3_dataset_3_1479.csv
22f3001694_GA4_dataset_4_720.csv
22f3001694_GA5_dataset_5_132.csv
22f3001694_GA6_dataset_5_132.csv
Buy_Computer.csv
dbscan.csv - Sheet1.csv
document.csv
IMDBMovieData.csv
kmeans.csv - Sheet1.csv

File Name: dbscan.csv - Sheet1.csv
Files of Type: CSV data files (*.csv)

Welcome to the Weka Explorer

STEP 3 – Cluster -> MakeDensityBasedCluster



sterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A

Class for wrapping a Clusterer to make it return a distribution and density

Fits normal distributions and discrete distributions within each cluster produced by the wrapped clusterer. Supports the NumberOfClustersRequestable interf
Clusterer does.

CAPABILITIES
Class -- No class

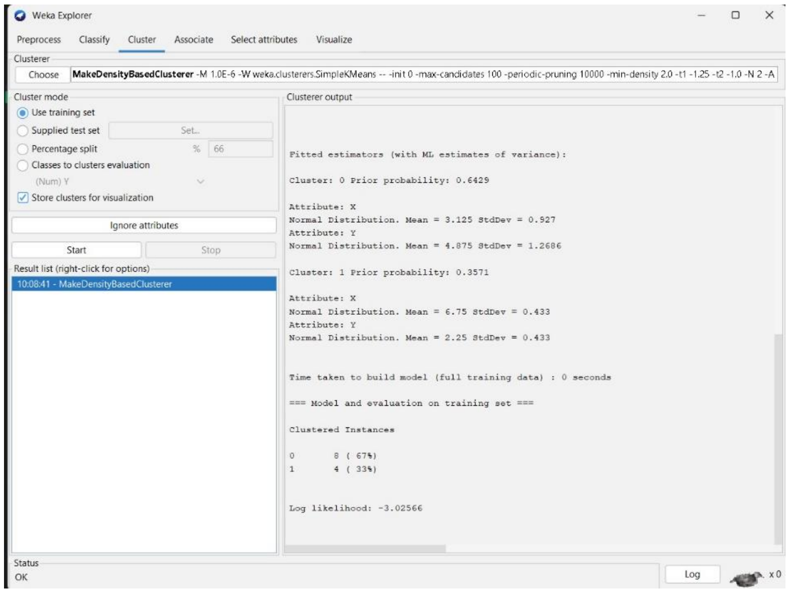
Attributes -- Binary attributes, Empty nominal attributes, Missing values, Nominal attributes, Numeric attributes, Unary attributes

Interfaces -- Randomizable, WeightedInstancesHandler

Additional
Minimum number of instances: 1

OK

<u>STEP 4</u> - Click ‘Start’.



Clusterer: **MakeDensityBasedClusterer** -M 1.0E-6 -W weka.clusterers.SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A

Cluster mode: Use training set [✓]; Percentage split % 66; Store clusters for visualization [✓]

Result list (right-click for options): 10:08:41 - MakeDensityBasedClusterer

Status: OK

Clusterer output:

```
Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probability: 0.6429

Attribute: X
Normal Distribution. Mean = 3.125 StdDev = 0.927
Attribute: Y
Normal Distribution. Mean = 4.875 StdDev = 1.2686

Cluster: 1 Prior probability: 0.3571

Attribute: X
Normal Distribution. Mean = 6.75 StdDev = 0.433
Attribute: Y
Normal Distribution. Mean = 2.25 StdDev = 0.433


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

=== Model and evaluation on training set ===

Clustered Instances

0       8 ( 67%)
1       4 ( 33%)


Log likelihood: -3.02566
```