Week 12

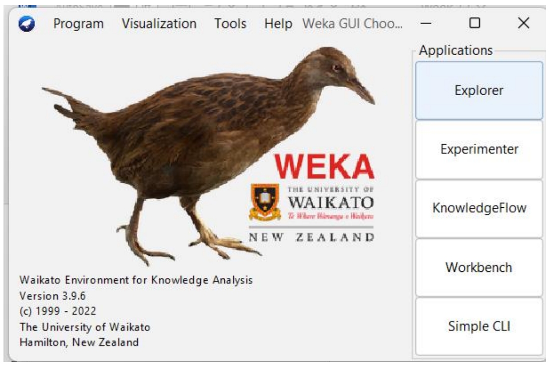
57.Consider Fisher' Iris data set provided, use scikit learn toolof machine learning python library. Use Weka tool to classify the dataset and frame the association rules and write a detailed statistical analysis report on the data.

To classify the Iris dataset and generate association rules using the **Weka** tool, follow the steps below. Weka is a comprehensive tool for machine learning, data mining, and pattern recognition, and it provides a GUI interface to perform tasks like classification and association rule mining.

Steps for Classifying the Iris Dataset and Generating Association Rules in Weka

.1. **Load the Iris Dataset into Weka**

1. **Open Weka**:The main window of Weka window open with several options.



Waikato Environment for Knowledge Analysis
Version 3.9.6
(c) 1999 - 2022
The University of Waikato
Hamilton, New Zealand

2. **Load the Dataset**:
   - Click on the **Explorer** button in the main window.
   - In the **Preprocess** tab, click on the **Open File** button.
   - Navigate to the Iris dataset (iris.arff or iris.csv). Use the built-in dataset in Weka.
   - The Iris dataset should now be loaded and displayed in the Weka GUI. The dataset consists of 150 instances and 5 attributes: 4 numerical features (sepal length, sepal width, petal length, petal width) and the class attribute (species).

3. Classify the Iris Dataset in Weka

You can classify the dataset using different classification algorithms in Weka.

Step 1: Choose a Classifier

1. **Go to the "Classify" tab**.
2. In the **Classifier** section, click the **Choose** button.
   - You will see various classification algorithms under different categories (e.g., **functions**, **trees**, **lazy**, **meta**, etc.).

For this example, we will use **J48 (C4.5 Decision Tree)**, which is commonly used for classification tasks.From the dropdown menu, navigate to **trees** > **J48**.

Step 2: Set the Parameters (Optional)

- You can leave the default settings for the **J48** classifier, but if you want to adjust parameters, click on the **"J48"** option in the Classifier box to set them.

Step 3: Train and Evaluate the Classifier

1. In the **Classify** tab, you can choose a **cross-validation** - Weka performs K-fold cross-validation (default is 10 folds).
2. Once you've chosen your evaluation method, click the **Start** button.Weka will run the classifier, and we will see a summary of the classification results

Step 4: View the Results(Classifier Output)

```
=== Run information ===
Scheme:     weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    iris
Instances:   150
Attributes:  5
             sepallength
             sepalwidth
             petallength
             petalwidth
             class
Test mode:   10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :      5
Size of the tree :       9
Time taken to build model: 0.03 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances        144          96     %
Incorrectly Classified Instances        6           4     %
Kappa statistic                     0.94
Mean absolute error                   0.035
Root mean squared error                 0.1586
Relative absolute error               7.8705 %
Root relative squared error            33.6353 %
Total Number of Instances             150
```

```
=== Detailed Accuracy By Class ===
TP Rate  FP Rate  Precision  Recall   F-Measure  MCC     ROC Area  PRC Area  Class
 0.980    0.000    1.000       0.980    0.990       0.985     0.990       0.987    Iris-setosa
 0.940    0.030    0.940       0.940    0.940       0.910     0.952       0.880    Iris-versicolor
0.960   0.030    0.941     0.960    0.950       0.925      0.961      0.905      Iris-virginica
Weighted Avg.   0.960    0.020    0.960     0.960    0.960     0.940    0.968    0.924
.
=== Confusion Matrix ===
a  b  c   <-- classified as
49   1     0 |  a = Iris-setosa
 0   47   3 |  b = Iris-versicolor
0    2    48 |  c = Iris-virginica
```

4. Generate Association Rules in Weka

We can generate **association rules** using the **Apriori** algorithm in Weka.

Step 1: Go to the "Associate" Tab

1. **Navigate to the "Associate" tab** in the Weka GUI.
2. Under the **Associator** section, click on **Choose** and select the **Apriori** algorithm (it is available under **associators** > **Apriori**)..

Step 2: Run the Apriori Algorithm

1. After setting the desired support and confidence, click the **Start** button.
2. Weka will run the **Apriori** algorithm and generate association rules from the dataset.

Step 4: View the Association Rules(Output)

```
=== Run information ===
Scheme:      weka.associations.Apriori
Relation:    iris-weka.filters.supervised.attribute.Discretize-Rfirst-last-precision6
Instances:   150
Attributes:  5
             sepallength
             sepalwidth
             petallength
             petalwidth
             class
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.3 (45 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 13
Size of set of large itemsets L(2): 11
Size of set of large itemsets L(3): 5
Size of set of large itemsets L(4): 1
```

```
Best rules found:

 1. petalwidth='(-inf-0.8]' 50 ==> petallength='(-inf-2.45]' 50    <conf:(1)> lift:(3) lev:(0.22) [33]
conv:(33.33)
 2. petallength='(-inf-2.45]' 50 ==> petalwidth='(-inf-0.8]' 50    <conf:(1)> lift:(3) lev:(0.22) [33]
conv:(33.33)
 3. class=Iris-setosa 50 ==> petallength='(-inf-2.45]' 50    <conf:(1)> lift:(3) lev:(0.22) [33] conv:(33.33)
 4. petallength='(-inf-2.45]' 50 ==> class=Iris-setosa 50    <conf:(1)> lift:(3) lev:(0.22) [33] conv:(33.33)
 5. class=Iris-setosa 50 ==> petalwidth='(-inf-0.8]' 50    <conf:(1)> lift:(3) lev:(0.22) [33] conv:(33.33)
 6. petalwidth='(-inf-0.8]' 50 ==> class=Iris-setosa 50    <conf:(1)> lift:(3) lev:(0.22) [33] conv:(33.33)
 7. petalwidth='(-inf-0.8]' class=Iris-setosa 50 ==> petallength='(-inf-2.45]' 50    <conf:(1)> lift:(3)
lev:(0.22) [33] conv:(33.33)
 8. petallength='(-inf-2.45]' class=Iris-setosa 50 ==> petalwidth='(-inf-0.8]' 50    <conf:(1)> lift:(3)
lev:(0.22) [33] conv:(33.33)
 9. petallength='(-inf-2.45]' petalwidth='(-inf-0.8]' 50 ==> class=Iris-setosa 50    <conf:(1)> lift:(3)
lev:(0.22) [33] conv:(33.33)
10. class=Iris-setosa 50 ==> petallength='(-inf-2.45]' petalwidth='(-inf-0.8]' 50    <conf:(1)> lift:(3)
lev:(0.22) [33] conv:(33.33)
```

6. Conclusion

Using **Weka**, we can easily classify the Iris dataset with a variety of machine learning algorithms (such as **J48** decision trees), and generate useful **association rules** with the **Apriori** algorithm.By combining classification with association rule mining, we gain both predictive insights and explanatory relationships in the dataset.

58.Consider the Heart dataset provided containing the samples of 1025 with 14 variables. To train the model to classify using Perceptron classifier using weka tool and write a detailed statistical analysis report on the data.

To perform classification using the **Perceptron** model on the **Heart Disease dataset** in **Weka** and obtain the performance metrics similar to the ones you've provided, follow these steps:

Step 1: Load the Heart Disease Dataset

1. **Open Weka** and go to the **Explorer** tab.In the **Preprocess** tab, click on **Open file** and load the Heart Disease dataset. If the dataset is in .csv or .arff format, ensure it's available on your machine and select it..

Step 2: Select the Perceptron Classifier

1. After loading the dataset, go to the **Classify** tab.Click the **Choose** button and navigate to functions -> Perceptron.

Step 3: Set the Evaluation Method To evaluate the model, select an evaluation method: Select the **"Cross-validation"** option and set the number of folds to 10.Once the method is selected, click on **Start** to train the model and evaluate its performance.

Step 4: View the Results(Classifier Output)

```
=== Run information ===
Scheme:       weka.classifiers.functions.MultilayerPerceptron
Relation:     heart
Instances:    1025
Attributes:   14
              age
              sex
              cp
              trestbps
              chol
              fbs
              restecg
              thalach
              exang
              oldpeak
              slope
              ca
              thal
              target
Test mode:    10-fold cross-validation



=== Classifier model (full training set) ===
Linear Node 0
    Inputs    Weights
    Threshold    0.356716723232184
    Node 1    -1.8501510460566466
    Node 2    1.8142247109958667
    Node 3    -0.9203766803872512
    Node 4    -1.579584991293276
    Node 5    0.5576244485957806
    Node 6    2.1321342071686122
    Node 7    -2.365195713356479
Sigmoid Node 1
    Inputs    Weights
    Threshold    -5.740594773278322
    Attrib age    3.1585710151678548
    Attrib sex    -5.959154886299894
    Attrib cp    -5.057127966284369
    Attrib trestbps    1.300565769735731
    Attrib chol    0.19253466591930263
    Attrib fbs    3.2456346057317775
    Attrib restecg    5.019756834425388
    Attrib thalach    -3.4201926294569978
    Attrib exang    9.257761972279788
    Attrib oldpeak    0.39004704515920074
    Attrib slope    -6.384520684143148
    Attrib ca    5.405235202600599
    Attrib thal    -6.9445111377039925
 …………………………………………………
Class
    Input
    Node 0
Time taken to build model: 0.45 seconds
=== Cross-validation ===
=== Summary ===
Correlation coefficient               0.8296
Mean absolute error                    0.1771
Root mean squared error                0.2899
Relative absolute error               35.3973 %
Root relative squared error           57.9381 %
Total Number of Instances             1025
```

59. Consider the Water Portability data set provided, apply data preprocessing steps and examine the applicability of clustering data using WEKA tool and justify the same. Use Weka and write a detailed statistical analysis report on the data.

Step 1: Understanding the Water Portability Dataset

The Water Portability dataset typically involves data related to the portability of water, which might contain information about water quality, whether it is safe for drinking, or various parameters affecting water safety. These datasets typically include attributes like pH level, turbidity, total dissolved solids (TDS), temperature, dissolved oxygen, etc.

The goal is to predict the portability of water or classify water into safe or unsafe categories, possibly by applying clustering techniques.

Step 2: Data Preprocessing in WEKA

Before applying clustering techniques, we must preprocess the data to ensure it is in the right format and ready for analysis. In WEKA, this involves several key steps:

Loading the Dataset: Open WEKA and go to the Explorer interface.Select Open file and load the Water Portability dataset (usually in .arff or .csv format).

Handling Missing Values: Inspect the dataset for any missing values. In WEKA, this can be done in the Preprocess tab.If there are missing values, you can either remove the rows with missing values or input them using the mean or median for numerical attributes or the mode for categorical attributes.

Step 3: Clustering the Data in WEKA

Once the data is preprocessed, we can apply clustering techniques. K-means is one of the most commonly used clustering algorithms. Selecting the Clustering Algorithm:

Go to the Classify tab and click on Choose. Under the Cluster option, select SimpleKMeans (for K-means clustering) or another clustering algorithm of your choice.

Setting Parameters: For SimpleKMeans, you will need to specify:

Number of Clusters (K): You can start with an arbitrary number, say 3 or 5, depending on the dataset. In practice, you can experiment with different values and evaluate the results.

Distance Function: Default is Euclidean distance, which is commonly used. You can choose other distance metrics if needed.

Running the Clustering Algorithm:

Click Start to run the clustering algorithm on the dataset.The output will show the number of clusters, the centroids (for K-means), and other cluster-related statistics.

Evaluation of Clusters: After clustering, examine the output. You'll see which instances belong to which clusters, the centroids (for K-means), and the sum of squared errors for each cluster. We can also visualize the clusters using the Visualize tab to gain insights into how well-separated the clusters are.

Clusterer Output

```
=== Run information ===
Scheme:      weka.clusterers.SimpleKMeans - "weka.core.EuclideanDistance -R first-last"
Relation:    WaterPotability
weka.filters.unsupervised.attribute.ReplaceMissingValues-
weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last
Instances:   2260
Attributes:  11
             region_area_
             ph
             Hardness
             Solids
             Chloramines
             Sulfate
             Conductivity
             Organic_carbon
             Trihalomethanes
             Turbidity
Ignored:
             Potability
Test mode:   Classes to clusters evaluation on training data

=== Clustering model (full training set) ===

kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 21588.0
Initial starting points (random):
Cluster 0:
region_area_1283,7.217393,230.968111,13579.85939,5.688316,318.708517,432.068538,12.475339
,63.561749,3.74662
Cluster 1:
region_area_2185,6.493764,255.924455,19821.64318,6.040153,299.12509,492.700709,12.269736,
89.592718,4.910911

Missing values globally replaced with mean/mode
```

```
Final cluster centroids:
                             Cluster#
Attribute                   Full Data              0                1
                            (2260.0)           (2259.0)           (1.0)
===================================================================
region_area_        region_area_1        region_area_1           region_area_2185
ph                  7.083561             7.083561                6.493764
Hardness            124.266124           124.266124              255.924455
Solids              321.247422           321.247422              19821.64318
Chloramines          0.047189             0.047189                6.040153
Sulfate             334.601013           334.601013              299.12509
Conductivity        211.724737           211.724737              492.700709
Organic_carbon        4.476899            4.476899               12.269736
Trihalomethanes     66.440129            66.440129                89.592718
Turbidity           3.266029             3.266029                 4.910911

Time taken to build model (full training data) : 0.01 seconds
=== Model and evaluation on training set ===
Clustered Instances

0      2259 (100%)
1         1 (  0%)

Class attribute: Potability
Classes to Clusters:

   0    1  <-- assigned to cluster
 1381    1 | 0
  878    0 | 1

Cluster 0 <-- 0
Cluster 1 <-- No class
Incorrectly clustered instances : 879.0     38.8938 %
```

Conclusion

The clustering analysis of the Water Portability dataset provides valuable insights into potential groupings of water quality. This unsupervised approach is effective in uncovering patterns that may not be immediately obvious and can serve as a useful first step in understanding water quality without the need for predefined labels

60. Use WEKA and experiment with the Association Rule Mining (A-priori),Agglomerative and Divisive Clustering

To experiment with different classifiers in WEKA, such as Association Rule Mining (Apriori), Agglomerative Clustering, and Divisive Clustering are as follows:

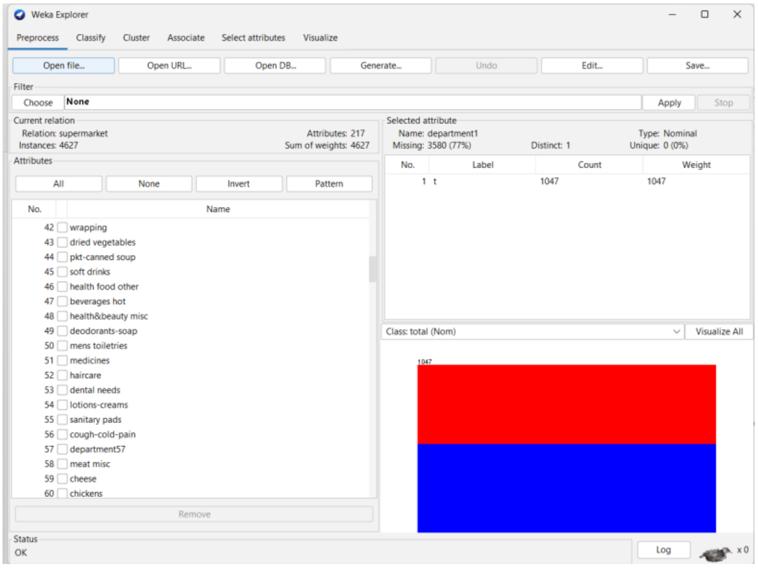
1. Association Rule Mining (Apriori) in WEKA

The Apriori algorithm is used for association rule mining, which is typically used for discovering interesting relationships or patterns among a set of items in large datasets (like market basket analysis). In WEKA, this can be done using the Apriori algorithm.

Steps to Apply Apriori in WEKA:

1. Open WEKA:
   - Launch WEKA and go to the Explorer tab.

2. Load the Dataset:
   - Click Open File to load a SuperMarket dataset.(c-programfiles-weka-data-supermarket) For association rule mining, we typically need a dataset with transactions or itemsets.



Filter: None

Current relation
Relation: supermarket   Attributes: 217
Instances: 4627   Sum of weights: 4627

Selected attribute
Name: department1   Type: Nominal
Missing: 3580 (77%)   Distinct: 1   Unique: 0 (0%)

No.	Label	Count	Weight
1	t	1047	1047

Attributes

No.	Name
42	wrapping
43	dried vegetables
44	pkt-canned soup
45	soft drinks
46	health food other
47	beverages hot
48	health&beauty misc
49	deodorants-soap
50	mens toiletries
51	medicines
52	haircare
53	dental needs
54	lotions-creams
55	sanitary pads
56	cough-cold-pain
57	department57
58	meat misc
59	cheese
60	chickens

Class: total (Nom)

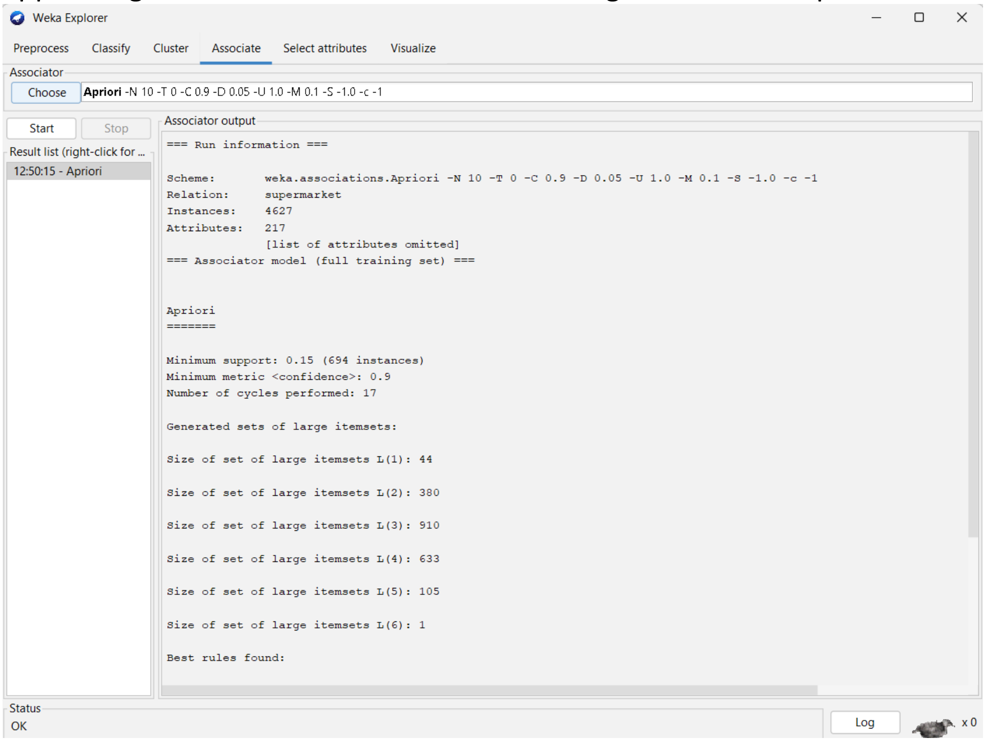
Status
OK

3. Select the Apriori Algorithm:
   - Go to the Classify tab in the WEKA Explorer.
   - Click the Choose button under the Classifier box.
   - Navigate to Association → Apriori.
4. Configure the Apriori Algorithm:

- Click on the name of the Apriori algorithm to open its parameters. You can modify parameters such as:
    - Support: The minimum support threshold for a rule to be considered.
    - Confidence: The minimum confidence level for a rule.
    - Num Rules: Number of rules to generate.
    - Lower Bound Min Support: A filter to control the minimum number of instances.

You can experiment with these parameters to explore the patterns in your dataset.

5. Run Apriori:
    - Click Start to run the Apriori algorithm on your dataset.
    - The results will show a list of association rules, indicating how often items appear together in transactions and how strong the relationships are.



Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

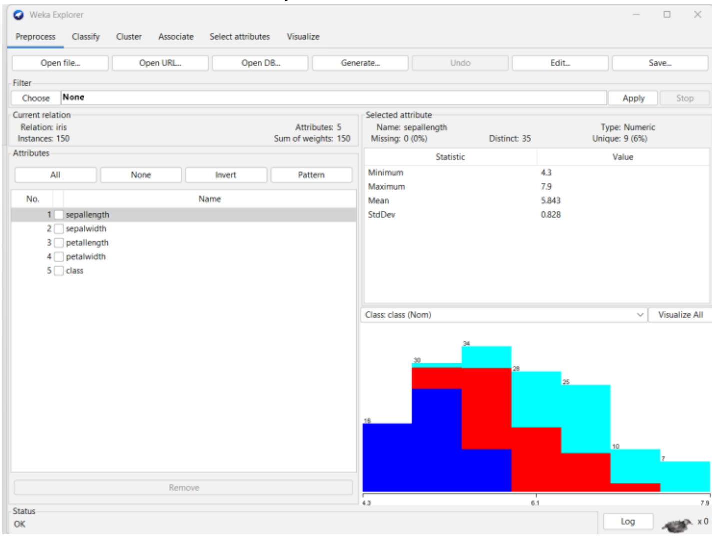
```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:
```

Agglomerative Clustering in WEKA

Agglomerative Clustering is a type of hierarchical clustering where clusters are merged iteratively.

Steps to Apply Agglomerative Clustering:

1. Open WEKA:
   - Launch WEKA and go to the Explorer tab.
2. Load the Dataset:
   - Click Open File and load iris dataset



Current relation
Relation: iris Attributes: 5
Instances: 150 Sum of weights: 150

Attributes

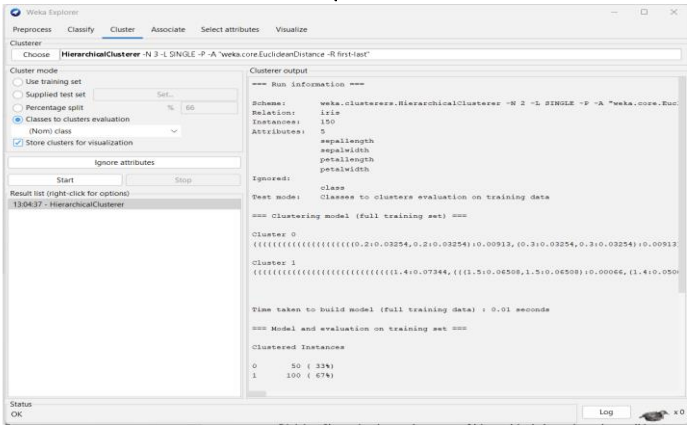
No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute
Name: sepallength Type: Numeric
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Status
OK

3. Select Agglomerative Clustering:
   - Go to the Cluster tab in WEKA.
   - Click the Choose button under Clusterer.
   - Navigate to Cluster → HierarchicalClusterer.
4. Configure Agglomerative Clustering:
   - Once HierarchicalClusterer is selected, you can click on its name to open its parameters.
   - Distance Function: You can choose between various distance measures like EuclideanDistance, ManhattanDistance, etc.
5. Run the Agglomerative Clustering:
   - Click Start to run the algorithm.
   - You will see a dendrogram (tree-like structure) showing the hierarchical relationship between clusters.



Clusterer
HierarchicalClusterer -N 3 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

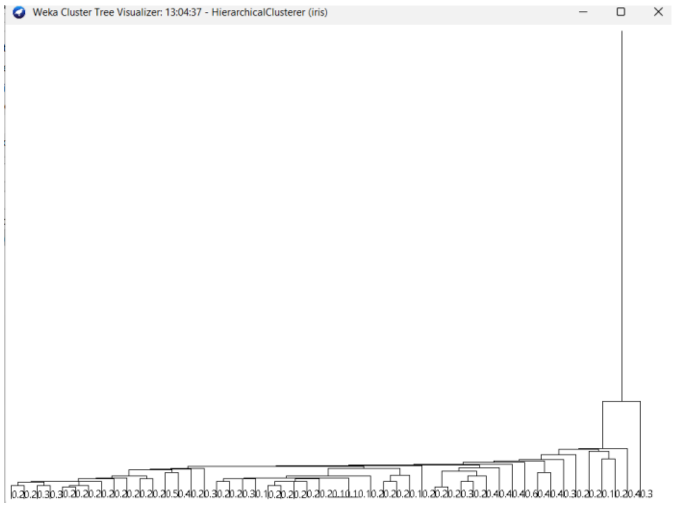
Cluster mode
Classes to clusters evaluation
(Nom) class

Result list (right-click for options)
13:04:37 - HierarchicalClusterer

Clusterer output
```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.Euc
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
Ignored:
              class
Test mode:    Classes to clusters evaluation on training data

=== Clustering model (full training set) ===

Cluster 0
((((((((((((((((((0.2:0.03254,0.2:0.03254):0.00913,(0.3:0.03254,0.3:0.03254):0.00913

Cluster 1
((((((((((((((((((((((((((1.4:0.07344,(((1.5:0.06508,1.5:0.06508):0.00066,(1.4:0.050


Time taken to build model (full training data) : 0.01 seconds

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

Clustered Instances

0       50 ( 33%)
1      100 ( 67%)
```

Status
OK

You’ll also see a dendrogram visualizing how the clusters are formed.


Weka Cluster Tree Visualizer: 13:04:37 - HierarchicalClusterer (iris)