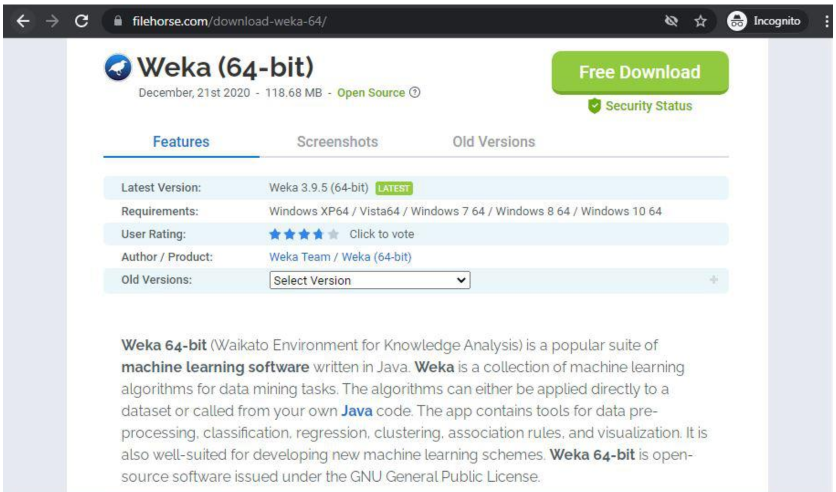
1. Study of WEKA tool and applying data mining techniques on following data sets in ARFF or CSV file Format.

Weka stands for Waikato Environment for Knowledge Analysis, it is software that is used in the data science field for data mining. It is free software. It is written in Java hence it can be run on any system supporting Java, so weka can be run on different operating systems like Windows, Linux, Mac, etc. Weka provides a collection of visualization tools that can be used for data analysis, cleaning, and predictive modeling. Weka can perform a number of tasks like data preprocessing, clustering, classification, regression, visualization, and feature selection.

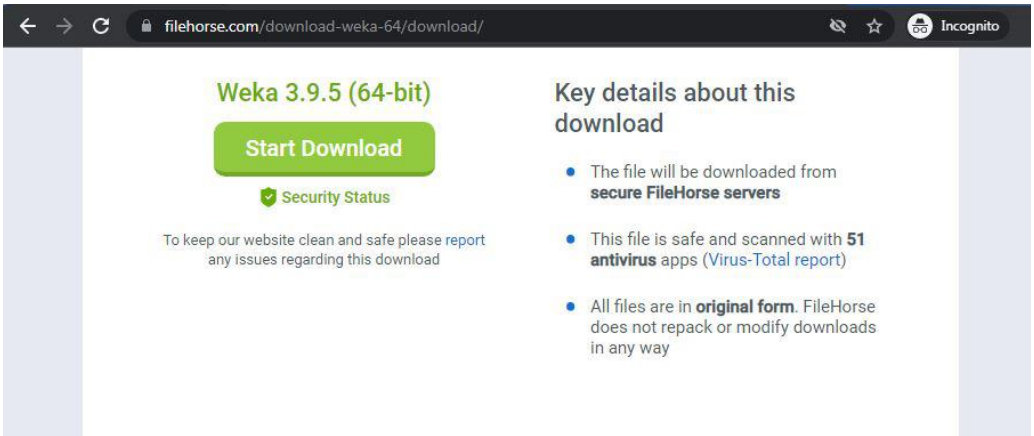
Installing Weka on Windows

Step 1: Visit this website using any web browser. Click on Free Download.

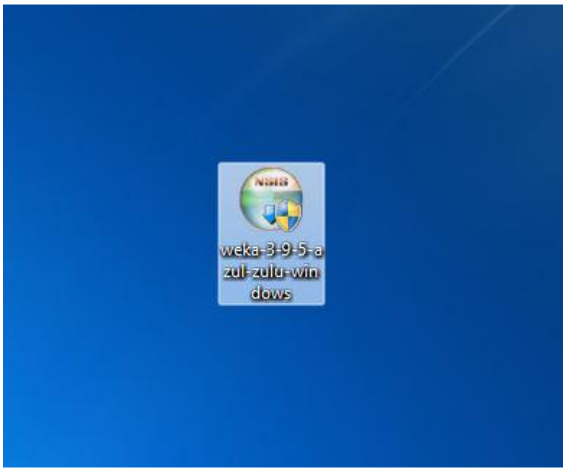
https://waikato.github.io/weka-wiki/downloading_weka/



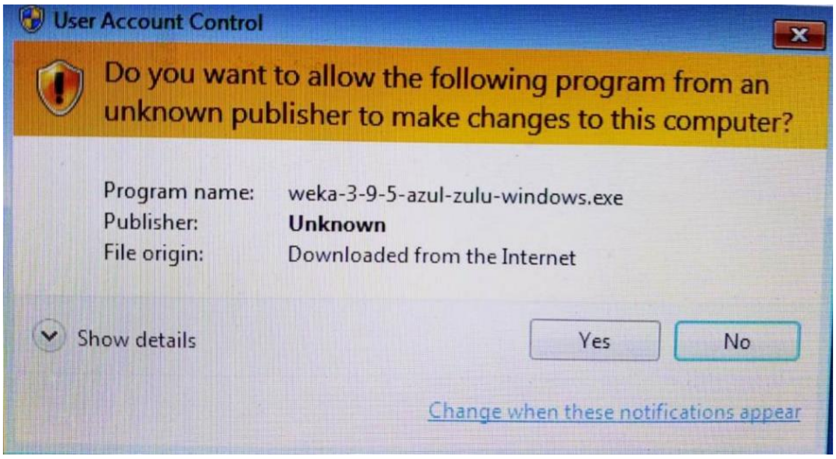
Step 2: It will redirect to a new webpage, click on Start Download. Downloading of the executable file will start shortly.



Step 3: Now check for the executable file in downloads in your system and run it.

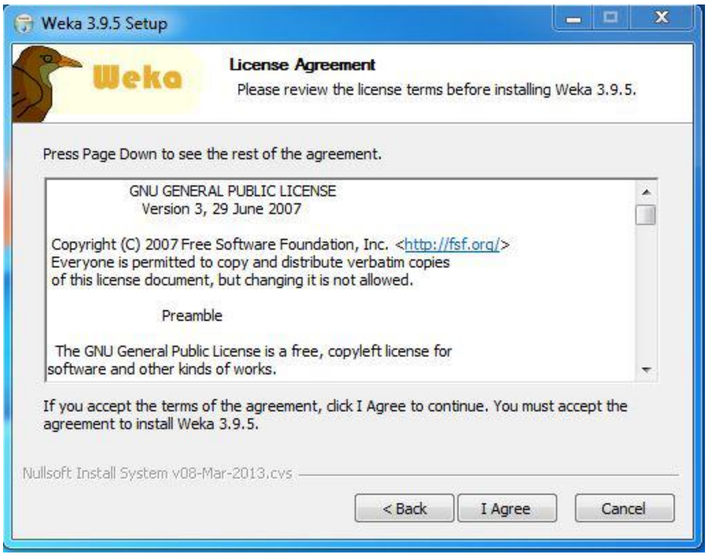


Step 4: It will prompt confirmation to make changes to your system. Click on Yes.



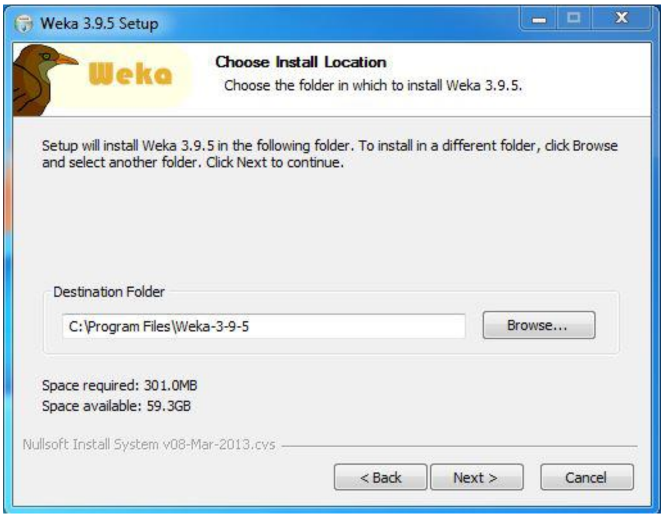
Step 5: Setup screen will appear, click on Next.

Step 6: The next screen will be of License Agreement, click on I Agree.



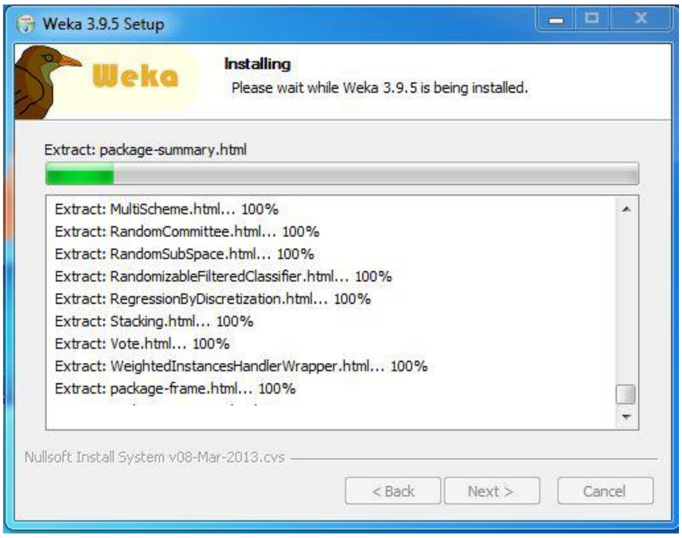
Step 7: Next screen is of choosing components, all components are already marked so don't change anything just click on the Install button.

Step 8: The next screen will be of installing location so choose the drive which will have sufficient memory space for installation. It needed a memory space of 301 MB.

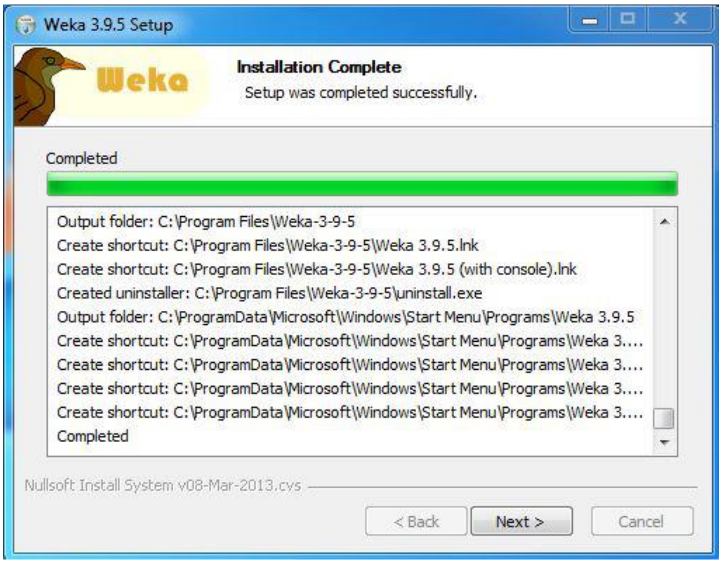


Step 9: Next screen will be of choosing the Start menu folder so don't do anything just click on Install Button.

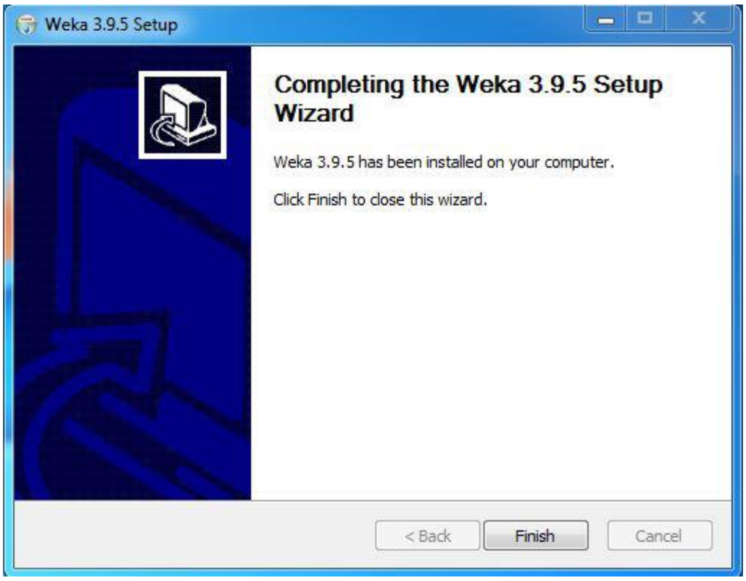
Step 10: After this installation process will start and will hardly take a minute to complete the installation.



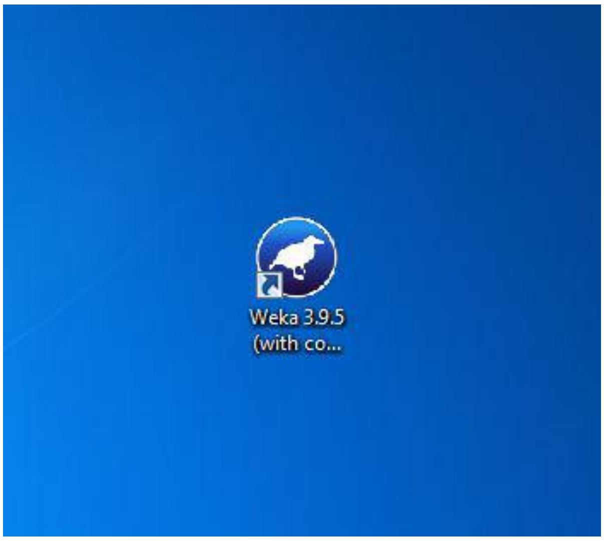
Step 11: Click on the Next button after the installation process is complete.



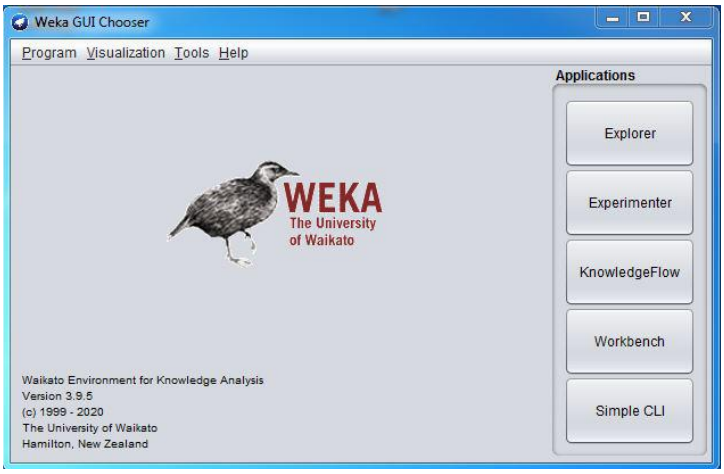
Step 12: Click on Finish to finish the installation process.



Step 13: Weka is successfully installed on the system and an icon is created on the desktop.



Step 14: Run the software and see the interface.

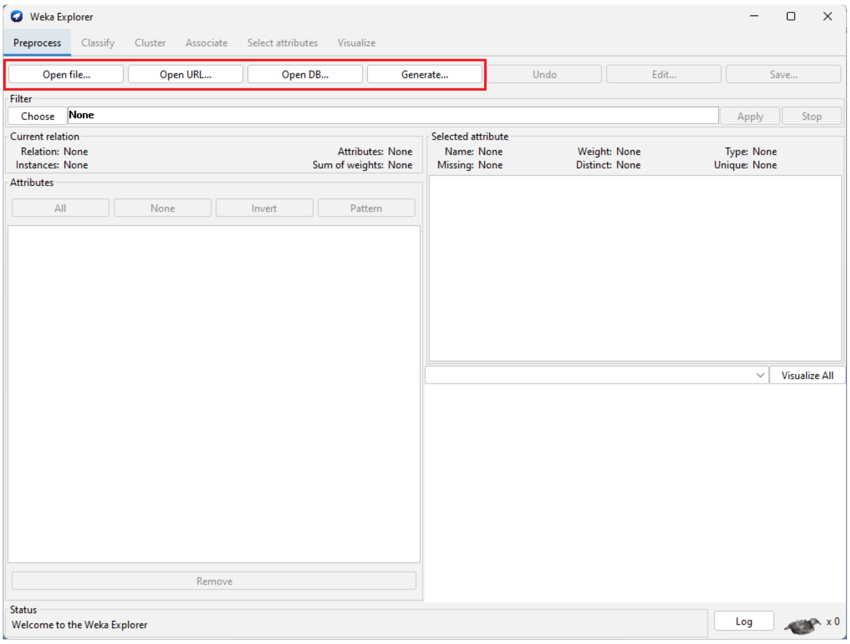


2. Implementation / Usage of WEKA for classification of datasets such as customer's data, weather forecasting data, agricultural data etc.

   In order to utilize Weka Explorer, it is essential to start by loading the data into the application. Multiple sources are available for data loading within Weka Explorer. Those are,

   1. Local file system
   2. Web
   3. Database
   4. Generate Artificial Data

   The diagram presented below provides a concise summary of the offerings provided by WEKA.



Various File Formats supported by WEKA :

In order to work with Weka Explorer, it is essential to **load the data** into the application. The data may in different formats such as CSV, Text, JSON and so on. WEKA supports a **wide range of file formats** to load the data.

The following is the complete list of file formats

- Arff data files(*.arff)
- Arff data files(*arff.gz)
- C4.5 data files(*.names)
- C4.5 data files(*.data)
- CSV data files(*.csv)
- JSON instance files(*.json)
- JSON instance files(*.json.gz)

- libsvm data files(*.libsvm)
- Matlab ASCII files(*.m)
- svm light data files(*.dat)
- Binary Serialized instances(*.bsi)
- XRFF data files(*.xrff)
- XRFF data files(*.xrff.gz)

As we see the WEKA supports various formats of data to load, among those formats the most commonly used data formats are **Arff data files(*.arff)** and **CSV data files(*.csv)**.

NOTE : The default data format of WEKA is **Arff data files (*.arff)**.

the ARFF file format:

An ARFF (**Attribute-Relation File Format**) file is an ASCII text file that describes a list of instances sharing a set of attributes. The ARFF file format has mainly **two sections**, those are

• **Header** section

• **Data** section

Header section:

The Header section of the ARFF file contains the **name of the relation**, a **list of the attributes** and their **types**.

@RELATION Declaration

The relation name is defined as the first line in the ARFF file.

format:

@RELATION <relation-name> - where <relation-name> is a string. The relation name must be quoted if the name includes spaces.

@ATTRIBUTE Declaration

The attribute specifies name of the attribute along with type.

format:

@ATTRIBUTE <attribute-name> <datatype> - where the <attribute-name> must start with an alphabet. The attribute name must be quoted if the name includes spaces.

Weka supports the following four datatypes:

1. Numeric attributes:

Numeric attributes can be real or integer numbers.

2. Nominal attributes:

Nominal values are defined by providing the possible values: { nominal-value1, nominal-value2, nominal-value3,… }

3. String attributes:

String attributes allow us to define attributes holding textual values.

4. Date attributes:

Date attribute defined as follows @ATTRIBUTE <name> date [<date-format>] - where <name> is the name for the attribute and <date-format> is an optional string. The default date-format string is yyyy-MM-dd'T'HH:mm:ss.

Example of Header Section:

% Title: Weather Dataset

@relation weather

@attribute Outlook {Sunny,Overcast,Rain}

@attribute Temperature numeric

@attribute Humidity numeric

@attribute Windy {True,False}

@attribute Play {Yes,No}

In the above example, The line which start with **%** are treated as comments.

@RELATION specifies the name of the relation.

@ATTRIBUTE specifies name of the attribute along with type and possible values.

Data section:

The Data section of the ARFF file contains the **list of data values (instance data)** separated by comma.

@data

Sunny,85,85,False,No

Sunny,80,90,True,No

Overcast,83,86,False,Yes

Rain,70,96,False,Yes

Rain,68,80,False,Yes

Rain,65,70,True,No

Overcast,?,65,True,Yes

Sunny,72,95,False,No

Sunny,69,70,False,Yes

Rain,75,?,False,Yes

Sunny,75,70,True,Yes

Overcast,?,90,True,Yes

Overcast,81,75,False,Yes

Rain,71,91,True,No

In the above, the symbol ? indicates missing values.

3. **Experiment to summarize and visualization of various datasets.**

Step 1: Launch WEKA

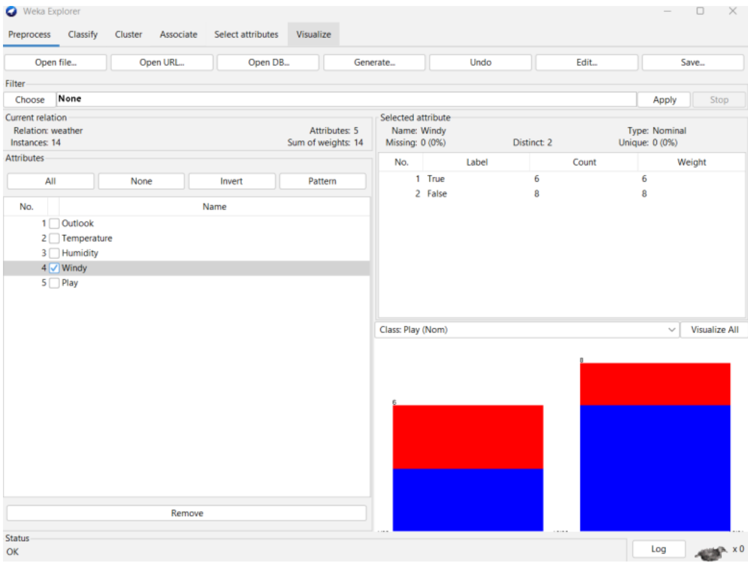
- Open WEKA GUI Chooser.
- Click on **Explorer** to open the main data mining environment.

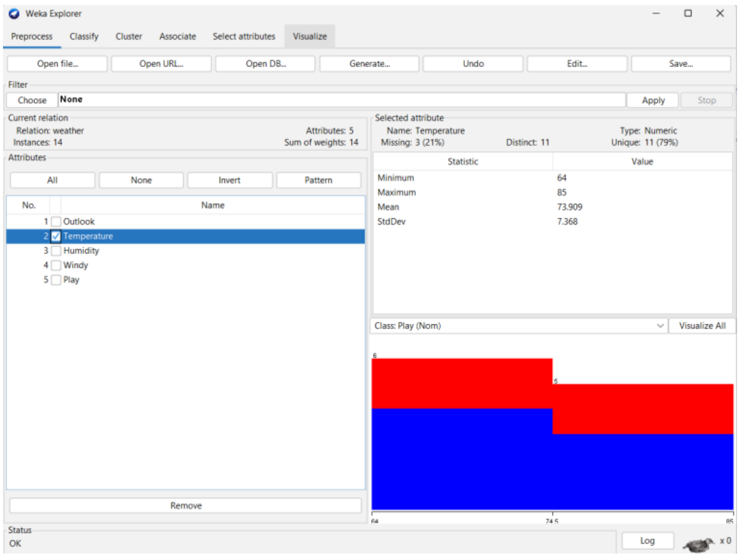
Step 2: Load Dataset

1. Click on **"Open file..."**.
2. Select a dataset file (.arff or .csv).
3. The dataset will load into the **Preprocess** tab.

For each attribute, WEKA shows:

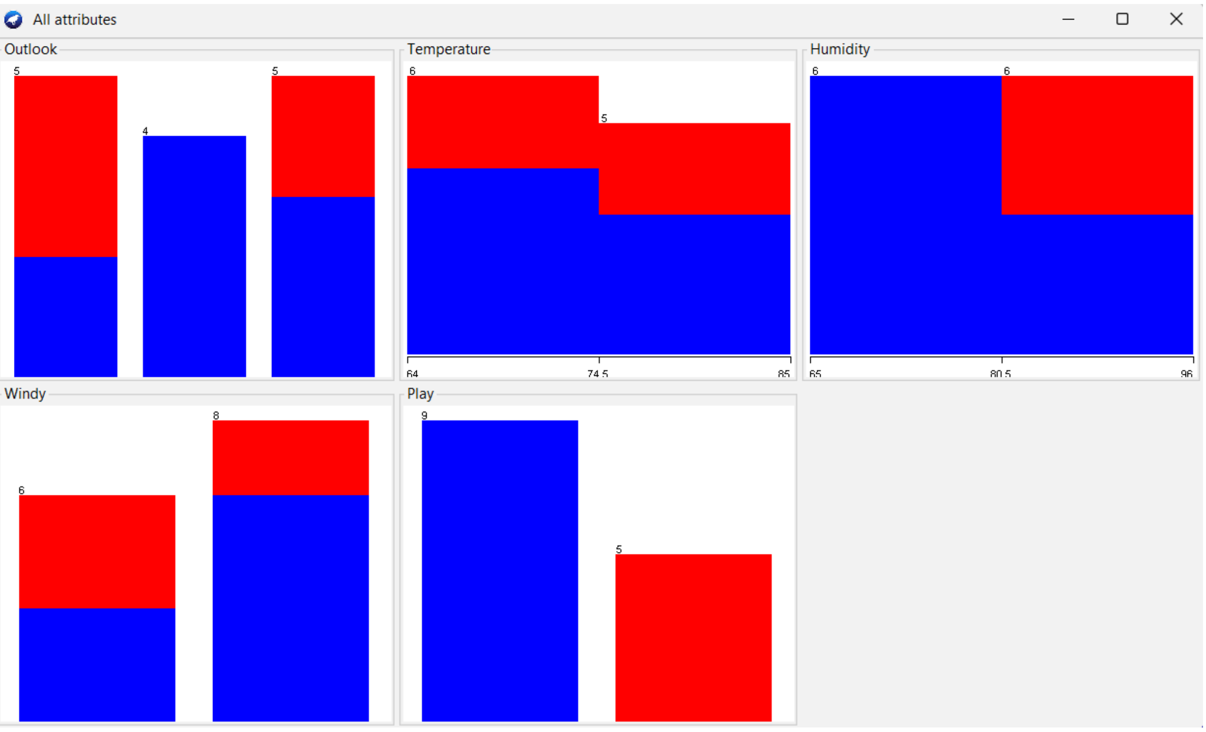
- **Type:** Nominal / Numeric
- **Missing Values**
- **Distinct Values**
- **Minimum, Maximum, Mean, Standard Deviation** (for numeric)
- **Class distribution** (for nominal/categorical)





Step 4: Visualize Data Distribution

- In the Preprocess tab, click on an attribute → it shows a **histogram** of the distribution.
- Color coding represents different classes (if any).



4. Experiment to demonstrate various data pre-processing techniques

Step 1: Open WEKA and Load Dataset

1. Launch **WEKA GUI Chooser**.
2. Click **Explorer**.
3. Click **Open file...** and load a dataset (.arff or .csv).
4. The dataset will appear in the **Preprocess** tab.

Step 2: View Summary Information

- In the **Preprocess** tab:
    - Select each attribute to see its **type**, **missing values**, **distinct values**, etc.

Step 3: Apply Data Pre-processing Techniques

a. Handling Missing Values:

Filter Path:

unsupervised → attribute → ReplaceMissingValues

Steps:

1. Click on **Choose** under Filter.
2. Select: unsupervised → attribute → ReplaceMissingValues
3. Click **Apply**.

This replaces missing values with:

- **Mean** for numeric attributes
- **Mode** for nominal attributes

b. Normalization: Rescales attributes to range [0,1]

Filter Path:

unsupervised → attribute → Normalize

Steps:

1. Click **Choose** → unsupervised → attribute → Normalize
2. Click **Apply**.

c. Standardization: Converts data to zero mean and unit variance

Filter Path:

unsupervised → attribute → Standardize

Steps:

1. Choose unsupervised → attribute → Standardize
2. Click **Apply**

d. Discretization: Converts numeric attributes to nominal

Filter Path:

unsupervised → attribute → Discretize

Steps:

1. Choose unsupervised → attribute → Discretize
2. Click **Apply**

e. Remove Unnecessary Attributes: Remove attributes that don't contribute to analysis, like IDs, serial numbers, timestamps, etc.

Filter Path:

unsupervised → attribute → Remove

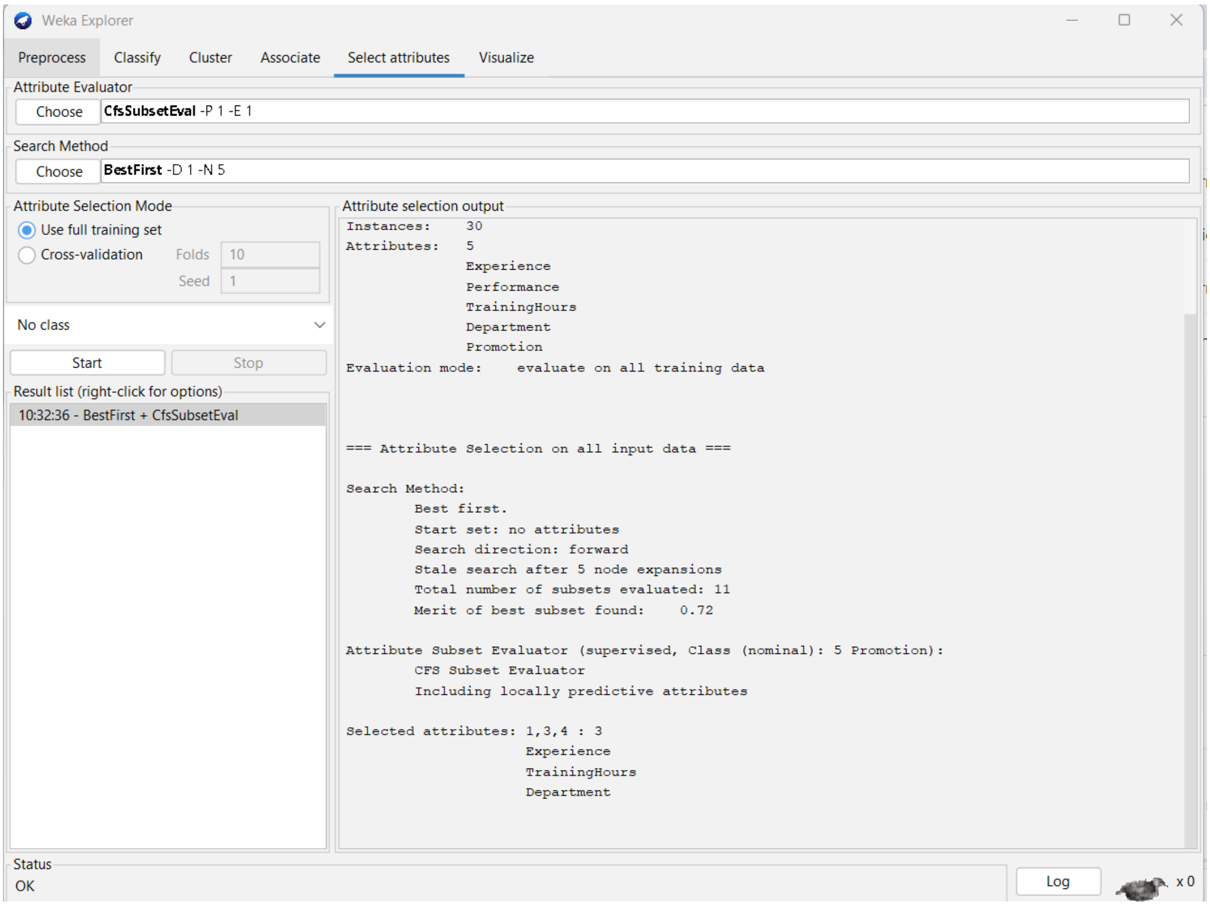
Steps:

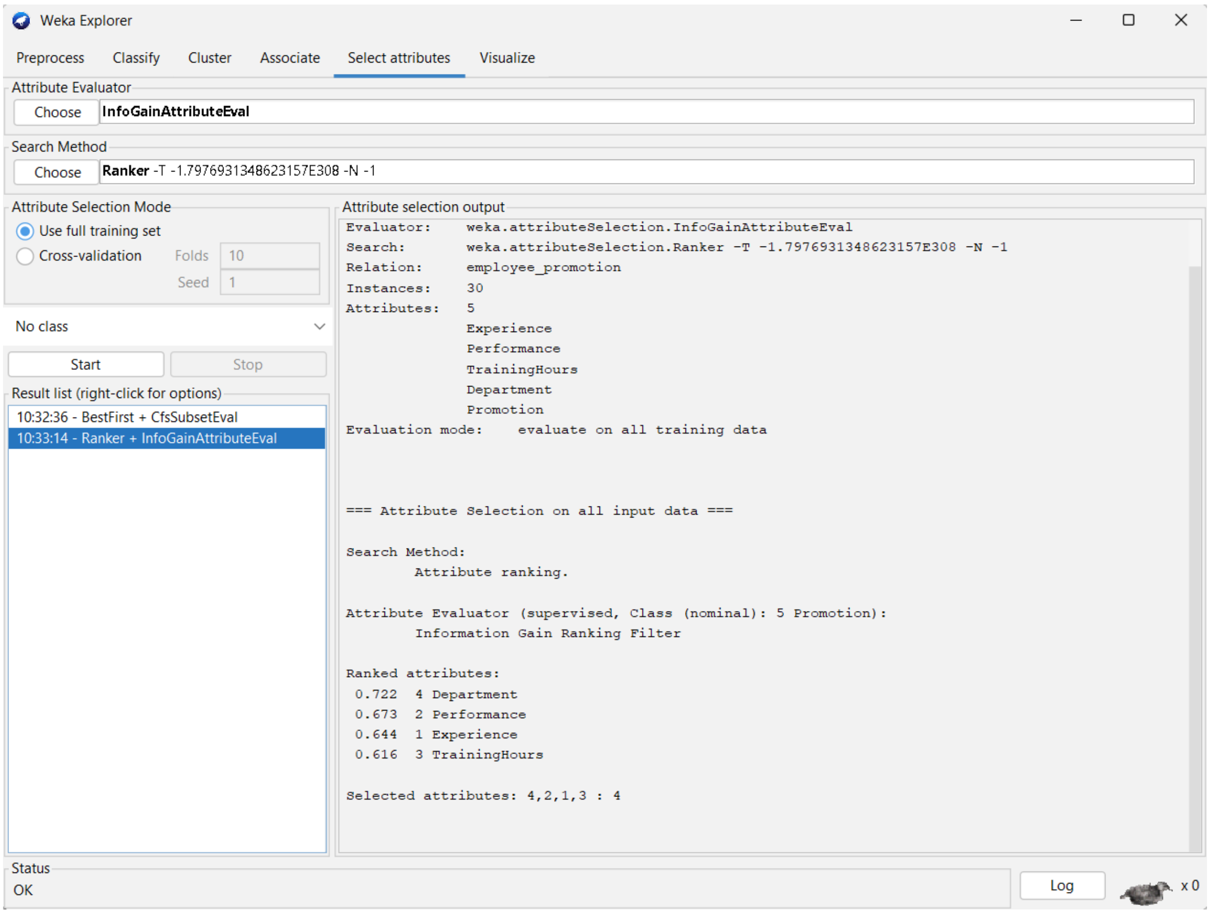
1. Click **Choose** → unsupervised → attribute → Remove
2. Click on the filter to set **attribute indices** to remove.
3. Click **Apply**

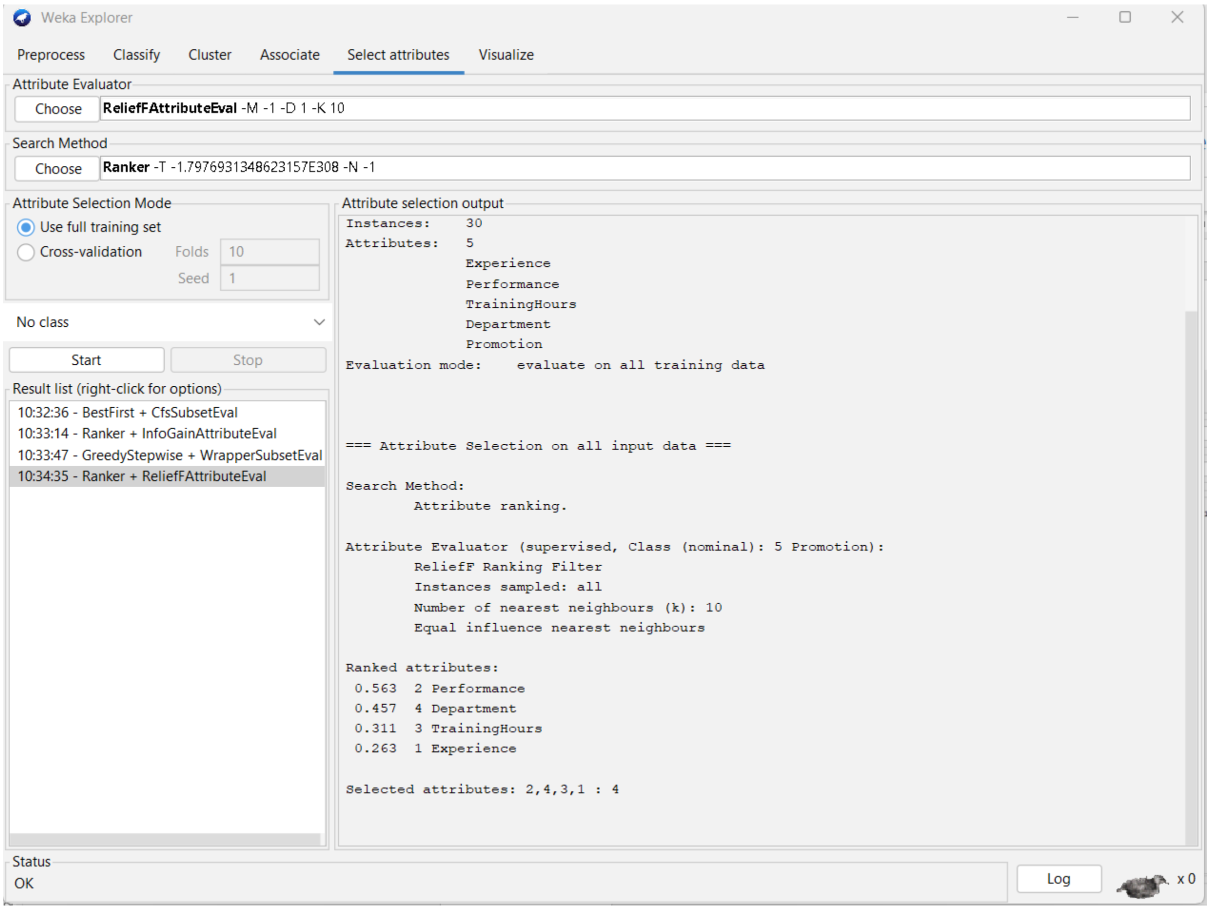
5. Experiment to select prominent feature subsets of various datasets.

- **Feature Importance Identification:**
It helps determine which features (attributes) in the dataset have the most influence on the output variable.
- **Dimensionality Reduction:**
Removes irrelevant or redundant features, reducing the number of input variables — this simplifies models and avoids overfitting.
- **Improved Model Performance:**
Using only the most significant features often increases classification or prediction accuracy and reduces computation time.
- **Better Interpretability:**
With fewer features, the resulting model becomes easier to understand and explain.
- **Preprocessing Step in Data Mining:**
Feature selection is a crucial step before applying algorithms like Decision Trees, SVM, Naïve Bayes, etc., improving their efficiency

1. Open WEKA → Explorer.
2. Load the dataset file.
3. Click on "Select attributes" tab.
4. Apply various evaluators (InfoGain, CfsSubsetEval, ReliefF).
5. Compare the selected feature subsets.




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Attribute Evaluator
Choose
InfoGainAttributeEval
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds
10
Seed
1
No class
Start
Stop
Result list (right-click for options)
10:32:36 - BestFirst + CfsSubsetEval
10:33:14 - Ranker + InfoGainAttributeEval
Attribute selection output
Evaluator: weka.attributeSelection.InfoGainAttributeEval
Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation: employee_promotion
Instances: 30
Attributes: 5
Experience
Performance
TrainingHours
Department
Promotion
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Attribute ranking.
Attribute Evaluator (supervised, Class (nominal): 5 Promotion):
Information Gain Ranking Filter
Ranked attributes:
0.722 4 Department
0.673 2 Performance
0.644 1 Experience
0.616 3 TrainingHours
Selected attributes: 4,2,1,3 : 4
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Attribute Evaluator
Choose
ReliefFAttributeEval -M -1 -D 1 -K 10
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds
10
Seed
1
No class
Start
Stop
Result list (right-click for options)
10:32:36 - BestFirst + CfsSubsetEval
10:33:14 - Ranker + InfoGainAttributeEval
10:33:47 - GreedyStepwise + WrapperSubsetEval
10:34:35 - Ranker + ReliefFAttributeEval
Attribute selection output
Instances: 30
Attributes: 5
Experience
Performance
TrainingHours
Department
Promotion
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Attribute ranking.
Attribute Evaluator (supervised, Class (nominal): 5 Promotion):
ReliefF Ranking Filter
Instances sampled: all
Number of nearest neighbours (k): 10
Equal influence nearest neighbours
Ranked attributes:
0.563 2 Performance
0.457 4 Department
0.311 3 TrainingHours
0.263 1 Experience
Selected attributes: 2,4,3,1 : 4
Status
OK
Log
x 0

6. Experiment to Evaluate Information Gain of an attribute in the student database

   Information Gain: Information Gain (IG) measures how much knowing an attribute reduces the uncertainty (entropy) about the class. It is based on Entropy, which measures impurity or randomness in the data.

   Step 1: Load Data

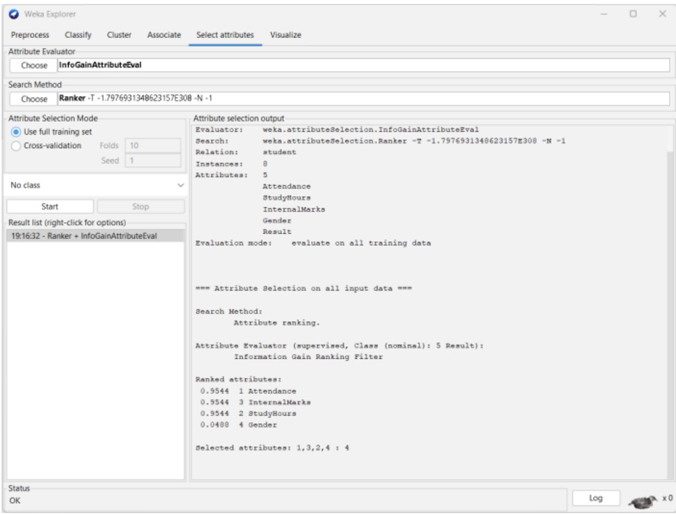
- Open Weka → Explorer → Preprocess
- Load student.arff
- Set Result as class attribute.

   Step 2: Select InfoGain

- Go to Select Attributes tab
- In Attribute Evaluator choose: InfoGainAttributeEval
- In Search Method choose: Ranker

   Step 3: Run

   Click Start → Weka gives you Information Gain values for each attribute.




```
@relation student
@attribute Attendance {Low, Medium, High}
@attribute StudyHours numeric
@attribute InternalMarks numeric
@attribute Gender {Male, Female}
@attribute Result {Pass, Fail}
@data
High,8,25,Male,Pass
Medium,5,20,Female,Pass
Low,2,12,Male,Fail
High,7,22,Female,Pass
Low,1,10,Female,Fail
Medium,4,18,Male,Pass
High,9,26,Female,Pass
Low,3,15,Male,Fail
```

7. Demonstration of classification rule process using j48 decision tree algorithm

J48 generates a tree based on the **attribute with the highest Information Gain Ratio** at each node. The tree can then be converted into **IF–THEN classification rules**.

Step 1: Open Weka Explorer

- Launch Weka → **Explorer**.

Step 2: Load Data

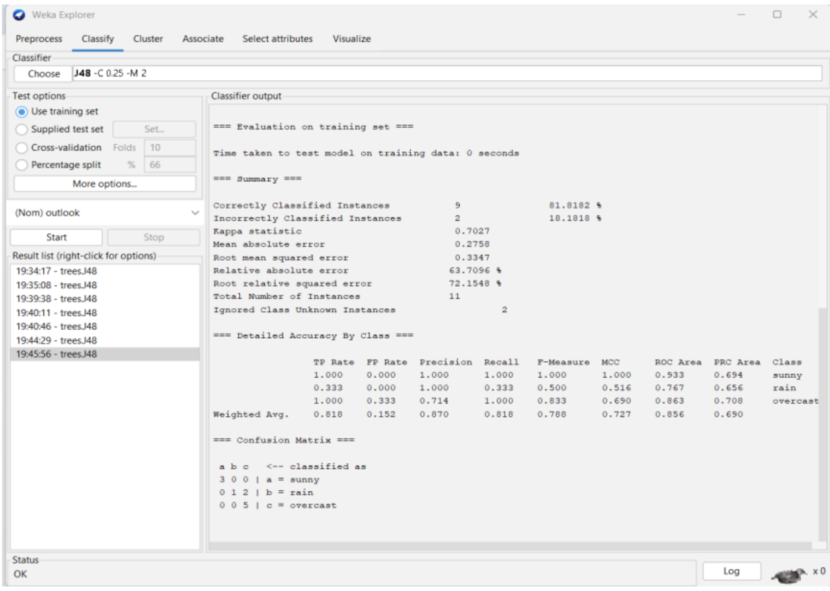
- Load student.arff.
- Go to **Classify tab**.
- Ensure Result is the class attribute.

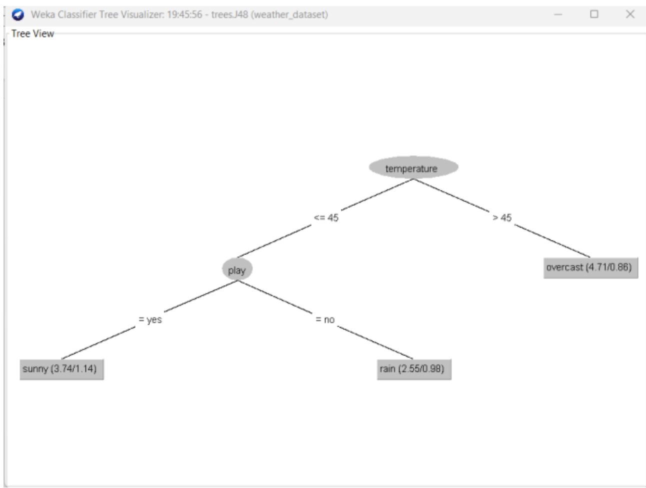
Step 3: Choose Classifier

- Click **Choose** → Select trees → J48.
- In "Test options", select **Use training set** (not cross-validation).

Step 4: Run the Classifier

- Click **Start**.
- Weka will output the **decision tree model** and evaluation metrics.



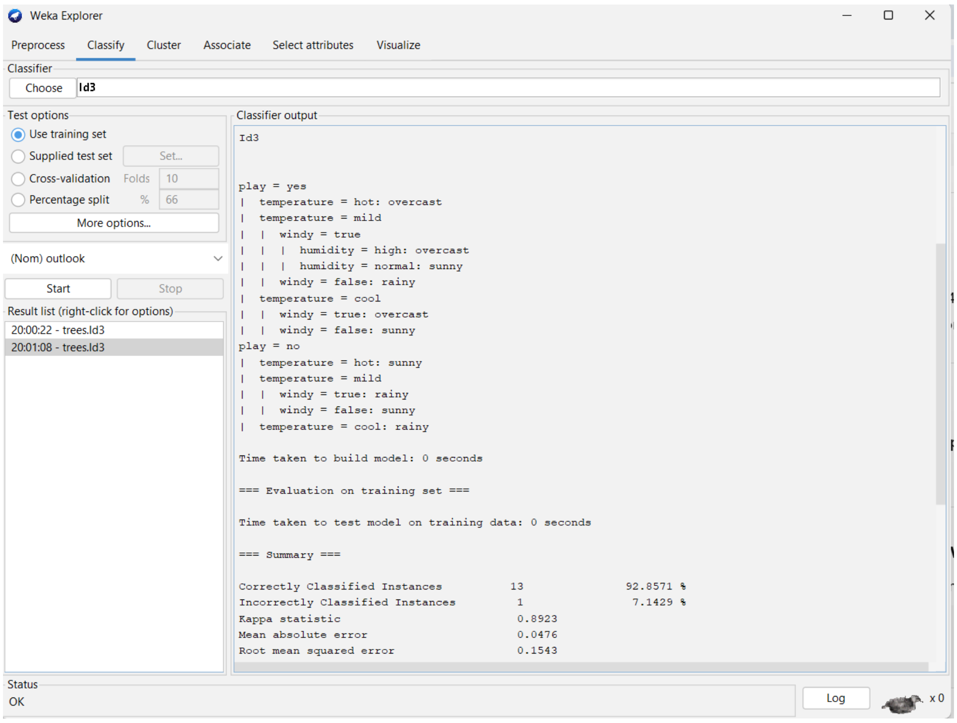


8. Demonstration of classification rule process using ID3 decision tree algorithm

ID3 (Iterative Dichotomiser 3)

- A decision tree algorithm developed by Quinlan.
- Builds the tree using **Information Gain** to select the best attribute at each step.
- Works well with **categorical data**, but cannot handle numeric values directly.
- Does **not support pruning**, so trees may overfit.

  - Open **Weka Explorer → Preprocess**.
  - Load dataset.
  - Go to **Classify → Choose → trees → Id3**
  - Run classifier.
  - Weka will display the decision tree and evaluation



9. Experiment to predict the class using the Bayesian classification

   Naive Bayes
   - A **probabilistic classifier** based on **Bayes' Theorem**.
   - Assumes attributes are **independent** given the class (hence "naïve").
   - Works well with both categorical and numeric data.
   - Fast, simple, and effective, even with small datasets.

Based on **Bayes' Theorem**:

$$P(Class|Data) = \frac{P(Data|Class) \times P(Class)}{P(Data)}$$

- In Weka, the most common is **Naïve Bayes**, which assumes that attributes are **conditionally independent** given the class.
- It works well on both categorical and numeric attributes.

Step 1: Load Dataset

- Open **Weka Explorer** → **Preprocess**.
- Load weather.arff.

Step 2: Set Class Attribute

- The last attribute play is the **class** (yes/no).

Step 3: Select Classifier

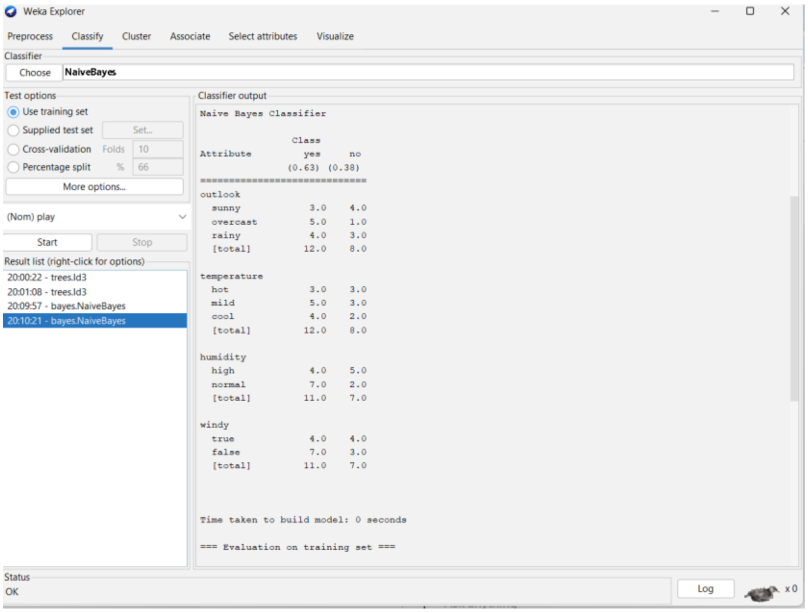
- Go to the **Classify tab**.
- Choose: bayes → NaiveBayes.

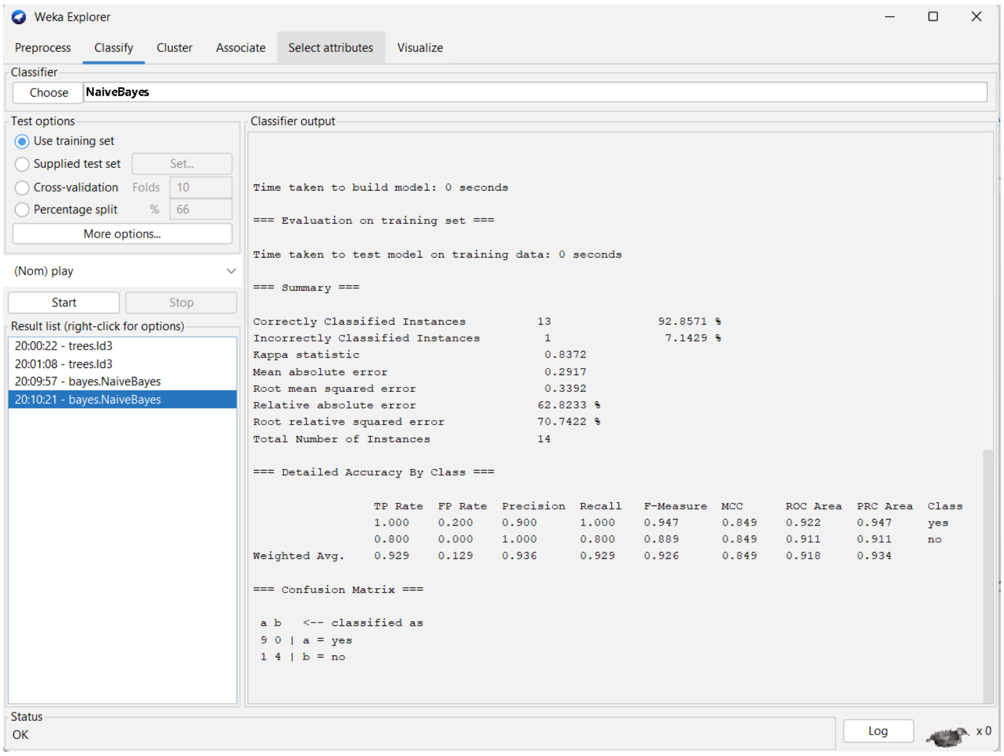
Step 4: Test Options

- Choose **10-fold cross-validation** (default), or **Use training set** for a small dataset.

Step 5: Run

- Click **Start**.

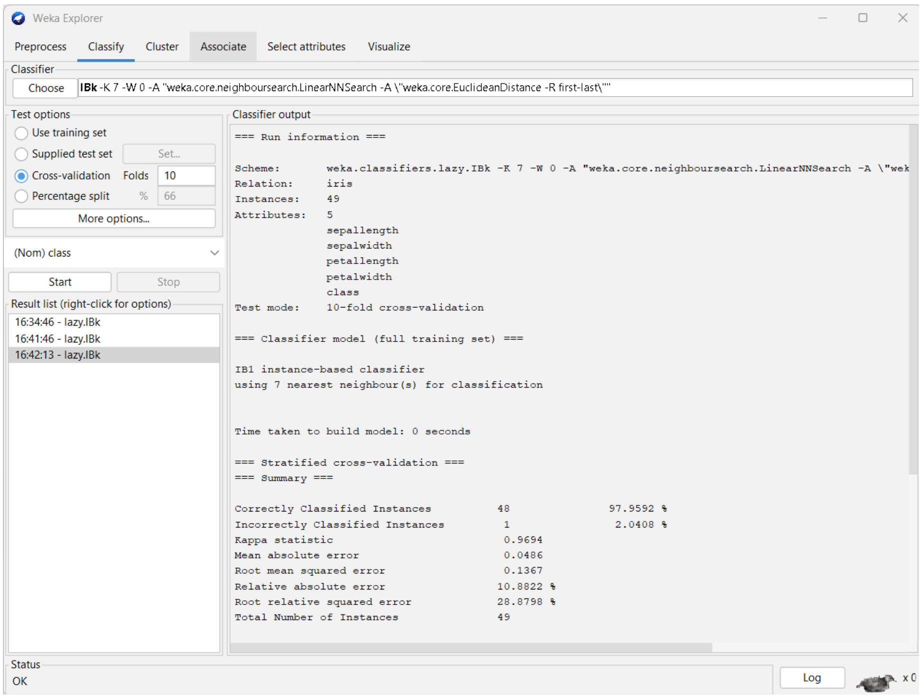


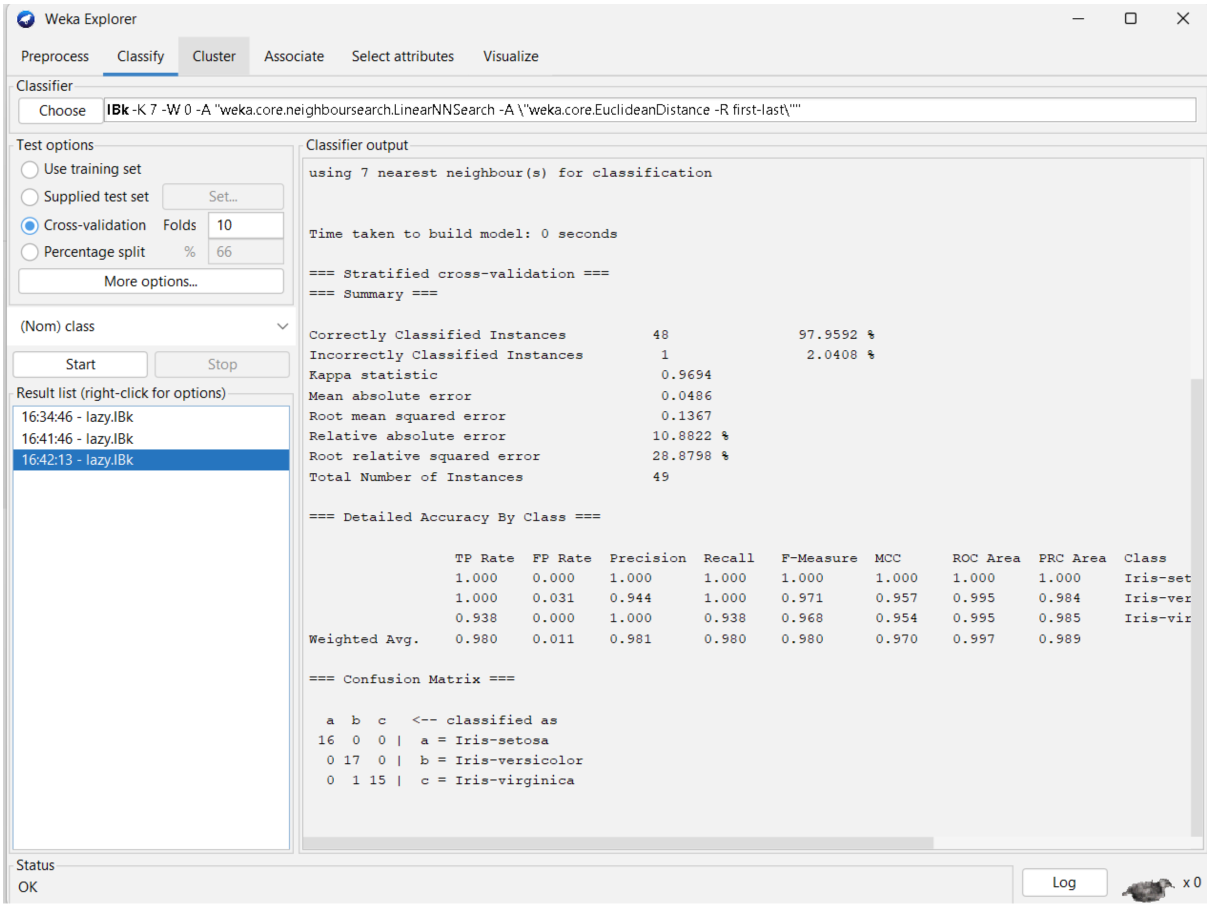

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) play
Start
Stop
Result list (right-click for options)
20:00:22 - trees.Id3
20:01:08 - trees.Id3
20:09:57 - bayes.NaiveBayes
20:10:21 - bayes.NaiveBayes
Classifier output
Time taken to build model: 0 seconds
=== Evaluation on training set ===
Time taken to test model on training data: 0 seconds
=== Summary ===
Correctly Classified Instances 13 92.8571 %
Incorrectly Classified Instances 1 7.1429 %
Kappa statistic 0.8372
Mean absolute error 0.2917
Root mean squared error 0.3392
Relative absolute error 62.8233 %
Root relative squared error 70.7422 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.200 0.900 1.000 0.947 0.849 0.922 0.947 yes
0.800 0.000 1.000 0.800 0.889 0.849 0.911 0.911 no
Weighted Avg. 0.929 0.129 0.936 0.929 0.926 0.849 0.918 0.934
=== Confusion Matrix ===
a b <-- classified as
9 0 | a = yes
1 4 | b = no
Status
OK
Log
x 0

10. Experiment to predict the class using the k-Nearest Neighbor classification

KNN: k-Nearest Neighbor (KNN) is a simple, supervised machine learning algorithm commonly used for both classification and regression tasks. It works on the principle of finding the *k* nearest data points to a query instance based on distance measures such as Euclidean distance and assigning the majority class (for classification) or the average value (for regression). Unlike other algorithms, KNN has no explicit training phase and is therefore known as an instance-based learning method. Its main strength lies in its simplicity, non-parametric nature, and effectiveness on small datasets with irregular decision boundaries, making it useful in applications such as pattern recognition, medical diagnosis, recommendation systems, and image classification. However, KNN has limitations: it becomes computationally expensive for large datasets, is sensitive to irrelevant or unscaled features, suffers from the curse of dimensionality in high-dimensional data, and its performance highly depends on the choice of *k*.

1. Open **WEKA** → **Explorer** → **Preprocess**.
2. Load **iris.arff**.
3. Go to **Classify** tab → **Choose** → lazy → IBk **(k-NN)**.
4. Try different **k values (1,3,5,7,9)**.
5. Use **10-fold Cross-Validation** for evaluation.
6. Record Accuracy, Confusion Matrix, and per-class Precision/Recall/F1.




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
IBk -K 7 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
16:34:46 - lazy.IBk
16:41:46 - lazy.IBk
16:42:13 - lazy.IBk
Classifier output
using 7 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 48 97.9592 %
Incorrectly Classified Instances 1 2.0408 %
Kappa statistic 0.9694
Mean absolute error 0.0486
Root mean squared error 0.1367
Relative absolute error 10.8822 %
Root relative squared error 28.8798 %
Total Number of Instances 49
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-set
1.000 0.031 0.944 1.000 0.971 0.957 0.995 0.984 Iris-ver
0.938 0.000 1.000 0.938 0.968 0.954 0.995 0.985 Iris-vir
Weighted Avg. 0.980 0.011 0.981 0.980 0.980 0.970 0.997 0.989
=== Confusion Matrix ===
a b c <-- classified as
16 0 0 | a = Iris-setosa
0 17 0 | b = Iris-versicolor
0 1 15 | c = Iris-virginica
Status
OK
Log
x 0

11. Experiment to implement weight & bias updating using the Back propagation Neural Network

A **Backpropagation Neural Network (BPNN)** is a supervised learning model consisting of input, hidden, and output layers.

It uses the **backpropagation algorithm** for training:

1. **Forward pass** – inputs are passed through the network to calculate outputs.
2. **Error computation** – difference between predicted and target values.
3. **Backward pass** – gradients of the error are propagated backward using the **chain rule**.
4. **Weight and bias update** – parameters are adjusted using Gradient Descent:

$$w_{new} = w_{old} - \eta \cdot \frac{\partial E}{\partial w}$$

$$b_{new} = b_{old} - \eta \cdot \frac{\partial E}{\partial b}$$

where **η** = learning rate, **E** = error function (e.g., Mean Squared Error).

Mul**tilayer Perceptron (MLP)** in WEKA is a feedforward neural network trained with **backpropagation**.

It has **input, hidden, and output layers**.

Backpropagation updates **weights and biases** iteratively to minimize error using **gradient descent**.

1. **Open WEKA Explorer.**

- Go to **Preprocess → Open file** → load a dataset (e.g., iris.arff or your custom dataset).
- Ensure the **class attribute** is set (usually last column).

2. **Go to Classify tab.**

- Click **Choose → functions → MultilayerPerceptron**.
- This is WEKA's backpropagation neural network.

3. **Configure the Neural Network.**

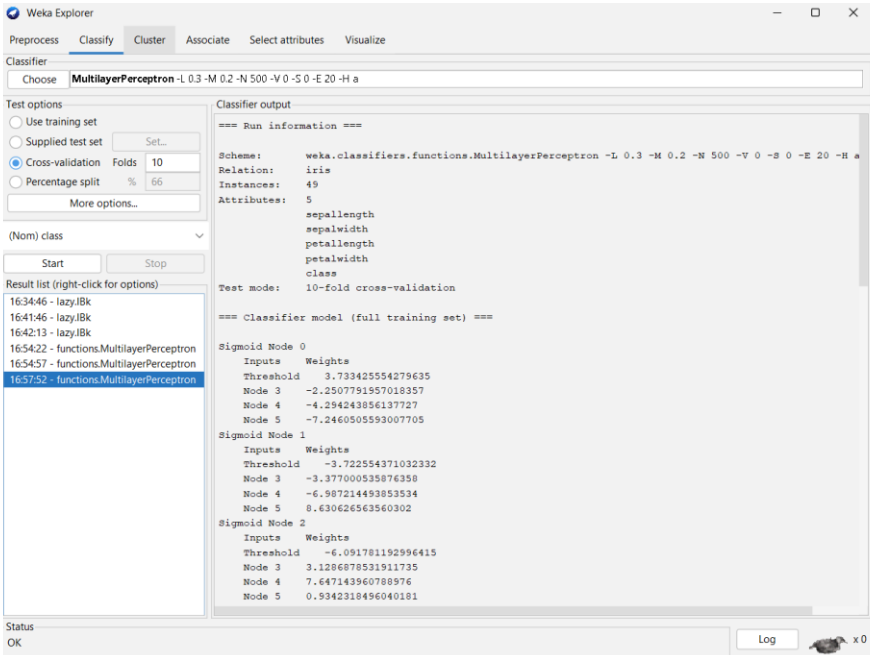
- Click on **MultilayerPerceptron** (blue text) to open parameters:
  - **Learning rate (η):** controls weight update step size (default 0.3).
  - **Momentum:** helps escape local minima by smoothing updates (default 0.2).
  - **Training time (epochs):** number of iterations (default 500).
  - **Hidden layers:** structure of hidden layer(s)
    - Use a = (attributes+classes)/2 (default)
    - You can try 1, 2, or 3 hidden units.
- Press **OK**.

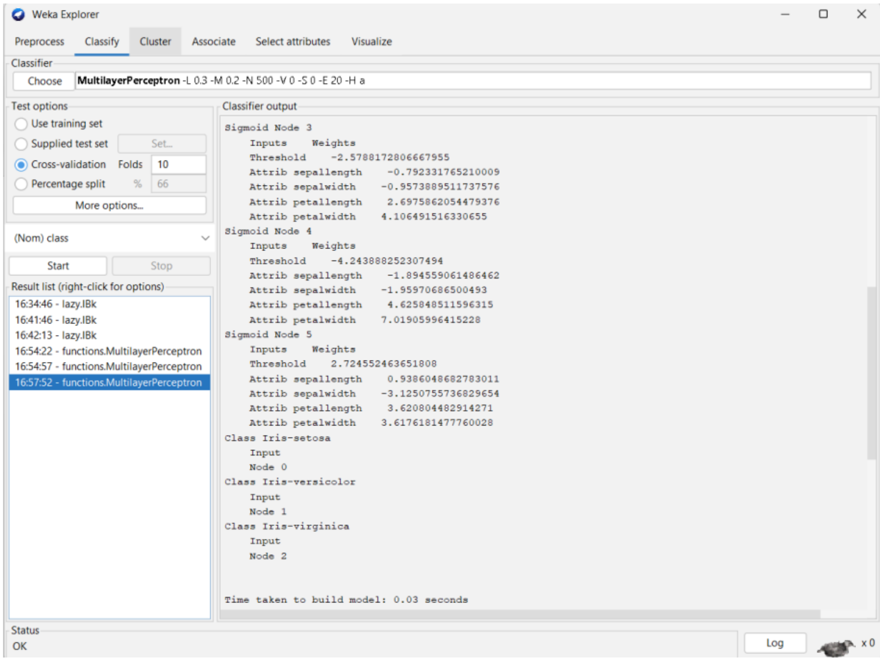
4. **Choose evaluation method.**

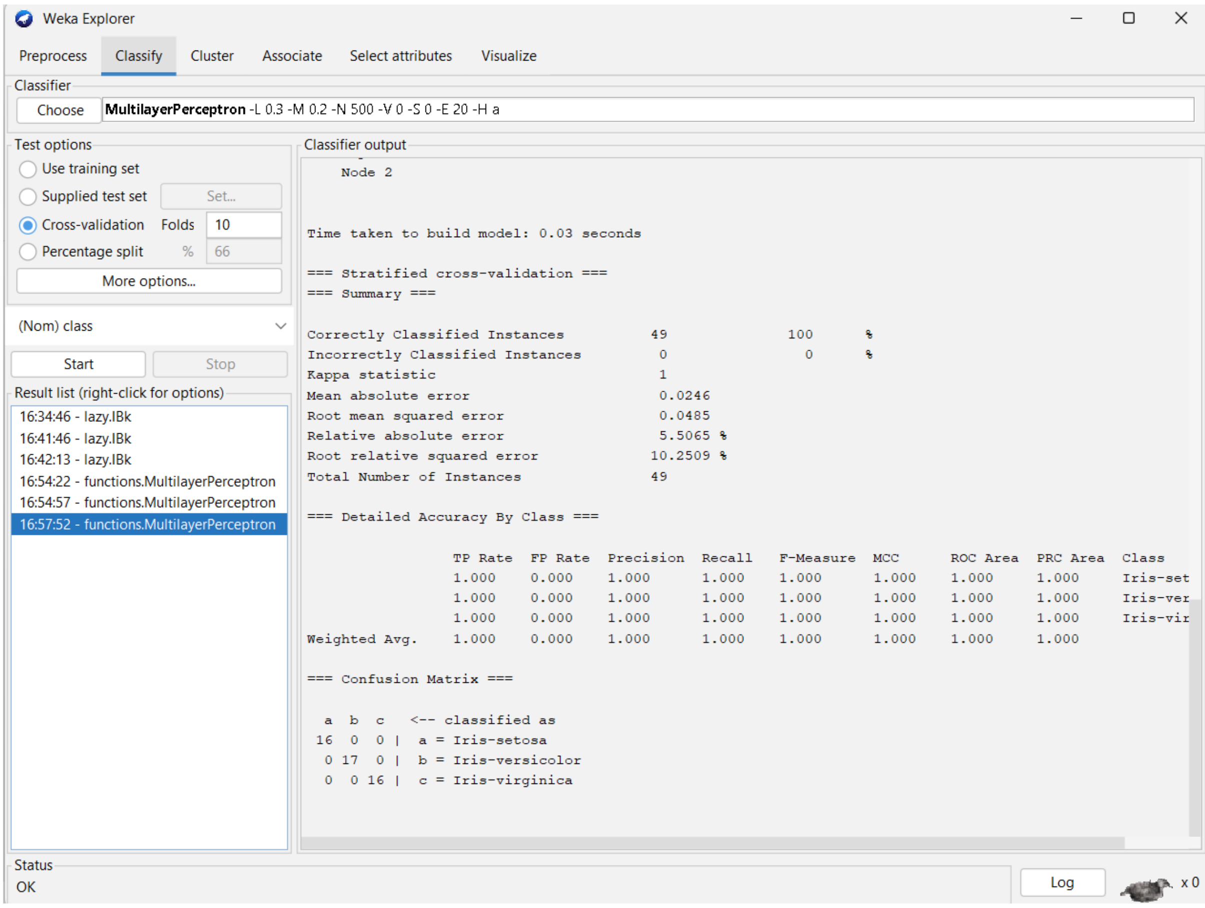
- At the top, select:
    - **Cross-validation (10 folds)** (recommended), or
    - **Percentage split (e.g., 70% train, 30% test)**.

5. **Run training.**

- Click **Start**.
- WEKA trains the network using backpropagation.
- The output shows **accuracy, confusion matrix, error rate**, etc.






Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
16:34:46 - lazy.IBk
16:41:46 - lazy.IBk
16:42:13 - lazy.IBk
16:54:22 - functions.MultilayerPerceptron
16:54:57 - functions.MultilayerPerceptron
16:57:52 - functions.MultilayerPerceptron
Classifier output
Node 2
Time taken to build model: 0.03 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 49 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0.0246
Root mean squared error 0.0485
Relative absolute error 5.5065 %
Root relative squared error 10.2509 %
Total Number of Instances 49
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-set
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-ver
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-vir
Weighted Avg. 1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000
=== Confusion Matrix ===
a b c <-- classified as
16 0 0 | a = Iris-setosa
0 17 0 | b = Iris-versicolor
0 0 16 | c = Iris-virginica
Status
OK
Log
x 0

12. Demonstration of clustering process using k-means algorithm

- **Clustering** is an **unsupervised learning** technique where data is grouped into clusters based on similarity.
- **K-Means** is the most widely used clustering algorithm.
- It works as follows:
  - ➢ Choose the number of clusters (**k**).
  - ➢ Randomly initialize **k centroids**.
  - ➢ Assign each data point to the **nearest centroid** (using distance measure, e.g., Euclidean).
  - ➢ Recalculate centroids as the mean of points in each cluster.
  - ➢ Repeat until centroids do not change significantly (convergence).

➢ **Open WEKA Explorer.**

1. Go to **Preprocess** → **Open file** → load a dataset (e.g., `iris.arff` or your custom dataset).
2. Remove the **class attribute** (optional) since clustering is unsupervised.

➢ **Go to the Cluster tab.**

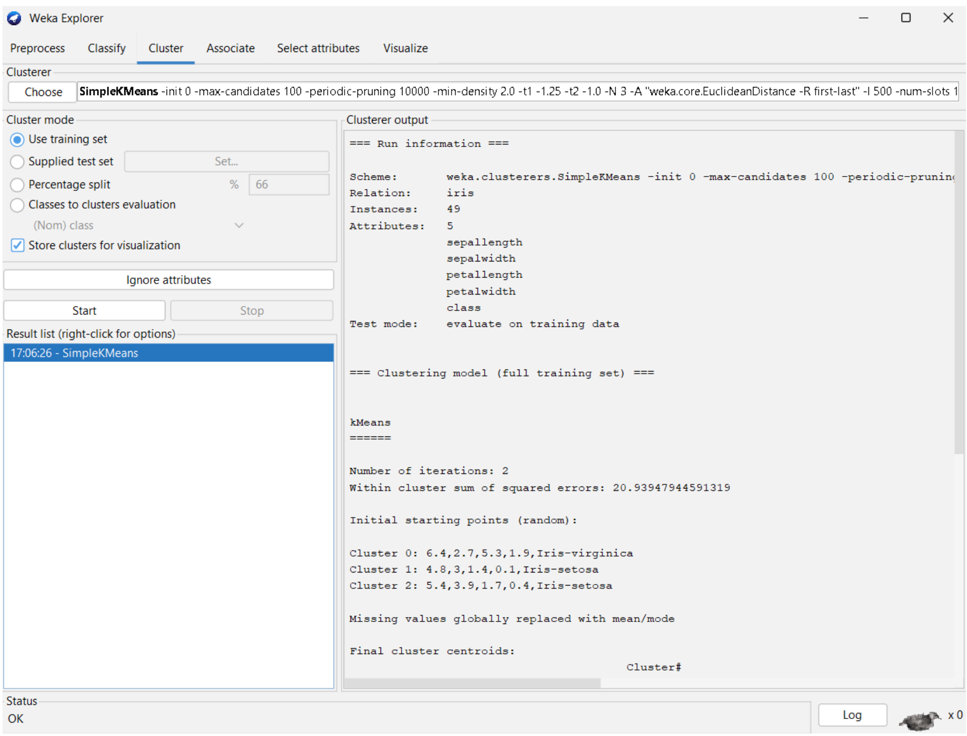
1. Click **Choose** → **SimpleKMeans**.

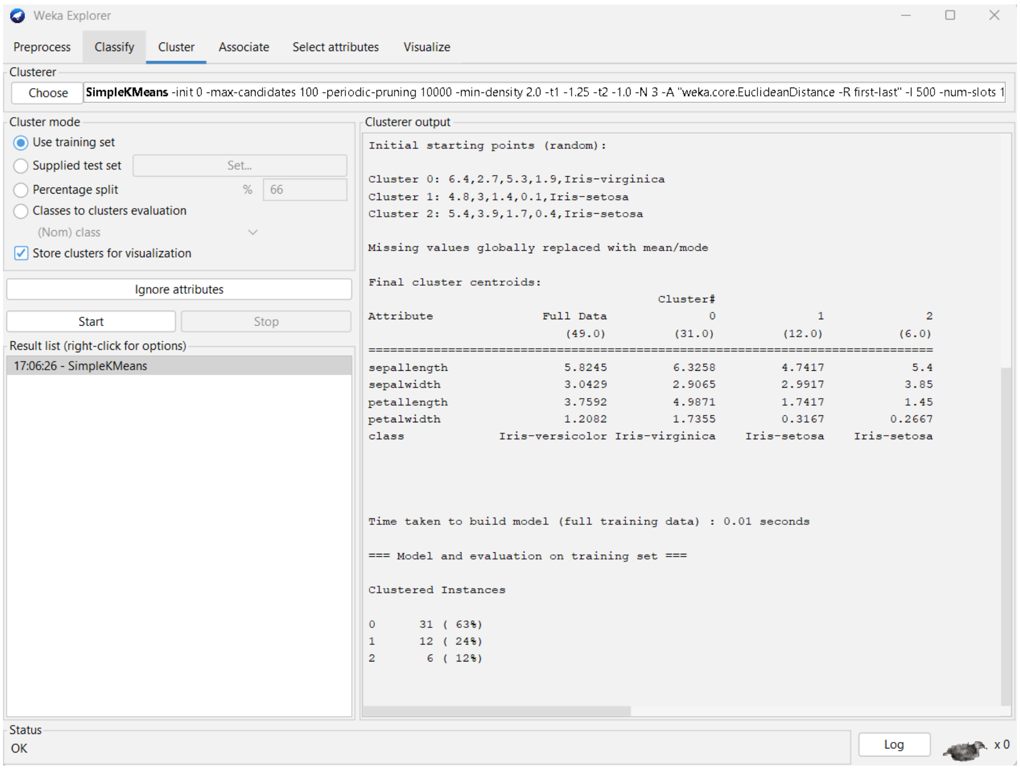
➢ **Configure K-Means parameters.**

1. Click **SimpleKMeans (blue text)** to open options:
   1. **NumClusters:** Set `k` (e.g., 3 for iris dataset).
   2. **Seed:** Random initialization (default 10).
   3. **Distance function:** EuclideanDistance (default).
2. Click **OK**.

➢ **Run the algorithm.**

1. At the top, select **Use training set** (to cluster all data).
2. Click **Start**.




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1
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)
17:06:26 - SimpleKMeans
Clusterer output
Initial starting points (random):
Cluster 0: 6.4,2.7,5.3,1.9,Iris-virginica
Cluster 1: 4.8,3,1.4,0.1,Iris-setosa
Cluster 2: 5.4,3.9,1.7,0.4,Iris-setosa
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1 2
(49.0) (31.0) (12.0) (6.0)
sepallength 5.8245 6.3258 4.7417 5.4
sepalwidth 3.0429 2.9065 2.9917 3.85
petallength 3.7592 4.9871 1.7417 1.45
petalwidth 1.2082 1.7355 0.3167 0.2667
class Iris-versicolor Iris-virginica Iris-setosa Iris-setosa
Time taken to build model (full training data) : 0.01 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 31 (63%)
1 12 (24%)
2 6 (12%)
Status
OK
Log
x 0

13. Demonstration of mining frequent patterns using Apriori algorithm

- The Apriori algorithm is a classic data mining algorithm used for association rule mining.
- It finds frequent itemsets in large transactional databases and then generates association rules from them.
- It uses support and confidence measures:
  - Support → How often an itemset appears in the dataset.
  - Confidence → How often items in Y appear in transactions that contain X (for rule X → Y).
- Widely used in market basket analysis (e.g., "If a customer buys Milk, they are likely to buy Bread")

1. Start Weka and open Explorer

- Launch the Weka GUI (double-click the weka.jar or use your Weka launcher).
- Click **Explorer**.

2. Load your dataset

- In the **Preprocess** tab click **Open file**

3. Switch to the Associate tab

- Click the **Associate** tab at the top of the Explorer window.

 Choose Apriori

- Click the **Choose** button (top left) and select: weka.associations.Apriori → **Apriori**.

 Set Apriori parameters

- Click the small text label that says Apriori (or double-click) to open the **configuration dialog**.
- Important parameters to set:
  - **lowerBoundMinSupport** (minimum support): enter a fraction such as 0.20 for 20% (for 20 transactions, 0.20 → requires itemset in at least 4 transactions).
  - **minMetric** (minimum confidence / metric): set 0.60 for 60% confidence (or use a value you need).
  - **numRules** (max number of rules to output): set something like 100 so you don't truncate useful rules.
- Notes: Weka typically accepts these as decimal fractions (0.2 = 20%). If the dialog tooltip shows a different format, follow the dialog's guidance.

 Run Apriori

- Click **OK** to close the parameter dialog.
- Click **Start** (lower left) to run Apriori on your loaded dataset.

 Read the output

Dataset:

@relation marketbasket

@attribute Milk {t, f}

@attribute Bread {t, f}

@attribute Butter {t, f}

@attribute Egg {t, f}

@attribute Apple {t, f}

@attribute Banana {t, f}

@attribute Sugar {t, f}

@data

t, t, t, f, f, f, f

t, t, f, t, f, f, f

t, f, f, f, t, t, f

f, t, t, f, f, f, t

t, t, f, f, t, f, f

f, t, f, t, f, f, t

t, f, f, f, f, t, t

f, t, t, t, f, f, f

t, t, f, f, f, t, f

f, f, f, f, t, t, t

t, t, t, f, t, f, f

f, t, f, t, f, f, f

t, f, f, f, t, f, t

t, t, t, t, f, f, f

f, t, f, f, f, t, t

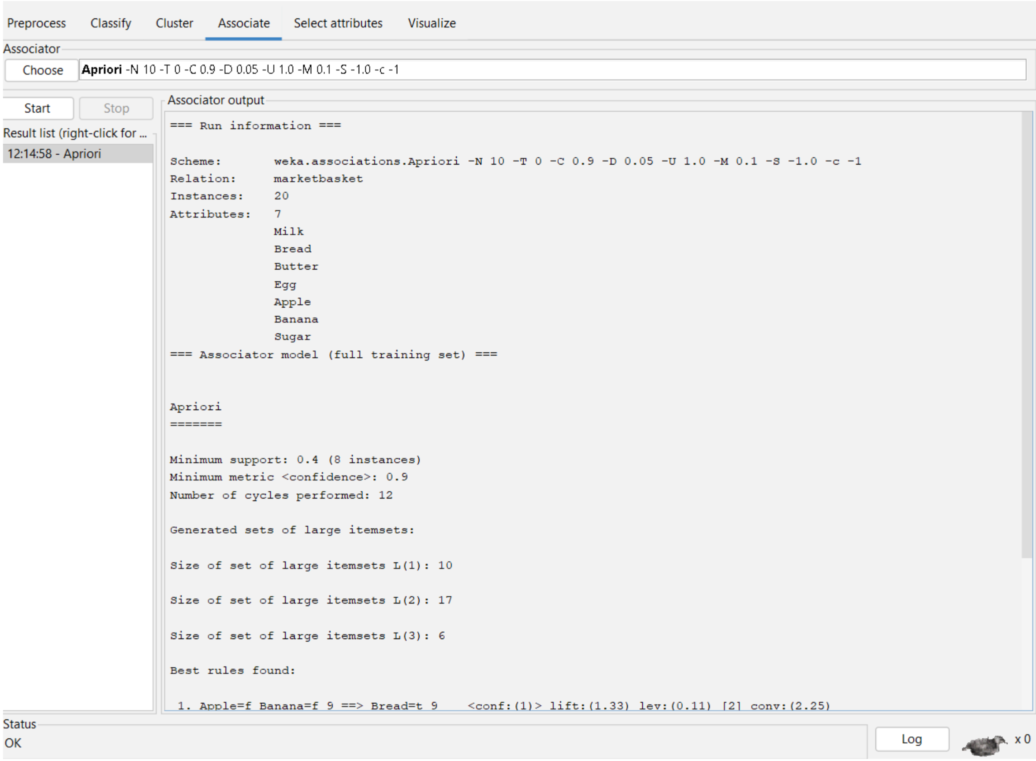
t, t, f, f, t, t, f

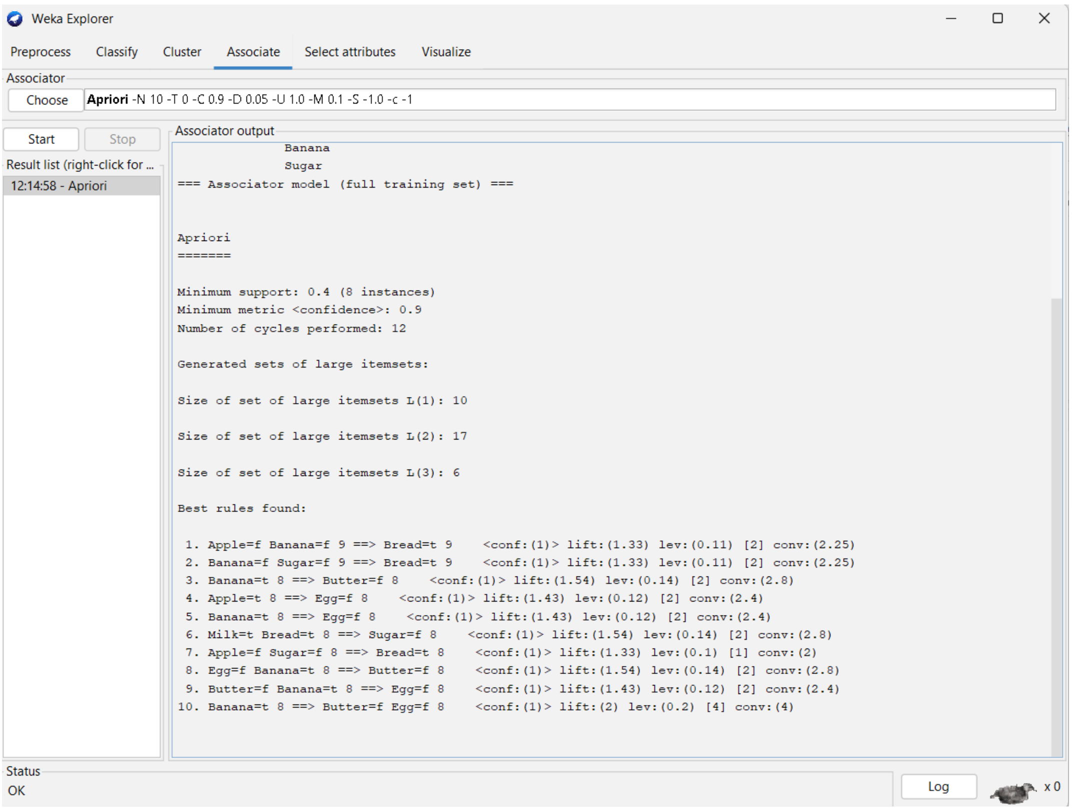
f, t, t, f, f, f, f

t, f, f, f, t, t, t

f, t, t, t, f, f, f

t, t, f, f, t, t, f


Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click for ...
12:14:58 - Apriori
Associator output
=== 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: marketbasket
Instances: 20
Attributes: 7
Milk
Bread
Butter
Egg
Apple
Banana
Sugar
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.4 (8 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 12
Generated sets of large itemsets:
Size of set of large itemsets L(1): 10
Size of set of large itemsets L(2): 17
Size of set of large itemsets L(3): 6
Best rules found:
1. Apple=f Banana=f 9 ==> Bread=t 9   <conf:(1)> lift:(1.33) lev:(0.11) [2] conv:(2.25)
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-click for ...
12:14:58 - Apriori
Associator output
Banana
Sugar
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.4 (8 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 12
Generated sets of large itemsets:
Size of set of large itemsets L(1): 10
Size of set of large itemsets L(2): 17
Size of set of large itemsets L(3): 6
Best rules found:
1. Apple=f Banana=f 9 ==> Bread=t 9   <conf:(1)> lift:(1.33) lev:(0.11) [2] conv:(2.25)
2. Banana=f Sugar=f 9 ==> Bread=t 9   <conf:(1)> lift:(1.33) lev:(0.11) [2] conv:(2.25)
3. Banana=t 8 ==> Butter=f 8   <conf:(1)> lift:(1.54) lev:(0.14) [2] conv:(2.8)
4. Apple=t 8 ==> Egg=f 8   <conf:(1)> lift:(1.43) lev:(0.12) [2] conv:(2.4)
5. Banana=t 8 ==> Egg=f 8   <conf:(1)> lift:(1.43) lev:(0.12) [2] conv:(2.4)
6. Milk=t Bread=t 8 ==> Sugar=f 8   <conf:(1)> lift:(1.54) lev:(0.14) [2] conv:(2.8)
7. Apple=f Sugar=f 8 ==> Bread=t 8   <conf:(1)> lift:(1.33) lev:(0.1) [1] conv:(2)
8. Egg=f Banana=t 8 ==> Butter=f 8   <conf:(1)> lift:(1.54) lev:(0.14) [2] conv:(2.8)
9. Butter=f Banana=t 8 ==> Egg=f 8   <conf:(1)> lift:(1.43) lev:(0.12) [2] conv:(2.4)
10. Banana=t 8 ==> Butter=f Egg=f 8   <conf:(1)> lift:(2) lev:(0.2) [4] conv:(4)
Status
OK
Log
x 0

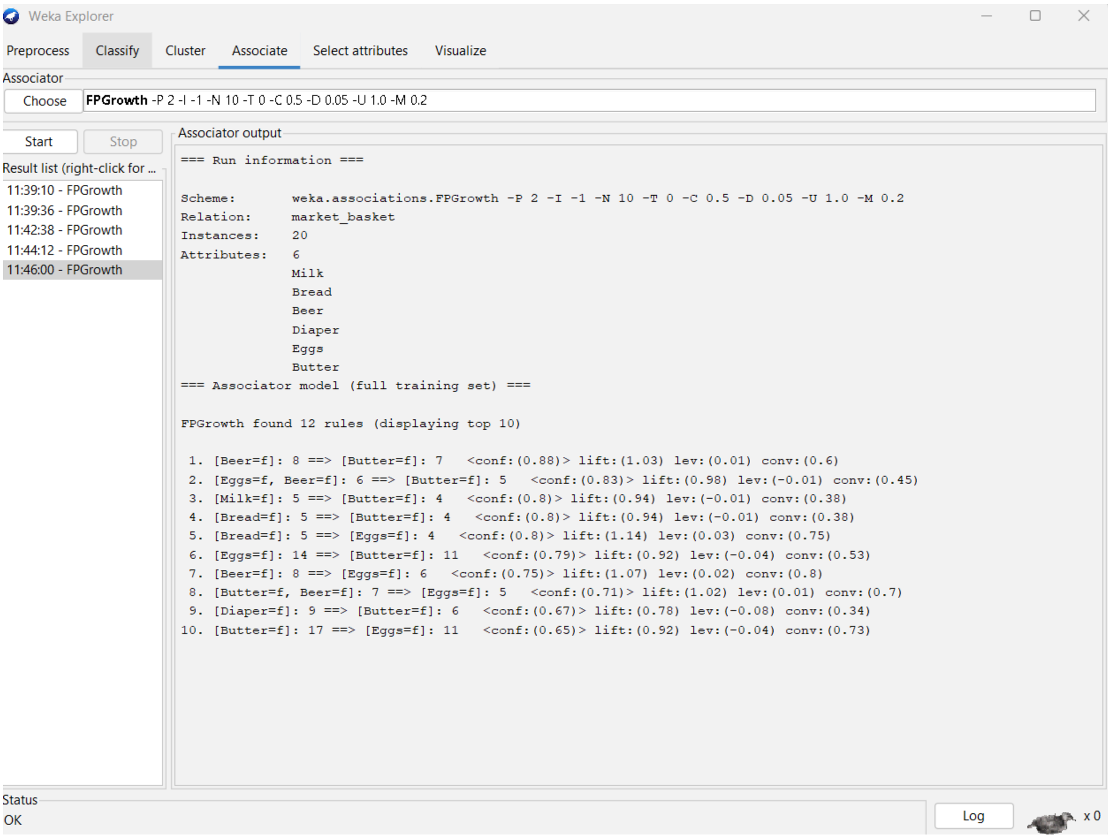
14. Demonstration of mining frequent patterns using FP-Growth algorithm

- **Frequent Pattern Growth (FP-Growth)** is an efficient algorithm for mining frequent itemsets without candidate generation (unlike Apriori).
- It compresses the dataset into a **Frequent Pattern Tree (FP-Tree)** and then recursively extracts frequent patterns.
- Works better than Apriori for **large datasets** because it avoids repeated database scans.

Procedure:

1. Prepare the dataset
2. Open Weka
3. Load the dataset
4. Go to Associate tab

   Click on Choose → select **weka.associations.FPGrowth**.
5. Select FP-Growth
6. Configure FP-Growth Parameters
   - **minSupport** → set as fraction (e.g., `0.2` = 20% support threshold).
   - **numRulesToFind** → set how many rules you want (e.g., `10`).
   - **metricType** → choose confidence, lift, or leverage (default is confidence).
   - **minMetric** → set minimum confidence (e.g., `0.5`).
7. Run FP-Growth



Dataset:

```
@relation market_basket

@attribute Milk {t,f}
@attribute Bread {t,f}
@attribute Beer {t,f}
@attribute Diaper {t,f}
@attribute Eggs {t,f}
@attribute Butter {t,f}

@data
t,t,t,f,f,f
t,t,t,t,f,f
t,t,f,t,f,f
f,t,t,t,t,f
t,f,t,t,f,f
t,t,f,f,t,f
t,t,t,f,t,f
t,f,f,t,f,f
t,t,f,f,f,t
f,t,t,f,t,f
t,f,t,t,f,f
t,t,f,f,f,f
f,t,t,t,f,f
t,t,f,t,f,f
t,t,t,t,f,f
f,t,f,t,f,f
t,f,t,f,f,t
t,t,f,t,t,f
t,f,t,f,t,f
f,t,t,f,f,t
```

15. Experiment to compare the performance of various data mining algorithms on the give database.
    1. Open **WEKA → Explorer → Open File → employee_promotion.arff**
    2. Set **Promotion** as the class attribute
    3. Run each algorithm:
        - **J48** (trees)
        - **NaiveBayes**
        - **IBk** (default k=1, then try k=3)
        - **SMO** (SVM)
    4. Compare metrics (Accuracy, Precision, Recall, F1).

Dataset:

```
@relation employee_promotion

@attribute Experience numeric
@attribute Performance {Low,Medium,High}
@attribute TrainingHours numeric
@attribute Department {HR,Sales,IT,Finance}
@attribute Promotion {Yes,No}

@data
2,Low,5,Sales,No
5,Medium,12,HR,No
7,High,20,IT,Yes
10,High,18,Finance,Yes
3,Low,6,Sales,No
8,High,15,IT,Yes
4,Medium,8,HR,No
6,Medium,10,Finance,Yes
1,Low,3,Sales,No
9,High,22,IT,Yes
2,Low,4,HR,No
5,Medium,9,Finance,Yes
7,High,19,IT,Yes
3,Medium,7,Sales,No
8,High,16,Finance,Yes
```

4,Low,5,HR,No

6,Medium,12,Sales,No

9,High,20,IT,Yes

10,High,21,Finance,Yes

2,Low,6,Sales,No

5,Medium,10,HR,No

7,High,18,IT,Yes

3,Medium,8,Finance,No

8,High,19,IT,Yes

4,Low,6,Sales,No

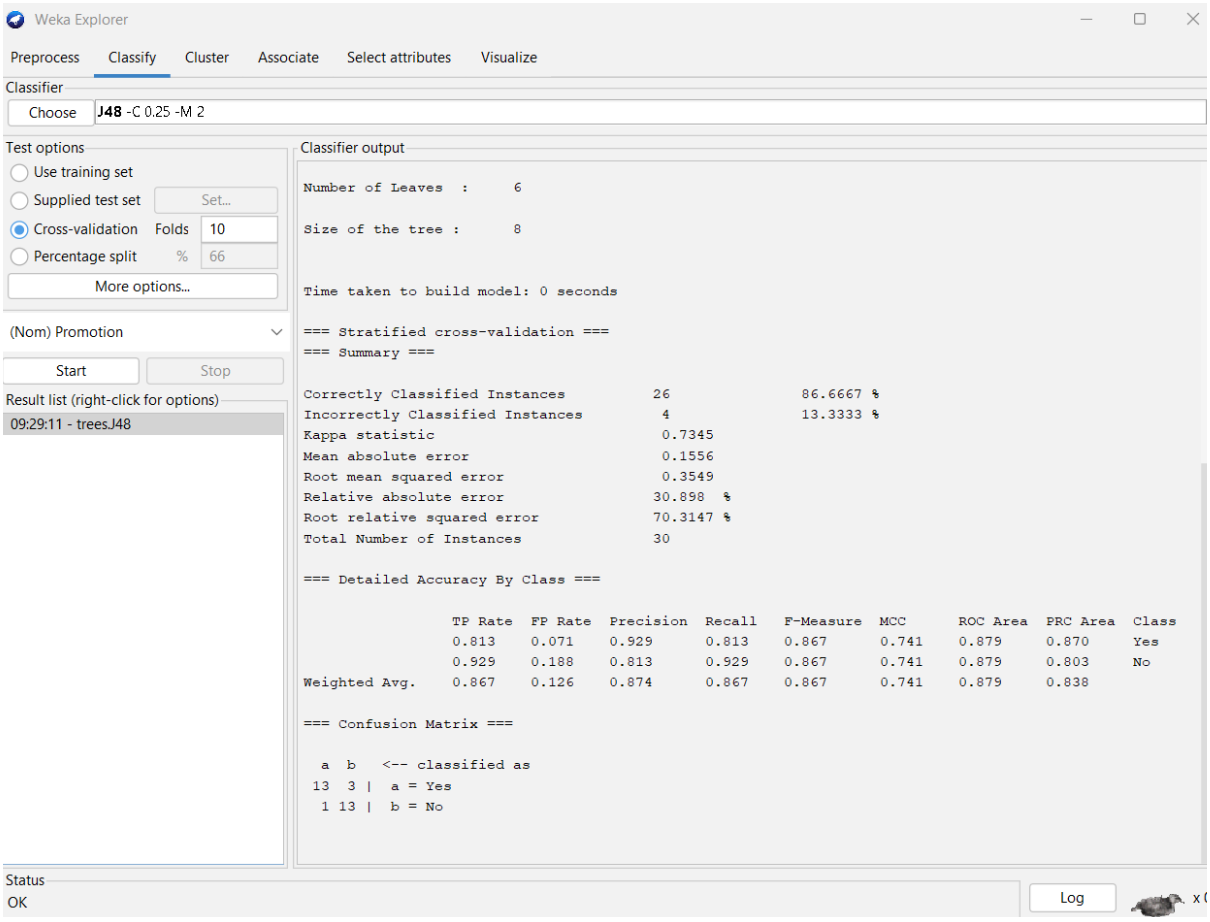
6,Medium,13,Finance,Yes

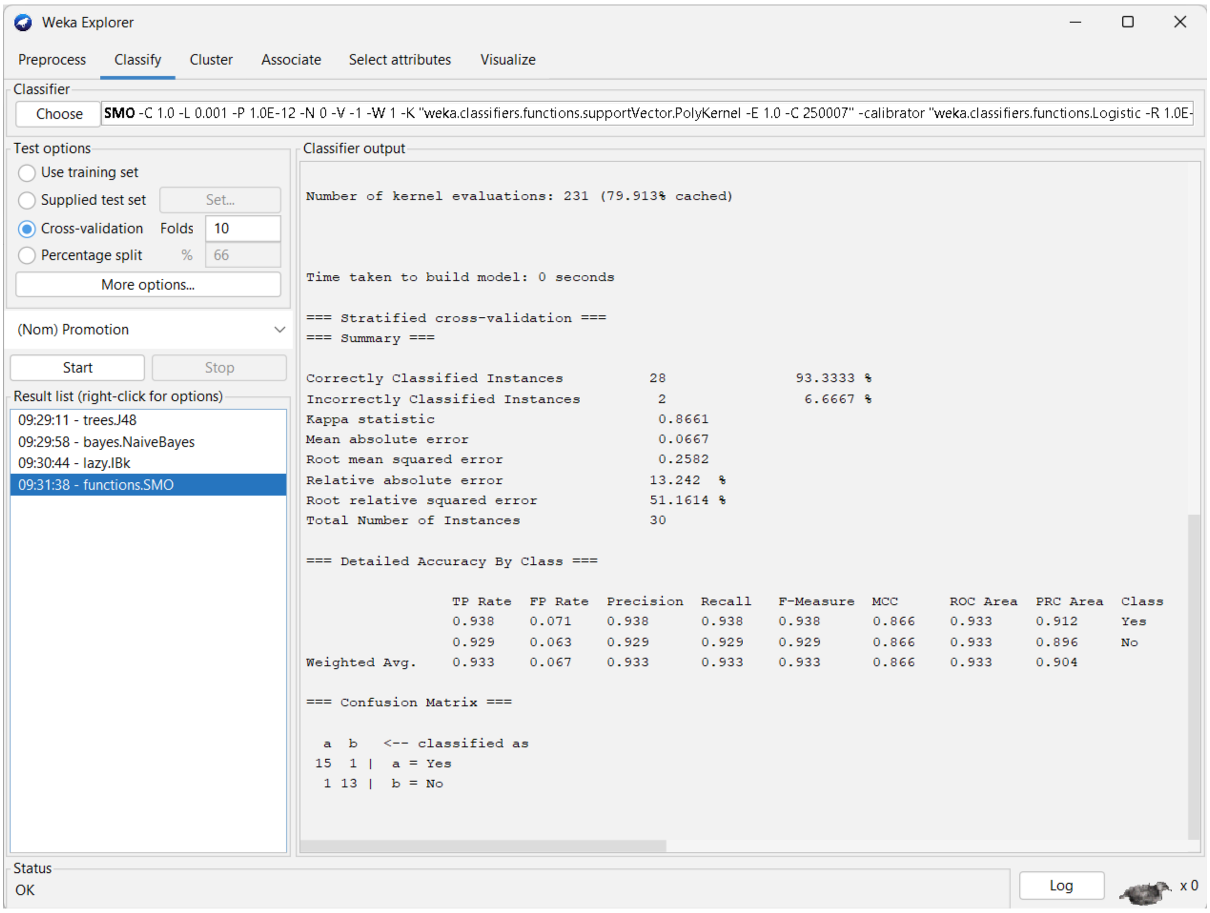
9,High,20,IT,Yes

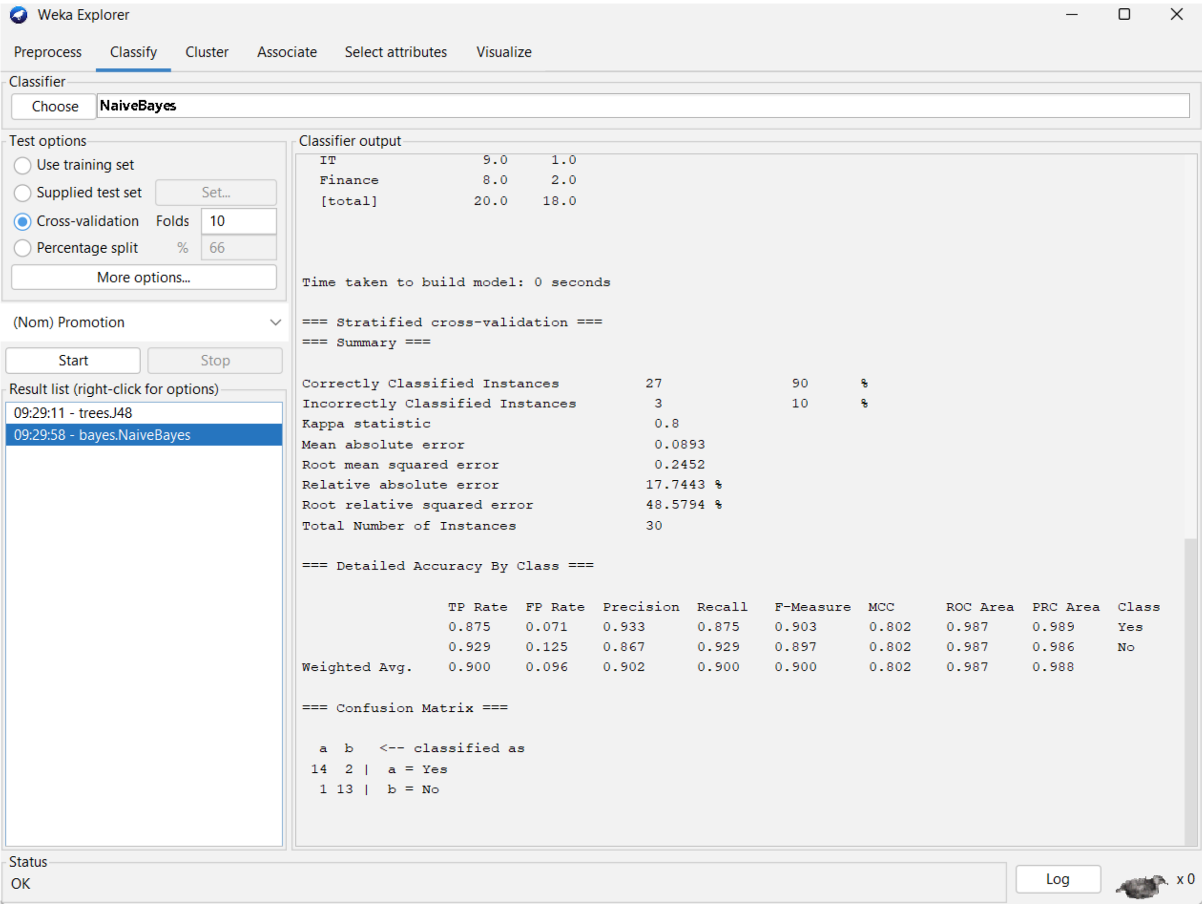
10,High,22,Finance,Yes

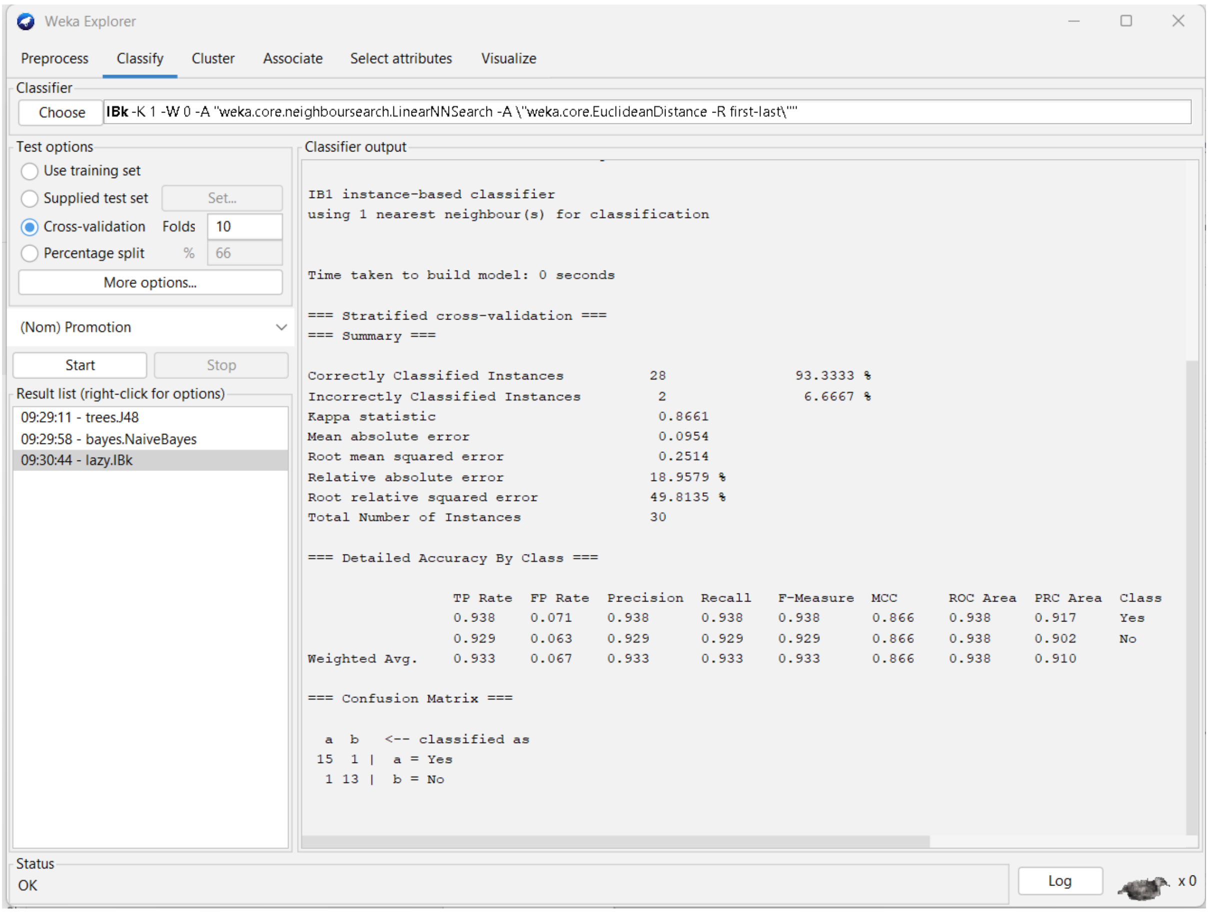
1,Low,4,Sales,No

7,Medium,12,HR,Yes




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Promotion
Start
Stop
Result list (right-click for options)
09:29:11 - trees.J48
09:29:58 - bayes.NaiveBayes
09:30:44 - lazy.IBk
09:31:38 - functions.SMO
Classifier output
Number of kernel evaluations: 231 (79.913% cached)
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 28 93.3333 %
Incorrectly Classified Instances 2 6.6667 %
Kappa statistic 0.8661
Mean absolute error 0.0667
Root mean squared error 0.2582
Relative absolute error 13.242 %
Root relative squared error 51.1614 %
Total Number of Instances 30
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.938 0.071 0.938 0.938 0.938 0.866 0.933 0.912 Yes
0.929 0.063 0.929 0.929 0.929 0.866 0.933 0.896 No
Weighted Avg. 0.933 0.067 0.933 0.933 0.933 0.866 0.933 0.904
=== Confusion Matrix ===
a b <-- classified as
15 1 | a = Yes
1 13 | b = No
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Promotion
Start
Stop
Result list (right-click for options)
09:29:11 - trees.J48
09:29:58 - bayes.NaiveBayes
Classifier output
IT 9.0 1.0
Finance 8.0 2.0
[total] 20.0 18.0
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 27 90 %
Incorrectly Classified Instances 3 10 %
Kappa statistic 0.8
Mean absolute error 0.0893
Root mean squared error 0.2452
Relative absolute error 17.7443 %
Root relative squared error 48.5794 %
Total Number of Instances 30
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.875 0.071 0.933 0.875 0.903 0.802 0.987 0.989 Yes
0.929 0.125 0.867 0.929 0.897 0.802 0.987 0.986 No
Weighted Avg. 0.900 0.096 0.902 0.900 0.900 0.802 0.987 0.988
=== Confusion Matrix ===
a b <-- classified as
14 2 | a = Yes
1 13 | b = No
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Promotion
Start
Stop
Result list (right-click for options)
09:29:11 - trees.J48
09:29:58 - bayes.NaiveBayes
09:30:44 - lazy.IBk
Classifier output
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 28 93.3333 %
Incorrectly Classified Instances 2 6.6667 %
Kappa statistic 0.8661
Mean absolute error 0.0954
Root mean squared error 0.2514
Relative absolute error 18.9579 %
Root relative squared error 49.8135 %
Total Number of Instances 30
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.938 0.071 0.938 0.938 0.938 0.866 0.938 0.917 Yes
0.929 0.063 0.929 0.929 0.929 0.866 0.938 0.902 No
Weighted Avg. 0.933 0.067 0.933 0.933 0.933 0.866 0.938 0.910
=== Confusion Matrix ===
a b <-- classified as
15 1 | a = Yes
1 13 | b = No
Status
OK
Log
x 0