

LAB PROGRAM - 3

Consider the 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

Use the data sources, like ARFF, XML ARFF files. Do the following

- i) Classify , Invoke MultiLayerPerception
- ii) Build neural network GUI as below
 - a) Beginning the process of editing the network to add a second hidden layer
 - b) The finished network with two hidden layers
- iii) Apply Lazy classifier, multi instance classifier
- iv) Apply any MetaLearning Algorithm
- v) Optimize base classifier's performance
- vi) Use clustering algorithm such as Cobweb, and Hierarchical Cluster
- vii) Select attribute by specifying an evaluator and a search method

i) Classify , Invoke MultiLayerPerception

- **Go to the "Classify" Tab:** This is where you can apply different classification algorithms.
- **Select MultiLayerPerceptron:** Under the "Classifier" section, click on the "Choose" button, then navigate to functions → MultilayerPerceptron.
- **Configure Parameters:** Click on the MultilayerPerceptron name to open the configuration panel. Here, you can set parameters like the number of hidden layers and epochs.
- **Run the Classifier:** Once the settings are configured, click "Start" to train the neural network with the MultilayerPerceptron.

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'MultilayerPerceptron'. The 'Test options' section shows 'Cross-validation' selected with 'Folds' set to 10. The 'Classifier output' section displays the following results:

Time taken to build model: 0.02 seconds

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

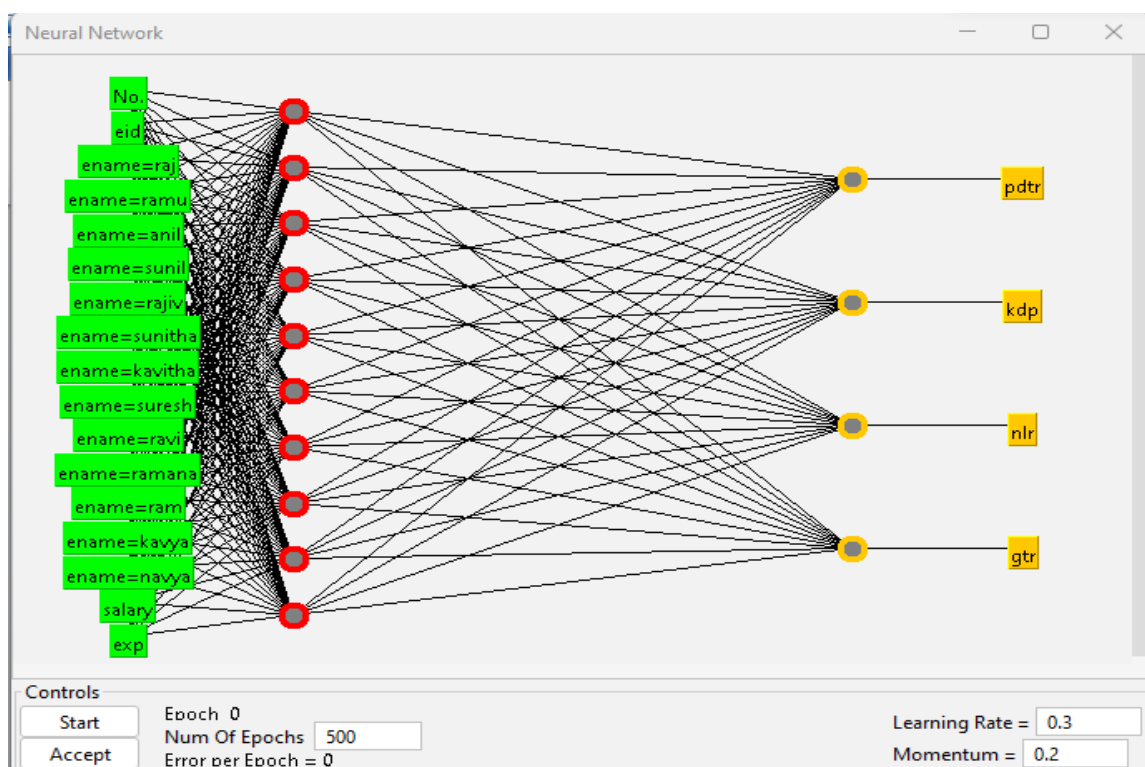
Metric	Value	Percentage
Correctly Classified Instances	3	23.0769 %
Incorrectly Classified Instances	10	76.9231 %
Kappa statistic	-0.1927	
Mean absolute error	0.3953	
Root mean squared error	0.5327	
Relative absolute error	107.2108 %	
Root relative squared error	121.9487 %	
Total Number of Instances	13	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.000	0.091	0.000	0.000	0.000	-0.123	0.727	0.325	pdtr
	0.500	0.714	0.375	0.500	0.429	-0.220	0.214	0.364	kdp
	0.000	0.182	0.000	0.000	0.000	-0.182	0.318	0.163	nlr
	0.000	0.200	0.000	0.000	0.000	-0.234	0.167	0.185	gtr
Weighted Avg.	0.231	0.418	0.173	0.231	0.198	-0.202	0.298	0.286	

=== Confusion Matrix ===

```
a b c d <-- classified as
0 1 1 0 | a = pdtr
1 3 1 1 | b = kdp
0 1 0 1 | c = nlr
0 3 0 0 | d = gtr
```

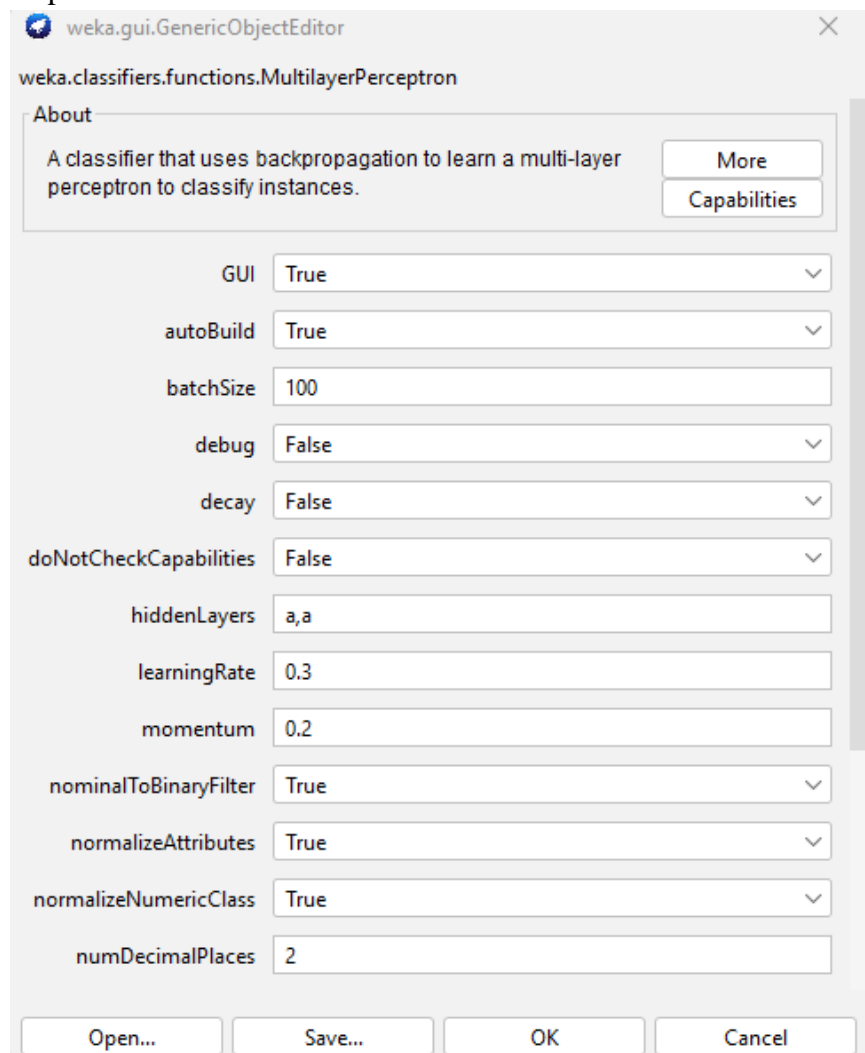


ii) Build neural network GUI as below

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

b) The finished network with two hidden layers

- **Edit the Network:** Modify the hiddenLayers parameter in the MultiLayerPerceptron setup to create a network with two hidden layers. Each "a" represents a hidden layer with adaptive size.
- **To add a second hidden layer,** set the hiddenLayers parameter to include two layers (e.g., "a,a" to specify two hidden layers with adaptive sizes).
- **Check the Final Network Structure:** After running the classifier, you can view the results and structure details under the "Result list" section.
- **Run the Classifier:** Once the settings are configured, click "Start" to train the neural network with the MultilayerPerceptron.



weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

More

Capabilities

GUI True

autoBuild True

batchSize 100

debug False

decay False

doNotCheckCapabilities False

hiddenLayers a,a

learningRate 0.3

momentum 0.2

nominalToBinaryFilter True

normalizeAttributes True

normalizeNumericClass True

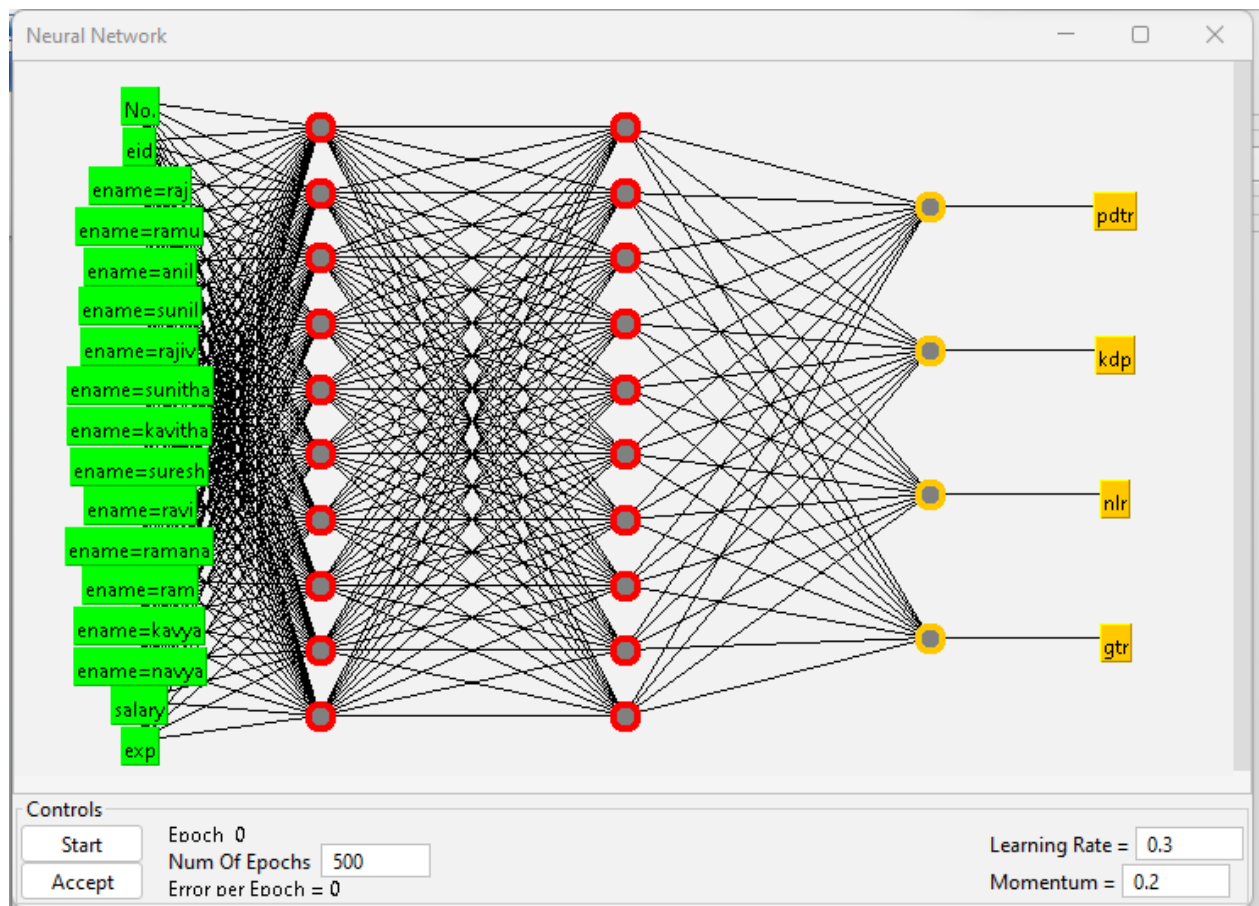
numDecimalPlaces 2

Open... Save... OK Cancel

Classifier output

```
=== Run information ===

Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H "a, a" -G -R
Relation:    lab 3 db
Instances:    13
Attributes:   6
              No.
              eid
              ename
              salary
              exp
              address
Test mode:    10-fold cross-validation
```



iii) Apply Lazy classifier, multi instance classifier

Lazy Classifier:

- **Choose a Lazy Classifier:** Click "Choose" under "Classify" and select lazy → IBk (k-nearest neighbors) as a lazy classifier.
- **Configure and Run:** Set parameters like the value of k and run the classifier by clicking "Start".

Preprocess **Classify** Cluster Associate Select attributes Visualize Auto-WEKA

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) address
 Start Stop

Result list (right-click for options)
 08:24:55 - functions.MultilayerPerceptron
 08:33:45 - functions.MultilayerPerceptron
 08:36:01 - functions.MultilayerPerceptron
 08:40:58 - functions.MultilayerPerceptron
 08:45:27 - lazy.IBk

Classifier output

Time taken to build model: 0 seconds

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

Correctly Classified Instances	5	38.4615 %
Incorrectly Classified Instances	8	61.5385 %
Kappa statistic	0.1034	
Mean absolute error	0.3248	
Root mean squared error	0.4906	
Relative absolute error	88.0921 %	
Root relative squared error	112.2973 %	
Total Number of Instances	13	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.000	0.000	?	0.000	?	?	0.523	0.160	pdtr
	0.667	0.286	0.667	0.667	0.667	0.381	0.702	0.614	kdp
	0.000	0.364	0.000	0.000	0.000	-0.284	0.545	0.250	nlr
	0.333	0.200	0.333	0.333	0.333	0.133	0.417	0.321	gtr
Weighted Avg.	0.385	0.234	?	0.385	?	?	0.585	0.420	

=== Confusion Matrix ===

```

a b c d <-- classified as
0 1 1 0 | a = pdtr
0 4 1 1 | b = kdp
0 1 0 1 | c = nlr
0 0 2 1 | d = gtr

```

Multi-Instance Classifier:

- **Choose a Multi-Instance Classifier:** Go to meta → MultiClassClassifier.
- **Set Parameters:** You can specify a base classifier within this setup.
- **Run:** Click "Start" to execute the multi-instance classifier.

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"'". The test options are set to 'Percentage split' with a percentage of 70. The classifier output shows the following summary:

```
=== Evaluation on test split ===
Time taken to test model on test split: 0 seconds

=== Summary ===
Correctly Classified Instances      3      75 %
Incorrectly Classified Instances    1      25 %
Kappa statistic                    0.5556
Mean absolute error                 0.2019
Root mean squared error             0.3229
Relative absolute error             56.7568 %
Root relative squared error        78.4679 %
Total Number of Instances          4

=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	?	0.000	?	?	?	?	?	?	pdtr
	0.667	0.000	1.000	0.667	0.800	0.577	0.833	0.917	kdp
	?	0.250	0.000	?	?	?	?	?	nlr
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	gtr
Weighted Avg.	0.750	0.000	1.000	0.750	0.850	0.683	0.875	0.938	

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

iv) Apply any MetaLearning Algorithm

- **Select a Meta-Classifier:** Meta-learning algorithms in Weka can be found under meta in the "Choose" menu. Examples include Bagging, Boosting, Stacking, and AdaBoostM1.
- **Configure and Run:** Choose one of the meta-learning algorithms, configure its parameters as needed, and click "Start".

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump'. The test options are set to 'Cross-validation' with 10 folds. The classifier output shows the following summary:

```
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      4      30.7692 %
Incorrectly Classified Instances    9      69.2308 %
Kappa statistic                    -0.1359
Mean absolute error                 0.3822
Root mean squared error             0.4797
Relative absolute error            103.6536 %
Root relative squared error        109.7977 %
Total Number of Instances          13

=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.000	0.091	0.000	0.000	0.000	-0.123	0.568	0.236	pdtr
	0.667	0.857	0.400	0.667	0.500	-0.225	0.167	0.344	kdp
	0.000	0.182	0.000	0.000	0.000	-0.182	0.091	0.167	nlr
	0.000	0.000	?	0.000	?	?	0.400	0.333	gtr
Weighted Avg.	0.308	0.438	?	0.308	?	?	0.271	0.297	

```
=== Confusion Matrix ===
 a b c d  <-- classified as
0 1 1 0 | a = pdtr
1 4 1 0 | b = kdp
0 2 0 0 | c = nlr
0 3 0 0 | d = gtr
```

v) Optimize base classifier's performance

- **Parameter Tuning:** Many classifiers allow for parameter tuning to improve their performance. You can adjust settings in the classifier configuration window.
- **Use Cross-Validation:** In the "Test options" section, select k-fold cross-validation (usually 10-fold) to ensure better performance measurement.

Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize Auto-WEKA

Classifier

Choose **AdaBoostM1** -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☒ Percentage split % 60

More options...

(Nom) address

Start Stop

Result list (right-click for options)

- 08:24:55 - functions.MultilayerPerceptron
- 08:33:45 - functions.MultilayerPerceptron
- 08:36:01 - functions.MultilayerPerceptron
- 08:40:58 - functions.MultilayerPerceptron
- 08:45:27 - lazy.IBk
- 08:49:47 - meta.MultiClassClassifier
- 08:53:56 - lazy.IBk
- 08:54:37 - meta.AdaBoostM1
- 08:54:44 - meta.AdaBoostM1
- 08:54:53 - meta.AdaBoostM1
- 08:57:49 - meta.AdaBoostM1
- 09:00:51 - meta.AdaBoostM1

Classifier output

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances	4	80	%
Incorrectly Classified Instances	1	20	%
Kappa statistic	0		
Mean absolute error	0.3333		
Root mean squared error	0.4082		
Relative absolute error	88.8889	%	
Root relative squared error	94.2809	%	
Total Number of Instances	5		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	?	0.000	?	?	?	?	?	?	pdtr
	1.000	1.000	0.800	1.000	0.889	?	0.500	0.800	kdp
	?	0.000	?	?	?	?	?	?	nlr
	0.000	0.000	?	0.000	?	?	0.500	0.200	gtr
Weighted Avg.	0.800	0.800	?	0.800	?	?	0.500	0.680	

=== Confusion Matrix ===

a b c d <-- classified as

```
0 0 0 0 | a = pdtr
0 4 0 0 | b = kdp
0 0 0 0 | c = nlr
0 1 0 0 | d = gtr
```

vi) Use clustering algorithm such as Cobweb, and Hierarchical Cluster

- **Go to the "Cluster" Tab:** Switch from "Classify" to "Cluster".
- **Cobweb:**

Select Cobweb from the list of clustering algorithms. Set parameters as needed, then click "Start" to cluster the dataset.

- **Hierarchical Cluster:**

Choose HierarchicalClusterer. Configure it (e.g., setting the distance function) and run the clustering.

Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize Auto-WEKA

Clusterer

Choose **HierarchicalClusterer** -N 2 -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) address

☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

- 09:01:54 - Cobweb
- 09:02:26 - HierarchicalClusterer

Clusterer output

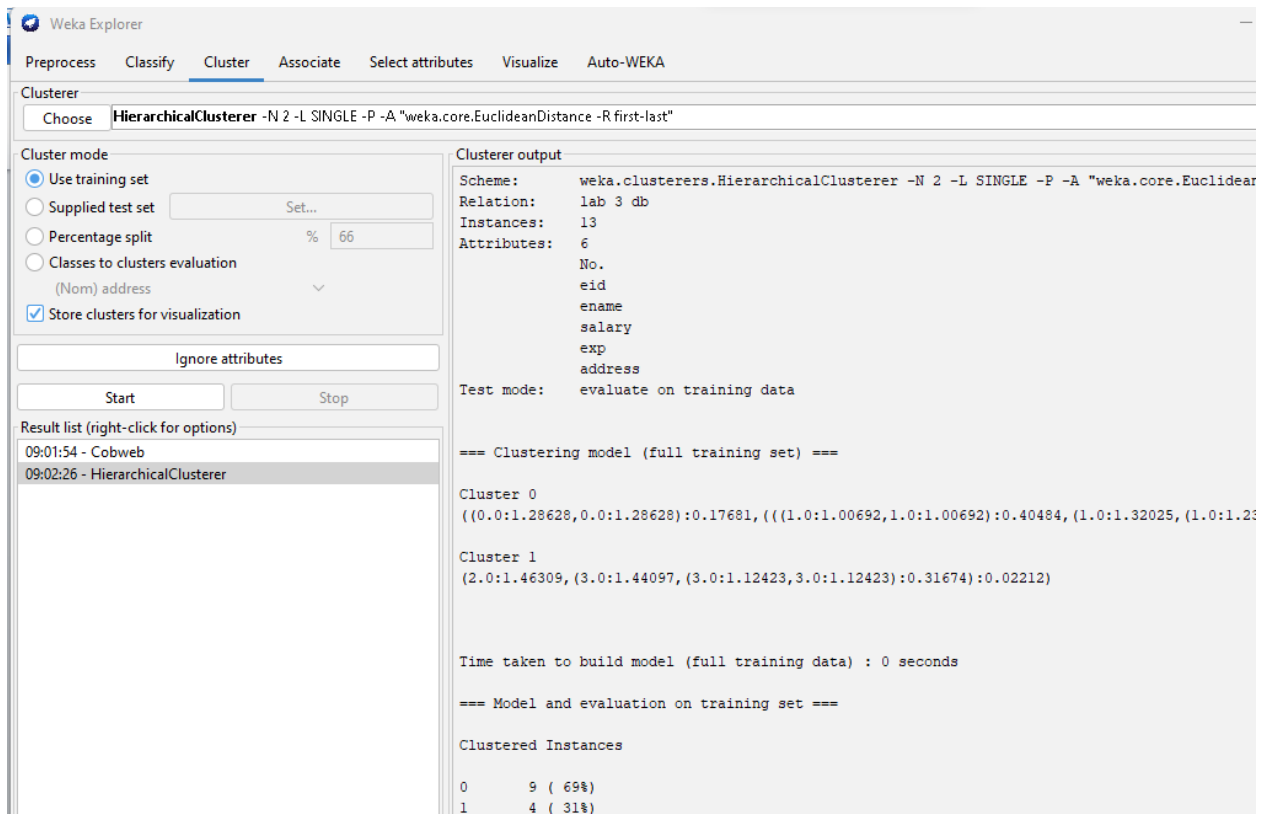
```
1 | leaf 11 [1]
node 0 [13]
1 | node 12 [2]
1 | leaf 13 [1]
1 | node 12 [2]
1 | leaf 14 [1]
node 0 [13]
1 | node 15 [3]
1 | leaf 16 [1]
1 | node 15 [3]
1 | leaf 17 [1]
1 | node 15 [3]
1 | leaf 18 [1]
```

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

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

Clustered Instances

2	1	(8%)
3	1	(8%)
4	1	(8%)
6	1	(8%)
7	1	(8%)
9	1	(8%)
10	1	(8%)
11	1	(8%)
13	1	(8%)
14	1	(8%)
16	1	(8%)
17	1	(8%)



vii) Select attribute by specifying an evaluator and a search method

- **Go to the "Select attributes" Tab:** In Weka's Explorer, click on "Select attributes".
- **Choose an Evaluator:** Under "Attribute Evaluator", choose an evaluator like CfsSubsetEval or InfoGainAttributeEval to measure attribute importance.
- **Choose a Search Method:** Select a search method like BestFirst or GreedyStepwise.
- **Apply and View Results:** Click "Start" to run the attribute selection process. The results will show the most important attributes based on the chosen evaluator and search method.

