NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPETA

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the bonafide record done by Mr/Ms _____bearing H.T.No _____ of ____ B.TECH __Semester in the _____laboratory is satisfactory completed

Staff In-charge **Head of the Department**

Submitted for the practical examination held on_____

Internal Examiner **External Examiner**

INDEX

S.No	Name of the Experiment	Page No.	Date of Conduction	Date of Submission	Marks Awarded	Signature of the Faculty
1.	To explore and understand the functionalities of the WEKA Data Mining/Machine Learning toolkit by installing and navigating its various interfaces (Explorer, Knowledge Flow, Experimenter, and Command-line). The objective is to analyze datasets (such as Weather and Iris) in ARFF format by examining their attributes, records, and class distribution, and by applying data visualization techniques (histograms, multi-dimensional plots) to gain insights into data preprocessing, classification, clustering, and association rule mining using WEKA.					
2.	Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets					

3.	Demonstrate performing classification on data sets Weka/R • Load each dataset and run 1d3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic. • Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix. • Load each dataset into Weka/R and perform Naïve-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained. • Plot RoC Curves • Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.					
4.	Demonstrate performing clustering of datasets.To perform **clustering** on a given dataset using					

	unsupervised learning techniques (such as **K-Means, Hierarchical**) in order to group the data into meaningful clusters based on similarities and patterns without prior class labels.					
5.	Demonstrate knowledge flow application on data sets into Weka/R • Develop a knowledge flow layout for finding strong association rules by using Apriori, FP Growth algorithms • Set up the knowledge flow to load an ARFF (batch mode) and perform a cross validation using J48 algorithm • Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree					
6.	Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observation					
7	Write a Python program to generate frequent item sets / association rules using Apriori algorithm					
8	Write a program to calculate chi-square value					

	using Python/R. Report your observation.					
9	Write a program to compute/display dissimilarity matrix (for your own dataset containing at least four instances with two attributes) using Python					

Week-1

1. Explore machine learning tool "WEKA"

1. **Explore WEKA Data Mining/Machine Learning Toolkit.**
2. **Downloading and/or installation of WEKA data mining toolkit.**
3. **Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface, Experimenter, command-line interface.**
4. **Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, Classify panel, Cluster panel, Associate panel and Visualize panel)**
5. **Study the arff file format Explore the available data sets in WEKA. Load a data set (ex. Weather dataset, Iris dataset, etc.)**
6. **Load each dataset and observe the following:**
    1. **List the attribute names and they types**
    2. **Number of records in each dataset**
    3. **Identify the class attribute (if any)**
    4. **Plot Histogram**
    5. **Determine the number of records for each class.**
    6. **Visualize the data in various dimensions**

Aim:

To explore and understand the functionalities of the WEKA Data Mining/Machine Learning toolkit by installing and navigating its various interfaces (Explorer, Knowledge Flow, Experimenter, and Command-line). The objective is to analyze datasets (such as Weather and Iris) in ARFF format by examining their attributes, records, and class distribution, and by applying data visualization techniques (histograms, multi-dimensional plots) to gain insights into data preprocessing, classification, clustering, and association rule mining using WEKA.

Features of WEKA Toolkit

1. Explorer Interface

- The **Explorer** is the primary graphical user interface of WEKA.
- It allows users to load datasets, preprocess data, select algorithms, build models, and evaluate results interactively.
- Provides tabs such as Preprocess, Classify, Cluster, Associate, Select attributes, and Visualize to streamline the machine learning workflow.

2. Knowledge Flow Interface

- The **Knowledge Flow** interface uses a drag-and-drop approach to design machine learning workflows.
- It visually represents the data processing pipeline, allowing users to connect components such as data sources, filters, classifiers, and visualizers.
- It is useful for automating repetitive tasks and experimenting with different data processing sequences.

3. Experimenter

- The **Experimenter** is designed for running and managing large-scale experiments.
- Supports systematic testing of multiple algorithms on multiple datasets.
- Provides tools for statistical analysis to compare algorithm performance rigorously.
- Useful for benchmarking and detailed experimentation.

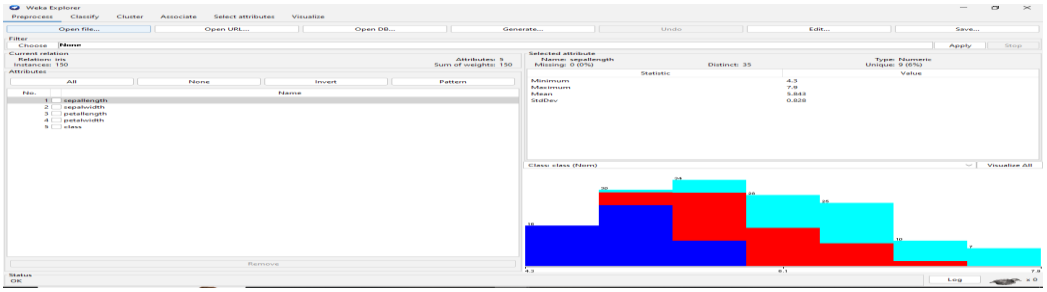
4. Command-Line Interface (CLI)

- WEKA also offers a **command-line interface** for users who prefer scripting and automation.
- Enables batch processing of datasets and integration into other systems.
- Supports running classifiers, filters, and other WEKA functionalities via command line arguments.

Navigating WEKA Explorer Panels

Preprocess Panel

- This panel allows users to **load datasets** and perform **data cleaning** or **filtering**.
- Users can apply filters to remove unwanted attributes, discretize numeric data, or select specific instances.
- It displays basic statistics about the dataset such as the number of instances, attributes, and data types.

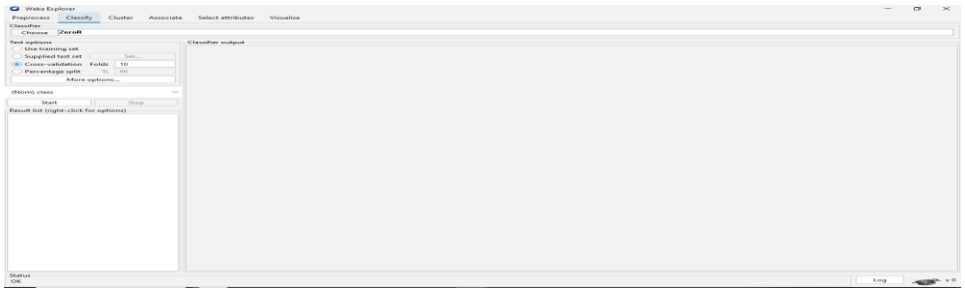


Select Attributes Panel

- Helps in **attribute selection** to improve model performance by identifying the most relevant features.
- Various attribute evaluators and search methods are available to choose the best subset of attributes.
- This reduces dimensionality and improves efficiency.

Classify Panel

- This panel is used to **build classification and regression models**.
- Users can select different algorithms, configure their parameters, and run cross-validation or training/testing.
- Results include accuracy, confusion matrix, and detailed performance metrics.



Cluster Panel

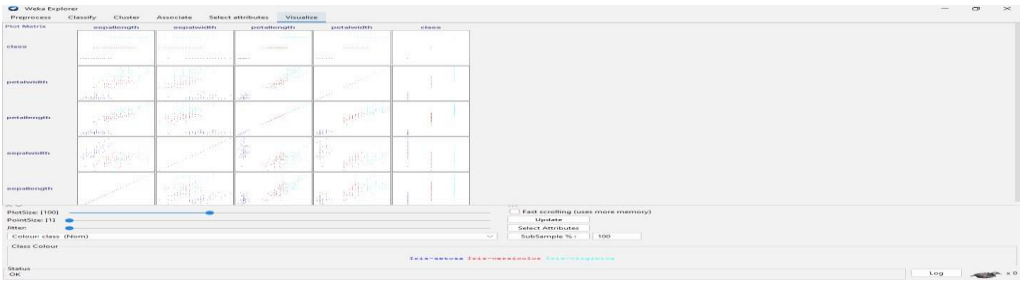
- Allows users to perform **clustering** tasks on datasets.
- Offers algorithms like k-means and EM clustering.
- Users can evaluate the clustering results and visualize clusters.

Associate Panel

- Used for **association rule mining**.
- Helps discover interesting relationships between attributes in the dataset.
- Users can set parameters to control rule generation such as support and confidence thresholds.

Visualize Panel

- Provides visualization tools for both raw data and model results.
- Users can plot scatter plots, histograms, and other charts to better understand data distribution and model behavior.
- Helpful in identifying patterns and outliers.



ARFF File Format

- **ARFF (Attribute-Relation File Format)** is WEKA's native data file format.
- It is a plain text format consisting of two sections:
  - **Header**: Defines the dataset's name (@relation) and lists attributes with their data types (@attribute).

```
@RELATION iris

@ATTRIBUTE sepallength  REAL
@ATTRIBUTE sepalwidth   REAL
@ATTRIBUTE petallength  REAL
@ATTRIBUTE petalwidth   REAL
@ATTRIBUTE class    {Iris-setosa,Iris-versicolor,Iris-virginica}
```

  - **Data**: Contains the actual data instances starting after the @data tag.

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
```

Exploring and Loading Datasets in WEKA

- WEKA comes bundled with several sample datasets such as **Iris**, **Weather**, **Diabetes**, etc.
- To explore datasets:
    - Open WEKA Explorer.
    - Click **Open file** in the Preprocess panel.
    - Navigate to the data folder within the WEKA installation directory.
    - Select a dataset like iris.arff or weather.arff.
- Loading a dataset allows you to view the attributes and instances and apply preprocessing, classification, clustering, and more.

Example: Loading the Iris Dataset

- Open WEKA Explorer.
- In the Preprocess panel, click **Open file**.
- Select the iris.arff file.
- The dataset loads and displays attributes such as sepal length, sepal width, petal length, petal width, and class.
- You can now proceed to classify, cluster, or visualize this dataset.

Loading and Exploring Datasets in WEKA

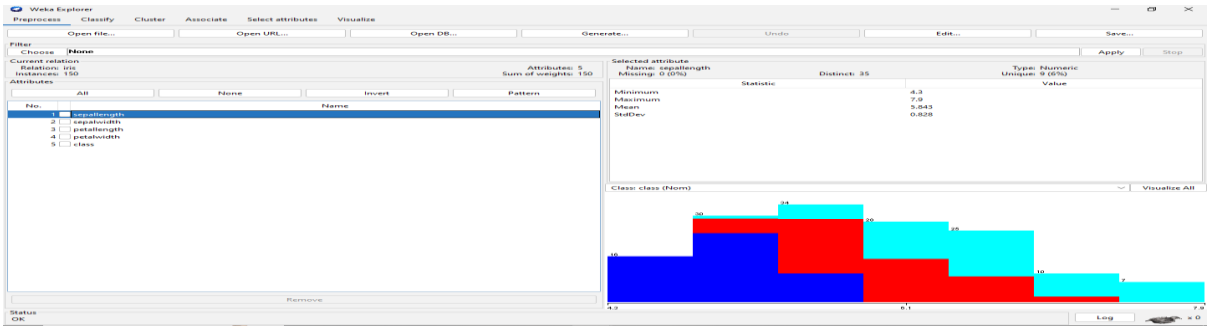
Datasets Used

- Iris dataset

Steps and Observations

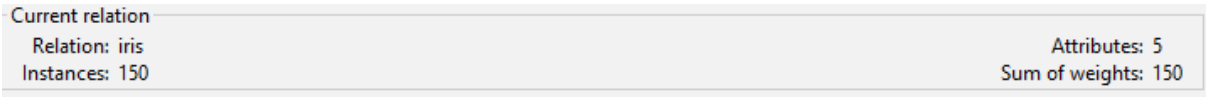
1. List Attribute Names and Their Types

7. Open the dataset in WEKA Explorer under the **Preprocess** panel.
8. The attribute list and types are displayed on the left side.



2. Number of Records in Each Dataset

- The number of instances (records) is shown at the top of the Preprocess panel after loading the dataset.



3. Identify the Class Attribute

- The class attribute is usually the target variable for classification.
- In WEKA, the class attribute is selected by default or can be chosen manually.

4. Plot Histogram

- In the Preprocess panel, select an attribute.
- Click on the **Visualize** button to view a histogram showing the distribution of values for that attribute.

5. Determine Number of Records for Each Class

- Select the class attribute in the Preprocess panel.
- The distribution of instances across classes is shown in the right pane, displaying counts for each class.

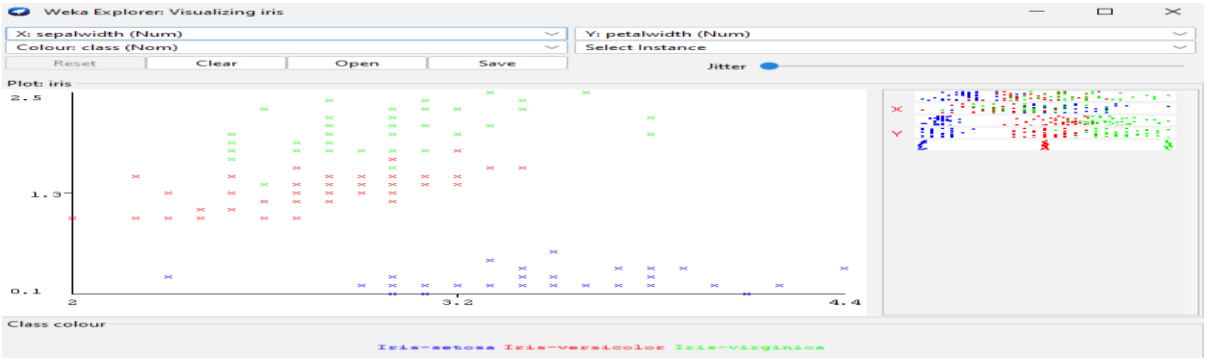
Selected attribute

Name: class
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	Iris-setosa	50	50
2	Iris-versicolor	50	50
3	Iris-virginica	50	50

6. Visualize Data in Various Dimensions

9. Go to the **Visualize** panel in WEKA Explorer.
10. This panel plots scatter plots between different pairs of attributes.
11. Use this to observe relationships, clusters, and class separability in 2D.



Viva Questions:

1. **What is WEKA, and why is it widely used in machine learning and data mining?**

2. **What is the difference between WEKA's Explorer, Knowledge Flow, Experimenter, and Command-line interfaces?**

3. **What is an ARFF file in WEKA, and how does it differ from a CSV file?**

4. **What are the panels available in the WEKA Explorer interface and their purposes?**

5. **If you load the Iris dataset in WEKA, what is the class attribute and how many records belong to each class?**

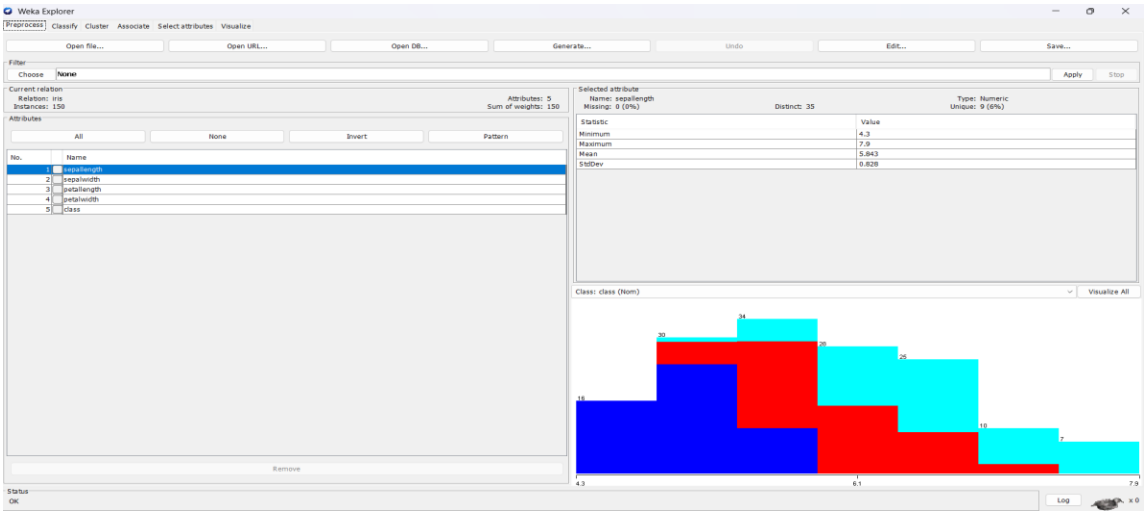
Week-2

Aim :

Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets

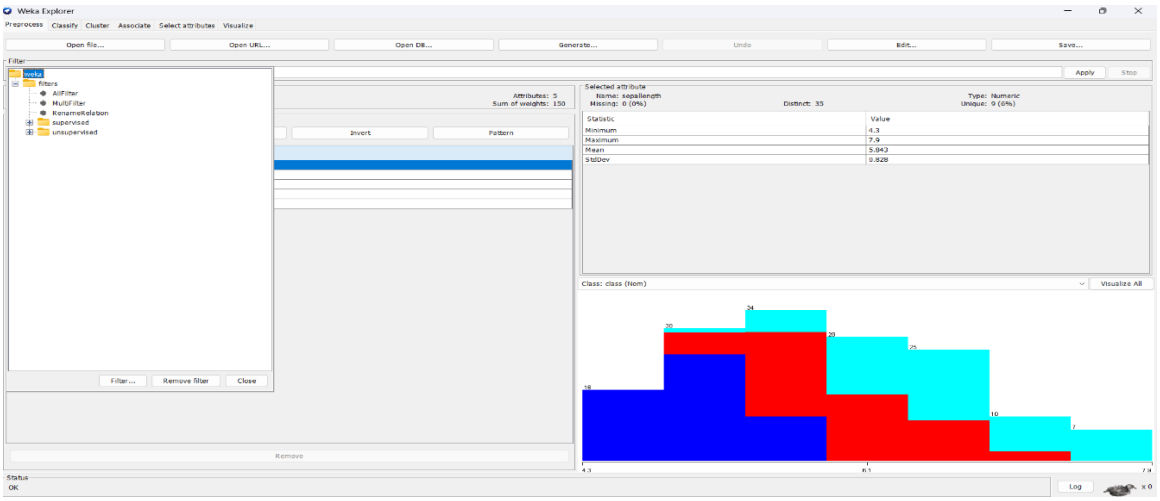
Data Preprocessing in Weka

- **Applying Discretization Filter**
  - Discretization converts numeric attributes into nominal ranges.

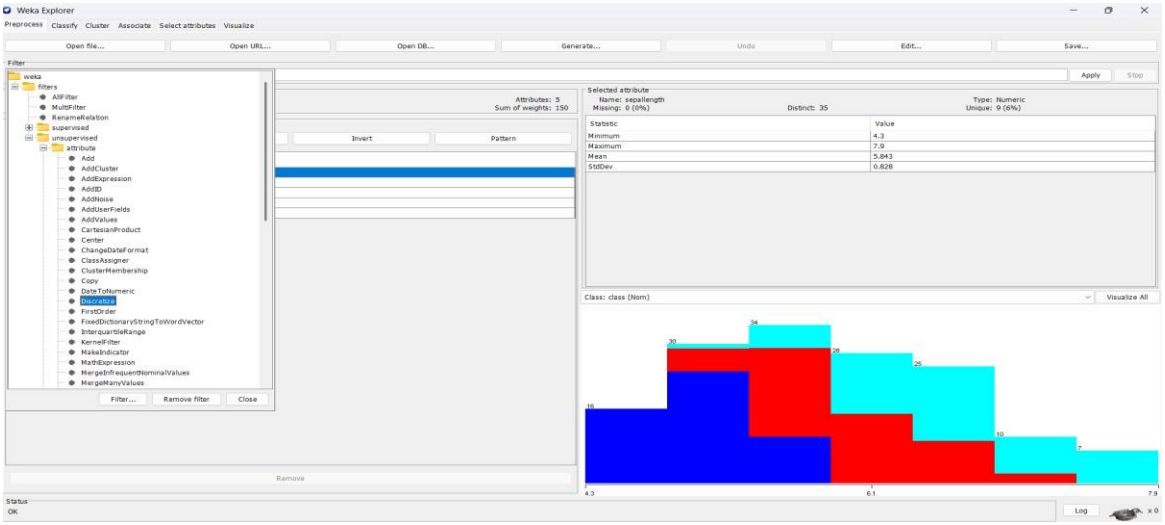


- This helps algorithms like Apriori which require categorical inputs

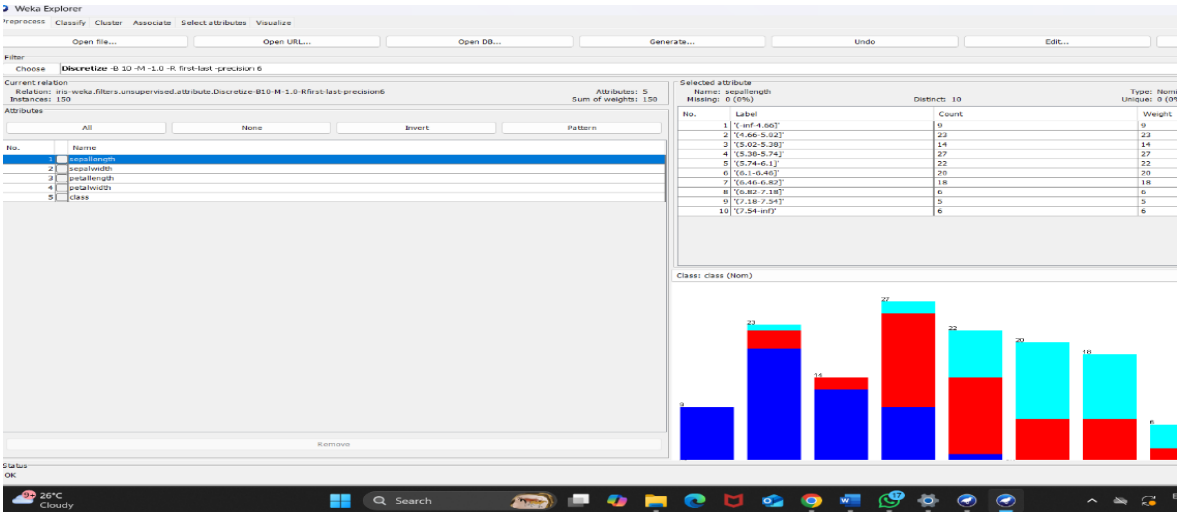
- Weka provides supervised and unsupervised discretization filters.



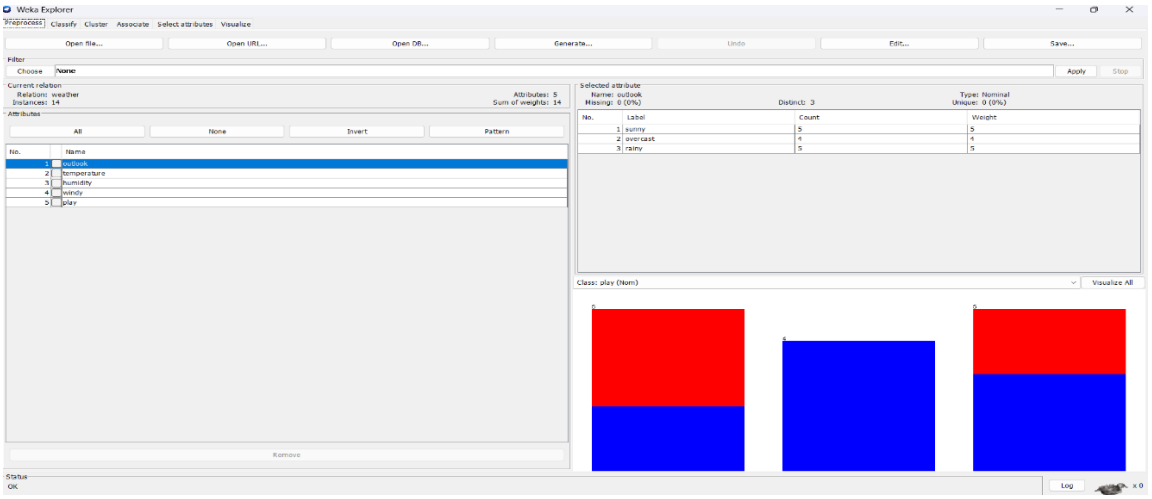
- The filter can be applied from the Preprocess panel.



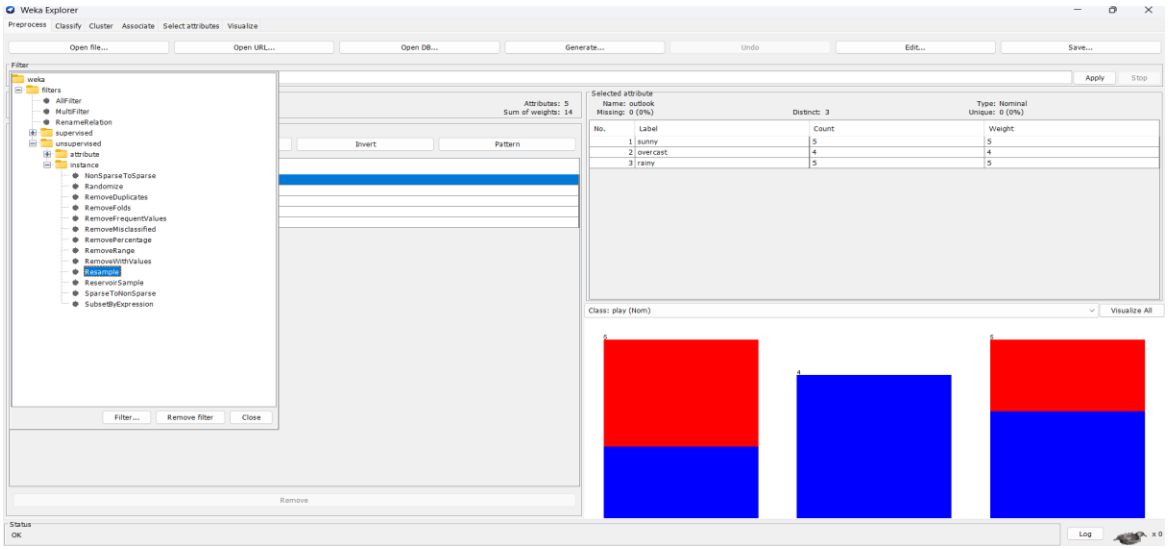
- Discretization improves interpretability in association rules.



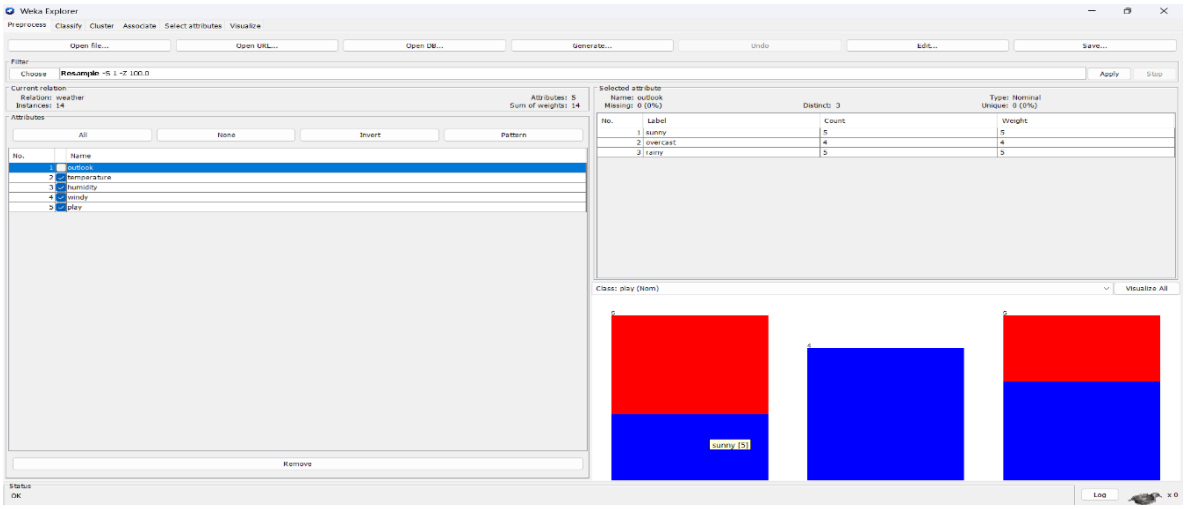
- **Applying Resample Filter**
  - Resample filter is used for random sampling of instances.



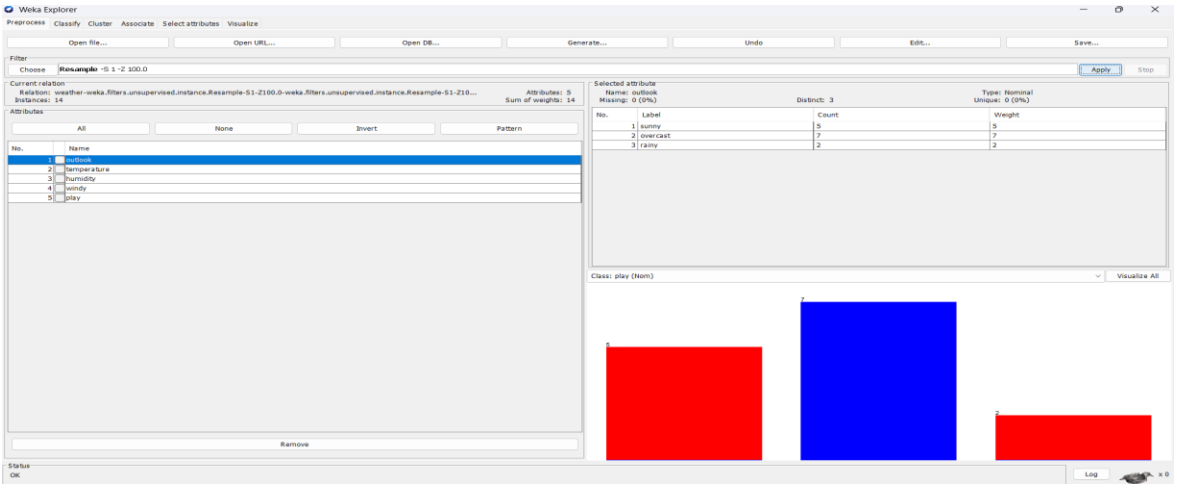
- It can be applied with or without replacement.



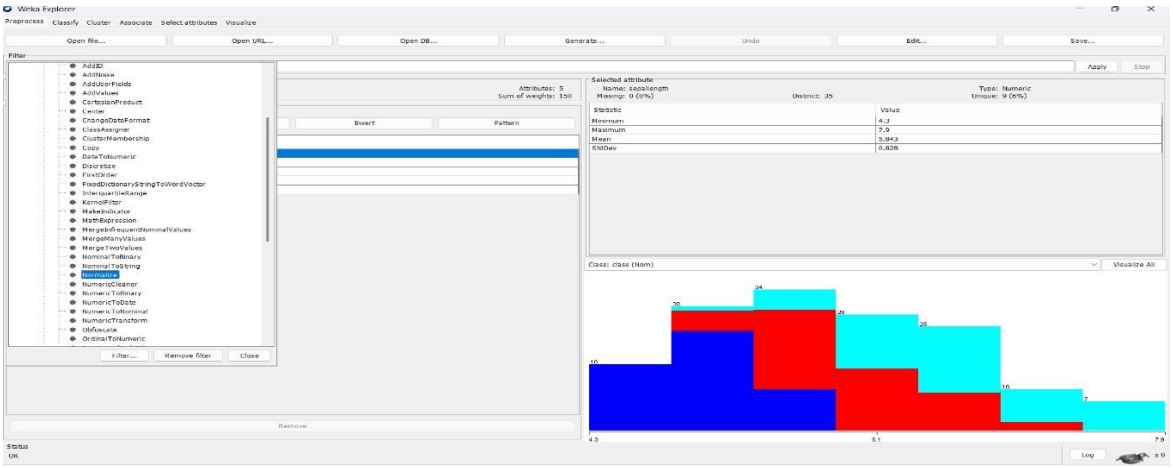
- It helps in balancing datasets with class imbalance.



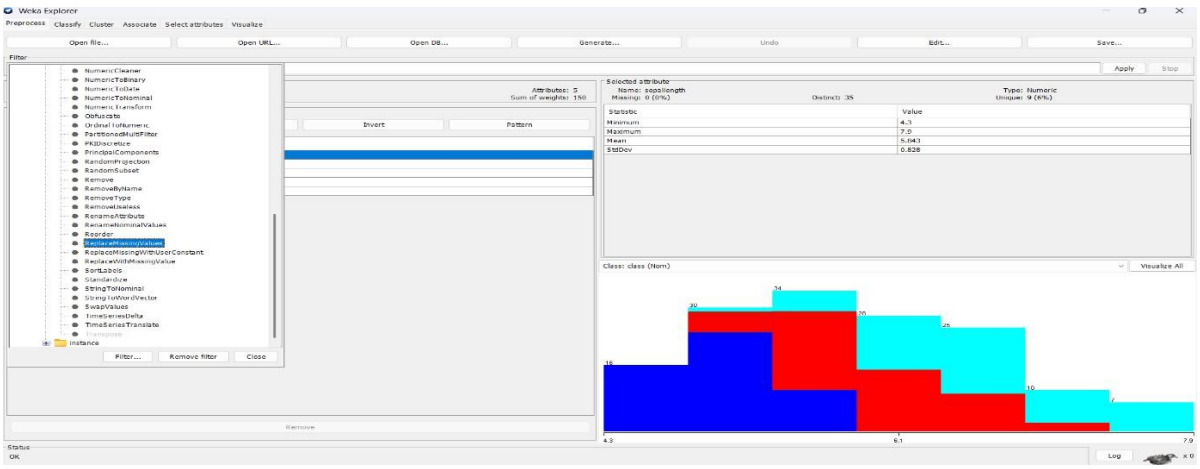
- Users can specify the percentage of data to retain.
- Resampling improves fairness in classification and rule mining.



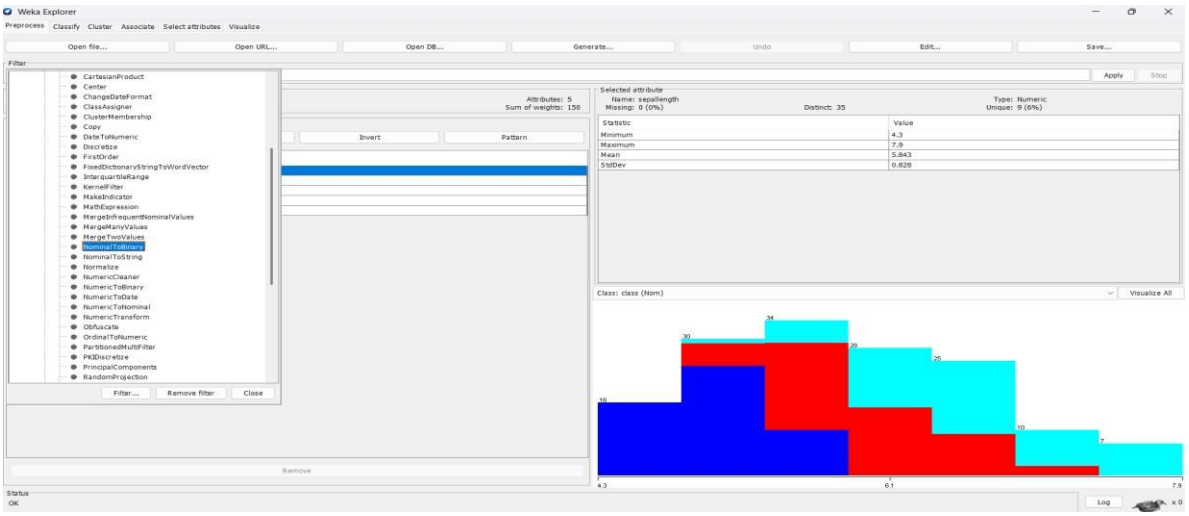
- **Other Filters in Weka**
  - Weka provides Normalize and Standardize filters for scaling data.



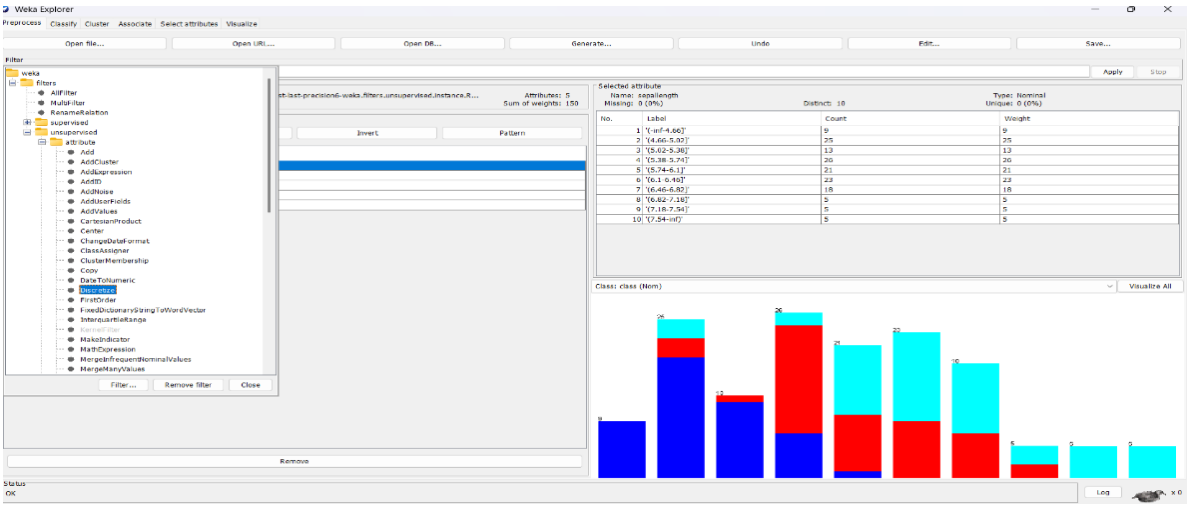
- Attribute removal filter allows removing unwanted fields.
- Replace missing values filter is used for data cleaning



- Nominal to Binary filter converts nominal attributes into binary form.

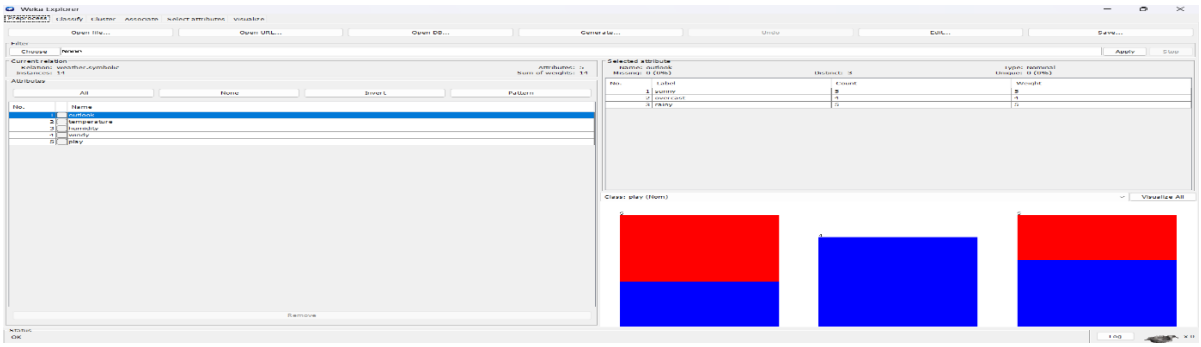


- These filters enhance the quality and usability of data.



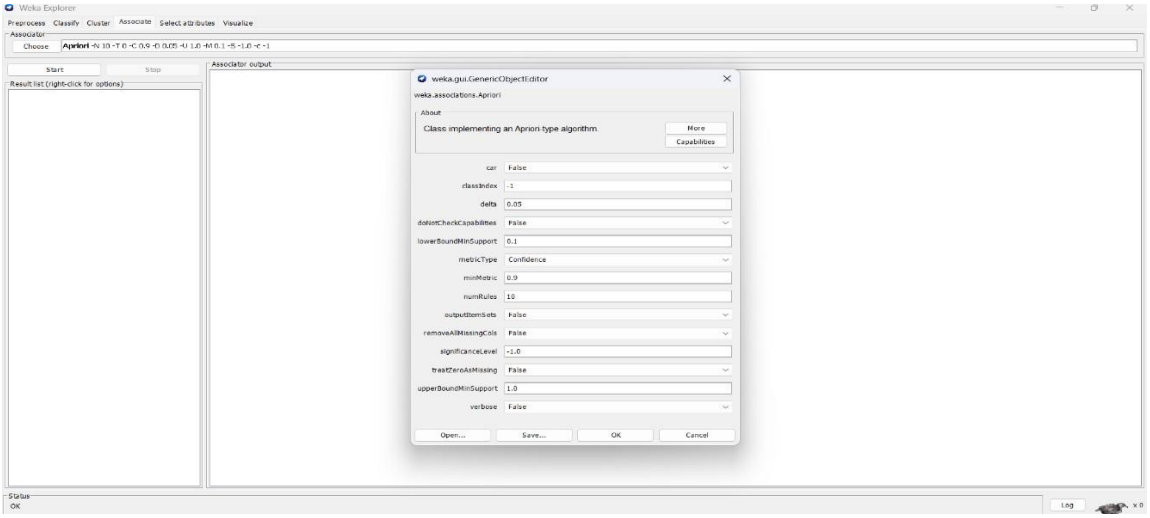
Association Rule Mining in Weka

- **Running Apriori on Weather Dataset**
  - Open weather.nominal dataset in Weka.



- Select **Associate** → **Apriori** algorithm.

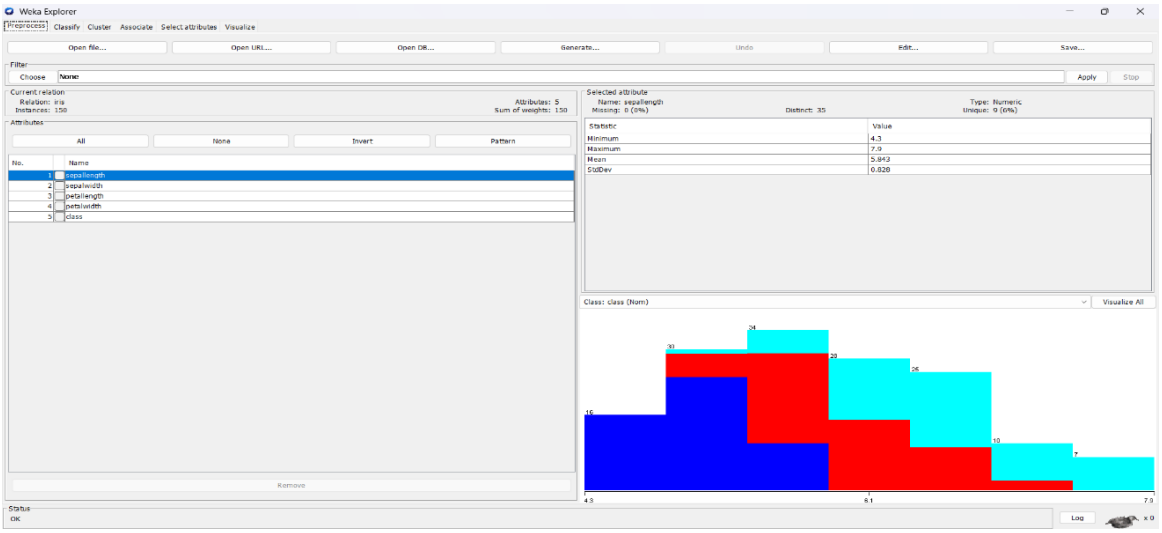
- Run with default support (0.1) and confidence (0.9).



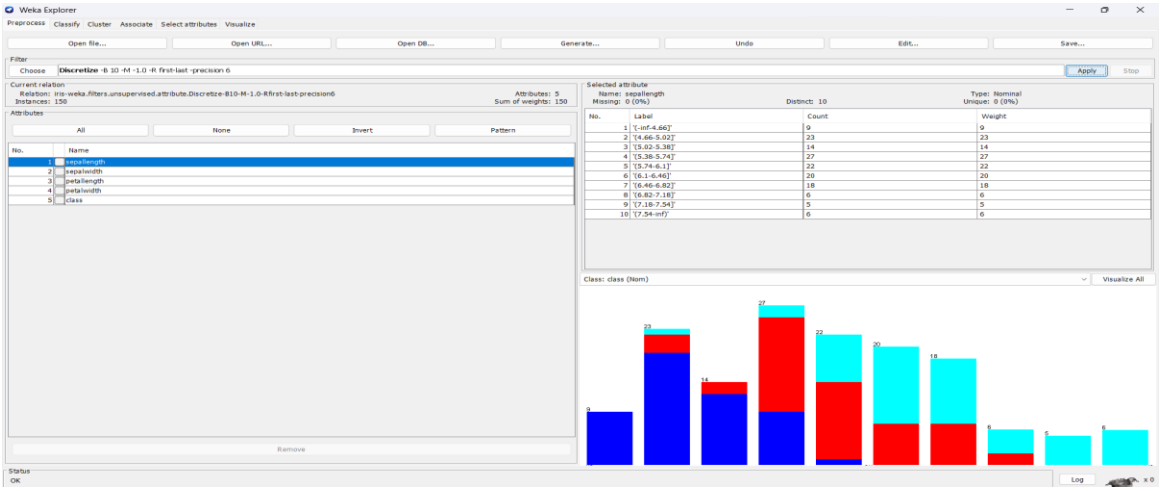
- Apriori generates rules like "If Outlook=Sunny then Play=No."
- Rules represent hidden relationships among attributes.

- **Running Apriori on Iris Dataset**

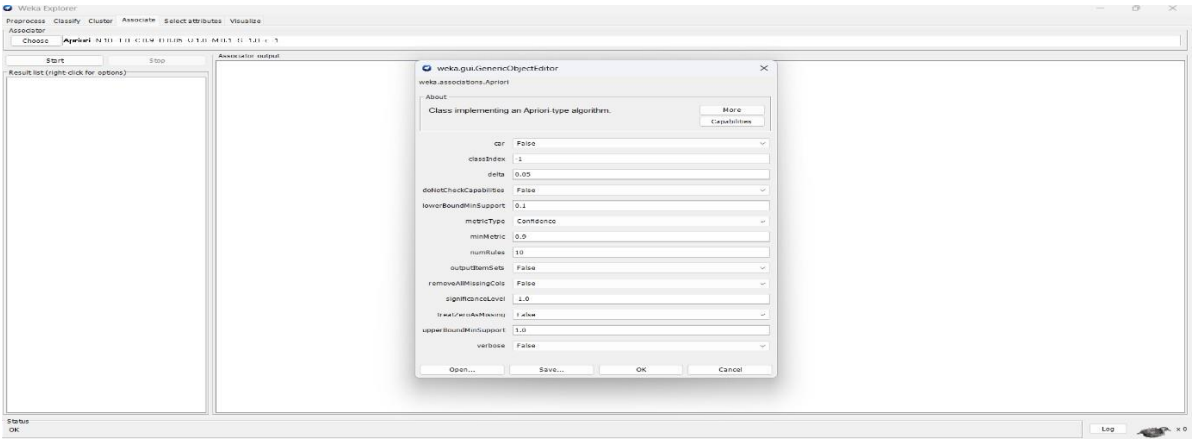
- Open iris.arff dataset in Preprocess panel.



- Apply discretization filter to convert numeric attributes.



- Run Apriori algorithm with minimum support 0.2 and confidence 0.8.

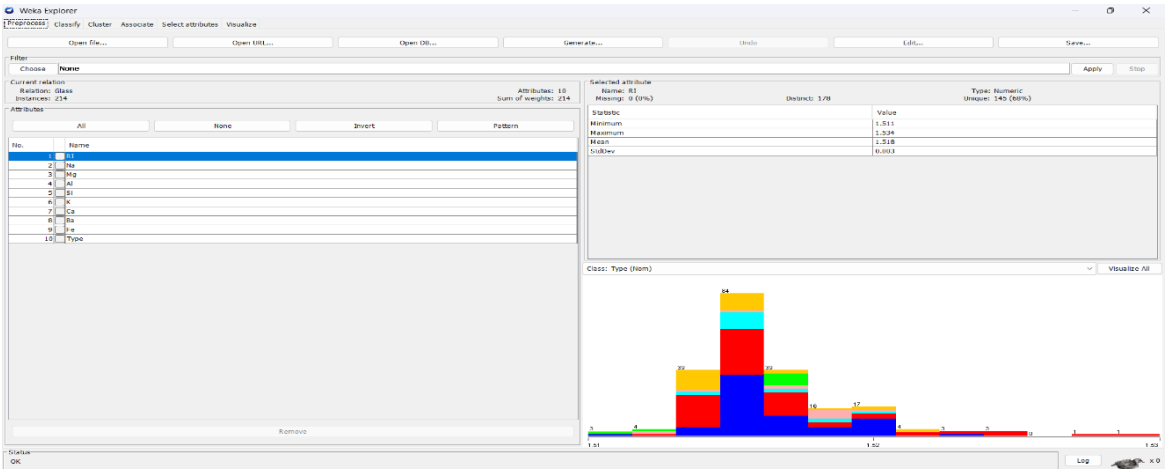


- Generated rules include attribute-value relations for flower types.

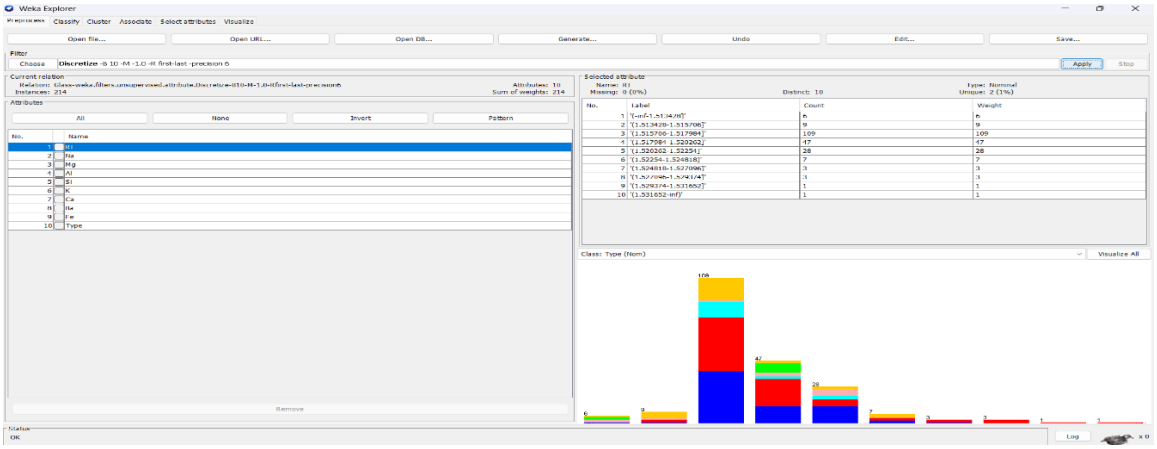
- Discretization helps Apriori produce meaningful categorical rules

2. Running Apriori on Glass Dataset

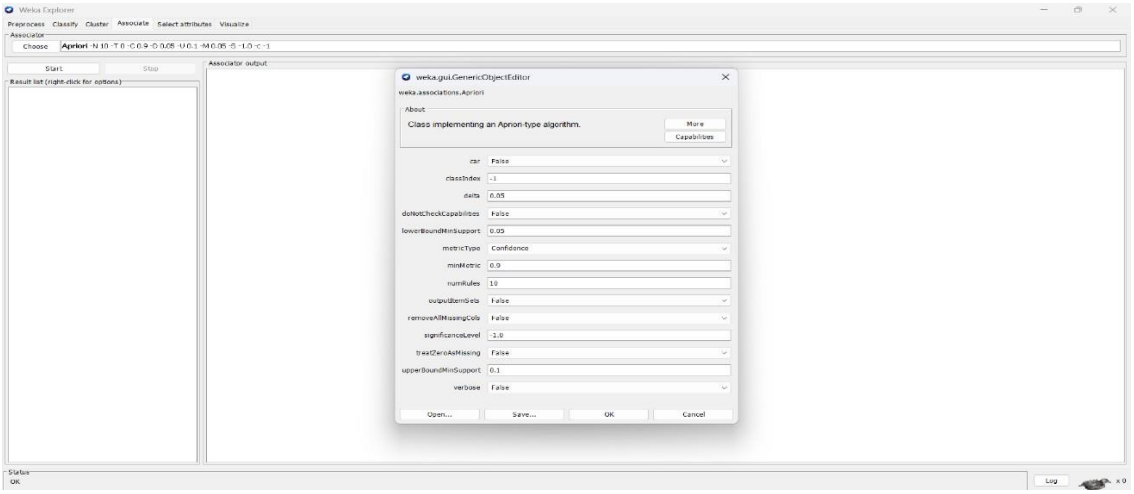
- Load glass.arff dataset into Weka.



- Since Glass has many numeric attributes, apply discretization.



- Run Apriori with lower support (0.05) due to large attribute set.



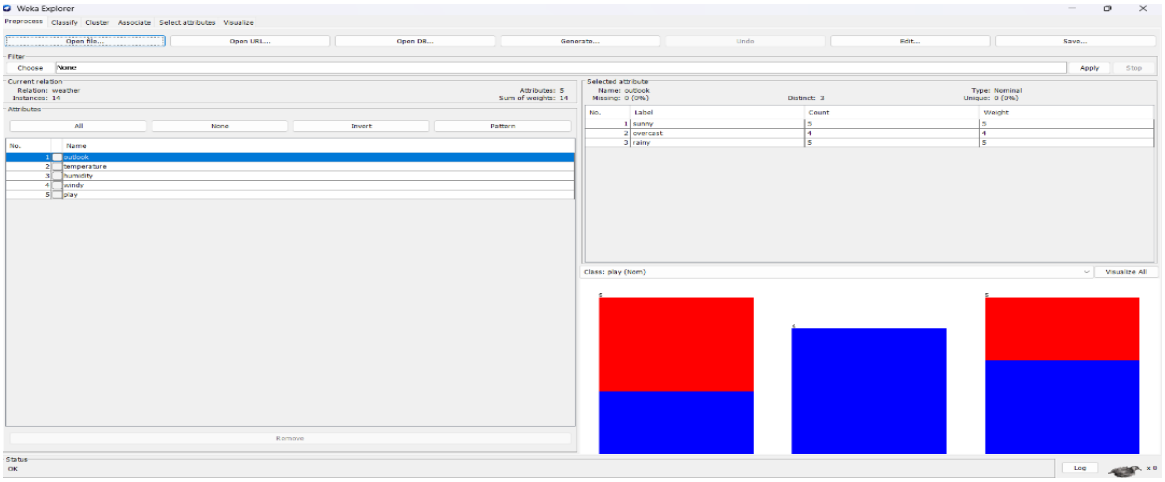
- Generated rules may highlight chemical compositions linked to glass type.

- Insights help identify key features that differentiate glass categories.

Effect of Discretization in Rule Generation

- **Without Discretization**

- Numeric attributes make rule generation difficult.

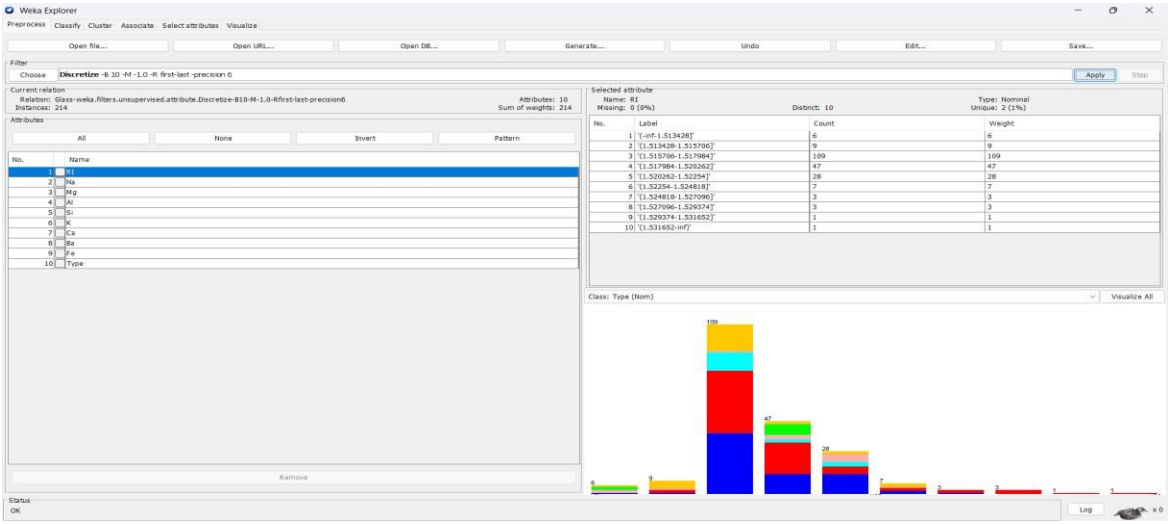


- Apriori may fail to produce useful rules.
- Continuous values reduce chances of frequent patterns.

```
                Si
                K
                Ca
                Ba
                Fe
                Type
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.3 (64 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 24

Size of set of large itemsets L(3): 9

Best rules found:

 1. Na='(12.725-13.39]' 93 ==> Ba='(-inf-0.315]' 93    <conf:(1)> lift:(1.16) lev:(0.06) [12] conv:(12.6)
 2. Na='(12.725-13.39]' K='(-inf-0.621]' 69 ==> Ba='(-inf-0.315]' 69    <conf:(1)> lift:(1.16) lev:(0.04) [9] conv:(9.35)
 3. RI='(1.515706-1.517984]' Mg='(3.143-3.592]' 65 ==> Ba='(-inf-0.315]' 65    <conf:(1)> lift:(1.16) lev:(0.04) [8] conv:(8.81)
 4. Na='(12.725-13.39]' Ca='(7.582-8.658]' 64 ==> Ba='(-inf-0.315]' 64    <conf:(1)> lift:(1.16) lev:(0.04) [8] conv:(8.67)
 5. Type=build wind non-float 76 ==> Ba='(-inf-0.315]' 75    <conf:(0.99)> lift:(1.14) lev:(0.04) [9] conv:(5.15)
 6. Type=build wind float 70 ==> Ba='(-inf-0.315]' 69    <conf:(0.99)> lift:(1.14) lev:(0.04) [8] conv:(4.74)
 7. Al='(1.253-1.574]' 81 ==> Ba='(-inf-0.315]' 75    <conf:(0.98)> lift:(1.13) lev:(0.04) [8] conv:(3.66)
 8. Al='(1.253-1.574]' K='(-inf-0.621]' 67 ==> Ba='(-inf-0.315]' 65    <conf:(0.97)> lift:(1.12) lev:(0.03) [7] conv:(3.03)
 9. Mg='(3.143-3.592]' 86 ==> Ba='(-inf-0.315]' 83    <conf:(0.97)> lift:(1.12) lev:(0.04) [8] conv:(2.91)
10. Type=build wind float 70 ==> K='(-inf-0.621]' 64    <conf:(0.91)> lift:(1.14) lev:(0.04) [7] conv:(1.96)
```

- Rules may be very specific with little generalization.
- Insights are limited and harder to interpret.

- **With Discretization**
  - Numeric attributes are converted into ranges (bins).



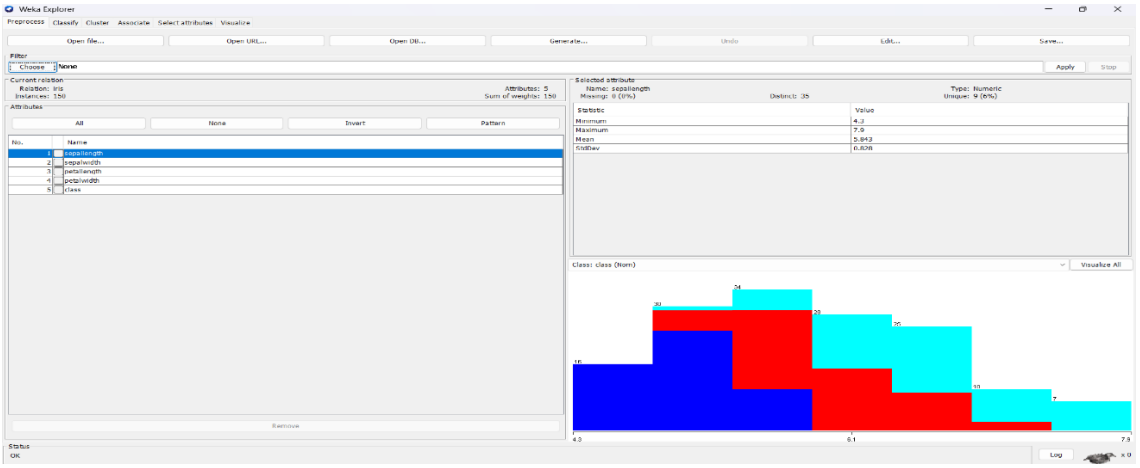
- Frequent patterns become easier to detect.

- Rules become simpler and more interpretable.
- Support and confidence of rules improve.

- Discretization clearly enhances Apriori performance.

Insights and Observations

- Preprocessing improves the quality of data for mining tasks.



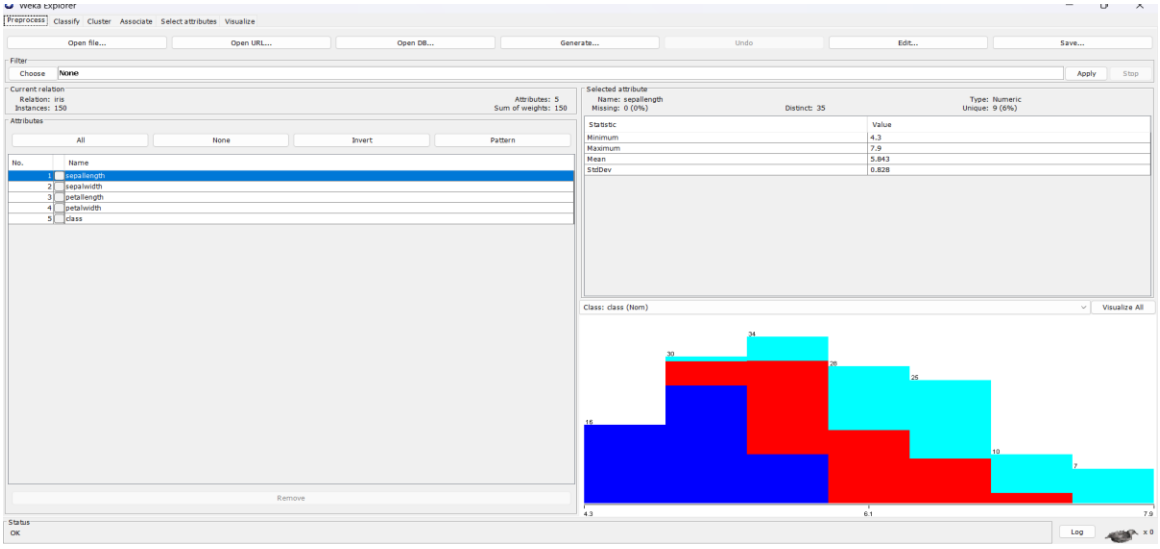
- Discretization significantly helps in Apriori rule generation.

- Weather dataset gives simple and interpretable rules.

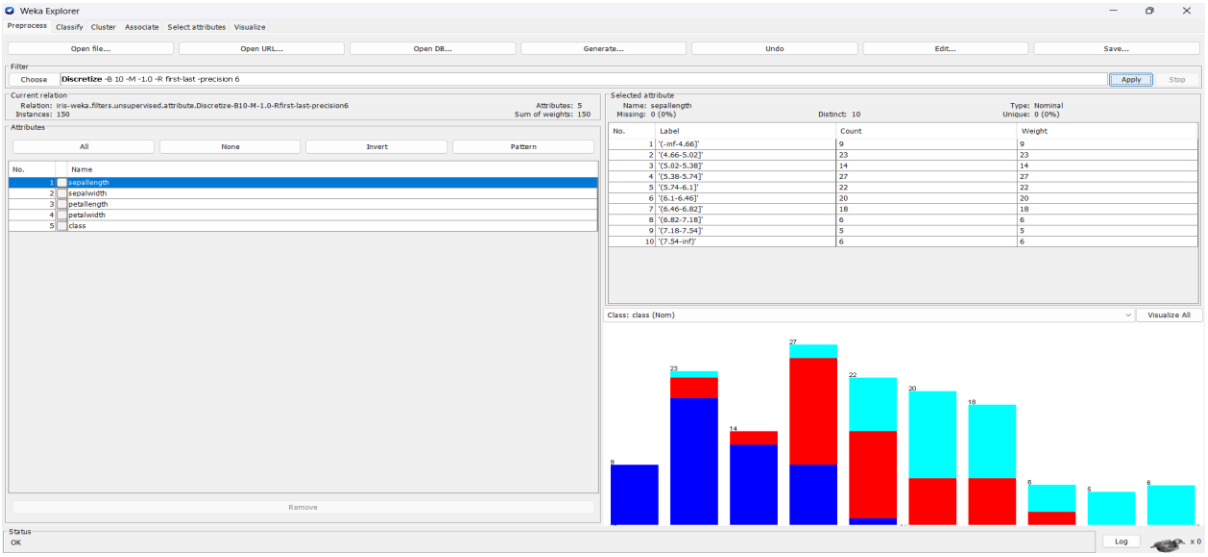
- Iris and Glass datasets generate complex rules after discretization.

- Overall, Apriori with discretized attributes produces more meaningful insights.

Before Decritization:



After Decritization:



Viva Questions

1. **Why is discretization important for Apriori in Weka?**

2. **What is the difference between support and confidence in association rule mining?**

3. **Why do we use resampling in preprocessing?**

4. **What happens if Apriori is applied on numeric attributes without preprocessing?**

5. **What is the main difference between Weather, Iris, and Glass datasets in rule mining?**

Week-3

3. Demonstrate performing classification on data sets Weka/R

- Load each dataset and run 1d3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.
- Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix.
- Load each dataset into Weka/R and perform Naïve-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained.
- Plot RoC Curves
- Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

AIM: To perform different classification algorithms on datasets in Weka (ID3, J48, Naïve Bayes, and k-NN), analyze entropy and Kappa statistic, extract rules, plot ROC curves, and compare classifier performance.

Exploring and Loading Datasets in WEKA

WEKA comes bundled with several sample datasets such as **Iris**, **Weather**, **Diabetes**, etc.

To explore datasets:

1. Open WEKA Explorer.
2. Click **Open file** in the Preprocess panel.
3. Navigate to the data folder within the WEKA installation directory.
4. Select a dataset like iris.arff or weather.arff.

Loading a dataset allows you to view the attributes and instances and apply preprocessing, classification, clustering, and more.

Example: Loading the Iris Dataset

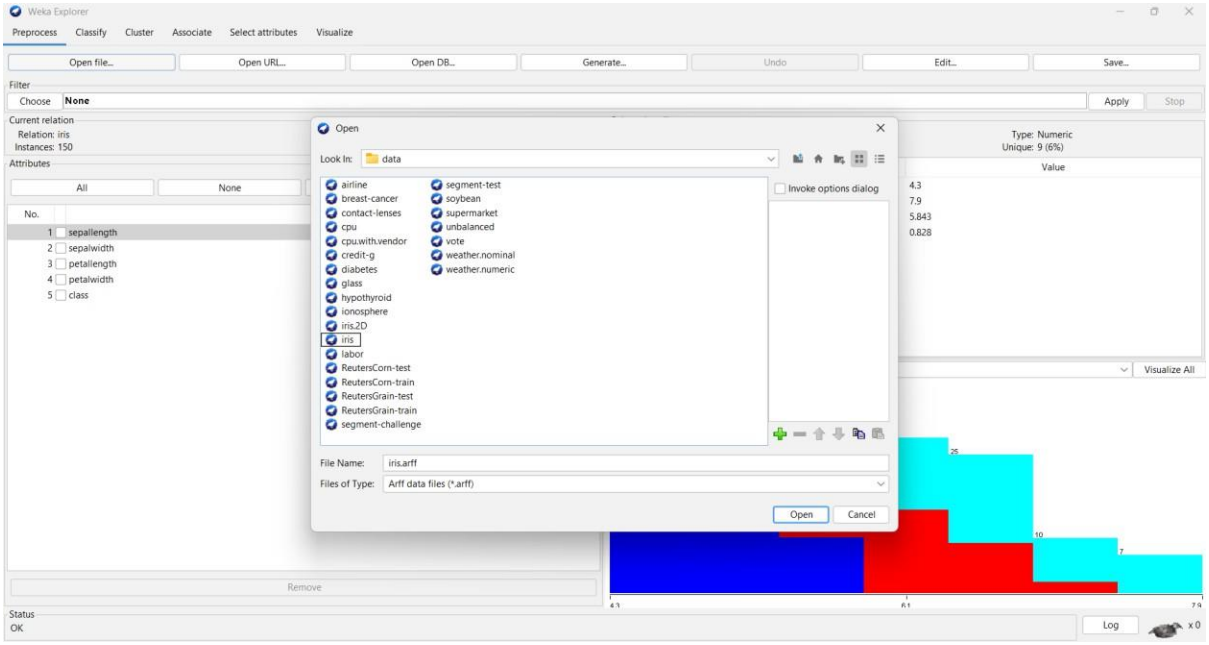
Open WEKA Explorer.

In the Preprocess panel, click **Open file**.

Select the iris.arff file.

The dataset loads and displays attributes such as sepal length, sepal width, petal length, petal width, and class.

You can now proceed to classify, cluster, or visualize this dataset.

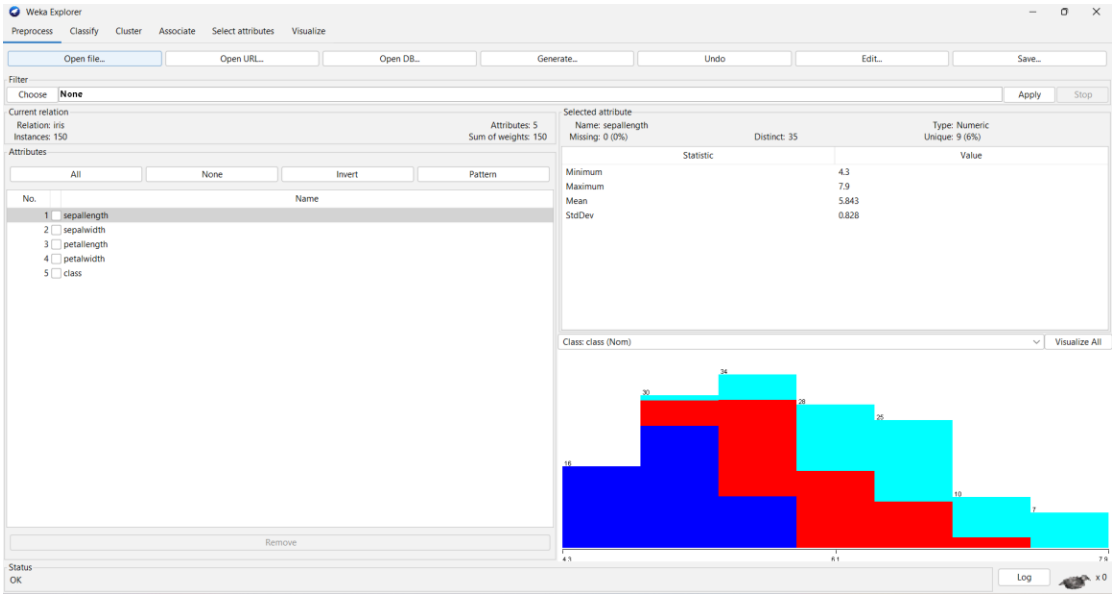


After loading the iris data set the visualization of the preprocess is as shown in the figure.

The attributes are sepallength , sepalwidth,petallength,petalwidth.

The value and statistic of the each attribute is as shown.

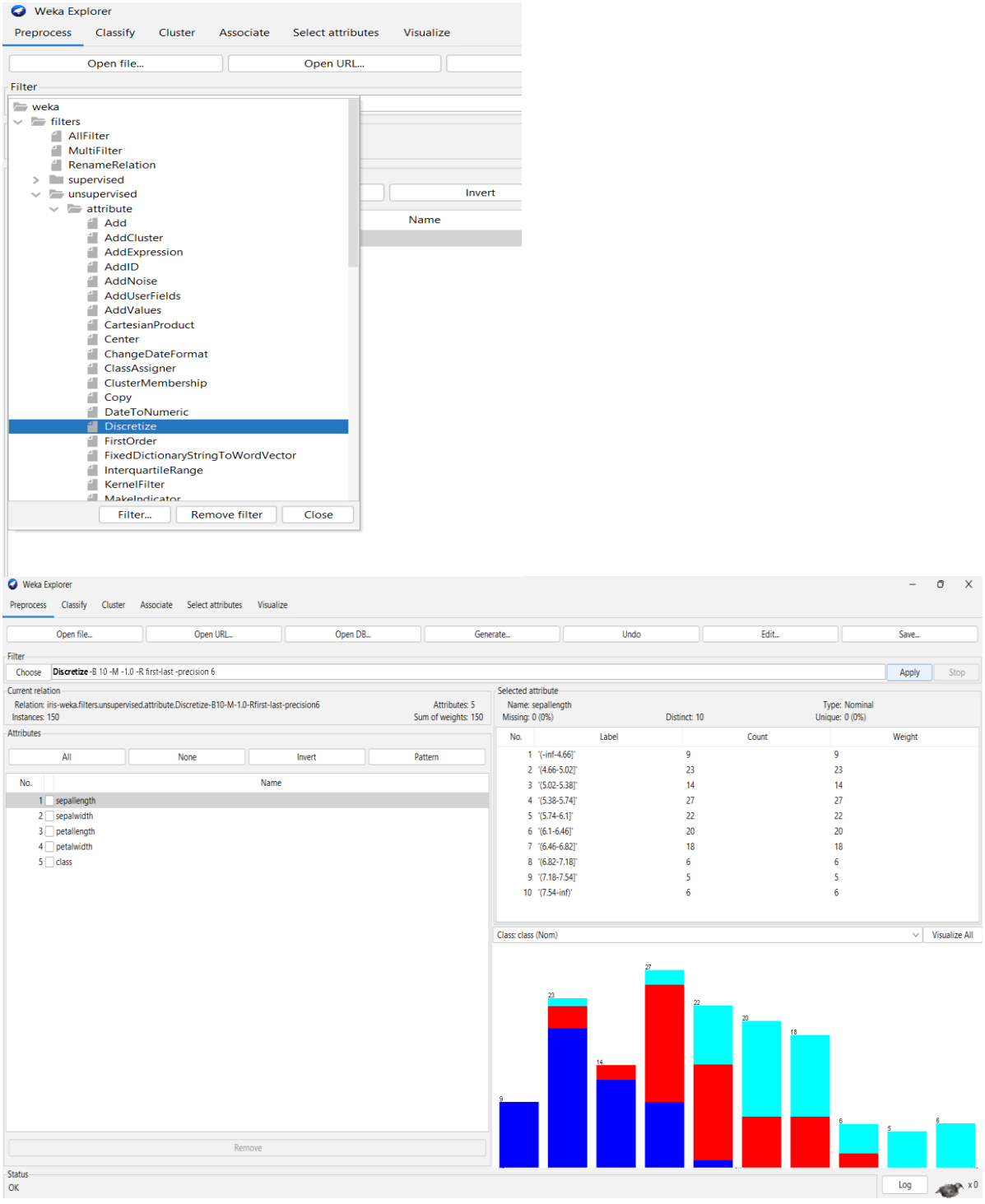
Histograms of each attribute of the data is shown below.



Discretizing the data:

- It is important to discritize the data because it involves transforming continuous numeric attributes into discrete, categorical (nominal) attributes.
- Open the filter.
- Select the Unsupervised data.

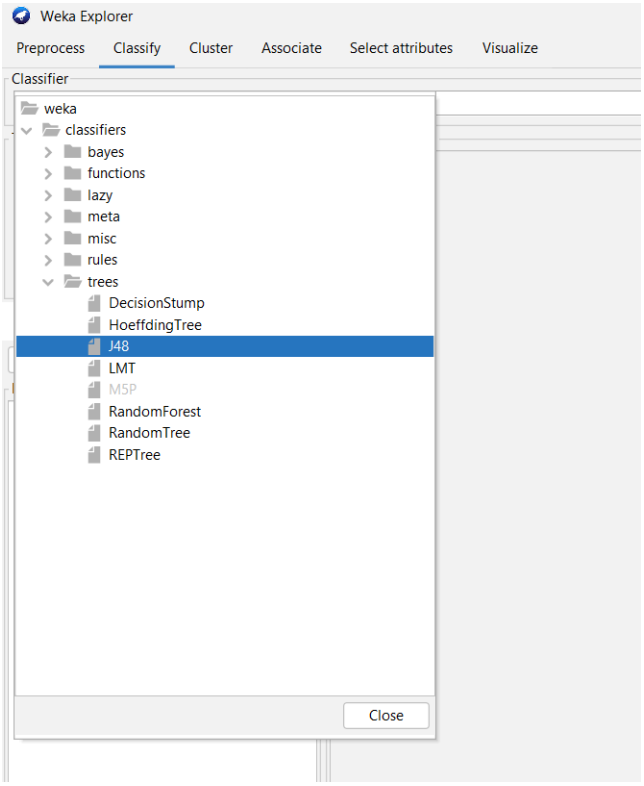
- Click on the discretize.
- Apply this filter on the data.



Filter Application: A "Discretize" filter has been applied to the "iris" dataset. This filter, Discretize -B 10 -M -1.0 -R first-last -precision 6, transforms numerical attributes into nominal (categorical) ones by dividing them into 10 bins.

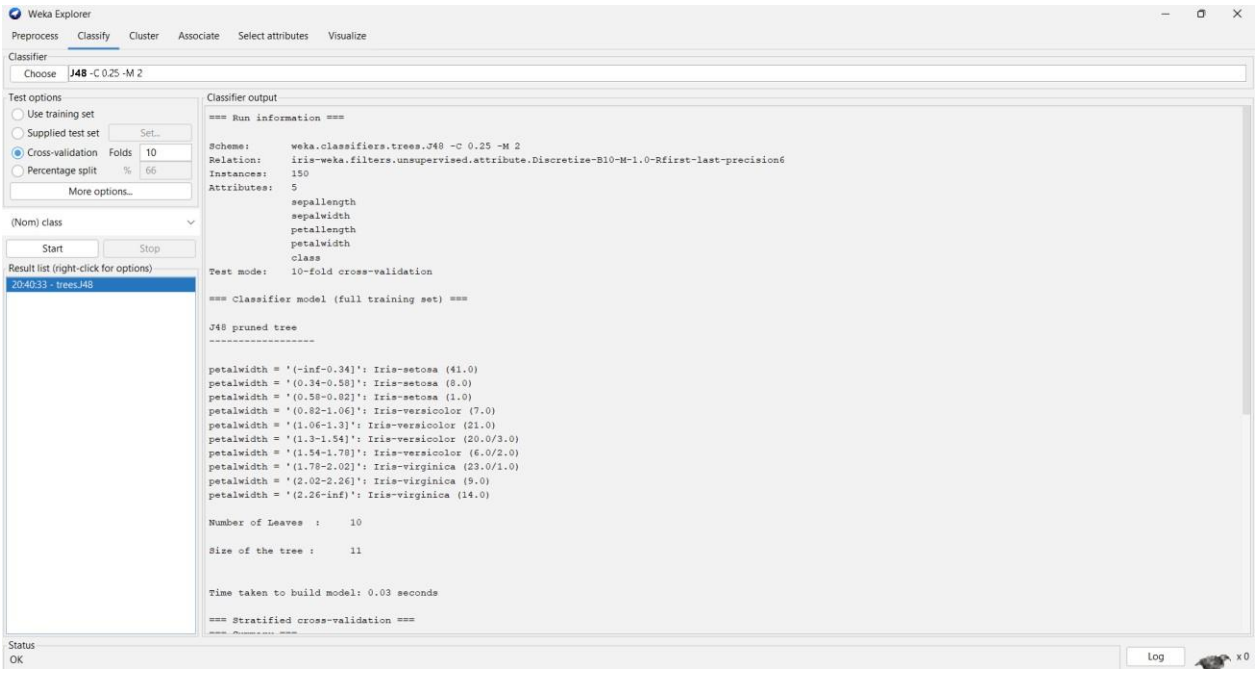
Classification on the data:

- To apply the classification on the data set, click on the classify panel.
- Select the different algorithms to perform on the data (J48,Id3 ,etc.., classification algorithms). Select the J48 algorithm as shown in fig.



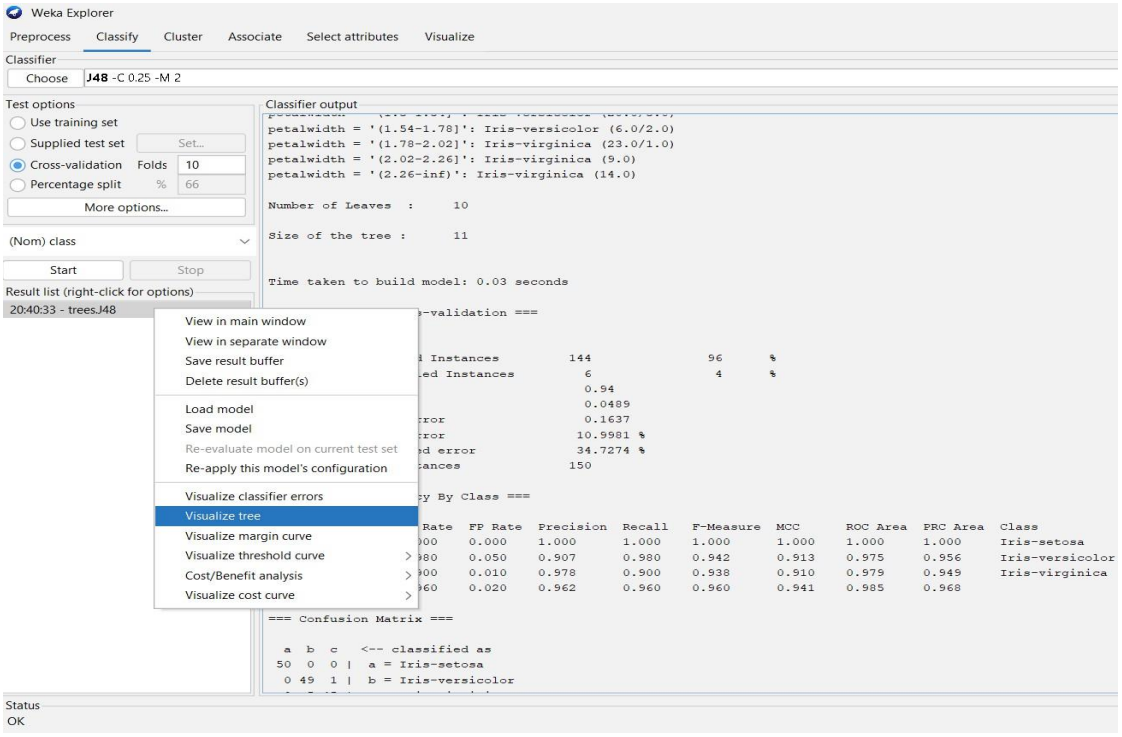
After applying the J48 algorithm, the default Cross-validation folds 10 in test options.

Click on the start button to see the classifier output.

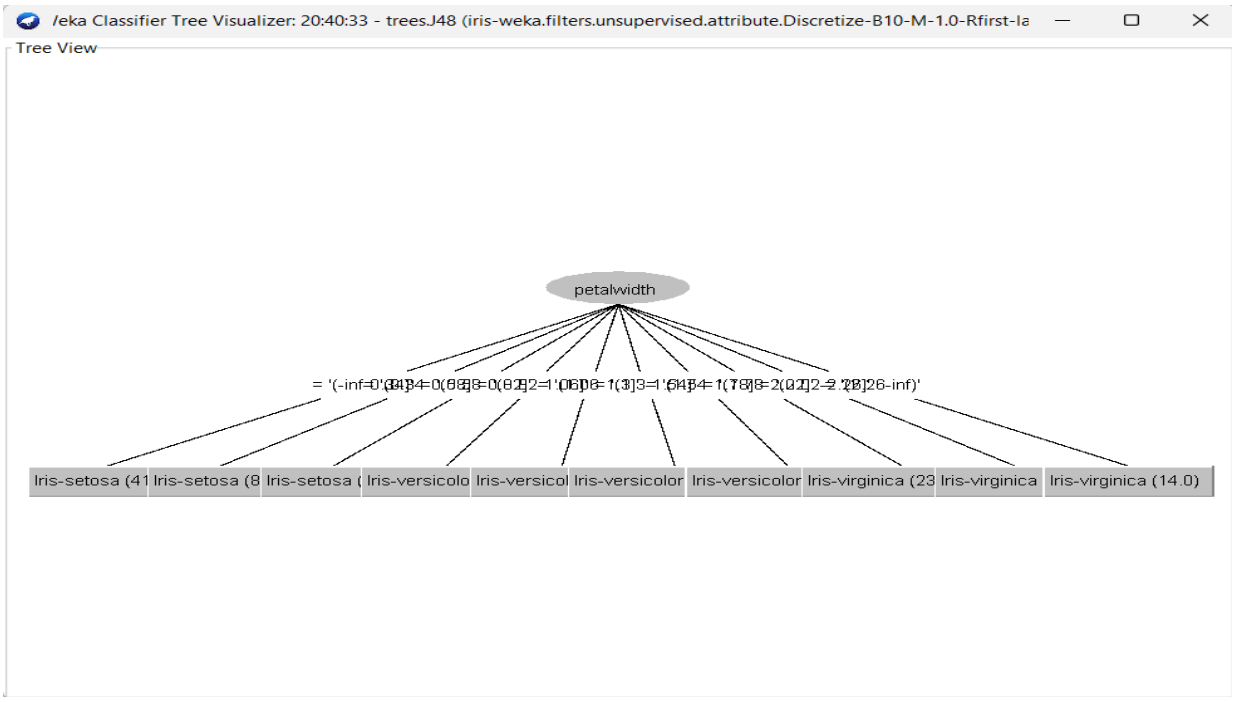


Visualization of a J48 algorithm in tree mode:

- In the result list , it shows the present running algorithm.
- Click on the right side to see more options.
- Click on the Visualize tree option to see the output in the tree mode.
- Do the process as shown in the figure.



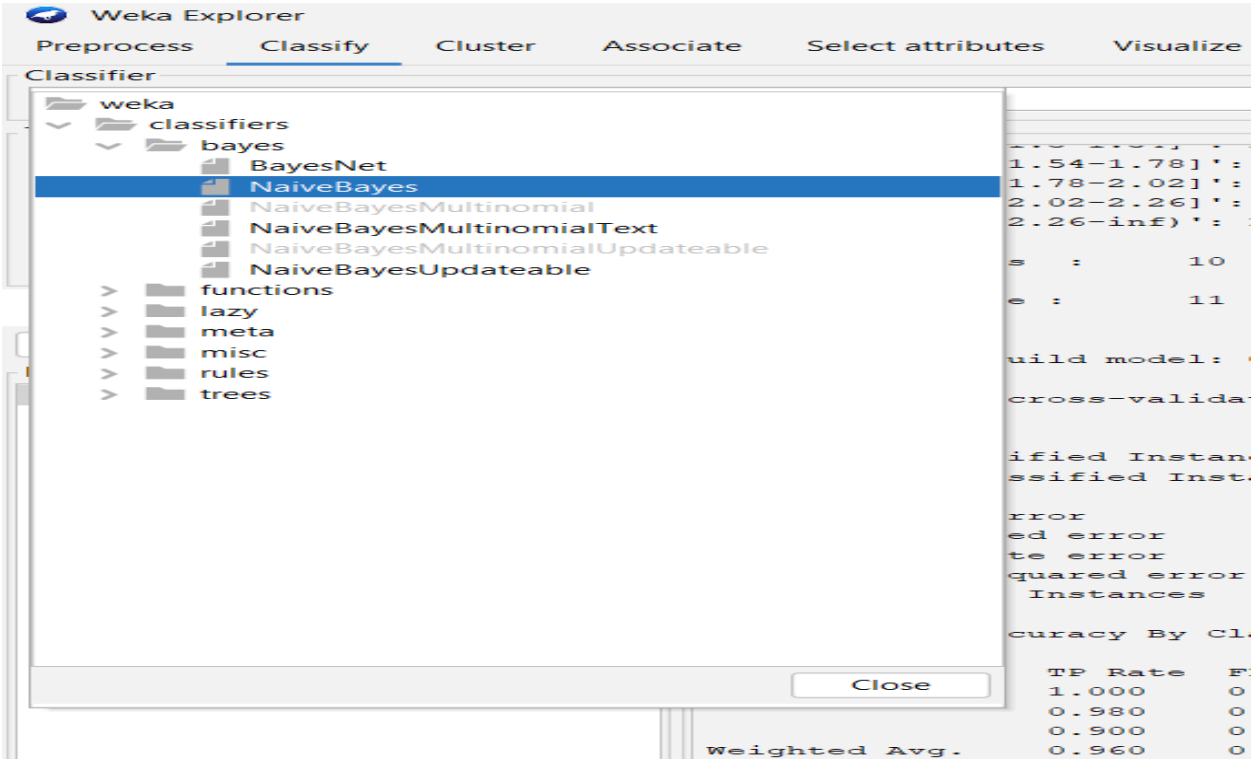
Classifier tree visualizer as shown in fig.



2. Load each dataset into Weka/R and perform Naïve-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained.

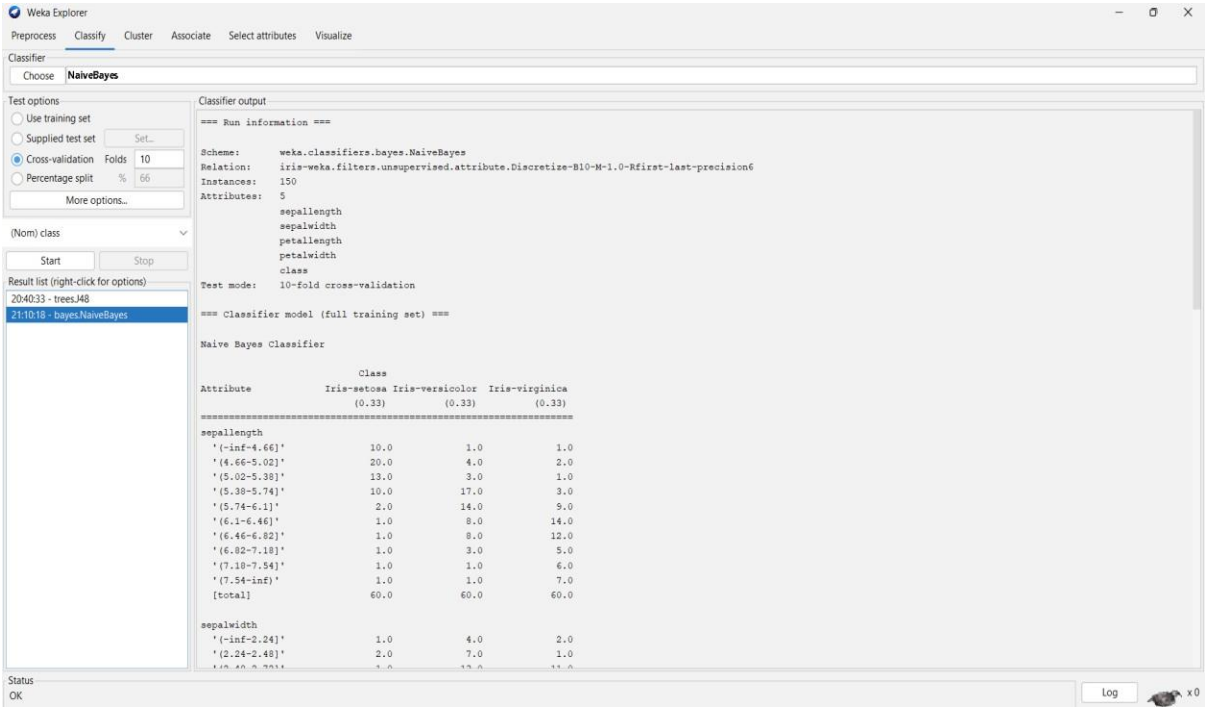
DATA SET:

- Performing the Naïve-bayes classification on the same data set (i.e.. is Iris data set).
- Here the data set Iris is already in discretization mode as we done in the above algorithm.
- Click on the classify panel.
- Select the Classifiers and open the bayes.
- In that bayes select the NaiveBayes algorithm.
- Do the process as shown in the figure.



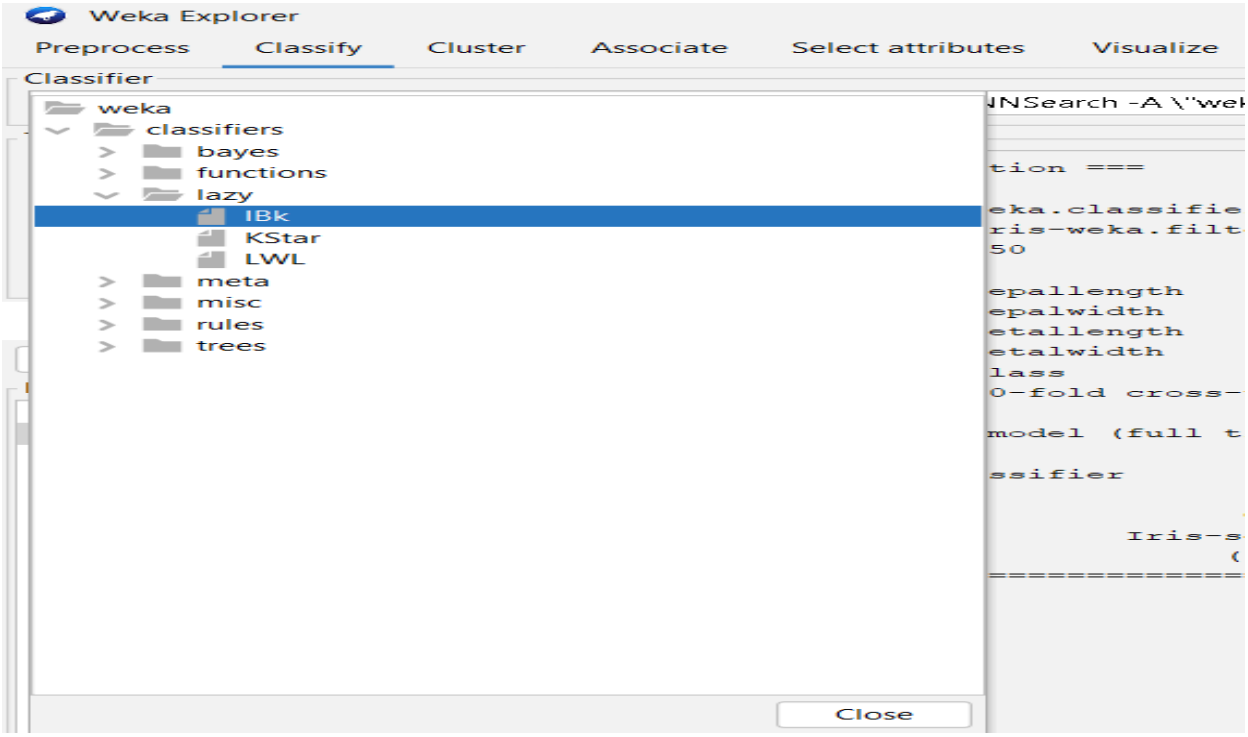
After choosing the Navie-Bayes Algorithm:

- In the test options the default cross validation folds is 10.
- Click on the start button to observe the classifier output for the Navie-bayes Algorithm.



-Nearest Neighbour Classification algorithm:

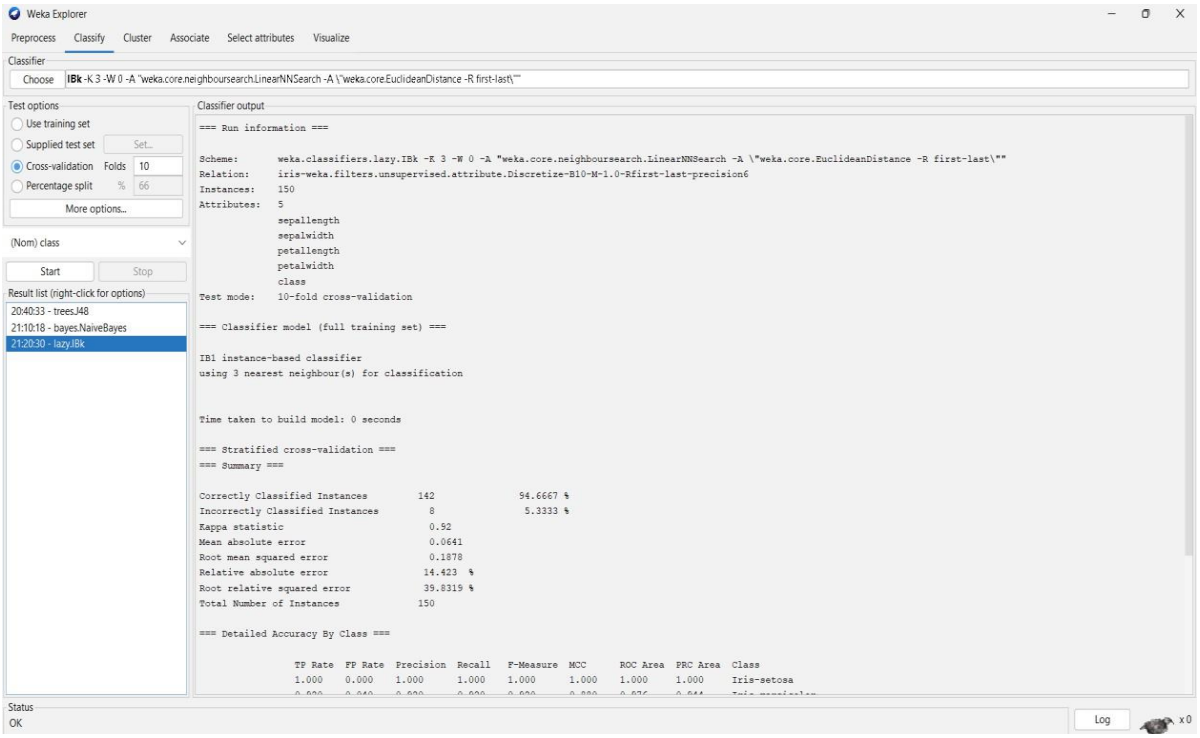
- **K** It is also one of the classification algorithm like J48,Id3 etc..
- Choose the IBk algorithm from the lazy classifiers.
- As shown in figure.



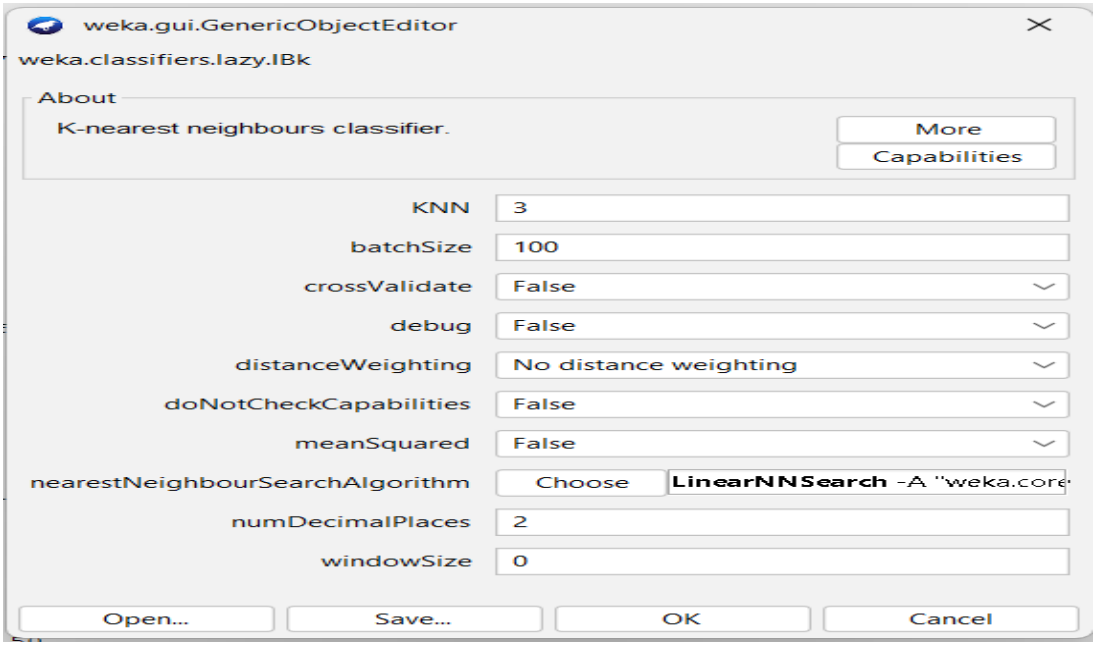
After choosing the IBk Algorithm

- In the test options the default cross validation folds is 10.

- Click on the start button to observe the classifier output for the IBk Algorithm.



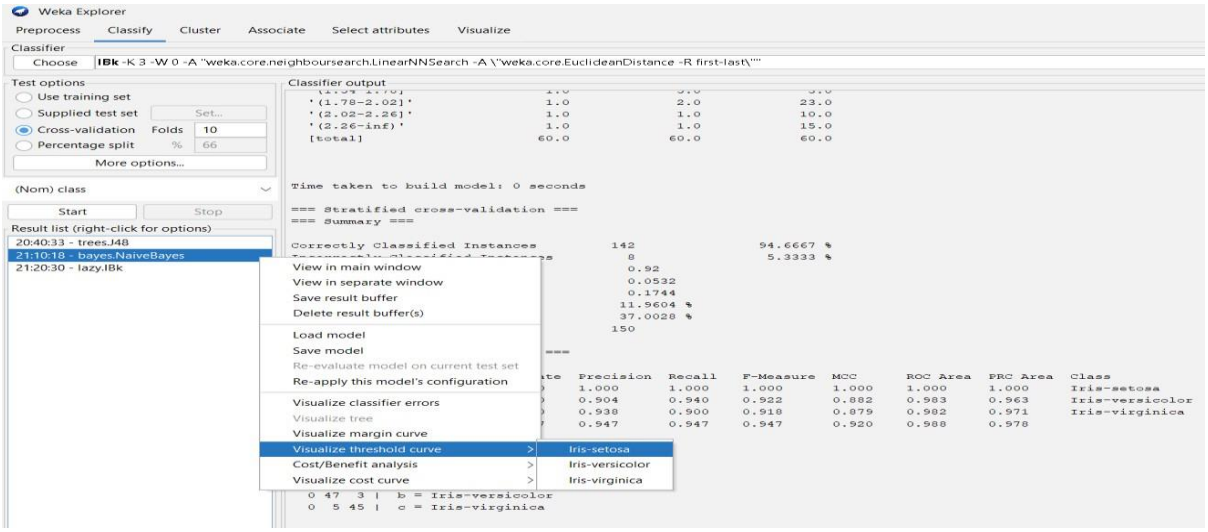
Here it is the Generic Object Editor for the classification on the iris data set with Ibk classification algorithm.



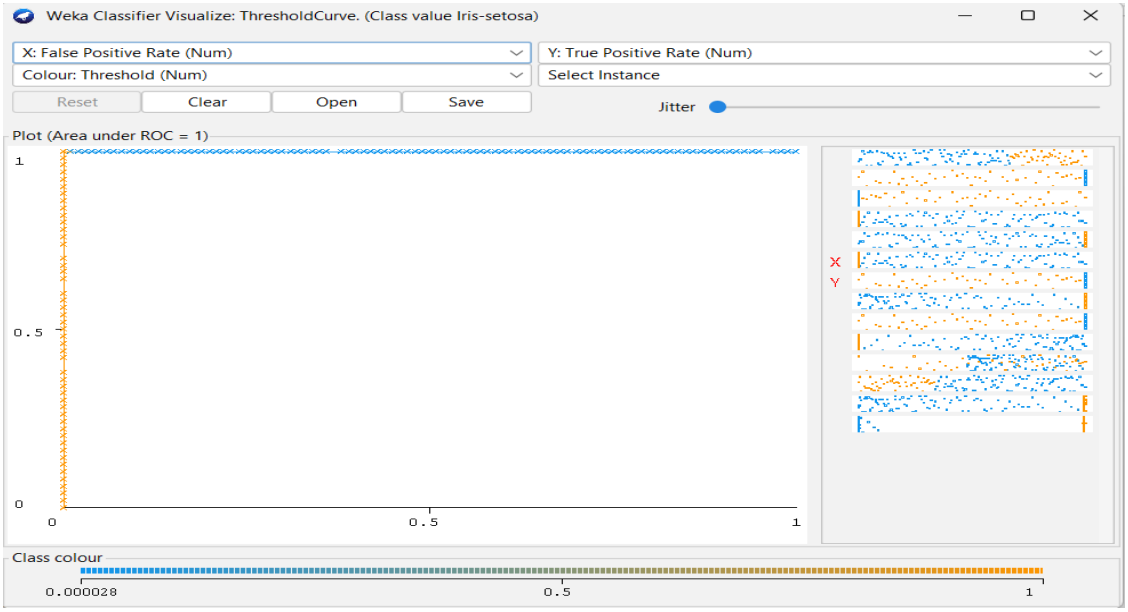
Plot RoC Curves

ROC Curve Visualization in Weka (IBk Classifier)

- After executing the IBk (k=3) classifier on the Iris dataset using 10-fold cross-validation, the results are listed under the Result list.
- By right-clicking on the classifier result, Weka provides multiple options for analyzing the model.

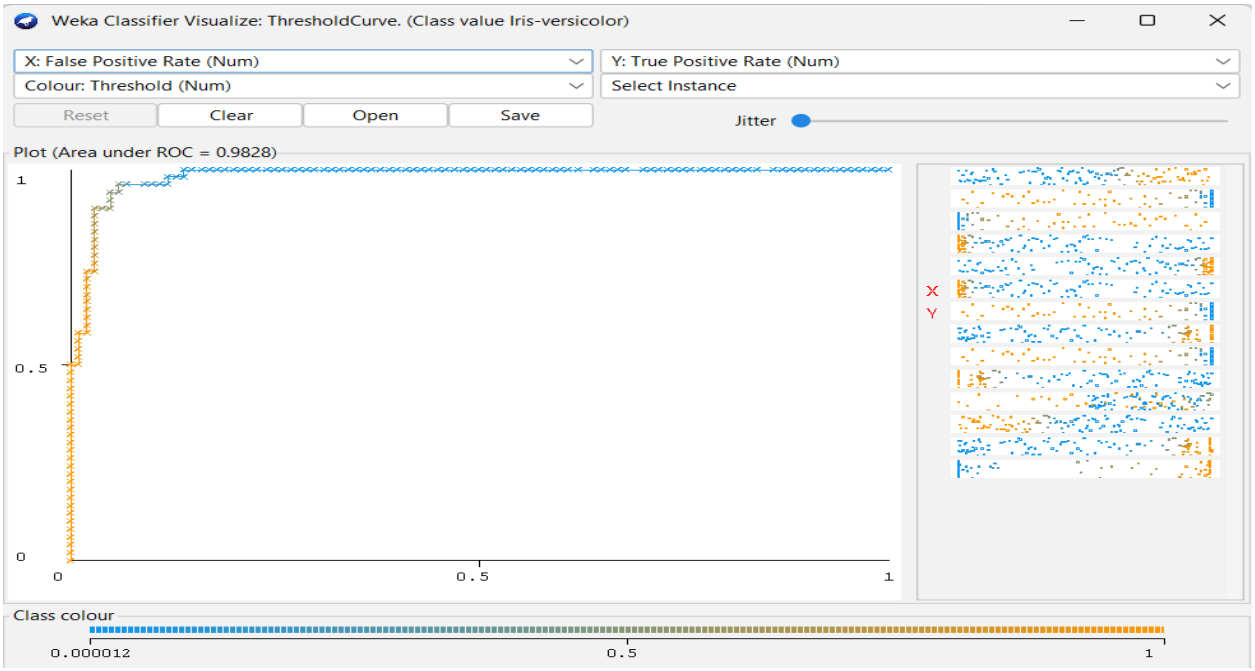


- Click on the Visualize threshold curve, in that select the Iris-setosa.
- **ROC Curve** plots the **True Positive Rate (TPR)** against the **False Positive Rate (FPR)**.

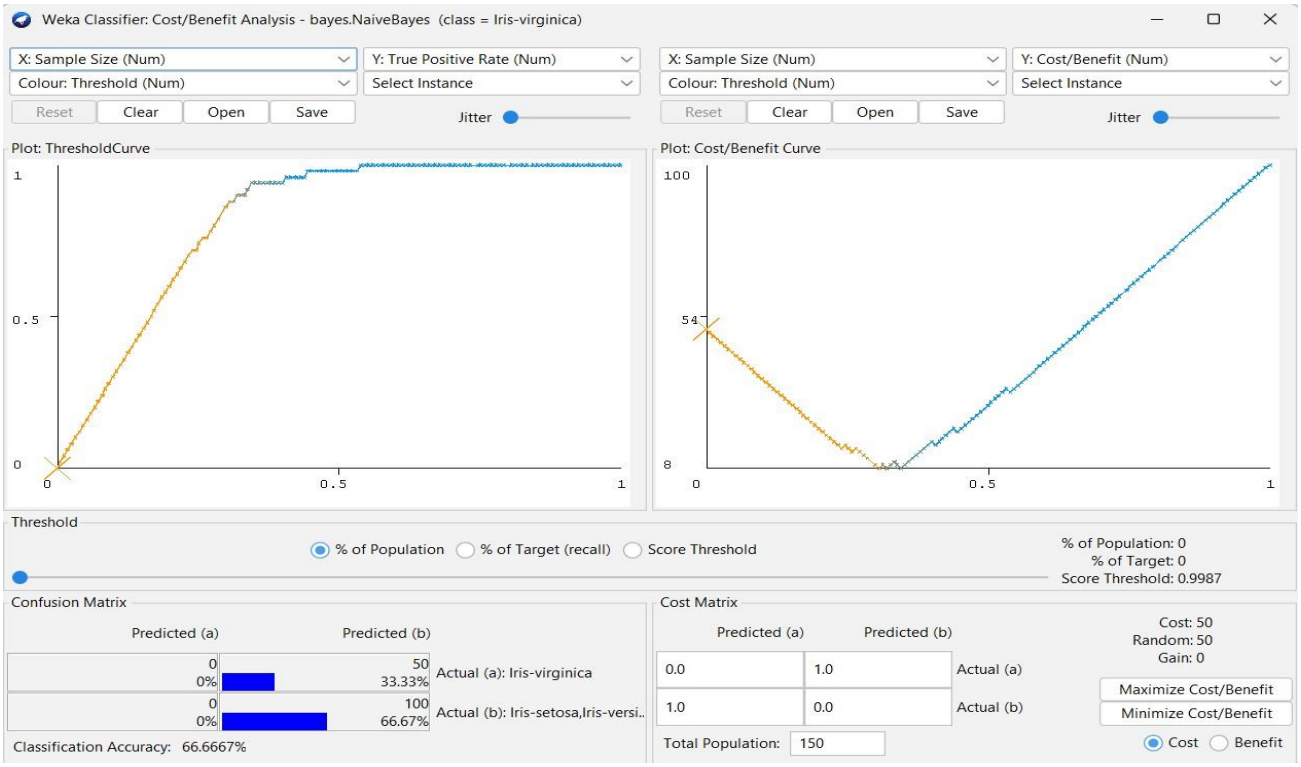


- It is used to evaluate the discriminative ability of a classifier.
- The **Area Under Curve (AUC)** values shown in the summary are very close to **1.0**, indicating that the classifier has high accuracy and strong class separation capability.

- Click on the Visualize threshold curve, in that select the Iris-versicolor.
- **ROC Curve** plots the **True Positive Rate (TPR)** against the **False Positive Rate (FPR)**.
- It is used to evaluate the discriminative ability of a classifier.
- The **Area Under Curve (AUC)** values shown in the summary are very close to **1.0**, indicating that the classifier has high accuracy and strong class separation capability.



- Click on the Visualize threshold curve, in that select the Iris-versicolor.
- **ROC Curve** plots the **True Positive Rate (TPR)** against the **False Positive Rate (FPR)**.
- It is used to evaluate the discriminative ability of a classifier.
- The **Area Under Curve (AUC)** values shown in the summary are very close to **1.0**, indicating that the classifier has high accuracy and strong class separation capability.



Viva Questions

1. What is the main difference between ID3 and J48 algorithms, and how do they construct decision trees?

ANS:_____

_____

2. What is the significance of entropy and information gain in building a decision tree classifier?

ANS:_____

_____

3,How do Naïve Bayes and k-Nearest Neighbour classifiers differ in terms of learning approach and assumptions?

ANS:_____

_____

4. What does the ROC curve represent, and how can you use it along with the confusion matrix to evaluate classifier performance?

ANS:_____

_____

_____

WEEK-4

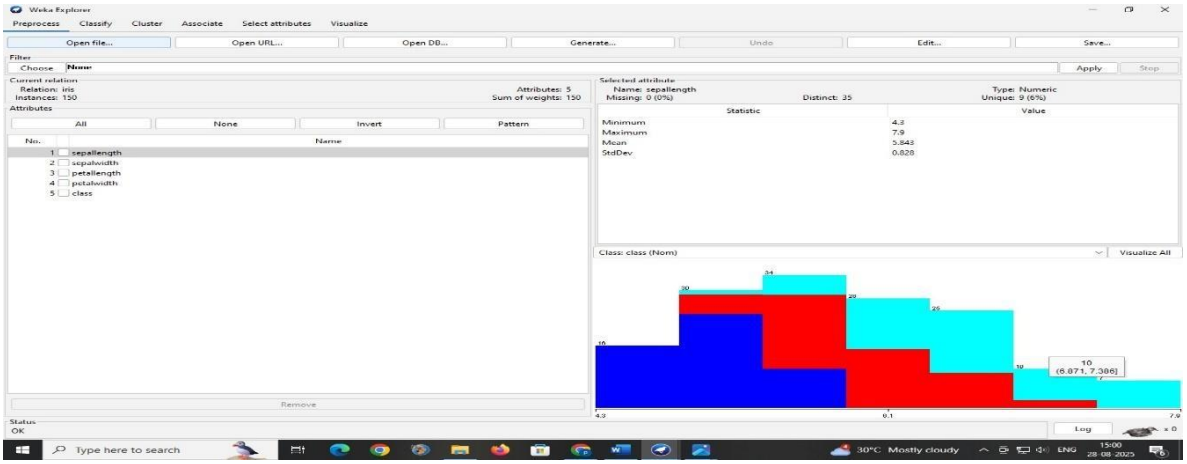
Aim:

Demonstrate performing clustering of datasets.To perform **clustering** on a given dataset using unsupervised learning techniques (such as **K-Means, Hierarchical**) in order to group the data into meaningful clusters based on similarities and patterns without prior class labels**.**

To Perform the clustering of datasets we have to follow these steps:

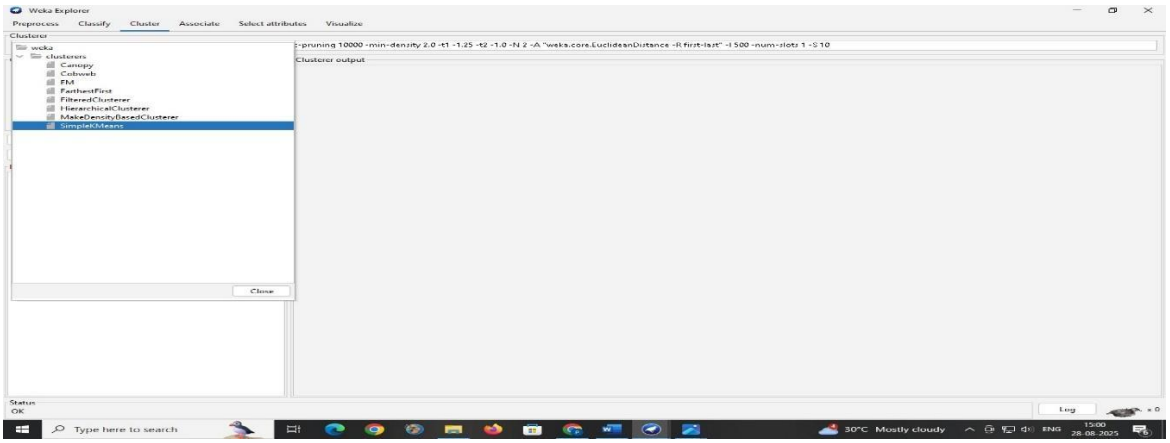
- ### Loading Dataset

- Open Weka Explorer → Preprocess panel.
- Load a dataset such as iris.arff or weather.numeric.arff.
- Weka displays the number of attributes and instances.
- Numeric datasets (like Iris) are best for clustering.

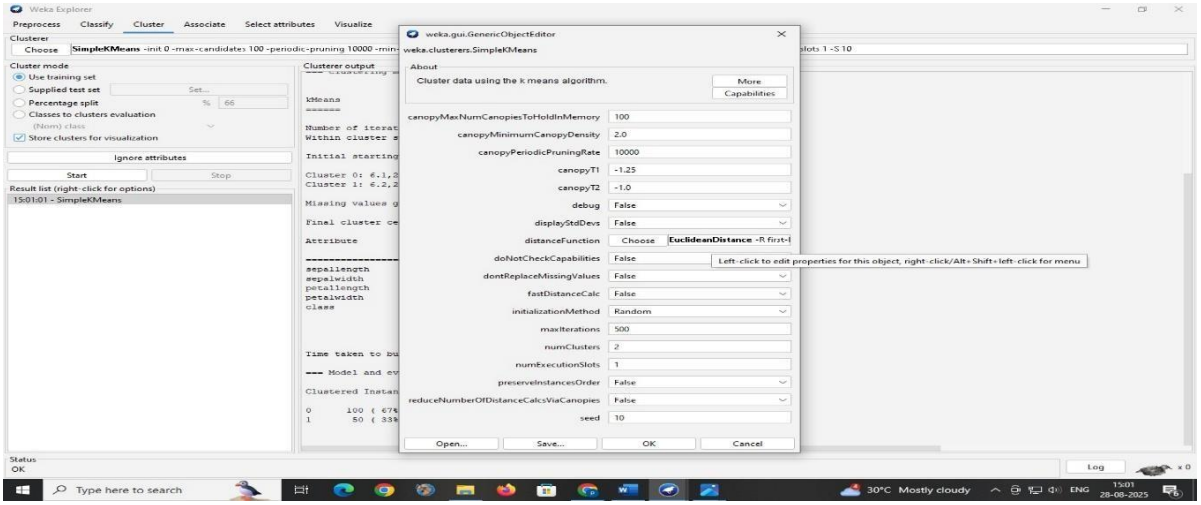


2. Running k-Means Clustering

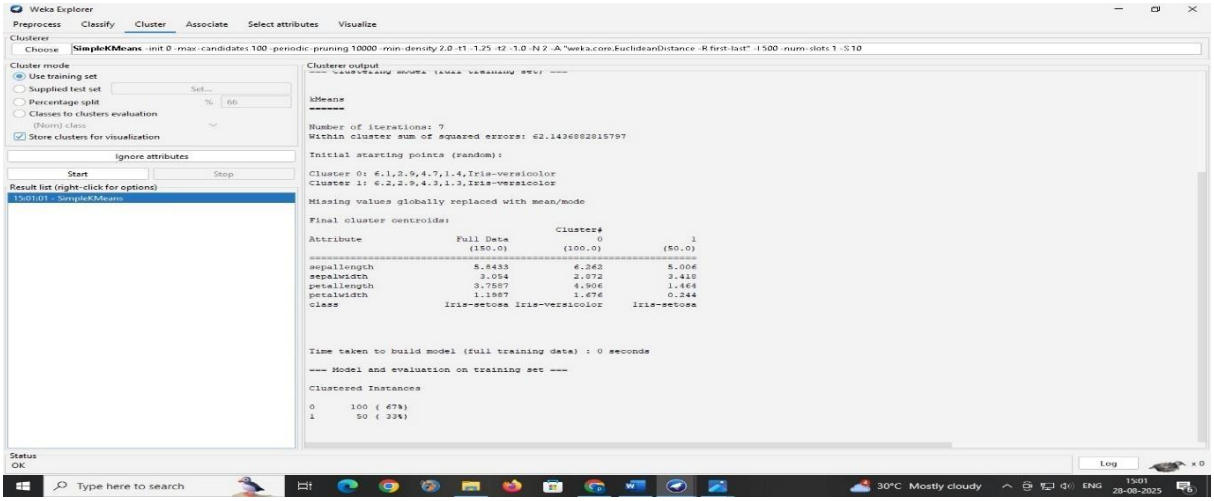
- Go to the Cluster panel in Weka Explorer.
- Select SimpleKMeans from the clustering algorithms.



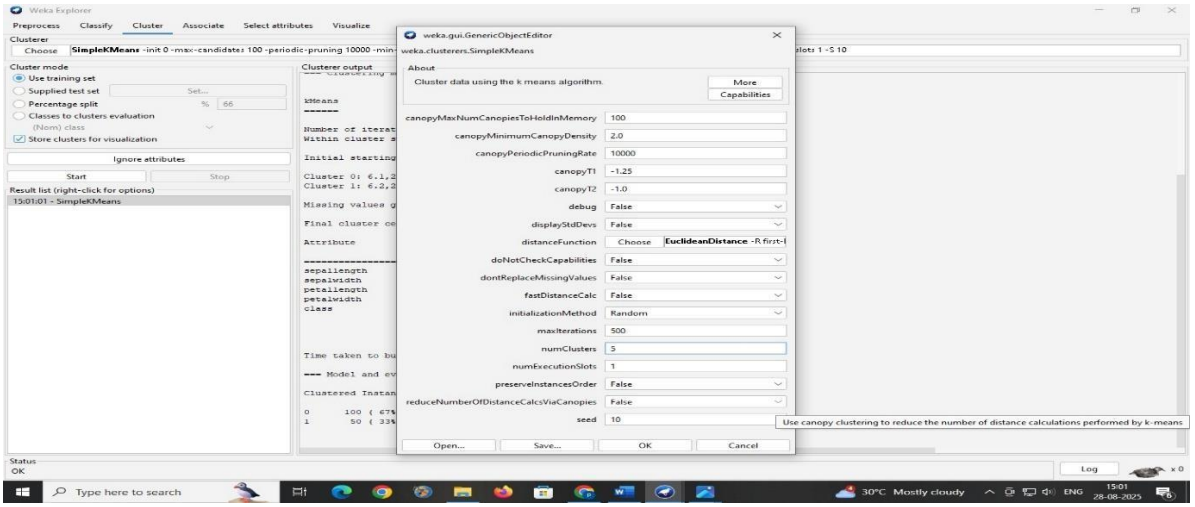
- Choose different values of k (e.g., 2, 3, 4) to test clustering.



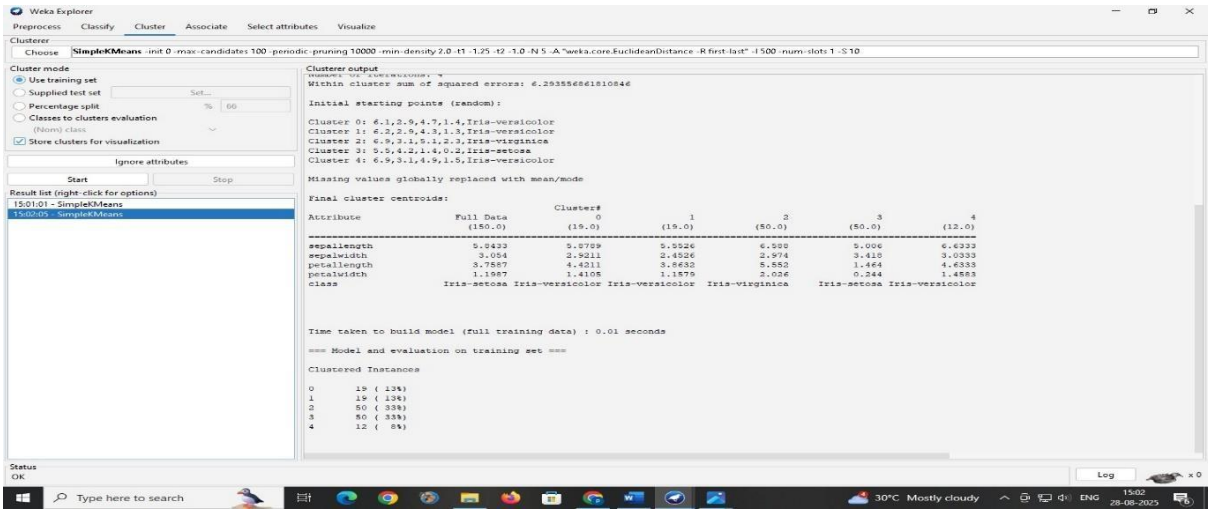
- Run the algorithm to generate cluster assignments.
- Output includes cluster centroids, number of instances per cluster, and Sum of Squared Errors (SSE).



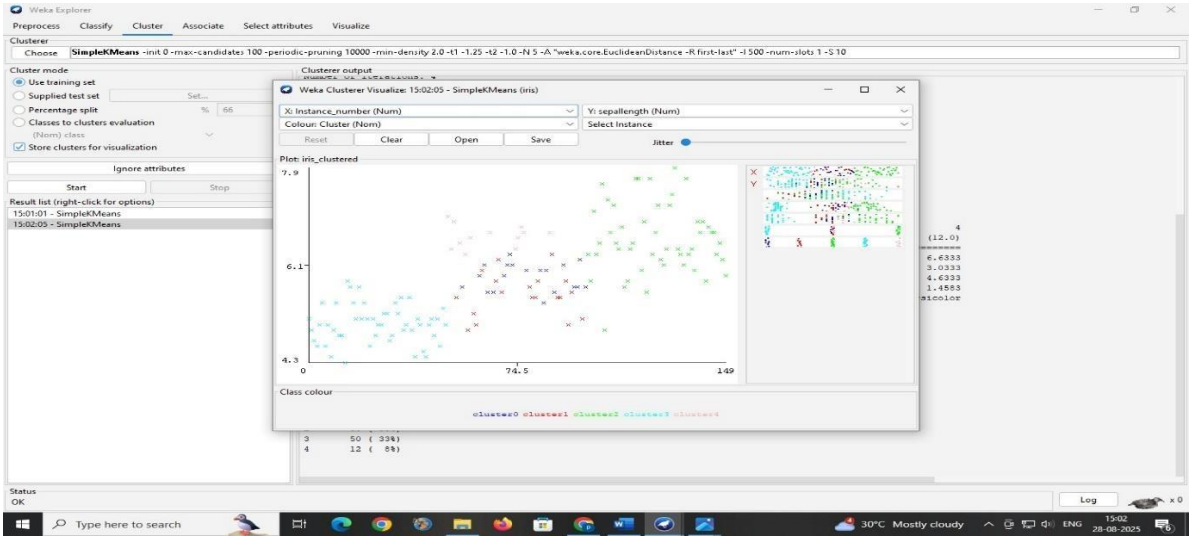
After Changing the values(numClusters:5):

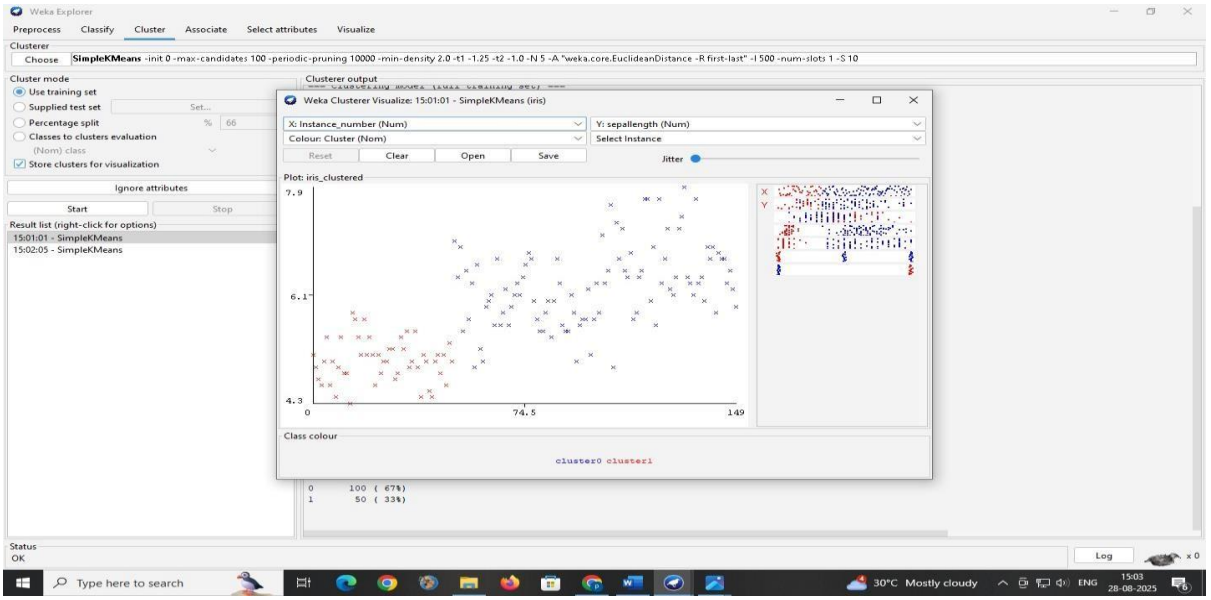


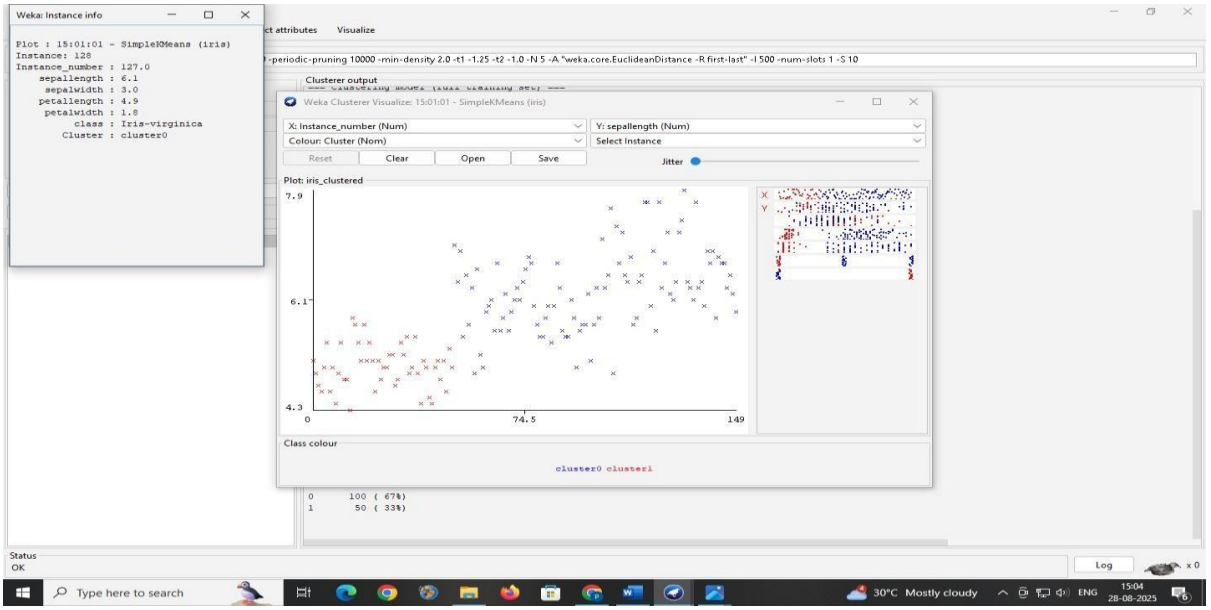
- Run the algorithm to generate cluster assignments.
- Output includes cluster centroids, number of instances per cluster, and Sum of Squared Errors (SSE).



- **Studying Cluster Results**
- For **iris dataset**, k=2 often matches the 3 flower species.
- he output shows **centroids**, which represent the mean values of attributes percluster.
- **SSE** measures how compact clusters are (lower is better).
- Misclassifications happen if clusters overlap in attribute space.
- Comparing clusters with the actual class attribute shows clustering accuracy.

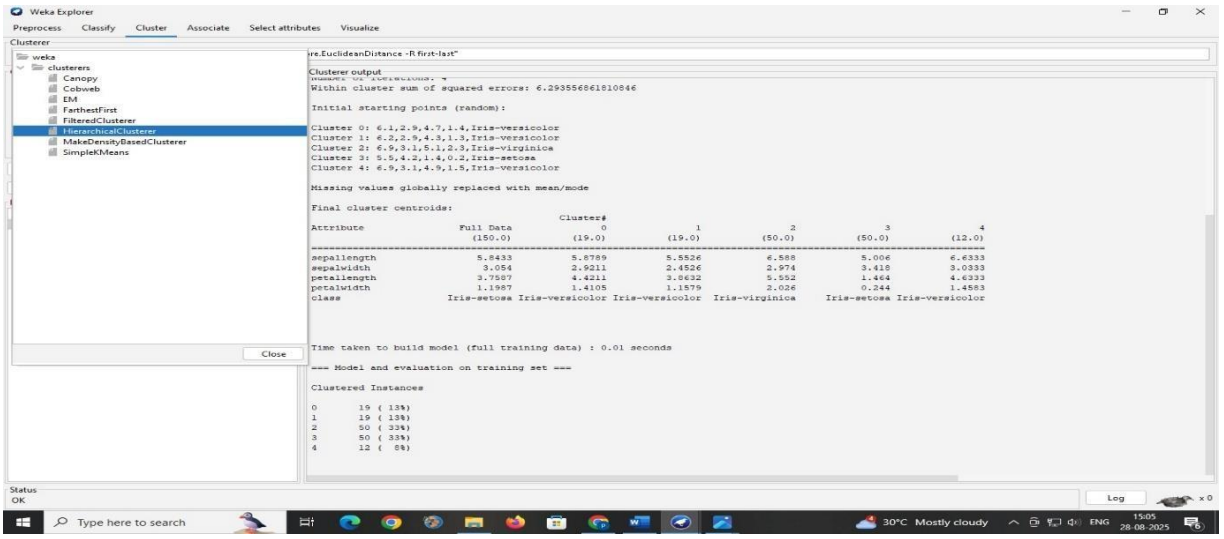




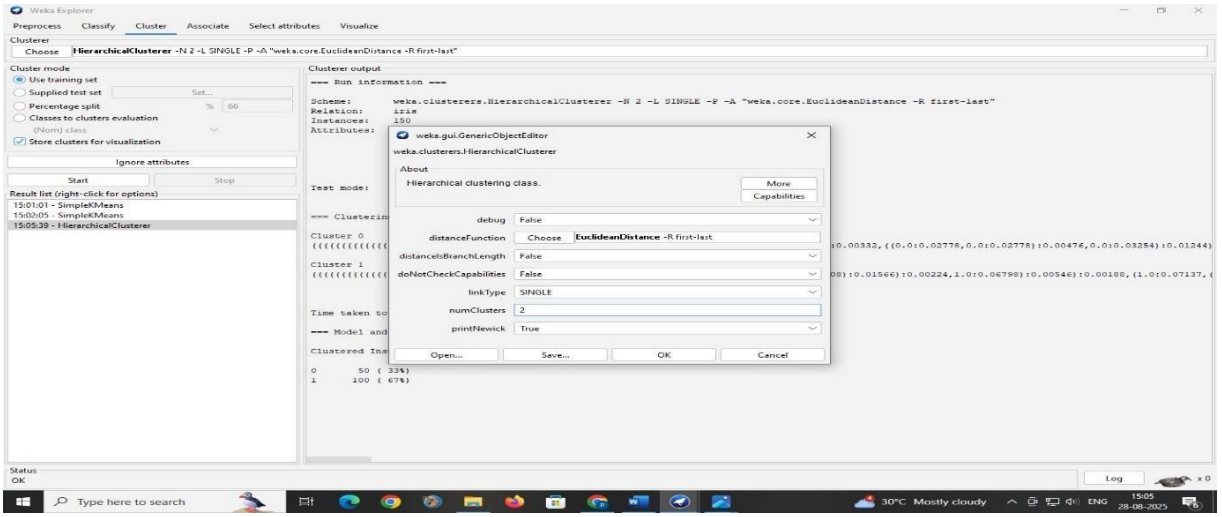


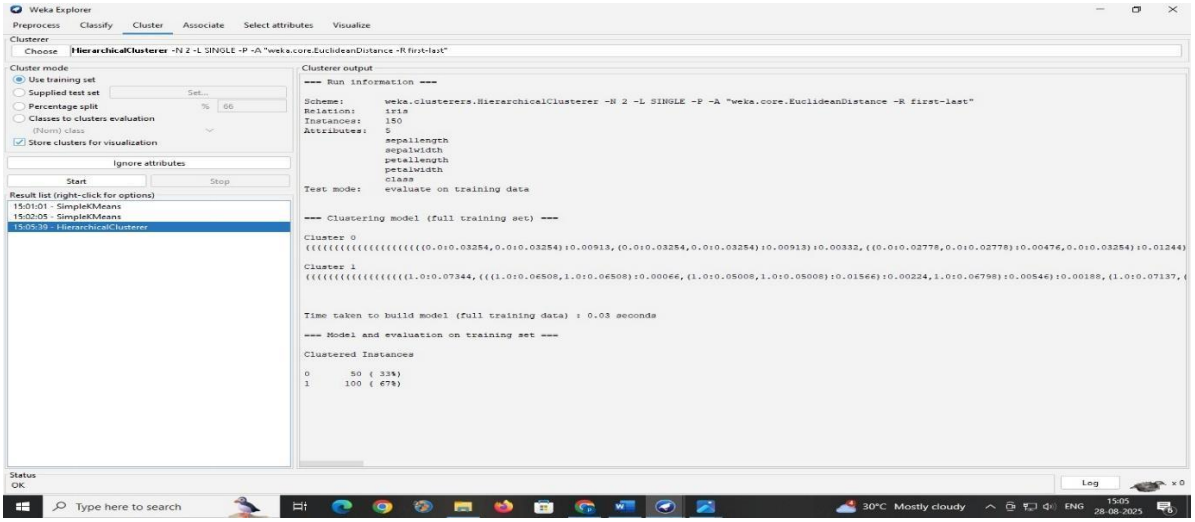
4. Exploring Other Clustering Techniques

- Weka provides other clustering algorithms like **EM (Expectation-Maximization)**, **Hierarchical clustering**, and **COBWEB**.
- EM automatically determines the number of clusters based on probability distribution.
- Hierarchical clustering builds a tree (dendrogram) of nested clusters.
- Different algorithms may produce different cluster structures.
- Comparing results helps understand which algorithm suits the dataset**.**

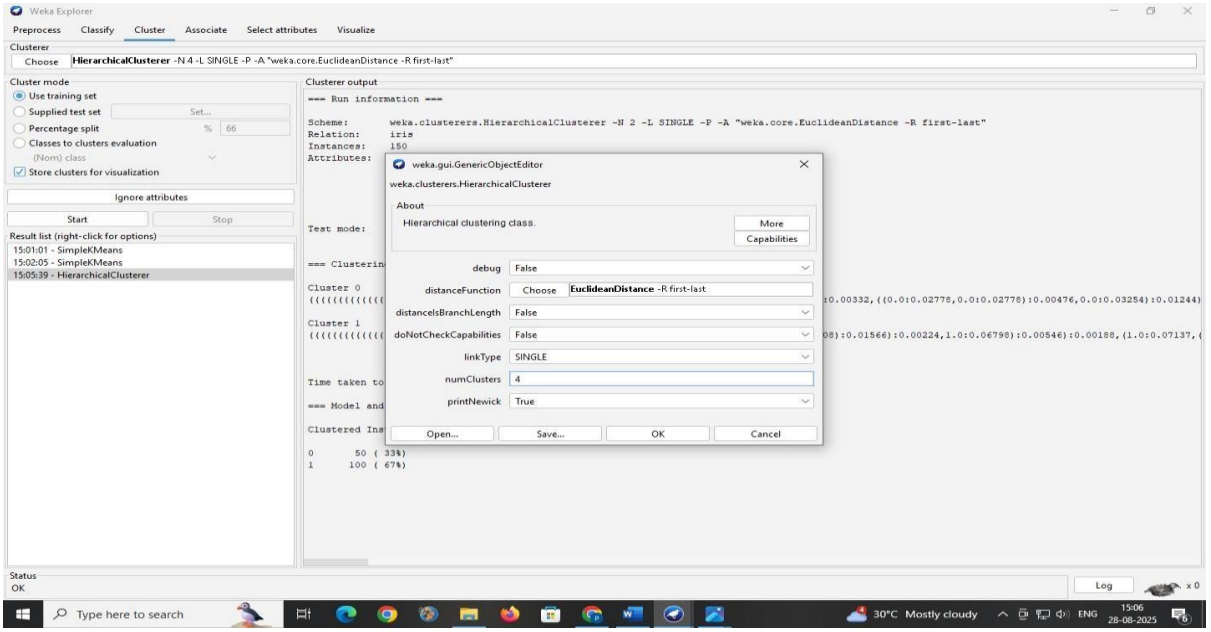


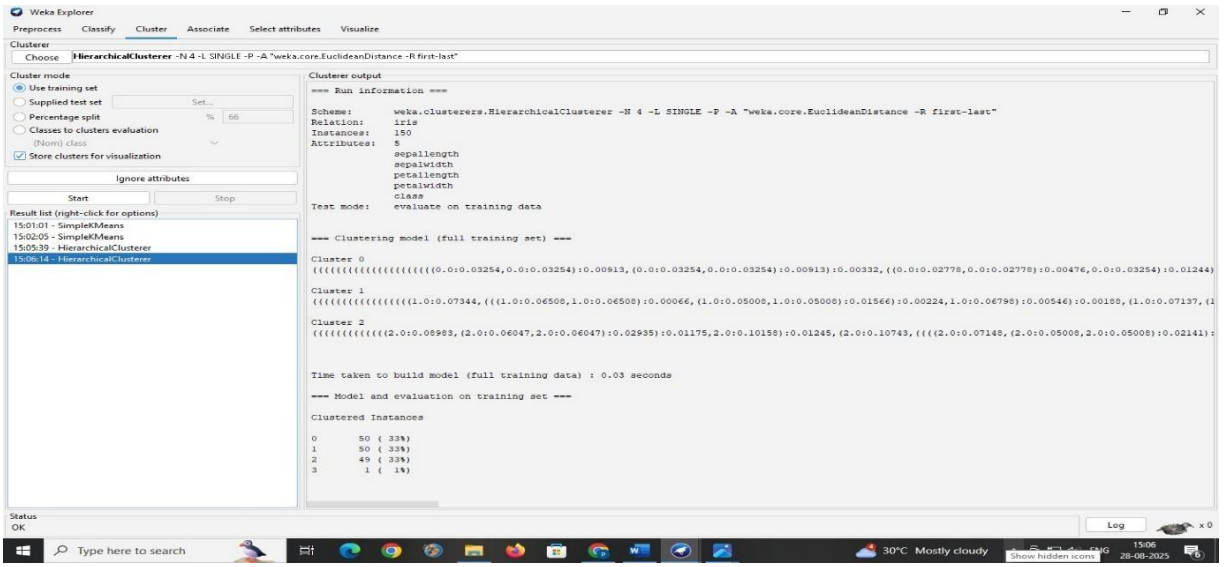
Before Changing the values(numClusters:2):





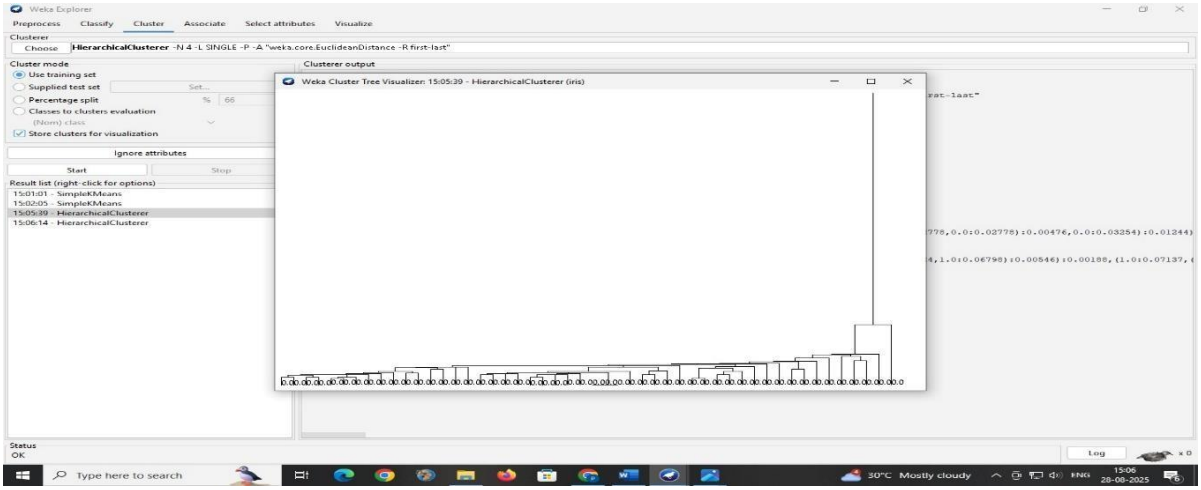
After Changing the values(numClusters:4):

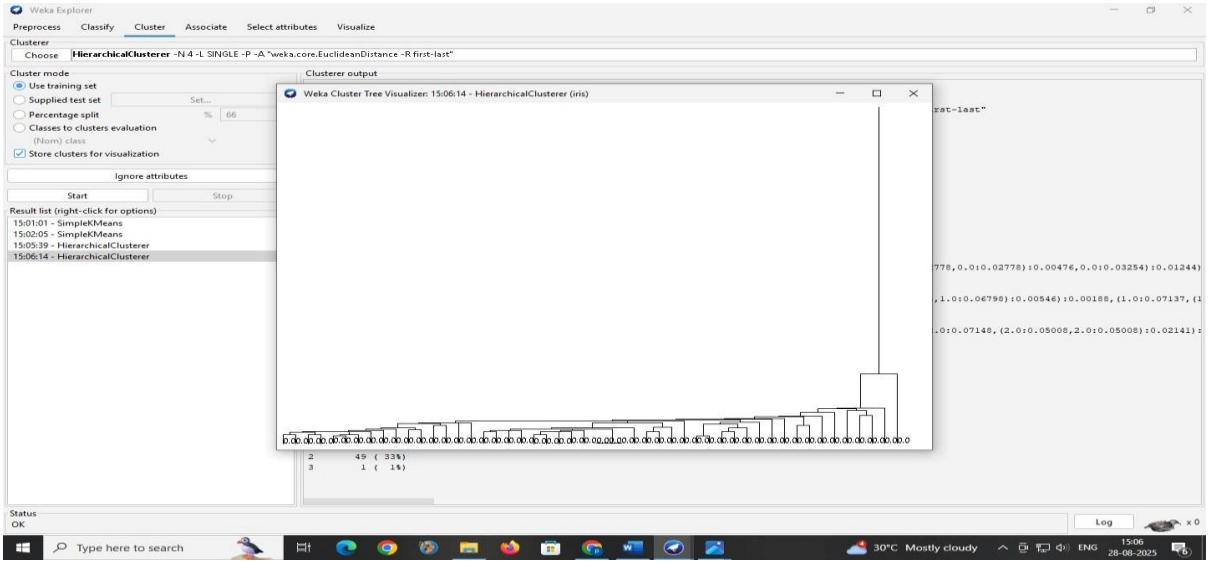




5. Visualizing Clusters & Insights

- Open the **Visualize panel** in Weka.
- Select attributes on X and Y axes to view **scatter plots of clusters**.
- Clusters are shown in different colors for clear separation.
- Well-separated datasets (like Iris) show clear clusters.
- Visualization confirms if chosen **k** correctly matches natural grouping.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
HierarchicalClusterer -N 4 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:01:01 - SimpleKMeans
15:02:05 - SimpleKMeans
15:05:39 - HierarchicalClusterer
15:06:14 - HierarchicalClusterer
Clusterer output
Weka Cluster Tree Visualizer: 15:05:39 - HierarchicalClusterer (iris)
Status
OK
Log
x 0
Type here to search
30°C Mostly cloudy
ENG
15:06
28-08-2025


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
HierarchicalClusterer -N 4 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:01:01 - SimpleKMeans
15:02:05 - SimpleKMeans
15:05:39 - HierarchicalClusterer
15:06:14 - HierarchicalClusterer
Clusterer output
Weka Cluster Tree Visualizer: 15:06:14 - HierarchicalClusterer (iris)
2 49 (33%)
3 1 (1%)
Status
OK
Log
x 0
Type here to search
30°C Mostly cloudy
ENG
15:06
28-08-2025

Viva Questions

1. **What is clustering, and how is it different from classification?**

2. **Explain how the k-means algorithm works and how the value of 'k' affects the clustering results.**

3. **What is the significance of the sum of squared errors (SSE) in clustering? How do you interpret it?**

4. **What are centroids, and why are they important in k-means clustering?**

5. **Can you name and explain at least two other clustering algorithms available in Weka or R besides k-means?**

Week-5

5. Demonstrate knowledge flow application on data sets into Weka/R

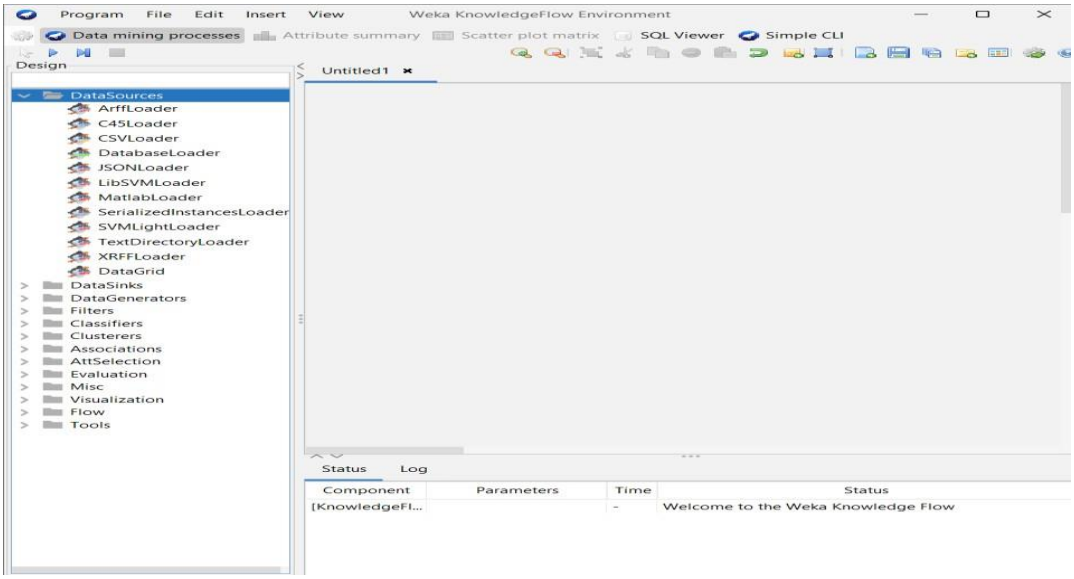
- Develop a knowledge flow layout for finding strong association rules by using Apriori, FP Growth algorithms
- Set up the knowledge flow to load an ARFF (batch mode) and perform a cross validation using J48 algorithm
- Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree

Develop a knowledge flow layout for finding strong association rules by using Apriori, FP Growth algorithms

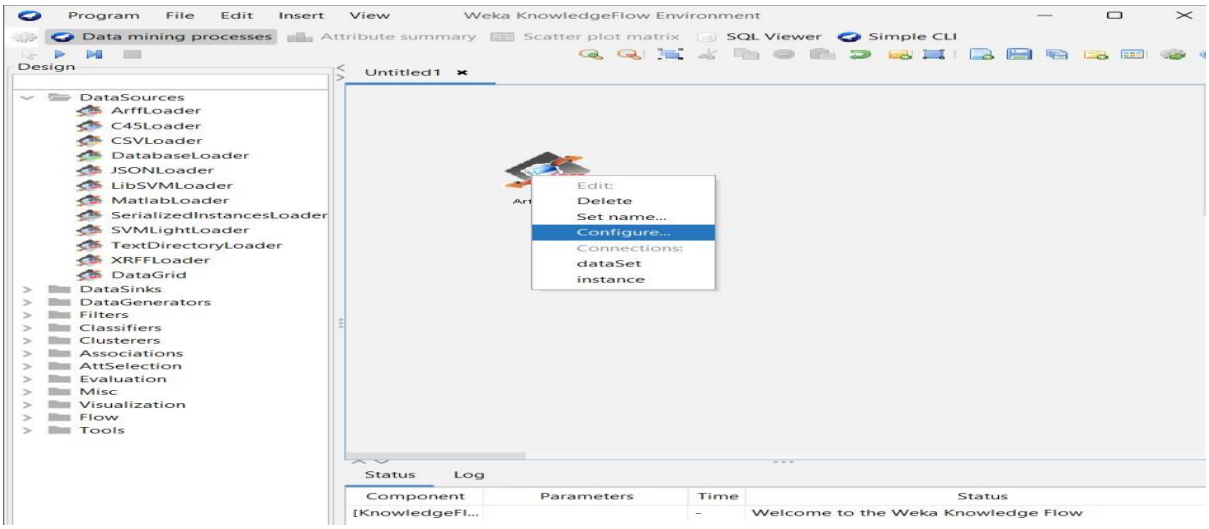
1. Open Knowledge Flow

Go to Tools → Knowledge Flow in Weka.

2. Load the dataset



- From the **Design** panel click **DataSources** → **ArffLoader** and click on the canvas to place it.
- Right-click the ArffLoader → **Configure** → browse and select your ARFF file.



3. From **Design** → **Filters** → **unsupervised** → **attribute** choose **Discretize** and place it on the canvas.

- Right-click **ArffLoader** → choose **dataSet** connection → click the Discretize node to connect `dataSet` → Discretize.
- Configure **Discretize** (right-click → Configure): pick number of bins or use default.

4. From **Design** → **Associators** pick **Apriori** and place it on the canvas. Also add **FPGrowth** (both under Associators).

- Right-click the Discretize (or ArffLoader if no filter) → select **dataSet** → click Apriori (connect dataset to Apriori).
- Repeat to connect same dataset to FPGrowth (so both get the same input in parallel).

5. Add two **Visualization** → **TextViewer** nodes (one per associator) to capture textual output.

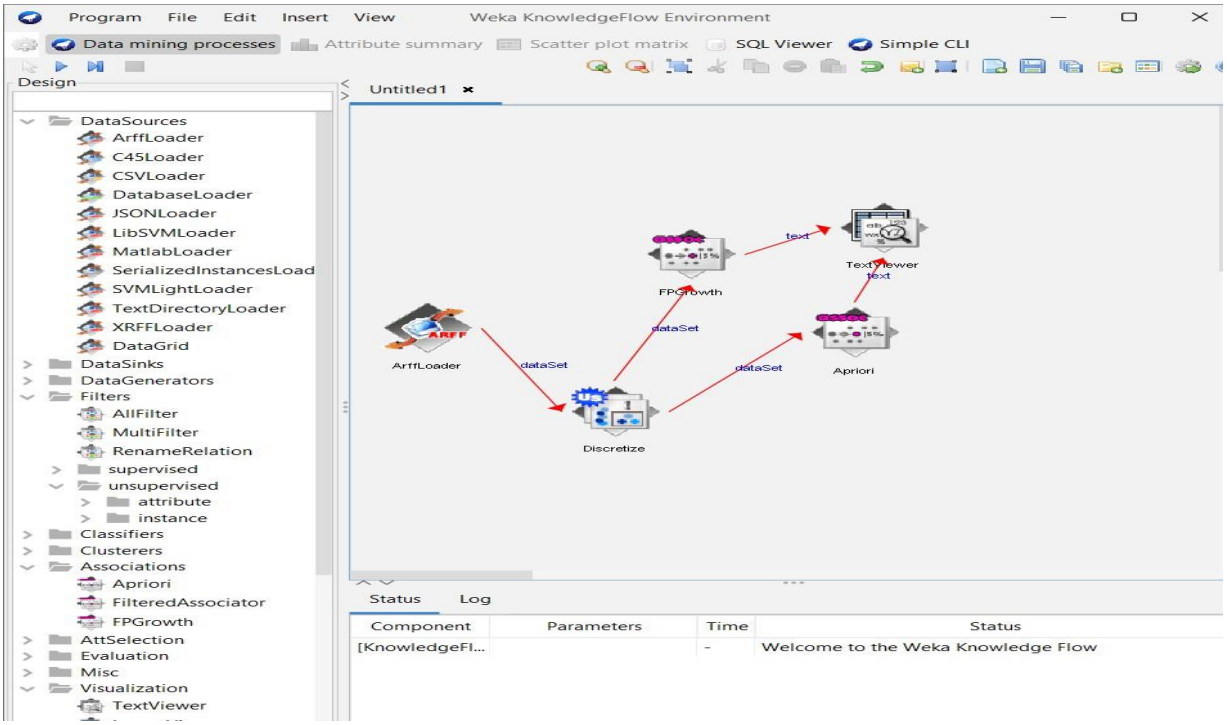
- Right-click Apriori → select **text** → click the TextViewer to connect.
- Right-click FPGrowth → select **text** → connect to the other TextViewer.

6. **Configure Apriori** (right-click → Configure):

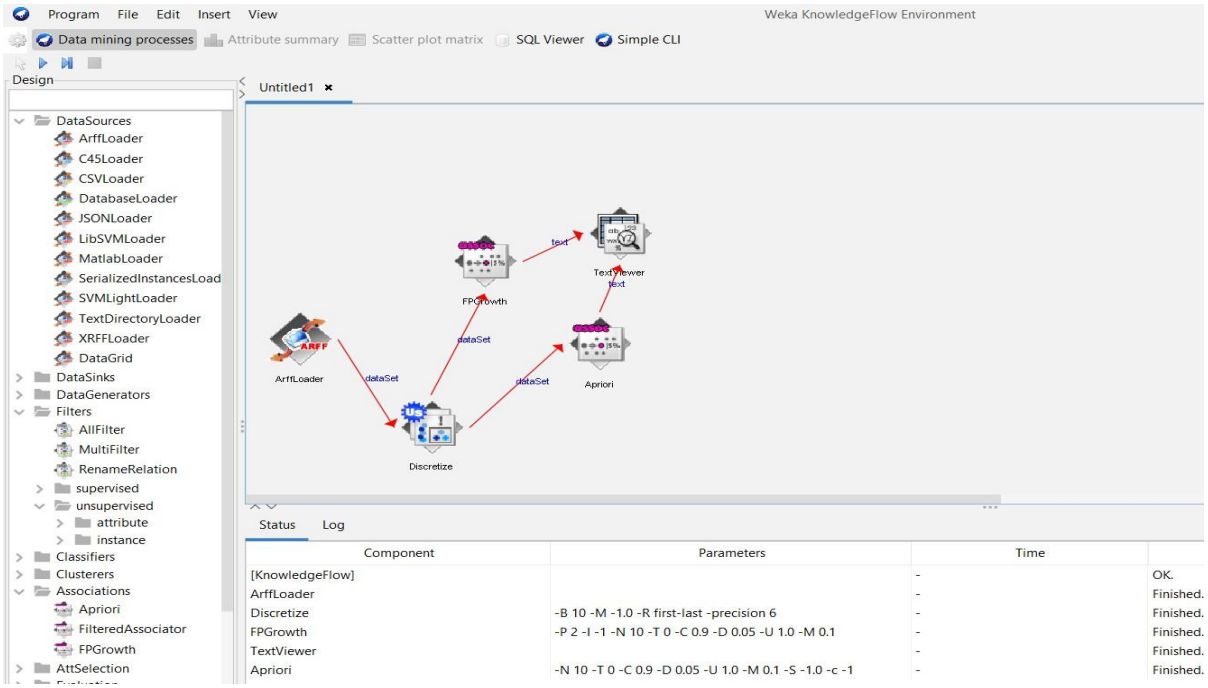
Useful options: **numRules** (-N) = how many rules to output (default 10), **minMetric** (-C) = min confidence (default 0.9)

7. **Configure FPGrowth** (right-click → Configure):

It has options for **minSupport** and **numRules** too — for dense datasets set minSupport higher; for sparse datasets lower it. Remember FPGrowth finds frequent itemsets.

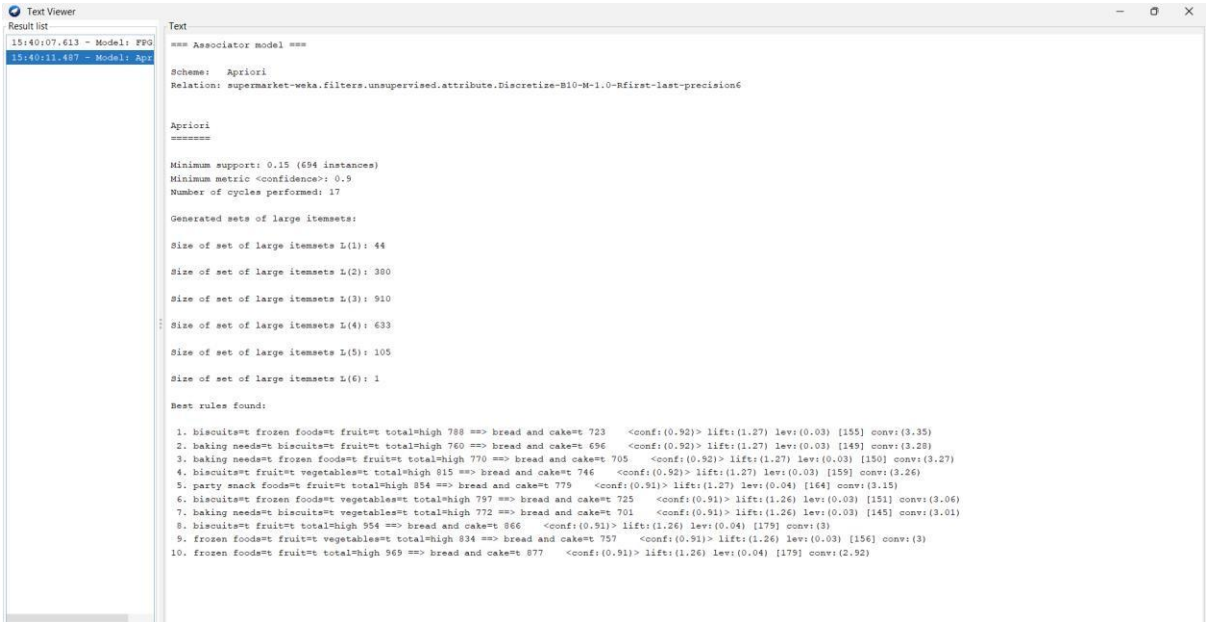


8. Press the **play** (Execute flow) button on the toolbar.

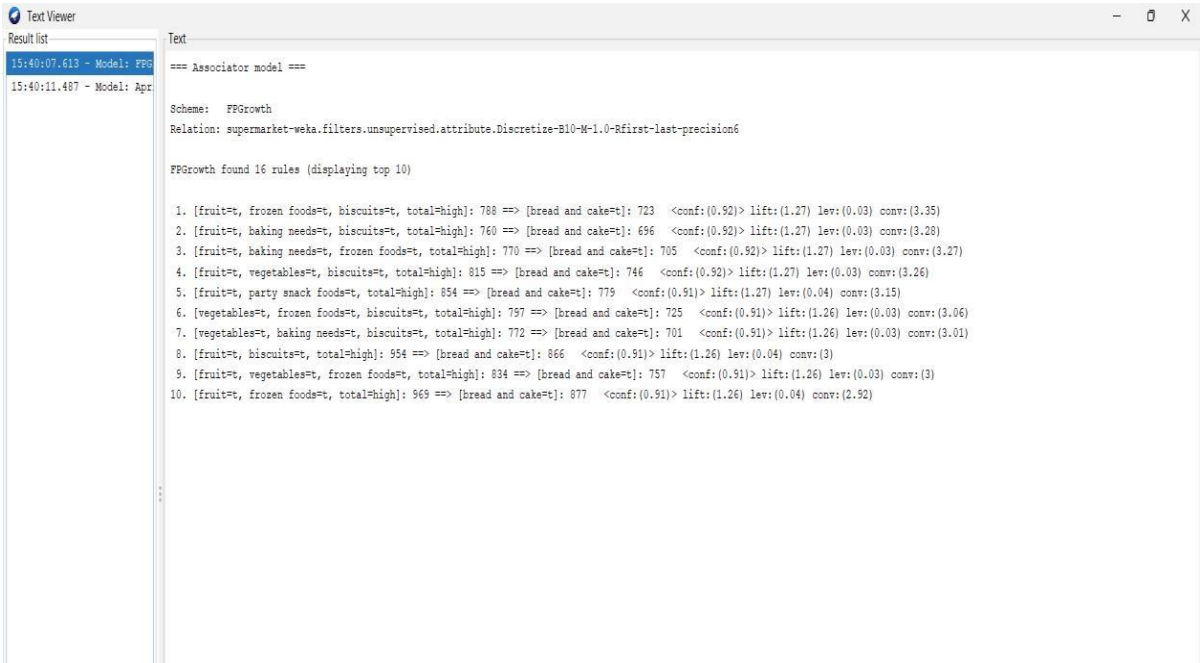


9. When finished, right-click each **TextViewer** → **Show results** to see the list of frequent itemsets and association rules. Interpret the rules by support and confidence.

Apriori will list **frequent itemsets** and then **rules** with metrics (support, confidence).



FPGrowth will list frequent itemsets and rules similarly



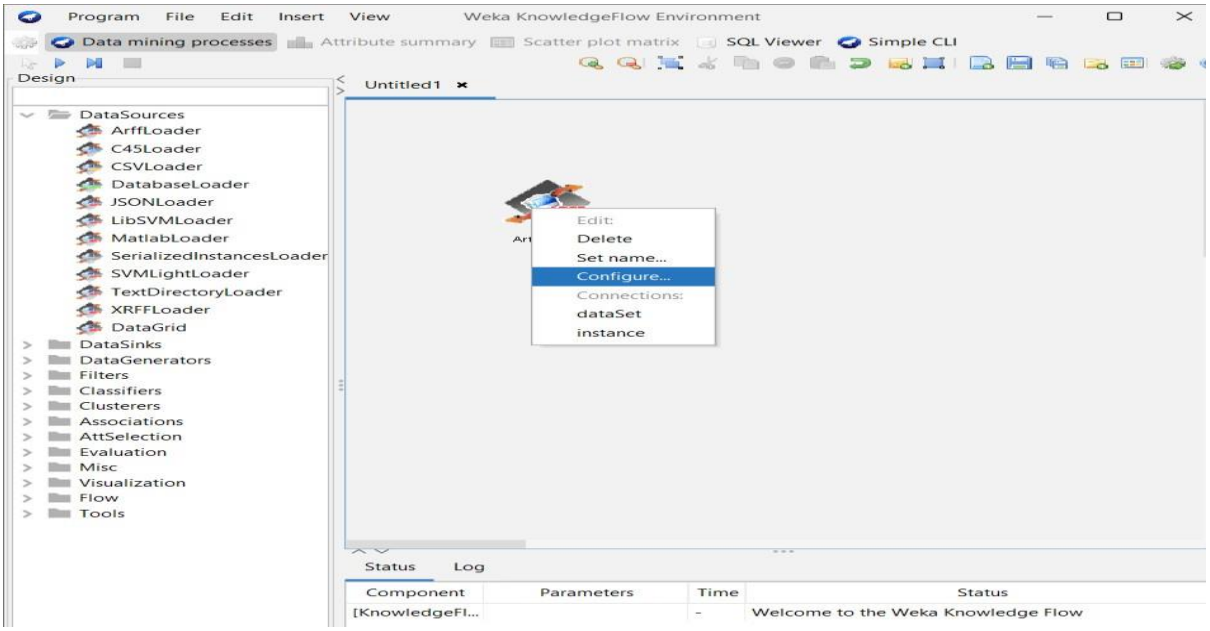
Set up the knowledge flow to load an ARFF (batch mode) and perform a cross validation using J48 algorithm

1. 1.Open Knowledge Flow

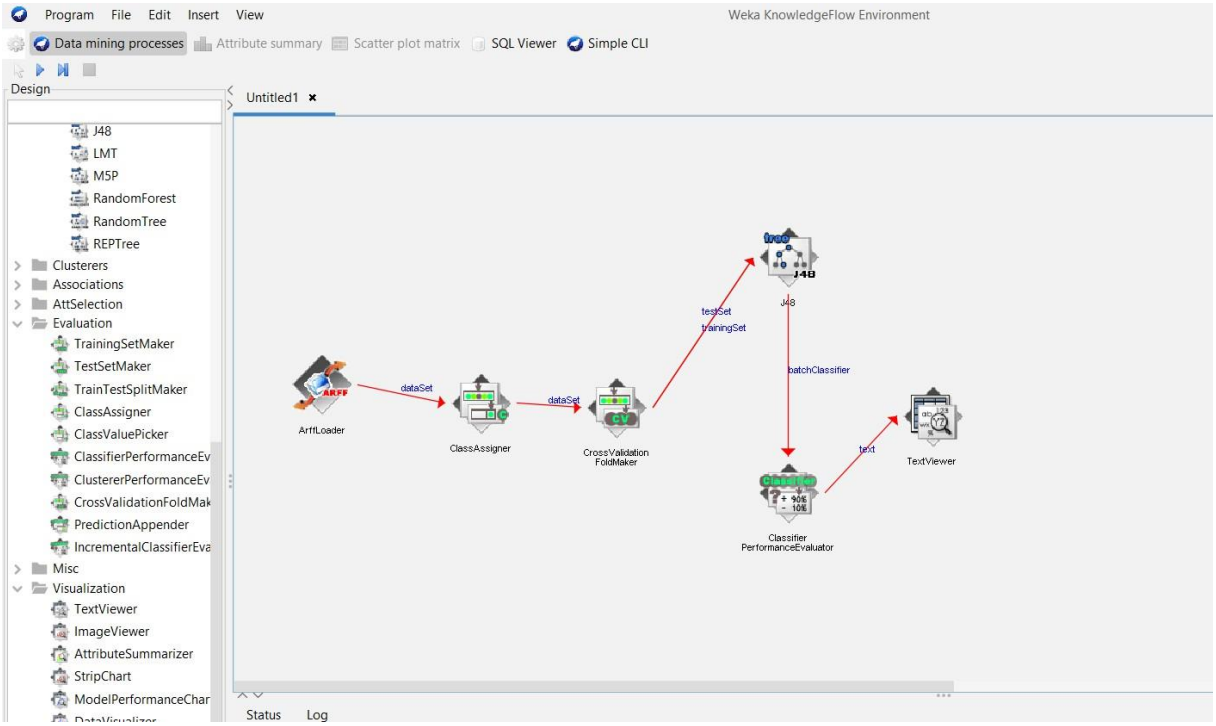
Go to Tools → Knowledge Flow in Weka.

2. Load the dataset.

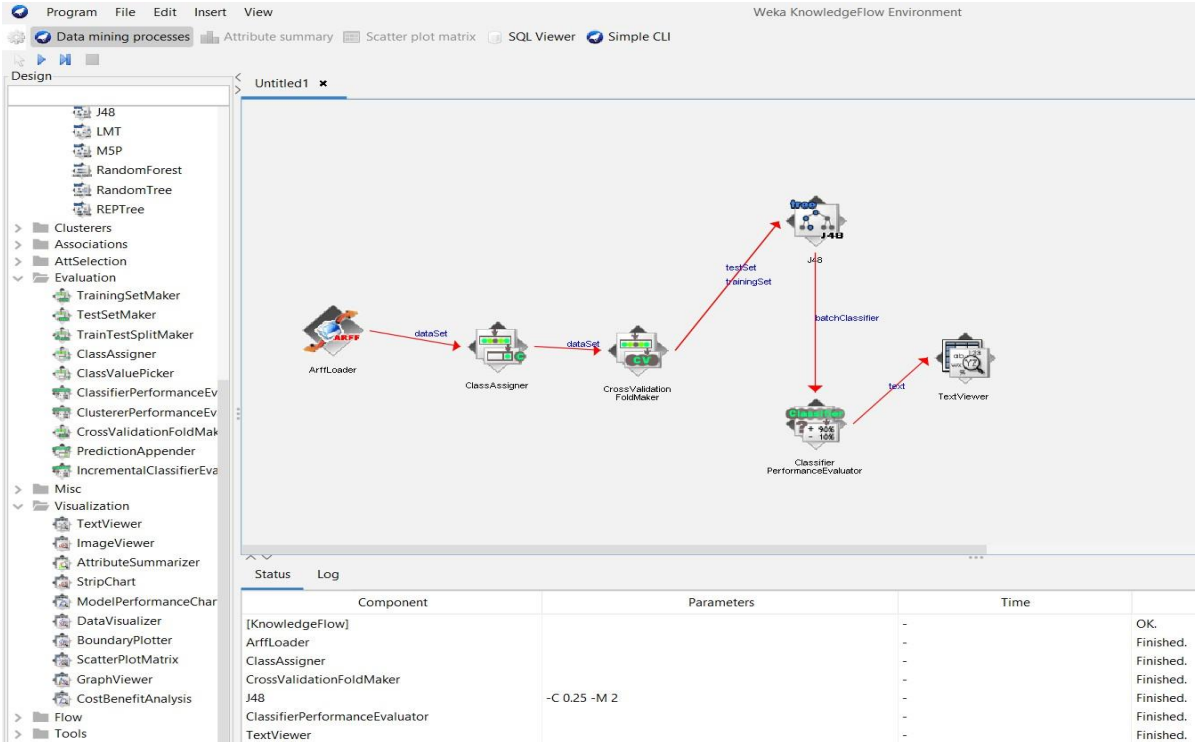
- From the **Design** panel click **DataSources** → **ArffLoader** and click on the canvas to place it.
- Right-click the ArffLoader → **Configure** → browse and select your ARFF file.



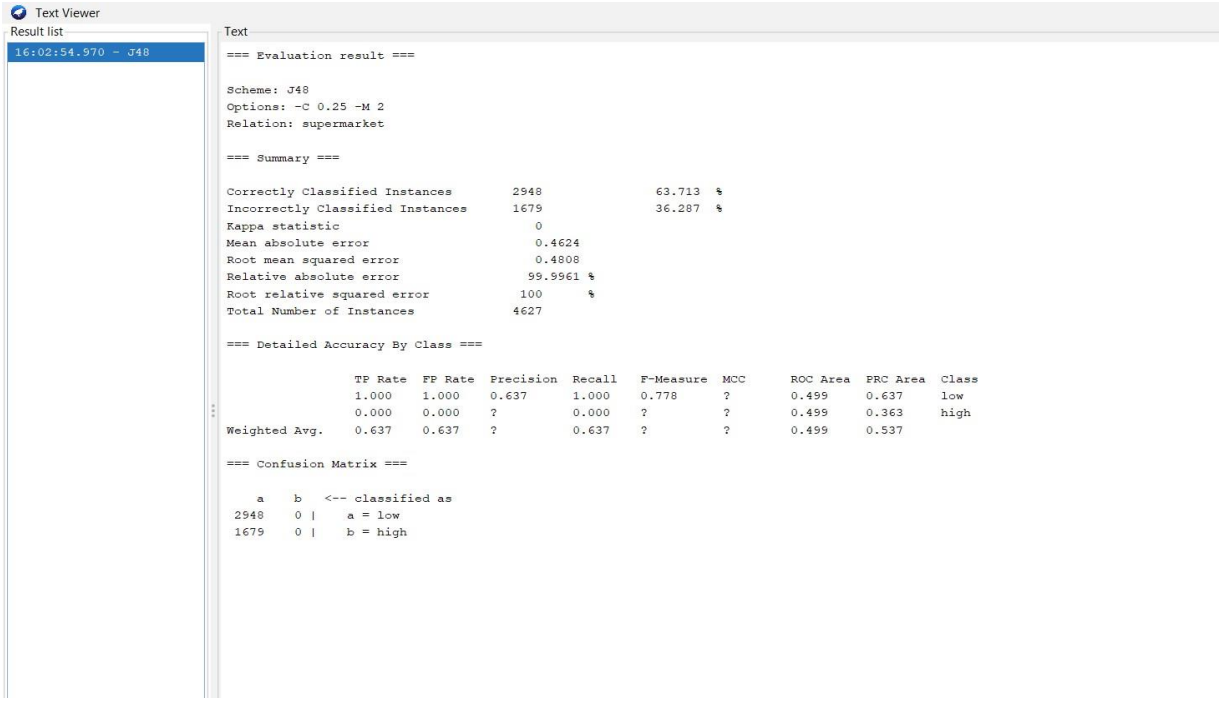
3. **Evaluation** → **ClassAssigner** → place on canvas.
    - Right-click ArffLoader → select **dataSet** → click ClassAssigner to connect.
    - Right-click ClassAssigner → **Configure** → set which attribute is the class (default last).
4. **Evaluation** → **CrossValidationFoldMaker** → place on canvas.
    - Right-click ClassAssigner → **dataSet** → click CrossValidationFoldMaker to connect.
    - Configure CrossValidationFoldMaker if you want different number of folds (default 10).
5. **Classifiers** → **trees** → **J48** → place on canvas.
    - Right-click CrossValidationFoldMaker → choose **trainingSet** → click J48 (connect).
    - Again right-click CrossValidationFoldMaker → choose **testSet** → click J48 (this creates two connections: trainingSet C testSet).
6. **Evaluation** → **ClassifierPerformanceEvaluator** → place on canvas.
    - Right-click J48 → select **batchClassifier** → click ClassifierPerformanceEvaluator to connect.
7. **Visualization** → **TextViewer** → place on canvas.
    - Right-click ClassifierPerformanceEvaluator → select **text** → connect to TextViewer.



8. Press the **play** (Execute flow) button on the toolbar.



9. Right-click the **TextViewer** → **Show results**. You will see cross-validation summary, fold results, confusion matrix, per-class metrics and global metrics (accuracy, kappa, etc.).



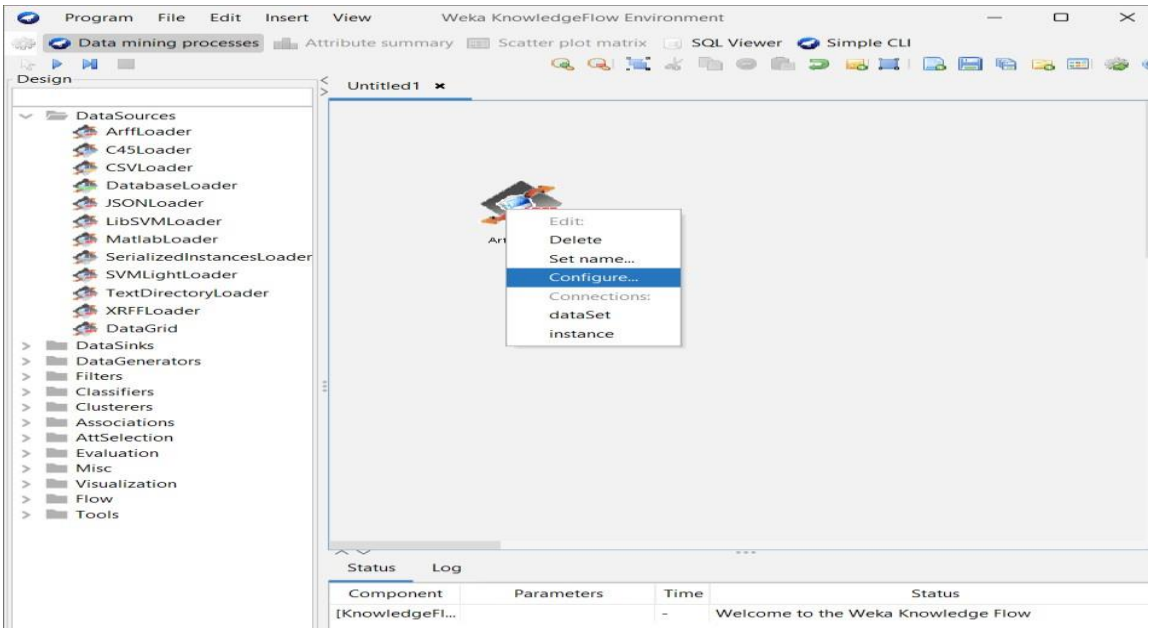
Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree

1. Start a fresh KnowledgeFlow canvas.

Go to Tools → Knowledge Flow in Weka.

2. Load the dataset.

- From the **Design** panel click **DataSources** → **ArffLoader** and click on the canvas to place it.
- Right-click the ArffLoader → **Configure** → browse and select your ARFF file.



3. **Evaluation** → **ClassAssigner** → place on canvas.

- Right-click ArffLoader → select **dataSet** → click ClassAssigner to connect.
- Right-click ClassAssigner → **Configure** → set which attribute is the class (default last).

4. **Evaluation** → **CrossValidationFoldMaker** → place on canvas.

- Right-click ClassAssigner → **dataSet** → click CrossValidationFoldMaker to connect.
- Configure CrossValidationFoldMaker if you want different number of folds (default 10).

5. **Classifiers** → **trees** → **J48** → place on canvas.

- Right-click CrossValidationFoldMaker → choose **trainingSet** → click J48 (connect).
- Again right-click CrossValidationFoldMaker → choose **testSet** → click J48 (this creates two connections: trainingSet C testSet).

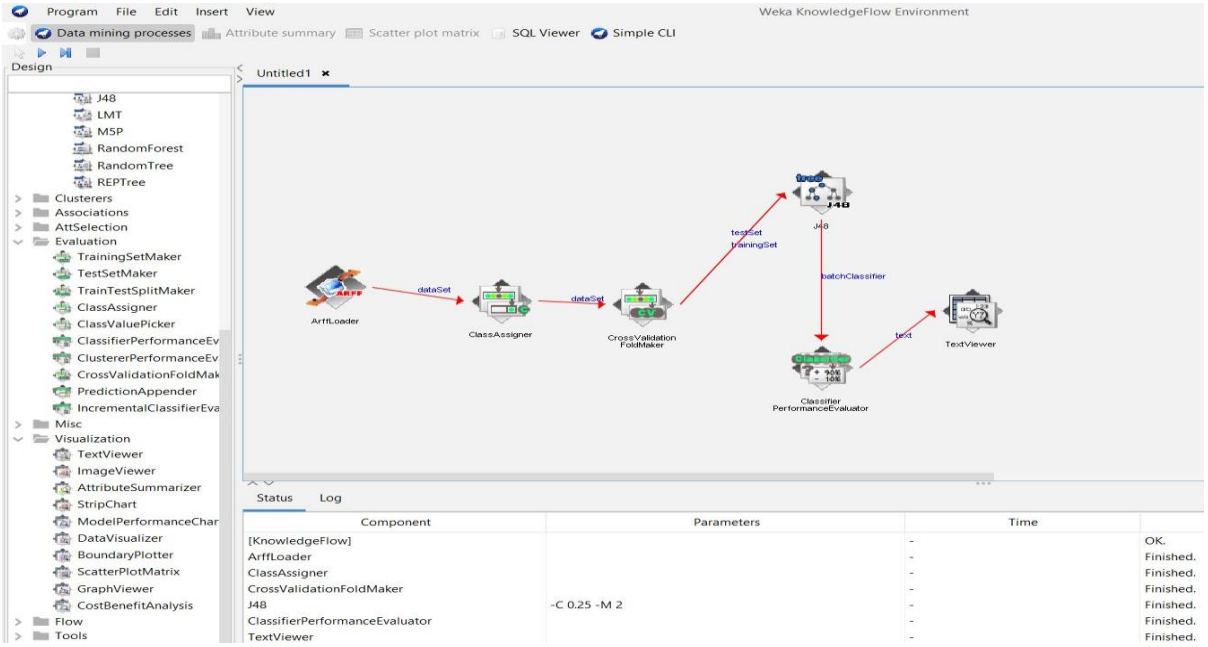
6. **Evaluation** → **ClassifierPerformanceEvaluator** → place on canvas.

Right-click J48 → select **batchClassifier** → click ClassifierPerformanceEvaluator to connect.

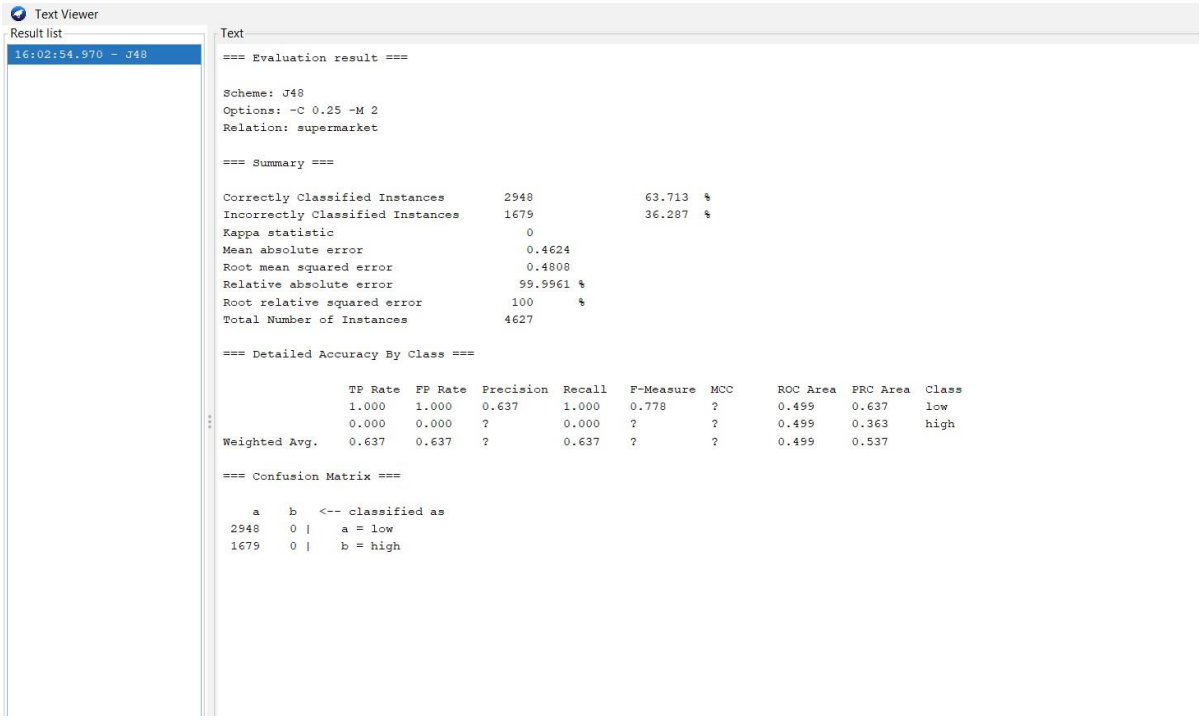
7. **Visualization** → **TextViewer** → place on canvas.

Right-click ClassifierPerformanceEvaluator → select **text** → connect to TextViewer.

8. Press the **play** (Execute flow) button on the toolbar.



9. Right-click the **TextViewer** → **Show results**. You will see cross-validation summary, fold results, confusion matrix, per-class metrics and global metrics (accuracy, kappa, etc.).



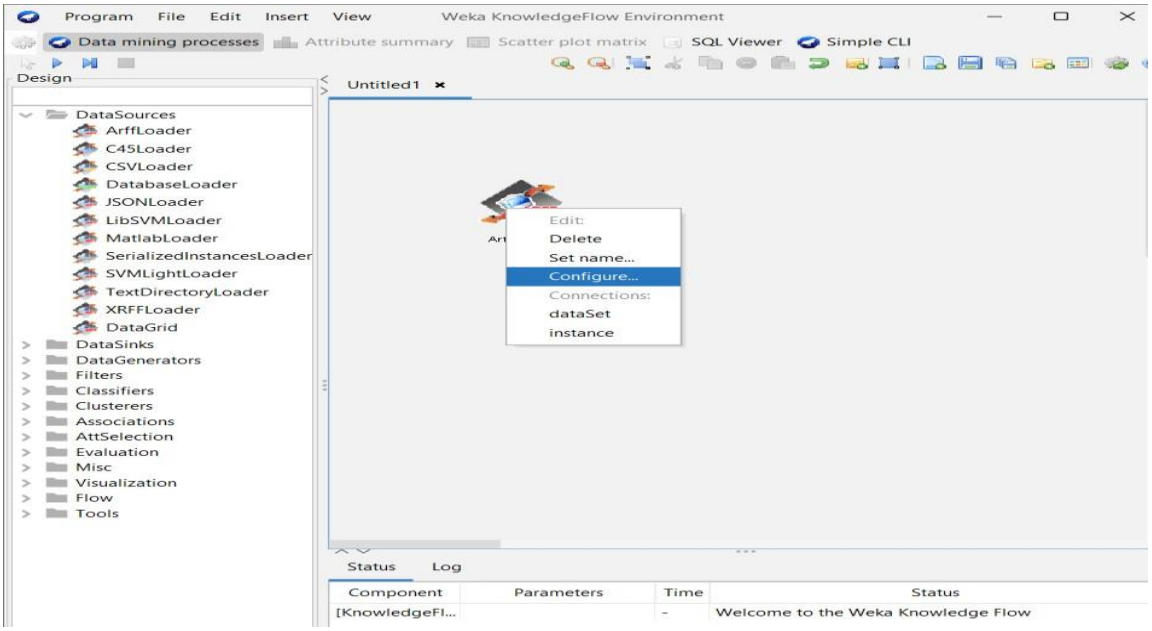
Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree

1. Start a fresh KnowledgeFlow canvas.

Go to Tools → Knowledge Flow in Weka.

2. Load the dataset.

From the **Design** panel click **DataSources** → **ArffLoader** and click on the canvas to place it.



Right-click the ArffLoader → **Configure** → browse and select your ARFF file.(eg.,vote.arff)

3. **Evaluation**→ **ClassAssigner** → place on canvas.

- Right-click ArffLoader → select **dataSet** → click ClassAssigner to connect.

4. **Evaluation**→ **CrossValidationFoldMaker** → place on canvas.

- Right-click ClassAssigner → select **dataSet** → click CrossValidationFoldMaker to connect.

5. **Classifiers** → **trees** → **J48** → place on canvas.

Right-click **CrossValidationFoldMaker** → **trainingSet** → **J48**.

Right-click **CrossValidationFoldMaker** → **testSet** → **J48**.

6. **Classifiers** → **trees** → **RandomForest** → place on canvas.

Right-click **CrossValidationFoldMaker** → **trainingSet** → **RandomForest**.

Right-click **CrossValidationFoldMaker** → **testSet** → **RandomForest**.

7. **Evaluation** → **ClassifierPerformanceEvaluator**, place on canvas.

Right-click **J48** → **batchClassifier** → **ClassifierPerformanceEvaluator**.

8. Add another ClassifierPerformanceEvaluator.

Evaluation → **ClassifierPerformanceEvaluator**, place on canvas.

Right-click **RandomForest** → **batchClassifier** → **ClassifierPerformanceEvaluator**.

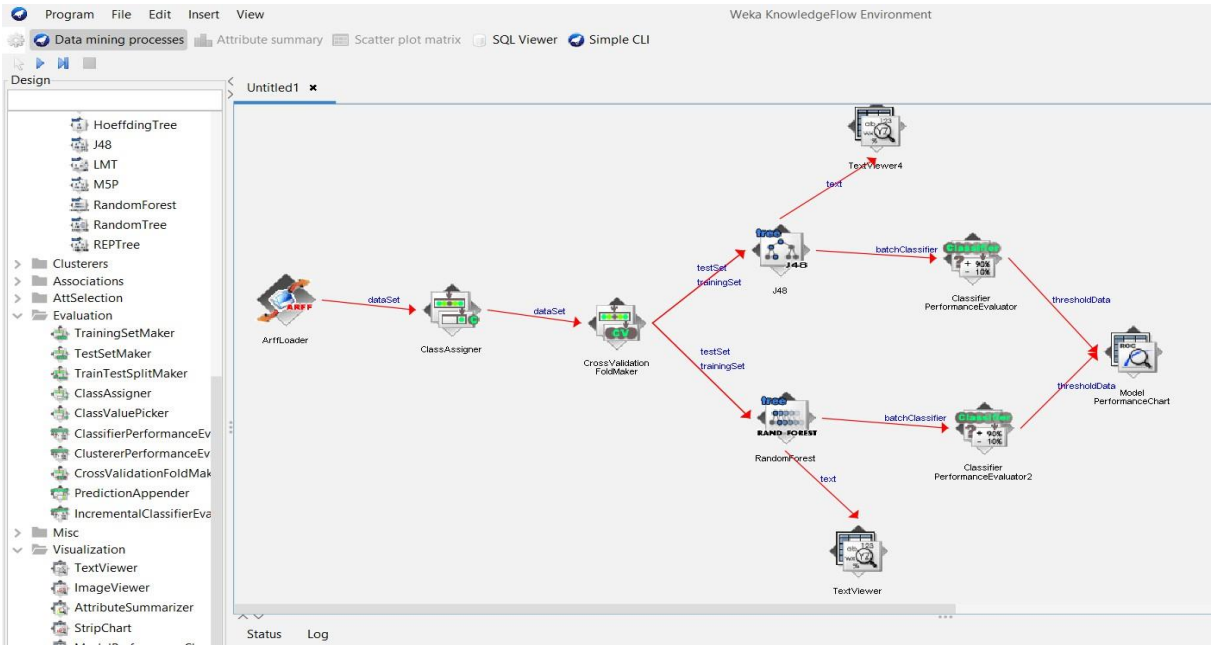
9. **Add ModelPerformanceChart and connect thresholdData**

- From **Visualization** → **ModelPerformanceChart**, place on canvas.
- Right-click **ClassifierPerformanceEvaluator (1)** → **thresholdData** → **click ModelPerformanceChart**.
- Right-click **ClassifierPerformanceEvaluator (2)** → **thresholdData** → **click the same ModelPerformanceChart**

10. **Visualization** → **TextViewer**, place **two** TextViewer nodes.

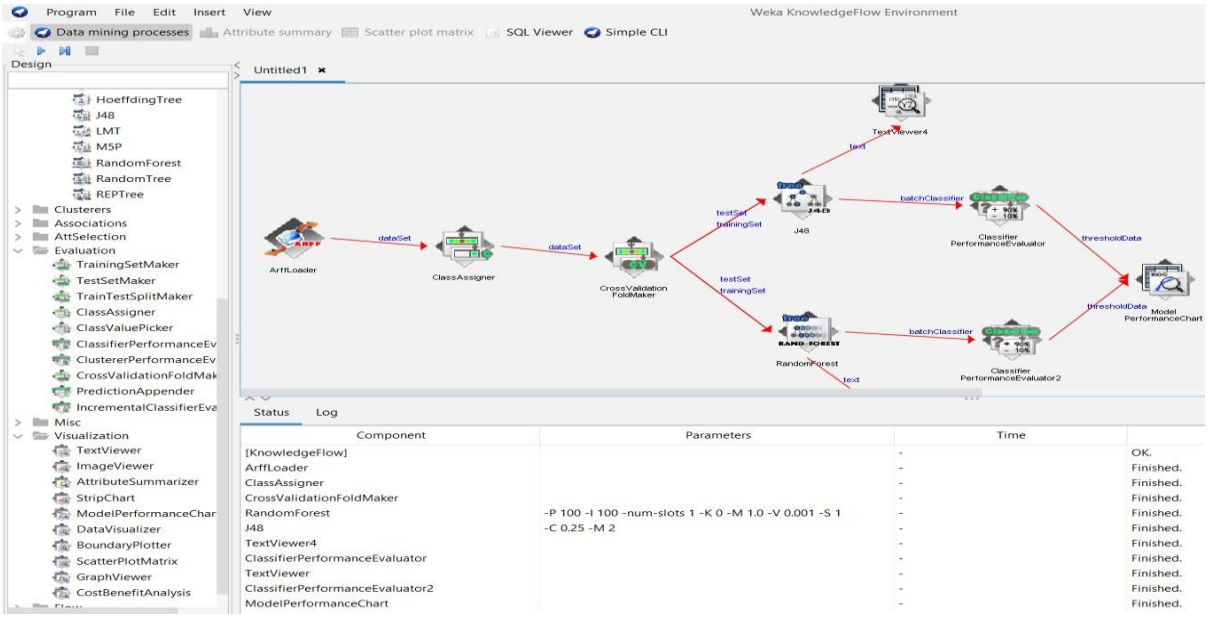
Right-click **J48** → **text** → **TextViewer (1)**.

Right-click **RandomForest** → **text** → **TextViewer (2)**.



8. Press the **play** (Execute flow) button on the toolbar.

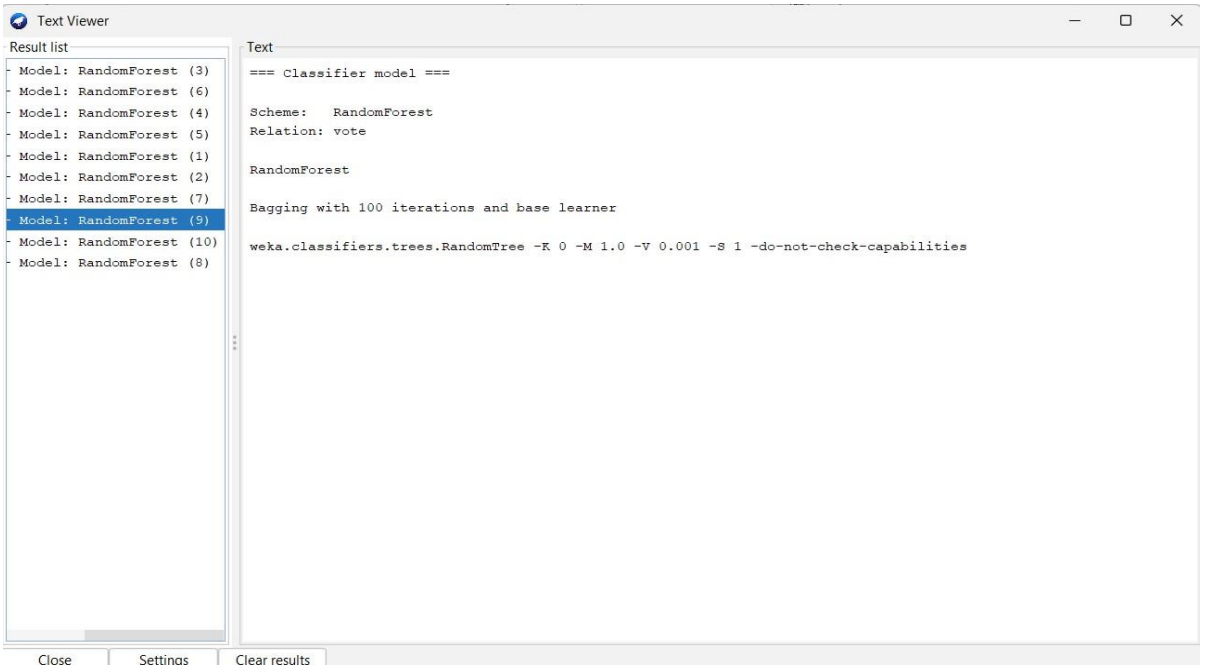
Wait until nodes (ArffLoader, evaluators, TextViewers, ModelPerformanceChart) finish / turn green.



Right-click **TextViewer (J48) → Show results**.

- You will see accuracy, confusion matrix, and **AUC for J48**.

Right-click **TextViewer (RandomForest) → Show results**.

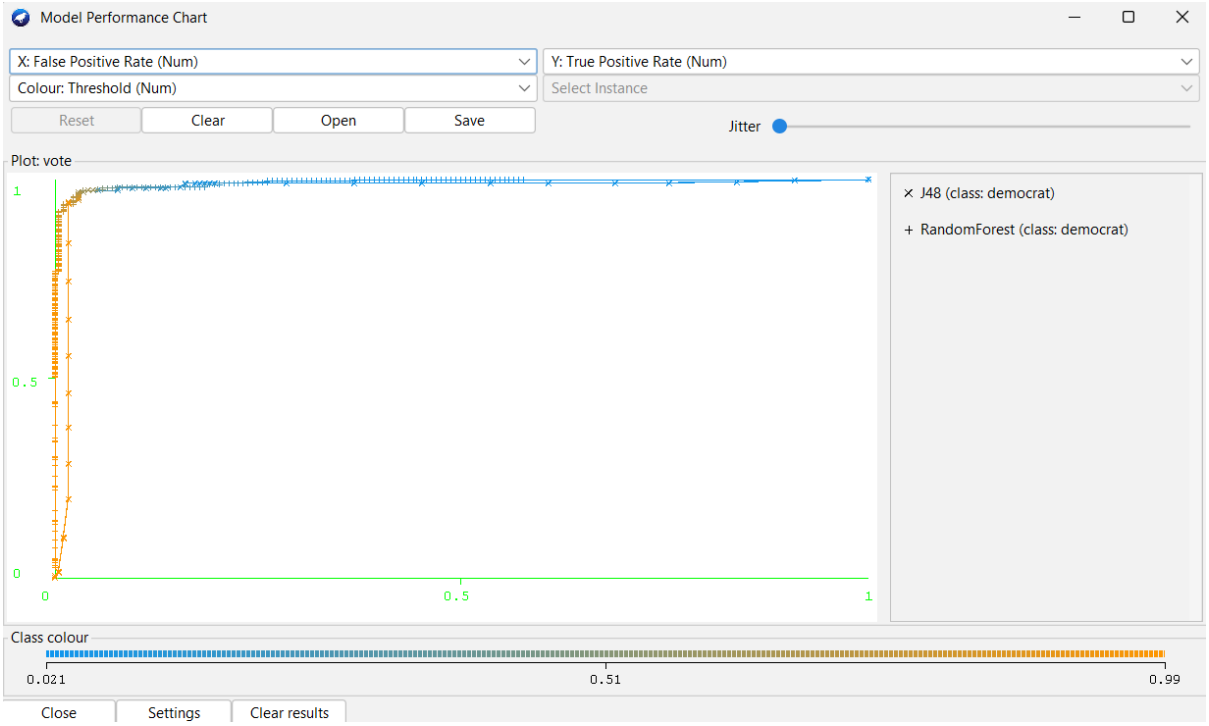


You will see accuracy, confusion matrix, and **AUC for RandomForest**.

Right-click on the **ModelPerformanceChart node**.

- Select **Show plot** (or sometimes **Show chart** depending on version).
- A window will appear showing **two ROC curves**:
- **J48 ROC curve** (in one color, e.g., blue).

  RandomForest ROC curve (in another color, e.g., red).



Viva Questions

1. **What is the purpose of using Knowledge Flow in Weka compared to the Explorer interface?**

ANS:_____

_____

2. **How do Apriori and FP-Growth algorithms differ in generating association rules, and when would you prefer one over the other?**

ANS:_____

_____

3. **Why is cross-validation used in J48 classification, and how does it improve model evaluation?**

ANS:_____

4. **What does it mean when two ROC curves (J48 vs Random Forest) overlap or diverge, and how do you interpret their performance from the same plot window?**

ANS:_____

_____

Week-6

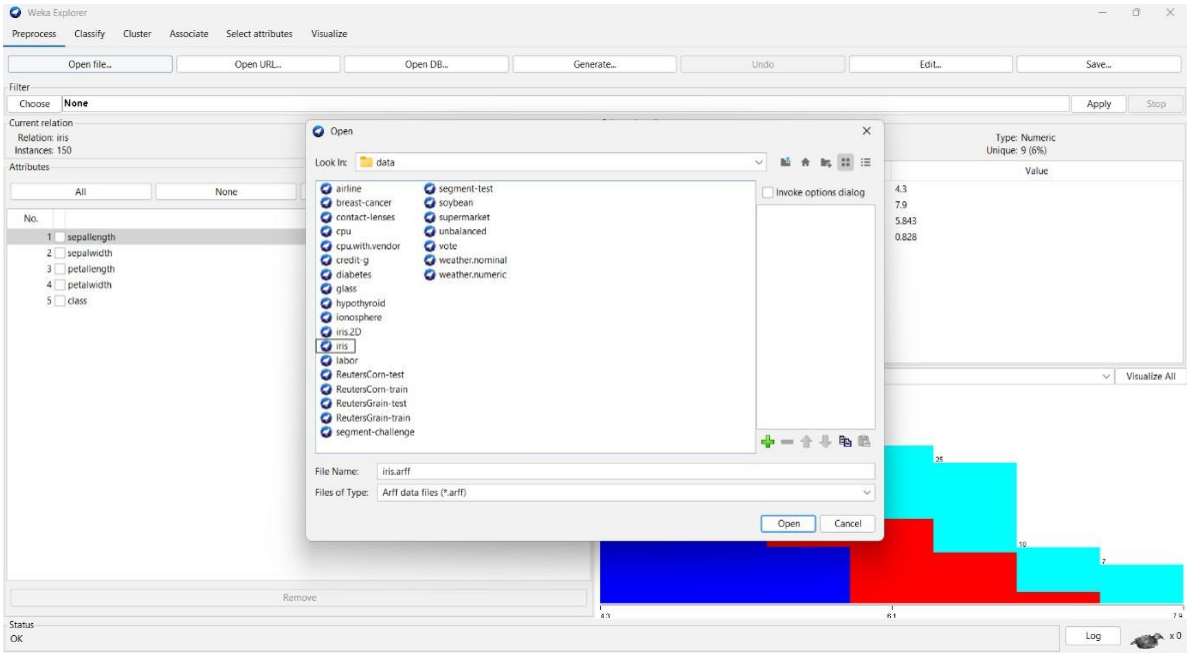
6. Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observation

AIM: Using the ZeroR technique on the Iris dataset by using the necessary preprocessing techniques.

1. Explorer Interface

- The **Explorer** is the primary graphical user interface of WEKA.
- It allows users to load datasets, preprocess data, select algorithms, build models, and evaluate results interactively.

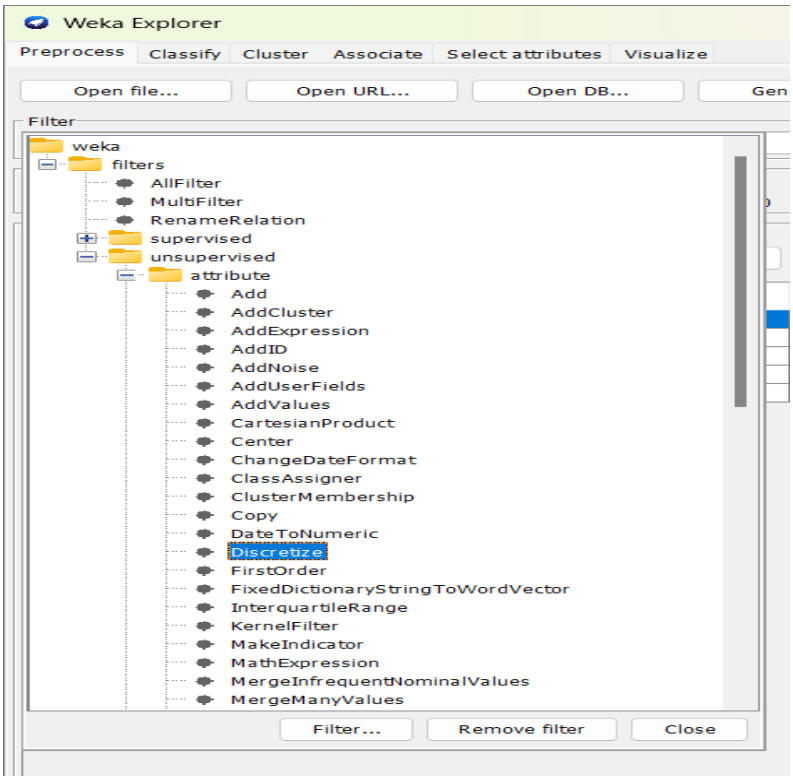
Loading the data set:



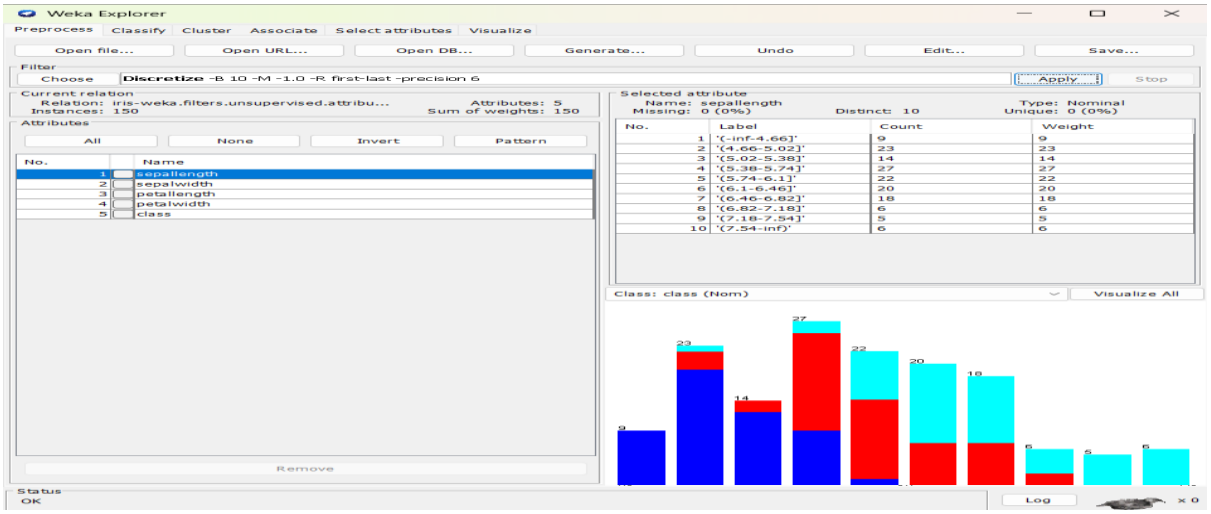
- Open WEKA Explorer.
- In the Preprocess panel, click **Open file**.
- Select the iris.arff file.
- The dataset loads and displays attributes such as sepal length, sepal width, petal length, petal width, and class.

Discretizing the data:

- Open the filter.
- Select the Unsupervised data.
- Click on the discretize.
- Apply this filter on the data.

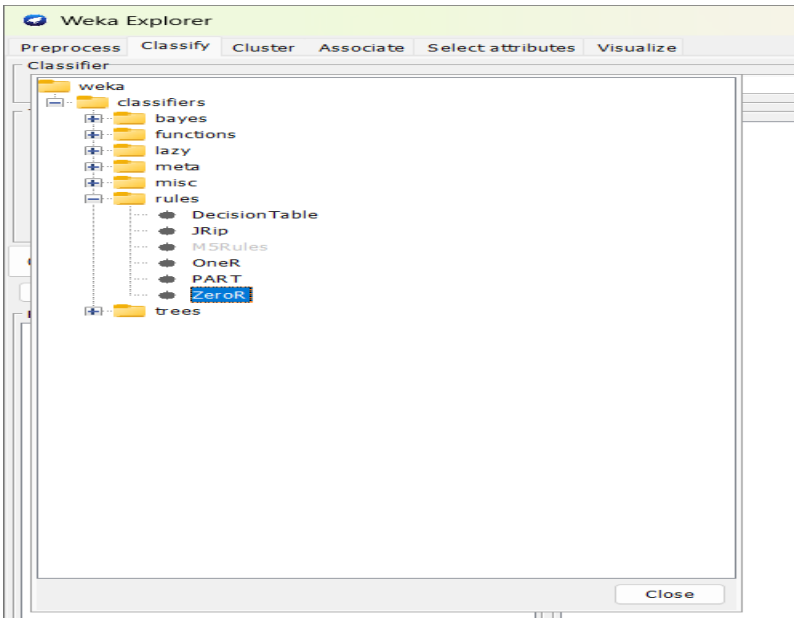


Observation of the data:

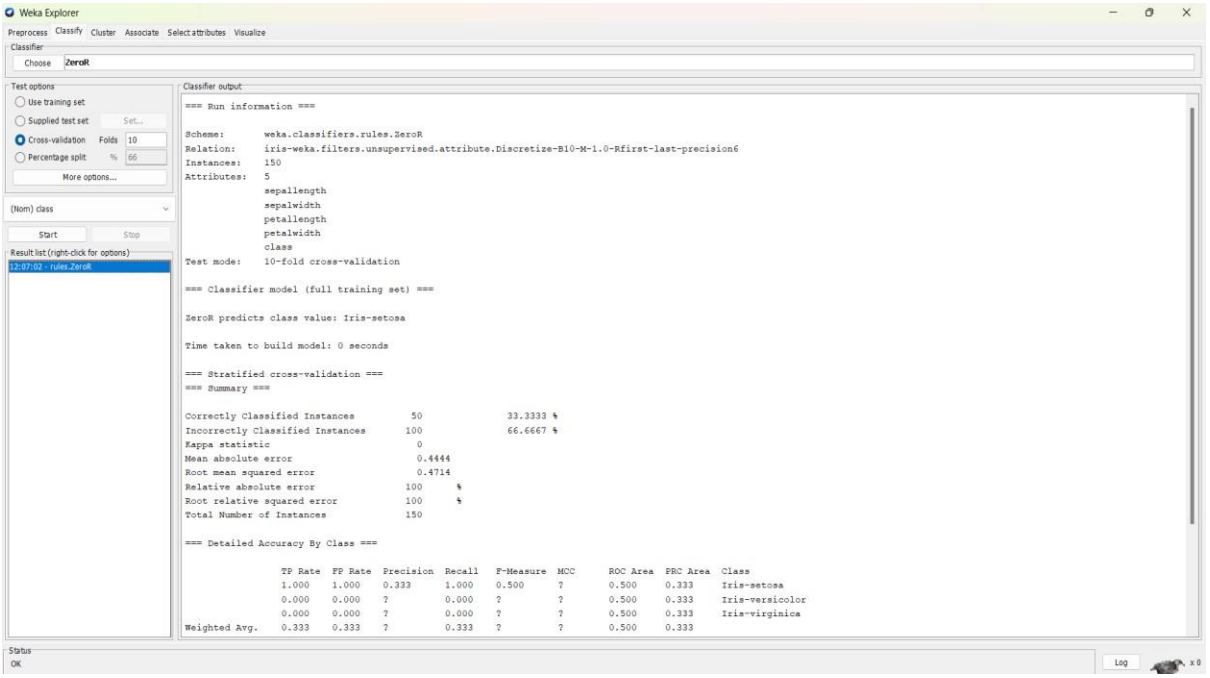


Classification on the data set:

- Click on the classify panel.
- To apply the rule ZeroR, click on the rules and select the ZeroR technique.



- Click on the choose button to select a technique in a classify.
- Test Options has the default Cross-validation with 10 folds.
- Click on the start button.
- Observe the output given by the classifier.



Viva Questions

- **What is the purpose of the ZeroR classifier in machine learning?**

  ANS:_____

  _____

- **How does ZeroR handle classification and regression problems differently?**

  ANS:_____

  _____

- **Why is ZeroR considered a baseline model, and what does its performance indicate about the dataset?**

  ANS:_____

  _____

- **If ZeroR achieves high accuracy on the Iris dataset, what does that imply about the class distribution?**

  ANS:_____

  _____

Week-7

7. Write a Python program to generate frequent item sets / association rules using Apriori algorithm

Program:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
# Sample Market Basket Data
# Each sublist represents a transaction, and elements within are items purchased
dataset = [
  ['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
  ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
  ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
  ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
  ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']
]
# 1. Data Preprocessing: Convert the list of transactions into a one-hot encoded DataFrame
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
print(te_ary)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)
# 2. Apply the Apriori Algorithm to find Frequent Itemsets
# min_support: Minimum support threshold for an itemset to be considered frequent
# use_colnames=True: Use item names as column headers instead of indices
```

```
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)


print("Frequent Itemsets:")
print(frequent_itemsets)
# 3. Generate Association Rules from Frequent Itemsets
# metric: The evaluation metric to use (e.g., 'lift', 'confidence', 'support')
# min_threshold: Minimum threshold for the chosen metric
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=.8)
# Display the results
print("\nAssociation Rules (sorted by lift):")
print(rules.sort_values(by='confidence', ascending=False))
```

Output:

```
[[False False False True False True True True True False True]
 [False False True True False True False True True False True]
 [ True False False True False True True False False False False]
 [False True False False False True True False False True True]
 [False True False True True True False False True False False]]
   Apple  Corn  Dill  Eggs  Ice cream  Kidney Beans  Milk  Nutmeg  Onion  \
0  False  False  False   True     False        True  True   True   True
1  False  False   True   True     False        True False   True   True
2   True  False  False   True     False        True  True  False  False
3  False   True  False  False     False        True  True  False  False
4  False   True  False   True      True        True False  False   True


   Unicorn  Yogurt
0   False    True
1   False    True
2   False   False
3    True    True
4   False   False
```

Frequent Itemsets:

	support	itemsets
0	0.2	(Apple)
1	0.4	(Corn)
2	0.2	(Dill)
3	0.8	(Eggs)
4	0.2	(Ice cream)
..	...	...
144	0.4	(Kidney Beans, Nutmeg, Eggs, Yogurt, Onion)
145	0.2	(Nutmeg, Eggs, Yogurt, Onion, Milk)
146	0.2	(Kidney Beans, Nutmeg, Yogurt, Onion, Milk)
147	0.2	(Kidney Beans, Nutmeg, Eggs, Dill, Yogurt, Onion)
148	0.2	(Kidney Beans, Nutmeg, Eggs, Yogurt, Onion, Milk)

[149 rows x 2 columns]

Association Rules (sorted by lift):

	antecedents	consequents	antecedent support \
629	(Onion, Milk)	(Yogurt, Nutmeg, Eggs, Kidney Beans)	0.2
0	(Apple)	(Eggs)	0.2
1	(Apple)	(Kidney Beans)	0.2
2	(Apple)	(Milk)	0.2
3	(Ice cream)	(Corn)	0.2
..	...	...	...
9	(Dill)	(Onion)	0.2
8	(Dill)	(Nutmeg)	0.2
7	(Dill)	(Kidney Beans)	0.2
6	(Dill)	(Eggs)	0.2
12	(Kidney Beans)	(Eggs)	1.0

	consequent support	support	confidence	lift	representativity \
629	0.4	0.2	1.0	2.500000	1.0
0	0.8	0.2	1.0	1.250000	1.0
1	1.0	0.2	1.0	1.000000	1.0
2	0.6	0.2	1.0	1.666667	1.0
3	0.4	0.2	1.0	2.500000	1.0
..	...	...	...	...	...
9	0.6	0.2	1.0	1.666667	1.0
8	0.4	0.2	1.0	2.500000	1.0
7	1.0	0.2	1.0	1.000000	1.0
6	0.8	0.2	1.0	1.250000	1.0
12	0.8	0.8	0.8	1.000000	1.0

	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
629	0.12	inf	0.75	0.500000	1.0	0.750000
0	0.04	inf	0.25	0.250000	1.0	0.625000
1	0.00	inf	0.00	0.200000	0.0	0.600000
2	0.08	inf	0.50	0.333333	1.0	0.666667
3	0.12	inf	0.75	0.500000	1.0	0.750000
..	...	...	...	...	...	...
9	0.08	inf	0.50	0.333333	1.0	0.666667
8	0.12	inf	0.75	0.500000	1.0	0.750000
7	0.00	inf	0.00	0.200000	0.0	0.600000
6	0.04	inf	0.25	0.250000	1.0	0.625000
12	0.00	1.0	0.00	0.800000	0.0	0.900000

Viva questions:

1. **What is the main purpose of using the Apriori algorithm in data mining?**

2. **What is the role of support and confidence in generating association rules?**

3. **Why is the TransactionEncoder used in the Apriori Python program?**

4. **What does the lift metric indicate in association rule mining?**

5. **How does changing the min_support value affect the output of the Apriori algorithm?**

Week-8

8. Write a program to calculate chi-square value using Python/R. Report your observation.

Program:

```
import pandas as pd

from scipy.stats import chi2_contingency


# Sample data (contingency table)

# Example: Relationship between gender and preference for a certain product
data = {'Cintent V': [23, 17],
    'Cintent A': [13, 35],

    'Cintent T': [30, 28]}

index = ['Male', 'Female']

contingency_table = pd.DataFrame(data, index=index)


print("Contingency Table:")
print(contingency_table)
# Perform the Chi-square test

chi2, p_value, dof, expected = chi2_contingency(contingency_table)
significance = 9.2

print("Expected frequencies table:")

print(pd.DataFrame(expected, columns=contingency_table.columns,
index=contingency_table.index))

print(f"Degrees of freedom: {dof}")
print(f"\nsignificance: {significance:.2f}")
print(f"\nChi-square statistic: {chi2:.2f}")
# Interpret the result
if chi2 < significance:
```

```
    print("\nReject the null hypothesis: There is a significant association between the variables.")
else:
    print("\nFail to reject the null hypothesis: There is no significant association between the variables.")
```

Output:

Contingency Table:

	Cintent V	Cintent A	Cintent T
Male	23	13	30
Female	17	35	28

Expected frequencies table:

	Cintent V	Cintent A	Cintent T
Male	18.082192	21.69863	26.219178
Female	21.917808	26.30137	31.780822

Degrees of freedom: 2

significance: 9.20

Chi-square statistic: 9.80

Viva questions:

1. **What is the main purpose of performing a Chi-square test in data analysis?**

2. **What are the key components of the Chi-square test output (χ^2, p-value, dof, expected)?**

3. **What does it mean when the Chi-square value is greater than the significance value?**

4. **Why do we use a contingency table in the Chi-square test?**

5. **How can the Chi-square test help determine the relationship between two categorical variables?**

Week-9

9. Write a program to compute/display dissimilarity matrix (for your own dataset containing at least four instances with two attributes) using Python

Program:

```
import pandas as pd
from scipy.spatial import distance_matrix
# Sample data points
data = [[0,2],[2,0],[3,1],[5,1]]
df = pd.DataFrame(data, columns=['X','Y'])
df
# Calculate distance matrix
pd.DataFrame(distance_matrix(df.values, df.values))
```

output:

	0	1	2	3
0	0.000000	2.828427	3.162278	5.099020
1	2.828427	0.000000	1.414214	3.162278
2	3.162278	1.414214	0.000000	2.000000
3	5.099020	3.162278	2.000000	0.000000

Viva questions:

1. **What is a dissimilarity (distance) matrix, and why is it important in data analysis?**

---

2. **Which distance metrics can be used to compute a dissimilarity matrix in Python?**

---

3. **What does each value in the distance matrix represent?**

4. **Why do we use the scipy.spatial.distance_matrix function instead of manual calculation?**

5. **How can the dissimilarity matrix be used in clustering or other machine learning tasks?**