

## DATA MINING AND VISUALIZATION LABORATORY

### 2. Experiment using WEKA tool.

Consider the following data set.

No.	eid Numeric	ename Nominal	salary Numeric	exp Numeric	address Nominal
1	101.0	raj	10000.0	4.0	pdtr
2	102.0	ramu	15000.0	5.0	pdtr
3	103.0	anil	12000.0	3.0	kdp
4	104.0	sunil	13000.0	3.0	kdp
5	105.0	rajiv	16000.0	6.0	kdp
6	106.0	sunitha	15000.0	5.0	nlr
7	107.0	kavitha	12000.0	3.0	nlr
8	108.0	suresh	11000.0	5.0	gtr
9	109.0	ravi	12000.0	3.0	gtr
10	110.0	ramana	11000.0	5.0	gtr
11	111.0	ram	12000.0	3.0	kdp
12	112.0	kavya	13000.0	4.0	kdp
13	113.0	navya	14000.0	5.0	kdp

i). Use the data sources, like ARFF, XML ARFF files.

**Prepare Dataset in ARFF Format:**

@relation employee

@attribute eid numeric

@attribute salary numeric

@attribute exp numeric

@attribute address {pdtr,kdp,nlr,gtr} % class attribute

@data

101.0, 15000.0, 4.0, pdtr

102.0, 15000.0, 5.0, kdp

103.0, 12000.0, 3.0, kdp

104.0, 13000.0, 6.0, kdp

105.0, 13000.0, 4.0, kdp

106.0, 14000.0, 6.0, nlr

107.0, 15000.0, 5.0, nlr

108.0, 12000.0, 3.0, gtr

109.0, 12000.0, 3.0, gtr

110.0, 13000.0, 4.0, kdp

111.0, 13000.0, 4.0, kdp

112.0, 14000.0, 5.0, kdp

113.0, 14000.0, 5.0, kdp

**Note:- Save file as employee.arff**

## Load Data in WEKA

- Open **WEKA Explorer**.
- Click **Open file**, select your employee.arff.

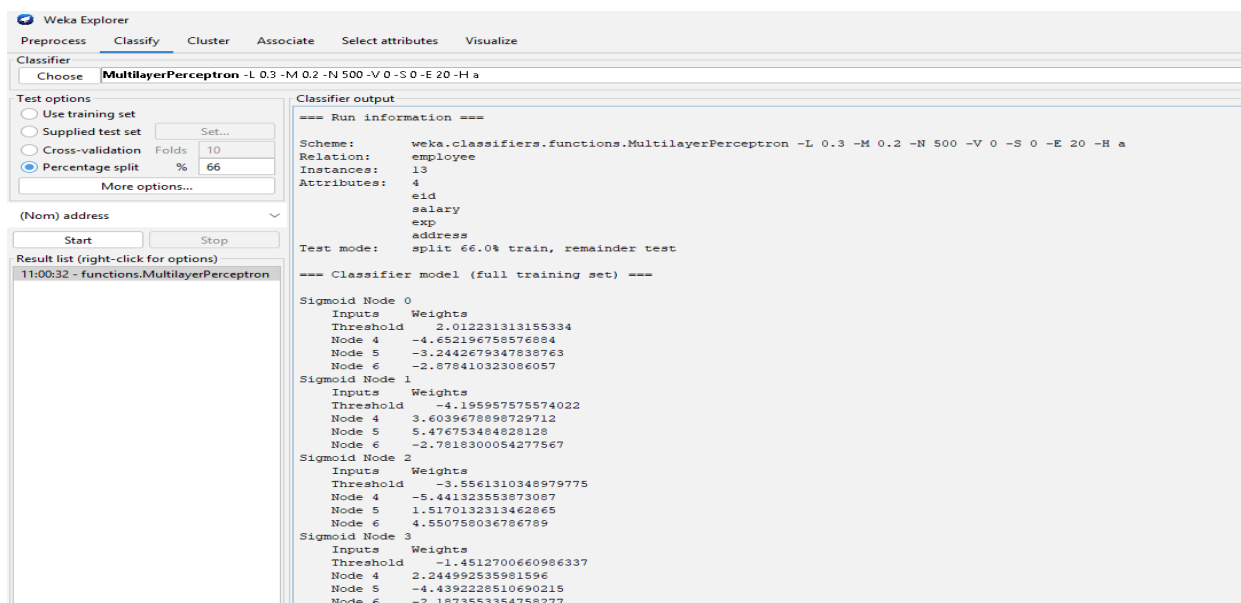
**After Preparing the Dataset Do the following:**

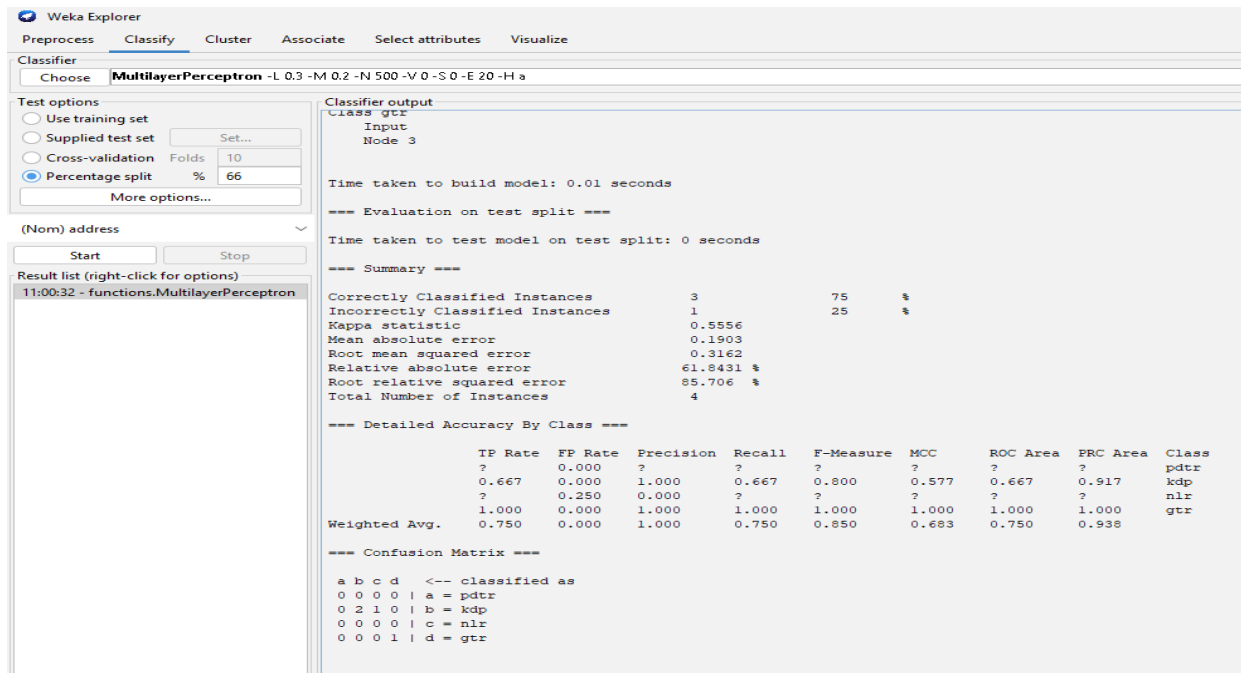
**a). Classify , Invoke MultiLayerPerception.**

**b). Build neural network GUI as below .**

### Build Neural Network

- Go to the **Classify** tab.
- Choose **Classifier > functions > MultilayerPerceptron**.
- Click **Start** to train with default parameters.
- View output, such as accuracy, confusion matrix.





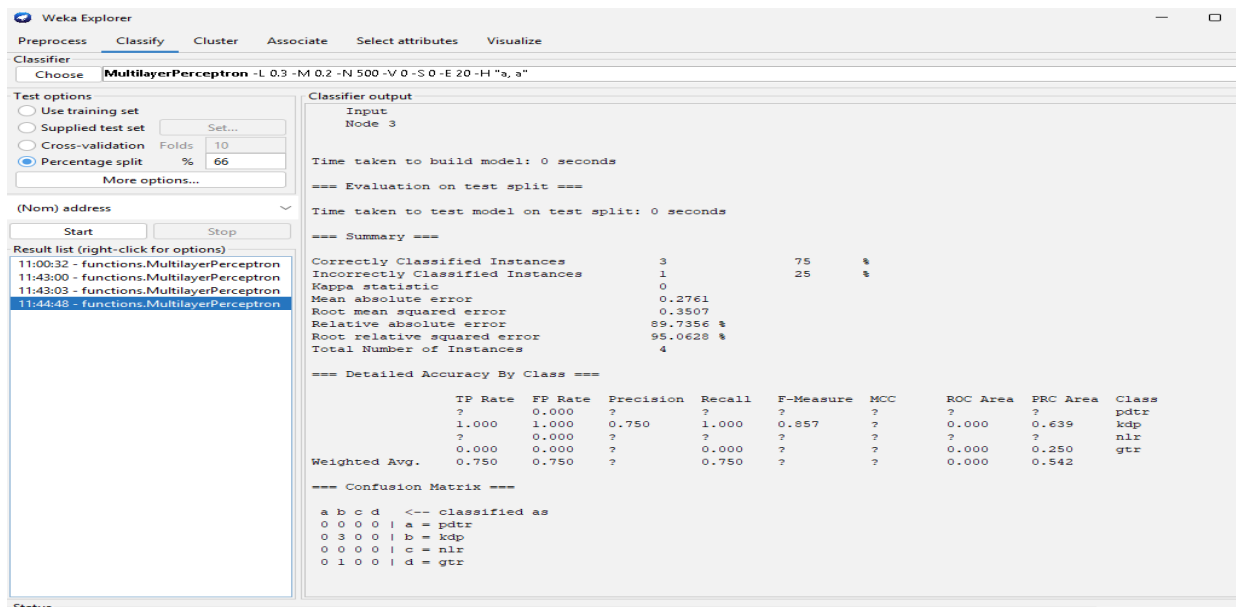
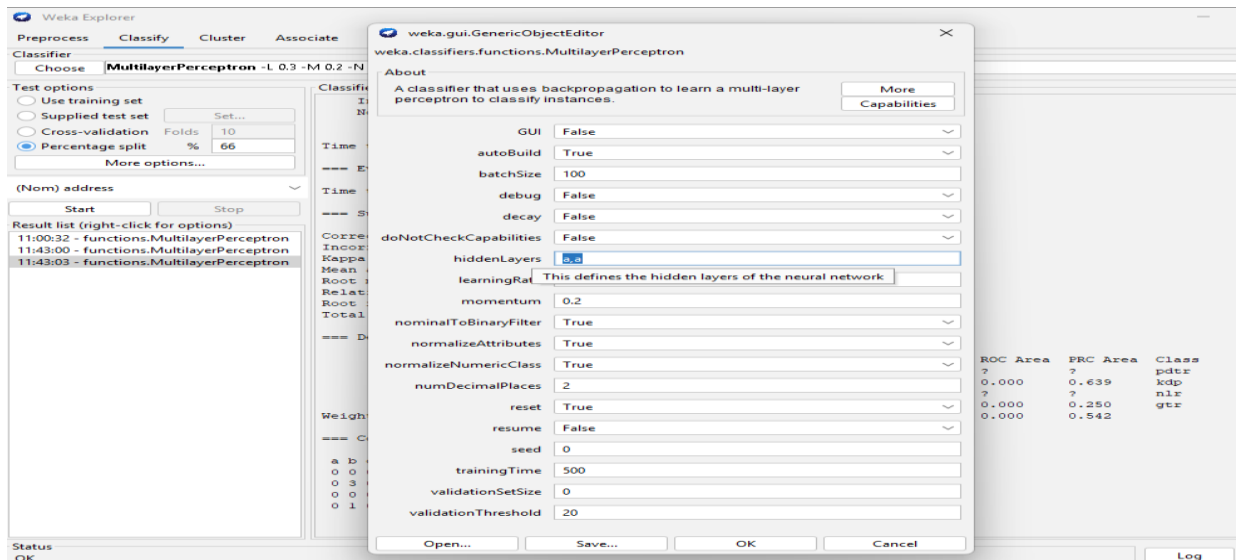
## ii) Beginning the process of editing the network to add a second hidden layer

### Edit network architecture:

- Click on the **MultilayerPerceptron** name → press **Edit**.
- In the “Hidden Layers” field, type a,a (meaning 2 hidden layers, each with number of nodes equal to (attribs + classes) / 2).
- Alternatively, specify exact nodes, e.g., 5,3.
- Click **OK** and **Start**.

## iii) The finished network with two hidden layers.

- Check output for improved accuracy and structure.



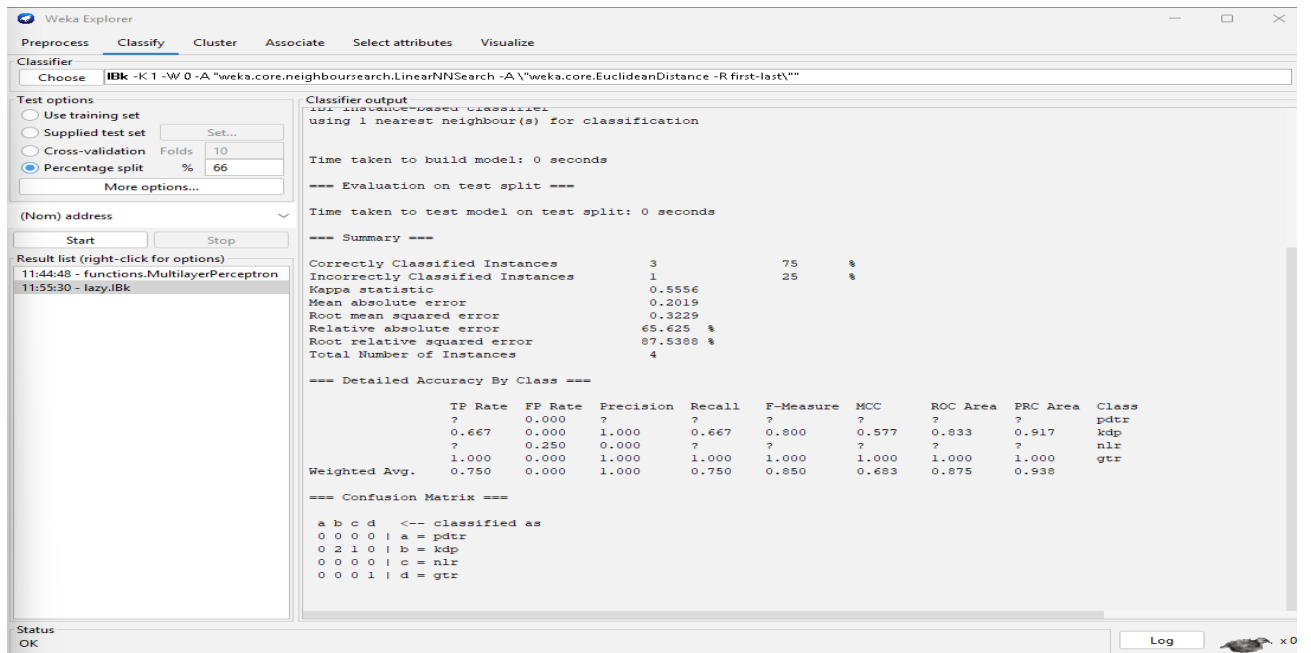
#### iv) Apply Lazy classifier, multi instance classifier.

##### Lazy Classifier

- Go to **Classify** tab.
- Choose **lazy > IBk** (k-NN).
- Click **Start** to train.

##### Multi-instance Classifier

- WEKA's **multi-instance classifiers** are available under **mi > MultiInstance....**
- Load dataset suitable for multi-instance (typically requires different format).
- Use **multi-instance** options if applicable.

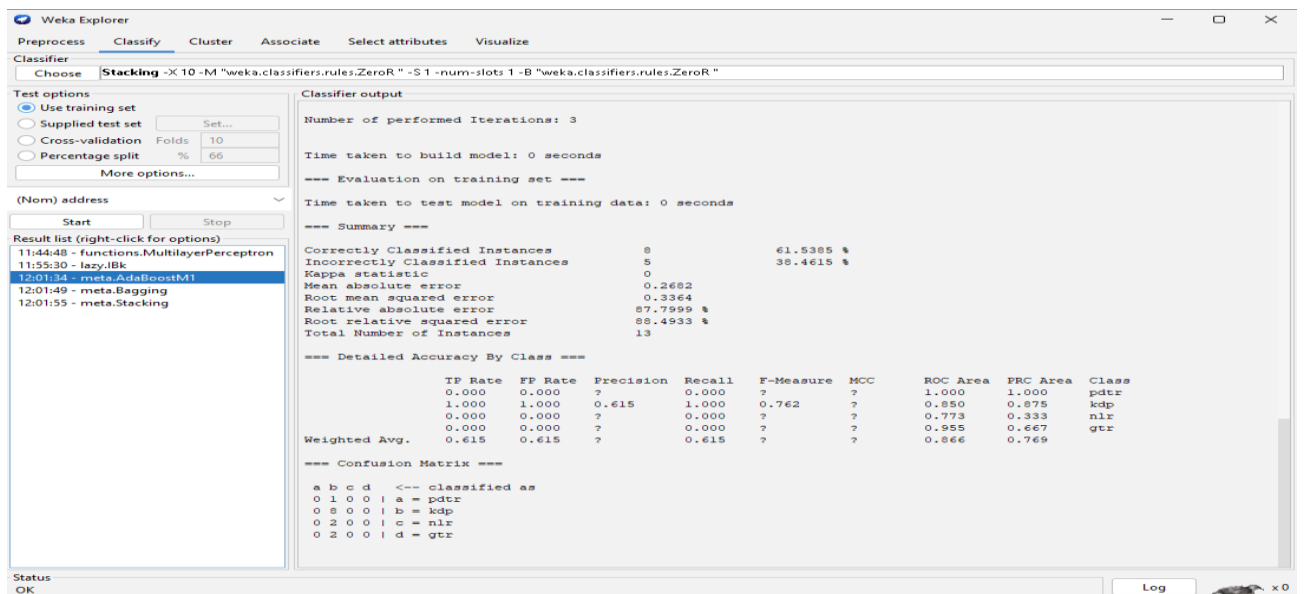


## v) Apply any MetaLearning Algorithm.

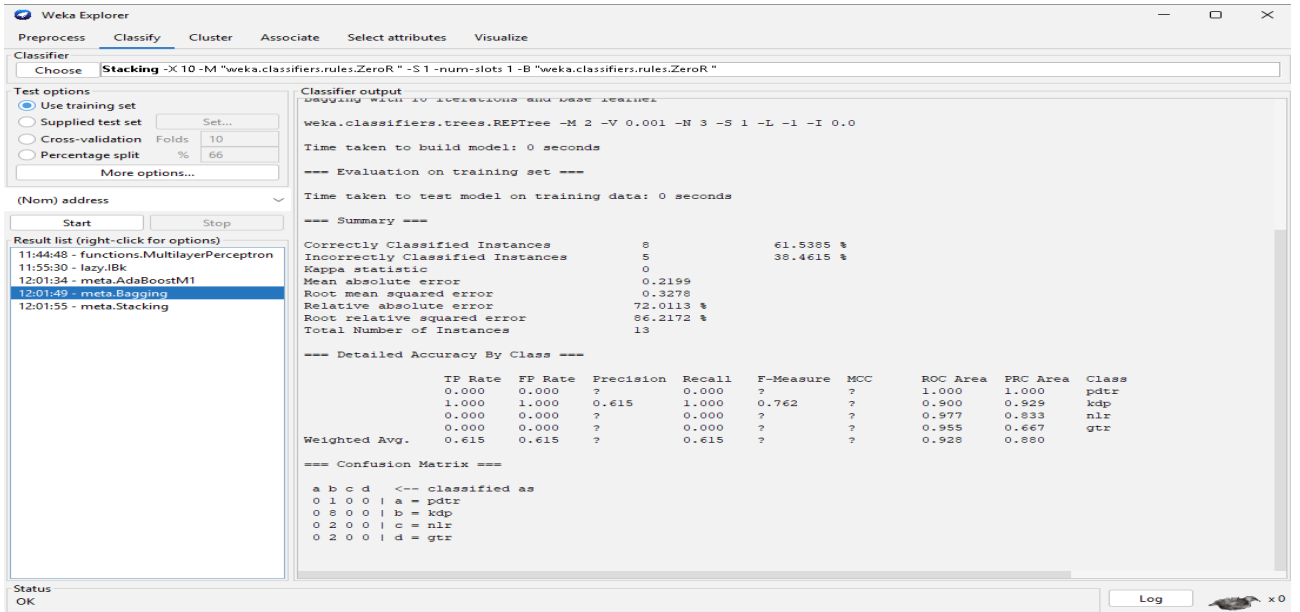
### Meta Classifiers

- Under **Classify**, select **meta**.
- Examples: **AdaBoostM1**, **Bagging**, **Stacking**.
- Select one, e.g., **AdaBoostM1**.
- Set base classifier (like J48).
- Click **Start**.

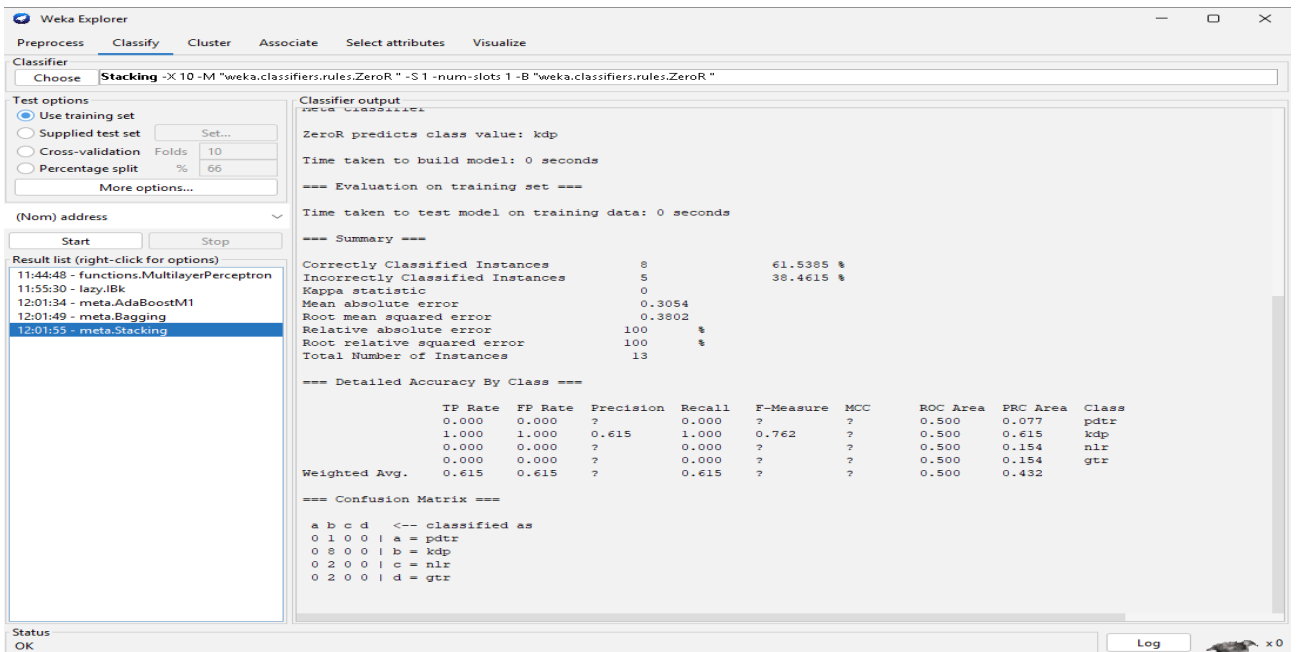
#### ➔ AdaBoostM1:



## ➔ Bagging:



### → Stacking:



**vi) Optimize base classifier's performance.**

Use **Grid Search** or **CV Parameter Selection**:

- Go to **Tools > CVPParameterSelection**.
- Choose base classifier (e.g., MultilayerPerceptron).
- Select parameters to tune (e.g., learning rate).

- Run to find optimal parameters.

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'MultilayerPerceptron - L 0.3 - M 0.2 - N 500 - V 0 - S 0 - E 20 - H a'. The 'Test options' section shows 'Use training set' selected. The 'Classifier output' section displays the following results:

Time taken to build model: 0 seconds  
Time taken to test model on training data: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances	12	92.3077 %
Incorrectly Classified Instances	1	7.6923 %
Kappa statistic	0.8725	
Mean absolute error	0.1134	
Root mean squared error	0.1921	
Relative absolute error	37.1428 %	
Root relative squared error	50.542 %	
Total Number of Instances	13	

=== 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	1.000	pdtr
0.875	0.000	1.000	0.875	0.875	0.933	0.854	0.950	0.975	kdp
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	nlr
Weighted Avg.	1.000	0.091	0.667	1.000	0.800	0.775	0.809	0.583	gtr

=== Confusion Matrix ===

```

a b c d <-- classified as
1 0 0 0 | a = pdtr
0 7 0 1 | b = kdp
0 0 2 0 | c = nlr
0 0 0 2 | d = gtr

```

The 'Result list' on the left shows the following entries:

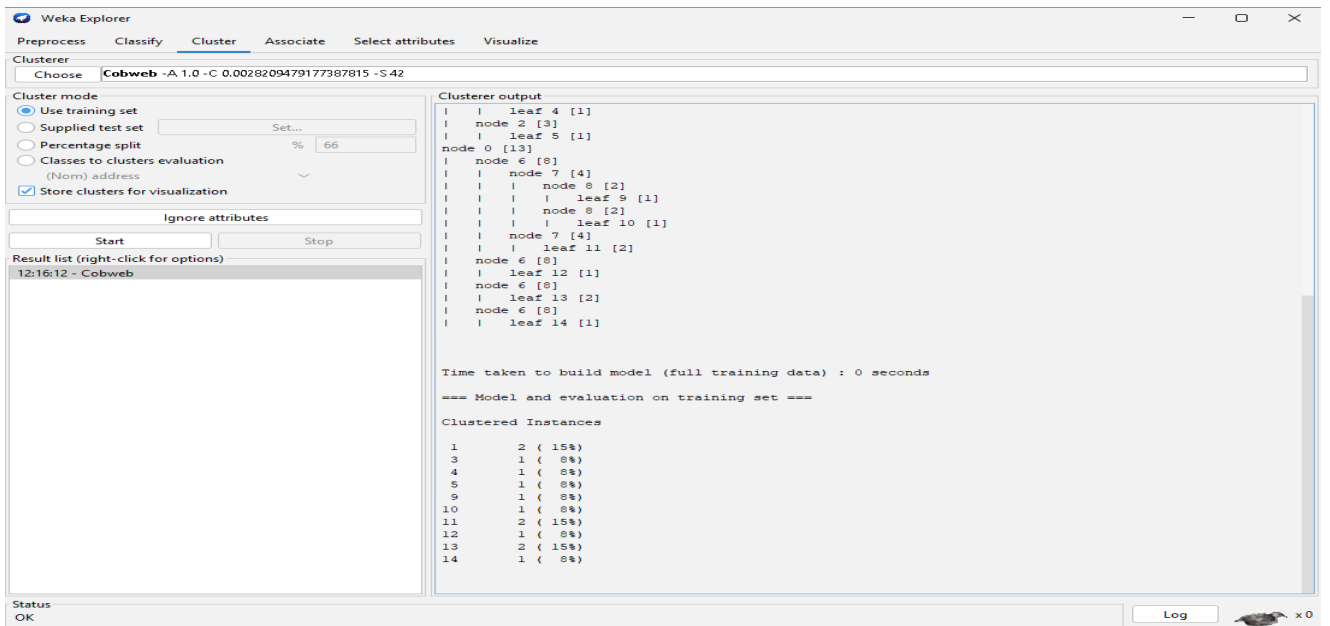
- 11:44:48 - functions.MultilayerPerceptron
- 11:55:30 - lazy.IBk
- 12:01:34 - meta.AdaBoostM1
- 12:01:49 - meta.Bagging
- 12:01:55 - meta.Stacking
- 12:13:56 - functions.MultilayerPerceptron

## vii) Use clustering algorithm such as Cobweb, and Hierarchical Cluster.

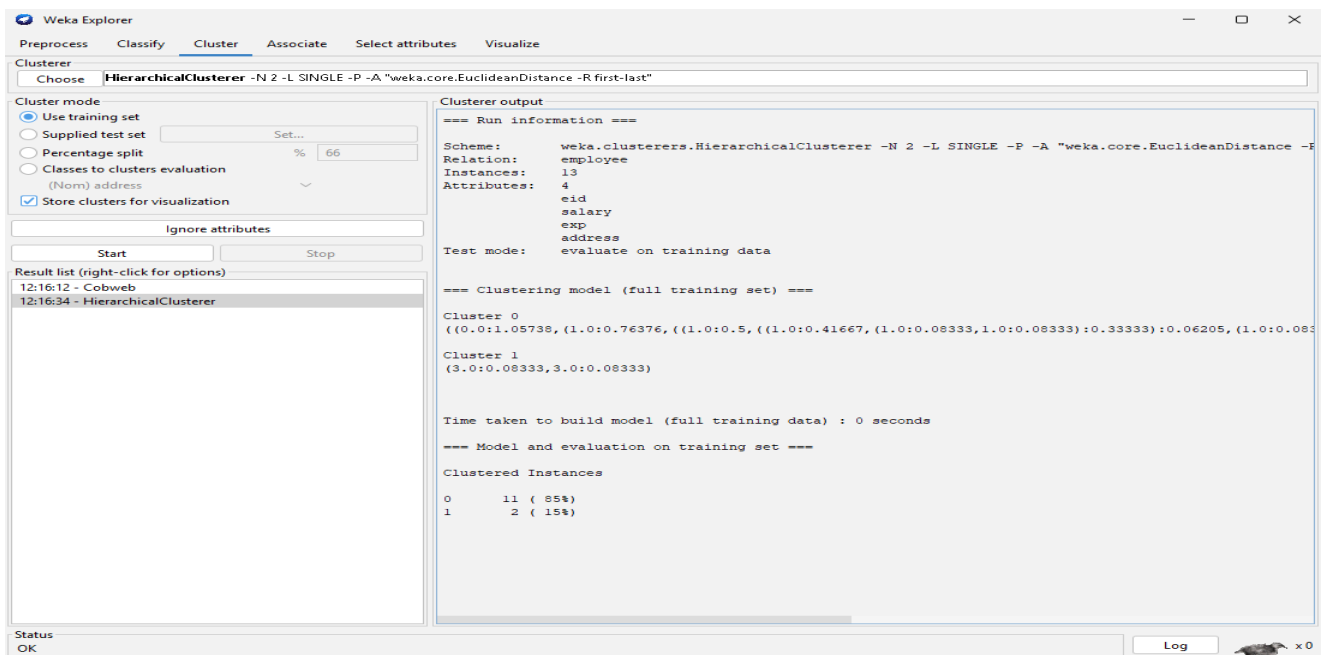
Clustering:

- Go to **Cluster** tab.
- Select **Cobweb**.
- Click **Start** to cluster dataset.
- Repeat with **HierarchicalClusterer**.
- View clusters formed.

➔ **Cobweb:**



➔ **HierarchicalClusterer:**

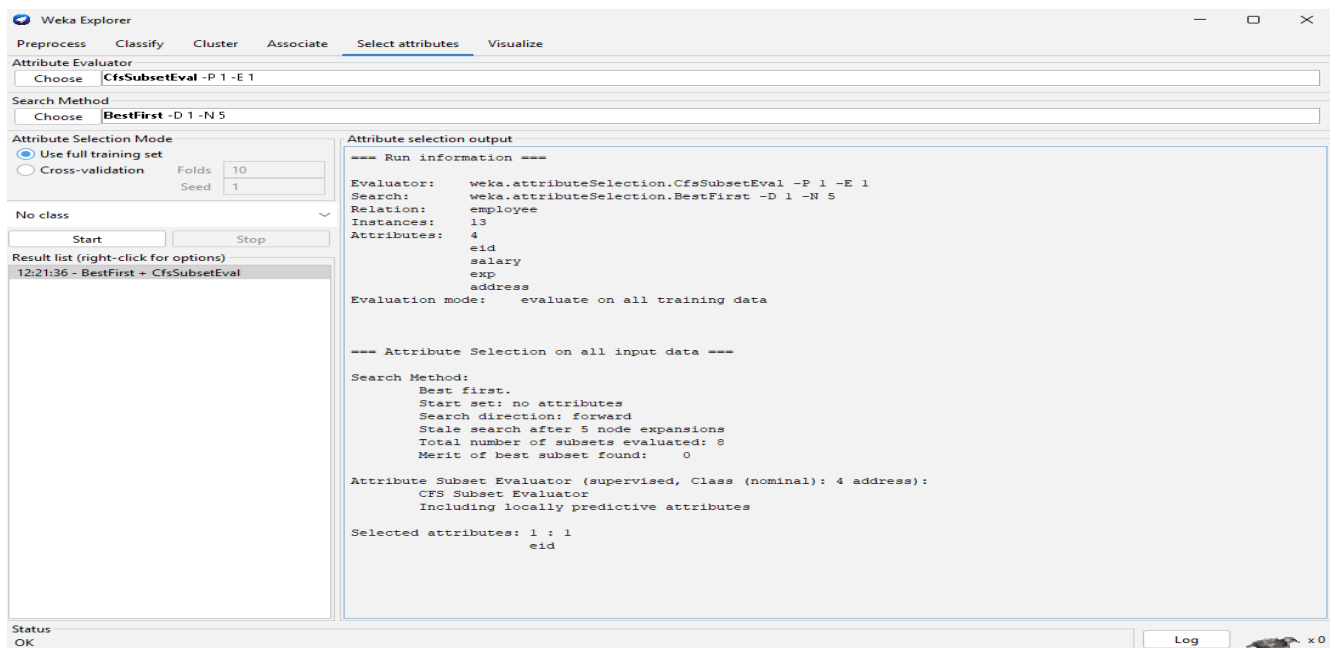




**viii) Select attribute by specifying an evaluator and a search method.**

### Select Attributes

- Go to **Select attributes** tab.
- Choose evaluator: e.g., **CfsSubsetEval**.
- Choose search method: e.g., **BestFirst**.
- Click **Start** to select important attributes.



**ix) Insert 30 more records in this file. Perform clustering on this new dataset. Identify optimum number of clusters. After identification of optimum number of clusters, prepare clustering on this number.**

- Edit your ARFF file to add 30 new employee records.
- Reload dataset in WEKA.
- Use clustering algorithms (e.g., **EM**, **k-Means**) to find clusters.
- Use **Cluster evaluation** to identify optimum number of clusters (e.g., look for highest silhouette score or lowest within cluster sum of squares).
- Perform clustering with optimum clusters.

**Weka Explorer**  
Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer  
Choose **EM** -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode  
☐ Use training set  
☐ Supplied test set Set...  
☐ Percentage split % 66  
☒ Classes to clusters evaluation (Nom) address  
☒ Store clusters for visualization

Ignore attributes  
Start Stop

Result list (right-click for options)  
 18:03:56 - SimpleKMeans  
 18:04:14 - EM

Clusterer output  
 Number of clusters selected by cross validation: 2  
 Number of iterations performed: 22

Attribute	Cluster 0 (0.65)	Cluster 1 (0.35)
eid		
mean	117.6276	130.2614
std. dev.	11.785	8.8423
salary		
mean	13699.0816	16315.6798
std. dev.	1011.6711	834.9244
exp		
mean	4.4941	8.2406
std. dev.	1.1907	1.2872

Time taken to build model (full training data) : 0.17 seconds  
 === Model and evaluation on training set ===

Clustered Instances

Cluster	Count	Percentage
0	28	65%
1	15	35%

Log likelihood: -14.24018

Class attribute: address  
 Classes to Clusters:

**Weka Explorer**  
Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer  
Choose **EM** -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode  
☐ Use training set  
☐ Supplied test set Set...  
☐ Percentage split % 66  
☒ Classes to clusters evaluation (Nom) address  
☒ Store clusters for visualization

Ignore attributes  
Start Stop

Result list (right-click for options)  
 18:03:56 - SimpleKMeans  
 18:04:14 - EM

Clusterer output  
 Final cluster centroids:

Attribute	Full Data (43.0)	Cluster 0 (24.0)	Cluster 1 (19.0)
eid	122	115.125	130.6842
salary	14604.6512	13520.8333	15973.6842
exp	5.7907	4.2917	7.6842

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

Clustered Instances

Cluster	Count	Percentage
0	24	56%
1	19	44%

Class attribute: address  
 Classes to Clusters:

```

0 1 <-- assigned to cluster
5 4 | pdtr
10 6 | kdp
5 4 | nlr
4 5 | gtr

Cluster 0 <-- kdp
Cluster 1 <-- gtr
  
```

Incorrectly clustered instances : 28.0 65.1163 %

Status  
OK

**x) Perform data analysis on the result obtained and prepare an analysis report for the same.**

## Analyze Results

- Review output clusters.
- Compare clusters with domain knowledge.
- Identify patterns (e.g., clusters by salary or experience).

- Prepare summary report discussing:
  - Cluster characteristics.
  - Accuracy of classifiers.
  - Attribute importance.
  - Insights for decision-making.

## **Result Analysis – Employee Dataset**

1. Algorithm Used: K-Means ( $k = 4$ )

2. Attributes: salary, experience, address

3. Cluster Summary:

- Cluster 1: Low salary (₹12k–₹13k), 2–4 yrs exp, kdp region
- Cluster 2: Mid salary (₹14k–₹15k), 5–6 yrs exp, pdtr region
- Cluster 3: High salary (₹16k–₹17k), 7–10 yrs exp, nlr/gtr

4. Key Patterns:

- Salary increases with experience
- Address influences salary range

5. Attribute Importance:

- Experience → highest impact
- Salary → moderate impact
- Address → low impact

6. Insights:

- HR can use results for pay scale design and promotions
- Useful for identifying training and career growth paths

7. Conclusion:

- Clustering logically grouped employees by experience & region

## Clustering Algorithm

- Go to the “Cluster” tab
- Choose algorithm:
- Cluster mode: Use training set
- Choose → SimpleKMeans
  - Click “SimpleKMeans” and set:
    - Number of clusters (k) = 4
    - Distance function = Euclidean
    - Keep other options default

The screenshot shows the Weka Explorer interface with the 'Cluster' tab selected. The 'SimpleKMeans' algorithm is chosen, and the 'Use training set' option is selected under 'Cluster mode'. The 'Store clusters for visualization' checkbox is checked. The 'Clusterer output' pane displays the following text:

```
kMeans
=====
Number of iterations: 6
Within cluster sum of squared errors: 26.475186308497758

Initial starting points (random):
Cluster 0: 104,13000,6,kdp
Cluster 1: 137,14000,5,gtr

Missing values globally replaced with mean/mode

Final cluster centroids:
Attribute      Full Data      Cluster#
              (43.0)        (26.0)        (17.0)
=====
eid            122      119.3846    127.5294
salary        14604.6512 14134.6154 15323.5294
exp            5.7907     5.1154     6.8235
address       kdp        kdp        gtr

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

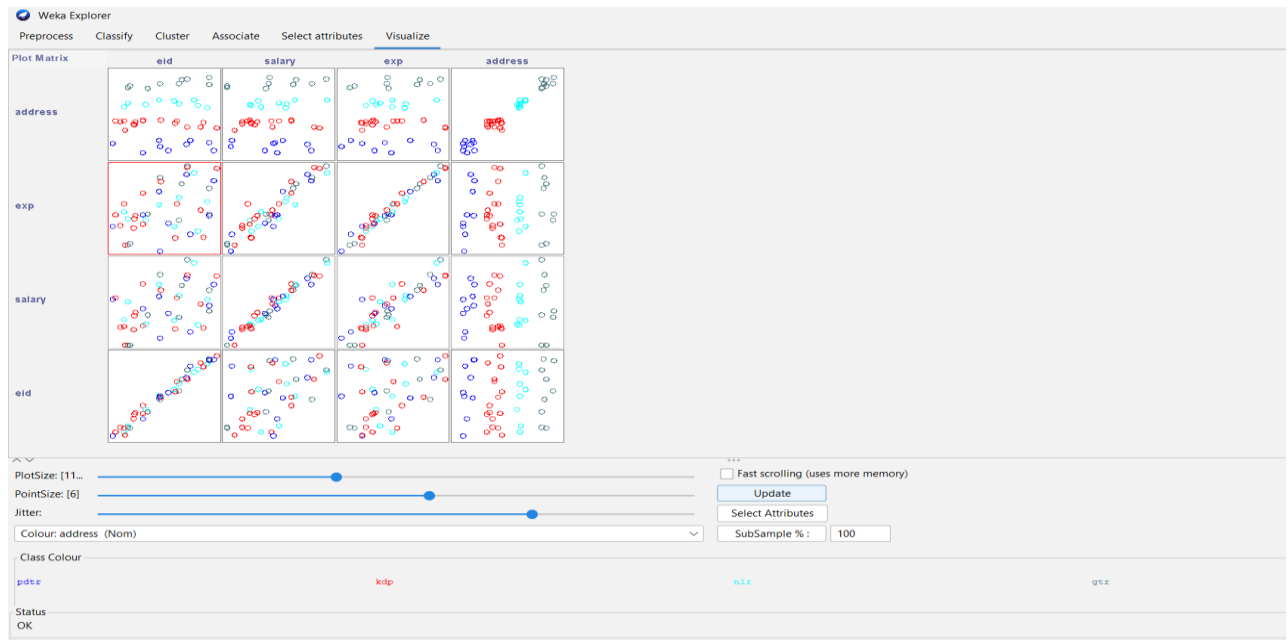
=== Model and evaluation on training set ===

Clustered Instances
0      26 ( 60%)
1      17 ( 40%)
```

The 'Result list' on the left shows three entries: '18:03:56 - SimpleKMeans', '18:04:14 - EM', and '18:24:03 - SimpleKMeans'.

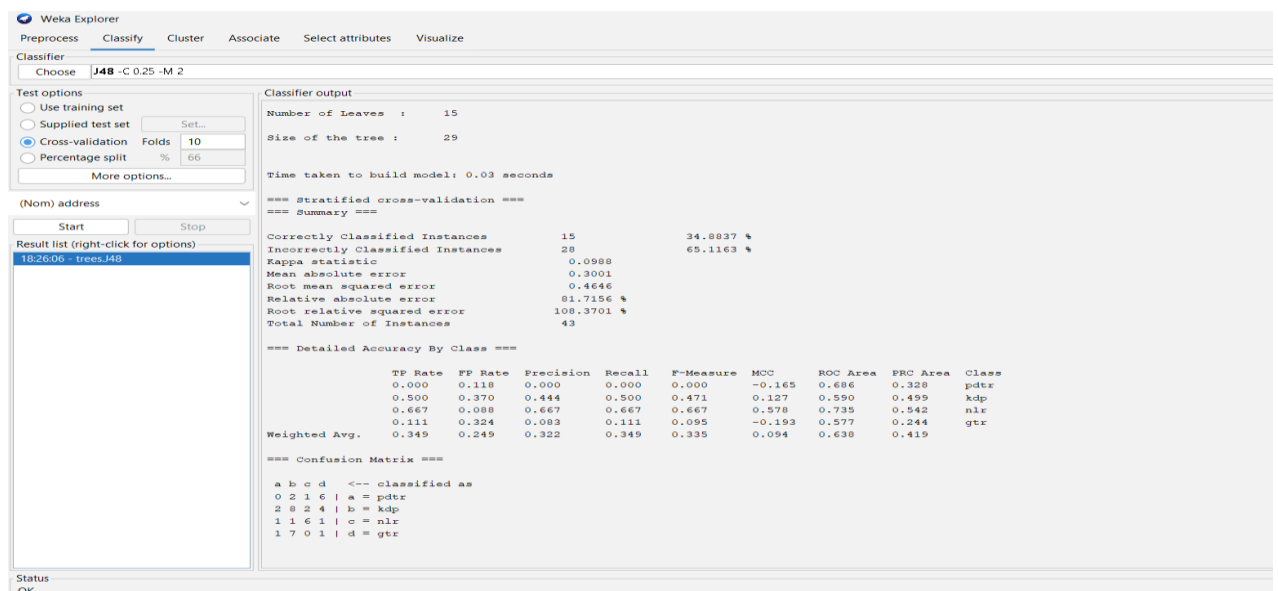
## Visualize the Clusters

- Click “Visualize” to view graphs (e.g., salary vs exp, color-coded by cluster).  
→ You’ll see natural grouping — higher salary with more experience.



## Evaluate with a Classifier:

- Go to Classify tab
- Choose algorithm → e.g., J48 (Decision Tree) or NaiveBayes
- Set class attribute = address
- Click Start → You'll get accuracy %, confusion matrix, and attribute importance.



**Weka Explorer**  
Preprocess Classify Cluster Associate Select attributes Visualize

Classifier Choose **NaiveBayes**

Test options  
☐ Use training set  
☐ Supplied test set  
☒ Cross-validation Folds **10**  
☐ Percentage split % **66**  
 More options...

(Nom) address  
 Start Stop

Result list (right-click for options)  
 18:26:06 - trees.J48  
 18:26:29 - bayes.NaiveBayes

Classifier output

Stat.	tree	nb	nb2	nb3
weight sum	5	16	5	5
precision	1	1	1	1

Time taken to build model: 0 seconds

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

Correctly Classified Instances	13	30.2326 %
Incorrectly Classified Instances	30	69.7674 %
Kappa statistic	-0.0023	
Mean absolute error	0.3767	
Root mean squared error	0.4675	
Relative absolute error	102.5942 %	
Root relative squared error	109.0433 %	
Total Number of Instances	43	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.000	0.088	0.000	0.000	0.000	-0.141	0.297	0.164	pdtr
	0.625	0.556	0.400	0.625	0.488	0.068	0.523	0.433	kdp
	0.111	0.206	0.125	0.111	0.118	-0.099	0.399	0.185	nlf
	0.222	0.147	0.286	0.222	0.250	0.083	0.435	0.229	gtr
Weighted Avg.	0.302	0.299	0.235	0.302	0.258	-0.008	0.431	0.282	

=== Confusion Matrix ===

a	b	c	d	<-- classified as
0	6	1	2	a = pdtr
0	10	4	2	b = kdp
2	5	1	1	c = nlf
1	4	2	2	d = gtr

## xi) Apply Apriori, Interpret the results.

- Go to **\*\*Associate\*\*** tab.
- Choose **\*\*Apriori\*\***.
- Click **\*\*Start\*\***.
- Interpret frequent itemsets, rules generated.

**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 ...)  
 18:39:32 - 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: employee-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last
Instances: 43
Attributes: 4
  eid
  salary
  exp
  address
  
```

=== Associator model (full training set) ===

Apriori  
 =====

Minimum support: 0.1 (4 instances)  
 Minimum metric <confidence>: 0.9  
 Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 14  
 Size of set of large itemsets L(2): 2

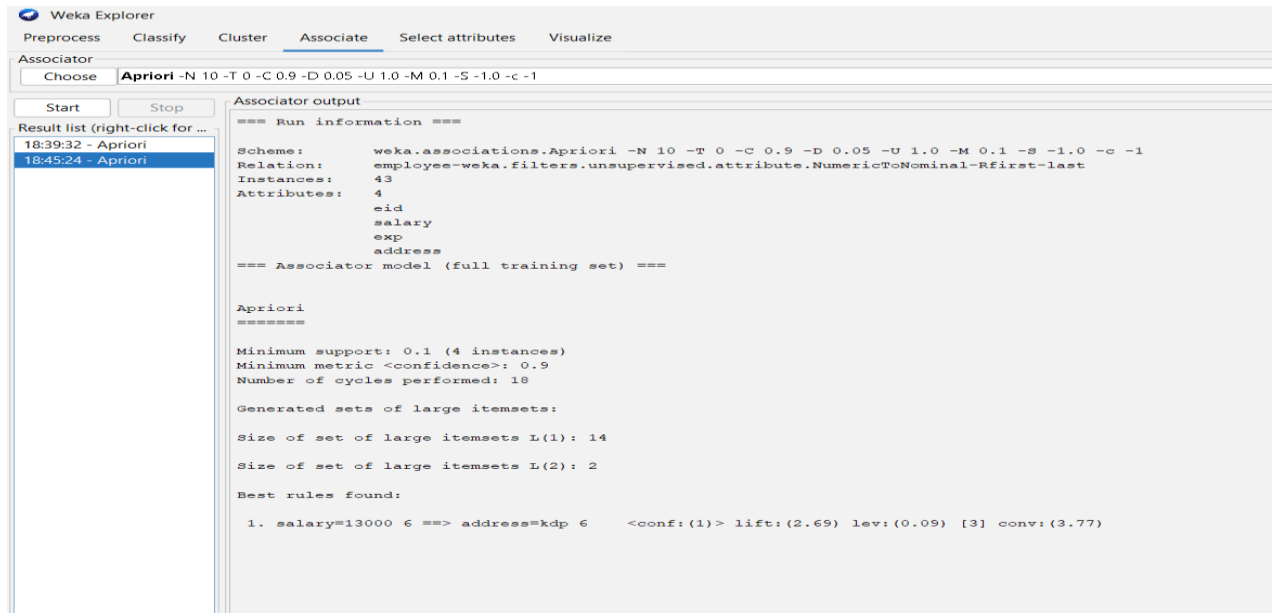
Best rules found:

```

1. salary=13000 6 ==> address=kdp 6 <conf: (1)> lift: (2.69) lev: (0.09) [3] conv: (3.77)
  
```

## xii) Apply Association rules and interpret the results.

- In Associate tab, Apriori shows association rules.
- Analyze rules with high confidence and support.
- Example: Salary = 15000 → Address = nlr (confidence 80%).



### Analysis of Association Rules:

- Used the Apriori algorithm in WEKA to generate association rules.
  - Focused on rules with high confidence and support to identify strong relationships.
  - Example: Salary = 15000 → Address = NLR (confidence = 80%)
- Indicates that employees earning ₹15,000 are highly likely (80%) to live in NLR.
- Association rules help in identifying patterns and correlations in the dataset.
  - Apriori efficiently discovered frequent itemsets and produced interpretable rules.

**xiii) Apply Association mining with the Apriori algorithm and find the best rules with threshold value of support of 50% and confidence of 70%.**

- Configure Apriori parameters:
- minSupport = 0.5
- minMetric (confidence) = 0.7
- Click Start.
- View best rules that satisfy thresholds.



**Steps to fix in WEKA**

1. Remove the eid attribute.
2. Discretize salary and exp into 3–4 bins.
3. Open **Apriori**:
  - **minSupport = 0.1**
  - **minMetric = 0.6**
  - **numRules = 10–20**
4. Click **Start** → you should see **rules like**:
  - Salary=13500-15000 → Address=nlr
  - Exp=5-7 → Address=kdp
  -



**xiv) Interpret the results obtained at Summary. Analyse Precision, Recall, F Score values.**

**After classification runs:**

- Check Classify output for Precision, Recall, F-Measure.
- Analyze which classes perform best.
- Summarize trade-offs (e.g., high precision but low recall).

**Classification Analysis**

- Check **Precision, Recall, F1-Score** in WEKA after classification.
- **Precision** → % of correct predictions for a class (high = fewer false positives).
- **Recall** → % of actual instances correctly identified (high = most positives captured).
- **F1-Score** → balance between precision and recall.
- Analyze **which classes perform best** and which perform poorly.
- **Trade-offs:**
  - High precision, low recall → strict, fewer predictions but mostly correct.
  - High recall, low precision → many predictions but more errors.
- Use these metrics to **evaluate model performance** and guide improvements.

**xv) Install R package at Weka environment and install rpart package.  
Implement decision tree.**

**Step 1: Install R on your system**

1. Download R from CRAN.
2. Install it with default settings.
3. Remember the installation path (you'll need it for WEKA).

**Step 2: Install RPlugin in WEKA**

1. Open **WEKA GUI**.
2. Go to **Tools** → **Package Manager**.
3. In the search box, type **RPlugin**.
4. Click **Install** and wait for it to finish.

5. Restart WEKA to activate the plugin.

### **Step 3: Install rpart package in R**

1. Open the **R console**.
2. Run:
3. `install.packages("rpart")`
4. Wait for the package to install. You can check by running:
5. `library(rpart)`

### **Step 4: Configure WEKA to use R**

1. In WEKA, go to **Tools** → **Options** → **R Plugin**.
2. Set the **path to your R executable** (e.g., C:\Program Files\R\R-4.4.2\bin\R.exe).
3. Click **Apply** and **OK**.

### **Step 5: Implement Decision Tree in WEKA**

1. Load your dataset in WEKA.
2. Go to **Classify** tab.
3. Click **Choose** → **trees** → **RPart**.
4. Set options if needed (e.g., min split, cp).
5. Click **Start** to run the classifier.

### **Step 6: View Output**

- After running, you'll see:
  - **Decision tree structure**
  - **Classification accuracy**
  - **Class-wise metrics** (Precision, Recall, F-Score)