



**Aurora's pg college  
Moosarambagh  
Mca department**

**MCA II/II SEM**

# **DATA MINING LAB MANUAL**

## Index

S.No	Experiment	Page no	Signature
1.	Demonstration of preprocessing on dataset student.arff		
2.	Demonstration of preprocessing on dataset labor.arff		
3.	Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm		
4.	Demonstration of Association rule process on dataset test.arff using apriori algorithm		
5.	Demonstration of classification rule process on dataset student.arff using j48 Algorithm		
6.	Demonstration of classification rule process on dataset employee.arff using j48 algorithm		
7.	Demonstration of classification rule process on dataset employee.arff using id3 algorithm		
8.	Demonstration of classification rule process on dataset employee.arff using naïve bayes algorithm		
9.	Demonstration of clustering rule process on dataset iris.arff using simple k-means		
10.	Demonstration of clustering rule process on dataset student.arff using simple k-means		

## **1. Demonstration of preprocessing on dataset student.arff**

**Aim:** This experiment illustrates some of the basic data preprocessing operations that can be performed using WEKA-Explorer. The sample dataset used for this example is the student data available in arff format.

Step1: Loading the data. We can load the dataset into weka by clicking on open button in preprocessing interface and selecting the appropriate file.

Step2: Once the data is loaded, weka will recognize the attributes and during the scan of the data weka will compute some basic strategies on each attribute. The left panel in the above figure shows the list of recognized attributes while the top panel indicates the names of the base relation or table and the current working relation (which are same initially).

Step3: Clicking on an attribute in the left panel will show the basic statistics on the attributes for the categorical attributes the frequency of each attribute value is shown, while for continuous attributes we can obtain min, max, mean, standard deviation and deviation etc.,

Step4: The visualization in the right button panel in the form of cross-tabulation across two attributes.

**Note:** we can select another attribute using the dropdown list.

Step5: Selecting or filtering attributes

**Removing an attribute-**When we need to remove an attribute, we can do this by using the attribute filters in weka. In the filter model panel, click on choose button, This will show a popup window with a list of available filters.

Scroll down the list and select the “weka.filters.unsupervised.attribute.remove” filters.

Step 6: a) Next click the textbox immediately to the right of the choose button. In the resulting dialog box enter the index of the attribute to be filtered out.

b) Make sure that invert selection option is set to false. The click OK now in the filter box. you will see “Remove-R-7”.

c) Click the apply button to apply filter to this data. This will remove the attribute and create new working relation.

d) Save the new working relation as an arff file by clicking save button on the top(button)panel.(student.arff)

## **Discretization**

1) Sometimes association rule mining can only be performed on categorical data. This requires performing discretization on numeric or continuous attributes. In the following example let us discretize age attribute.

① Let us divide the values of age attribute into three bins (intervals).

② First load the dataset into weka (student.arff)

③ Select the age attribute.

④ Activate filter-dialog box and select “WEKA.filters.unsupervised.attribute.discretize” from the list.

⑤ To change the defaults for the filters, click on the box immediately to the right of the choose button.

⑥ We enter the index for the attribute to be discretized. In this case the attribute is age. So we must enter ‘1’ corresponding to the age attribute.

⑦ Enter ‘3’ as the number of bins. Leave the remaining field values as they are.

⑧ Click OK button.

⑨ Click apply in the filter panel. This will result in a new working relation with the selected attribute partitioned into 3 bins.

⑩ Save the new working relation in a file called student-data-discretized.arff

## **Dataset student .arff**

@relation student

@attribute age {<30,30-40,>40}

@attribute income {low, medium, high}

@attribute student {yes, no}

@attribute credit-rating {fair, excellent}

@attribute buyspc {yes, no}

@data

%

<30, high, no, fair, no

<30, high, no, excellent, no

30-40, high, no, fair, yes

>40, medium, no, fair, yes

>40, low, yes, fair, yes

>40, low, yes, excellent, no

30-40, low, yes, excellent, yes

<30, medium, no, fair, no

<30, low, yes, fair, no

>40, medium, yes, fair, yes

<30, medium, yes, excellent, yes

30-40, medium, no, excellent, yes

30-40, high, yes, fair, yes

>40, medium, no, excellent, no

%

The following screenshot shows the effect of discretization.

The screenshot shows the Weka Explorer interface. The 'Filter' menu is open, and 'Discretize -B 10 -M -1.0 -R first-last' is selected. The 'Current relation' is 'tbuk-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last-weka.filter...', with 14 instances and 5 attributes. The 'Attributes' list includes 'age', 'income', 'student', 'creditrating', and 'buyspc'. The 'Selected attribute' is 'student', which is a nominal attribute with 2 distinct values: 'yes' (count 7) and 'no' (count 7). The 'Class' is 'age (Nom)'. The visualization shows two stacked bar charts. The left chart shows the distribution of 'age' for 'yes' and 'no' students, and the right chart shows the distribution of 'age' for 'yes' and 'no' students. The bars are colored cyan, red, and blue from top to bottom.

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter: Choose **Discretize -B 10 -M -1.0 -R first-last** Apply

Current relation  
Relation: tbuk-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last-weka.filter...  
Instances: 14 | Attributes: 5

Attributes  
All | None | Invert | Pattern

No.	Name
1	<input checked="" type="checkbox"/> age
2	<input checked="" type="checkbox"/> income
3	<input checked="" type="checkbox"/> student
4	<input type="checkbox"/> creditrating
5	<input type="checkbox"/> buyspc

Remove

Selected attribute  
Name: student | Missing: 0 (0%) | Distinct: 2 | Type: Nominal | Unique: 0 (0%)

No.	Label	Count
1	yes	7
2	no	7

Class: age (Nom) Visualize All

7 7

Status: OK | Log | x 0

start | Weka GUI Chooser | Weka Explorer | ALEKHYA (G:) | tbuk - Notepad | 1:52 PM

## **2. Demonstration of preprocessing on dataset labor.arff**

**Aim:** This experiment illustrates some of the basic data preprocessing operations that can be performed using WEKA-Explorer. The sample dataset used for this example is the labor data available in arff format.

Step1:Loading the data. We can load the dataset into weka by clicking on open button in preprocessing interface and selecting the appropriate file.

Step2:Once the data is loaded, weka will recognize the attributes and during the scan of the data weka will compute some basic strategies on each attribute. The left panel in the above figure shows the list of recognized attributes while the top panel indicates the names of the base relation or table and the current working relation (which are same initially).

Step3:Clicking on an attribute in the left panel will show the basic statistics on the attributes for the categorical attributes the frequency of each attribute value is shown, while for continuous attributes we can obtain min, max, mean, standard deviation and deviation etc.,

Step4:The visualization in the right button panel in the form of cross-tabulation across two attributes.

**Note:**we can select another attribute using the dropdown list.

Step5:Selecting or filtering attributes

**Removing an attribute-**When we need to remove an attribute,we can do this by using the attribute filters in weka.In the filter model panel,click on choose button,This will show a popup window with a list of available filters.

Scroll down the list and select the “weka.filters.unsupervised.attribute.remove” filters.

Step 6:a)Next click the textbox immediately to the right of the choose button.In the resulting dialog box enter the index of the attribute to be filtered out.

b)Make sure that invert selection option is set to false.The click OK now in the filter box.you will see “Remove-R-7”.

c)Click the apply button to apply filter to this data.This will remove the attribute and create new working relation.

d)Save the new working relation as an arff file by clicking save button on the top(button)panel.(labor.arff)

## **Discretization**

1) Sometimes association rule mining can only be performed on categorical data. This requires performing discretization on numeric or continuous attributes. In the following example let us discretize duration attribute.

① Let us divide the values of duration attribute into three bins (intervals).

② First load the dataset into weka (labor.arff)

③ Select the duration attribute.

④ Activate filter-dialog box and select “WEKA.filters.unsupervised.attribute.discretize” from the list.

⑤ To change the defaults for the filters, click on the box immediately to the right of the choose button.

⑥ We enter the index for the attribute to be discretized. In this case the attribute is duration. So we must enter ‘1’ corresponding to the duration attribute.

⑦ Enter ‘1’ as the number of bins. Leave the remaining field values as they are.

⑧ Click OK button.

⑨ Click apply in the filter panel. This will result in a new working relation with the selected attribute partitioned into 1 bin.

⑩ Save the new working relation in a file called labor-data-discretized.arff

**Dataset labor.arff**



Viewer

Relation: labor-neg-data

No.	duration Numeric	wage-increase-first-year Numeric	wage-increase-second-year Numeric	wage-increase-third-year Numeric	cost-of-living-adjustment Nominal	working-hours Numeric	pension Nominal	standby-pay Numeric	shift-differential Numeric	educa
1	1.0	5.0				40.0			2.0	
2	2.0	4.5		5.8		35.0	ret_allw			yes
3						38.0	empl_c...		5.0	
4	3.0	3.7		4.0	5.0	tc				yes
5	3.0	4.5		4.5	5.0		40.0			
6	2.0	2.0		2.5			35.0			6.0
7	3.0	4.0		5.0	5.0	tc		empl_c...		yes
8	3.0	6.9		4.8	2.3		40.0			3.0
9	2.0	3.0		7.0			38.0		12.0	25.0
10	1.0	5.7			none		40.0	empl_c...		4.0
11	3.0	3.5		4.0	4.6	none		36.0		3.0
12	2.0	6.4		6.4			38.0			4.0
13	2.0	3.5		4.0	none		40.0			2.0
14	3.0	3.5		4.0	5.1	tcf		37.0		4.0
15	1.0	3.0			none		36.0			10.0
16	2.0	4.5		4.0	none		37.0	empl_c...		
17	1.0	2.8					35.0			2.0
18	1.0	2.1			tc		40.0	ret_allw	2.0	3.0
19	1.0	2.0			none		38.0	none		yes
20	2.0	4.0		5.0	tcf		35.0		13.0	5.0
21	2.0	4.3		4.4			38.0			4.0
22	2.0	2.5		3.0			40.0	none		
23	3.0	3.5		4.0	4.6	tcf		27.0		
24	2.0	4.5		4.0			40.0			4.0
25	1.0	6.0					38.0		8.0	3.0
26	3.0	2.0		2.0	2.0	none		40.0	none	
27	2.0	4.5		4.5	tcf					yes
28	2.0	3.0		3.0	none		33.0			yes
29	2.0	5.0		4.0	none		37.0			5.0
30	3.0	2.0		2.5			35.0	none		
31	3.0	4.5		4.5	5.0	none		40.0		no
32	3.0	3.0		2.0	2.5	tc		40.0	none	5.0
33	2.0	2.5		2.5			38.0	empl_c...		
34	2.0	4.0		5.0	none		40.0	none		3.0
35	3.0	2.0		2.5	2.1	tc		40.0	none	2.0
36	2.0	2.0		2.0	none		40.0	none		no
37	1.0	2.0			tc		40.0	ret_allw	4.0	0.0
38	1.0	2.8			none		38.0	empl_c...	2.0	3.0

Right click (or left+alt) for context menu

Undo OK

start start Weka GUI Chooser Weka Explorer Weka Classifier Tree ... prep labor - Paint ALEKHYA (G:)

The following screenshot shows the effect of discretization

The screenshot shows the Weka Explorer interface with the 'Discretize' filter applied to the 'duration' attribute. The filter settings are: 'Discretize -B 10 -M 1.0 -R first-last'. The current relation is 'labor-neg-data-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last-...' with 57 instances and 17 attributes.

The 'Selected attribute' section shows the 'duration' attribute has been converted to a nominal type with 3 distinct values. The resulting bins and their counts are as follows:

No.	Label	Count
1	'(-inf-1.2]'	10
2	'(1.2-1.4]'	0
3	'(1.4-1.6]'	0
4	'(1.6-1.8]'	0
5	'(1.8-2]'	27
6	'(2-2.2]'	0
7	'(2.2-2.4]'	0
8	'(2.4-2.6]'	0
9	'(2.6-2.8]'	0
10	'(2.8-inf)'	19

The 'Class: class (Nom)' visualization shows a stacked bar chart with two classes (blue and red) across the bins. The counts for each class are:

Bin No.	Class 1 (Blue)	Class 2 (Red)
1	5	5
2	0	0
3	0	0
4	0	0
5	15	12
6	0	0
7	0	0
8	0	0
9	0	0
10	5	14

The status bar shows 'OK' and the system tray includes 'Log' and a mouse cursor icon. The taskbar at the bottom shows 'start', 'Weka GUI Chooser', 'Weka Explorer', 'tbuk - Notepad', and the time '1:58 PM'.

### 3. Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm

**Aim:** This experiment illustrates some of the basic elements of association rule mining using WEKA. The sample dataset used for this example is contactlenses.arff

Step1: Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized. In this example it is age attribute.

Step2: Clicking on the associate tab will bring up the interface for association rule algorithm.

Step3: We will use apriori algorithm. This is the default algorithm.

Step4: Inorder to change the parameters for the run (example support, confidence etc) we click on the text box immediately to the right of the choose button.

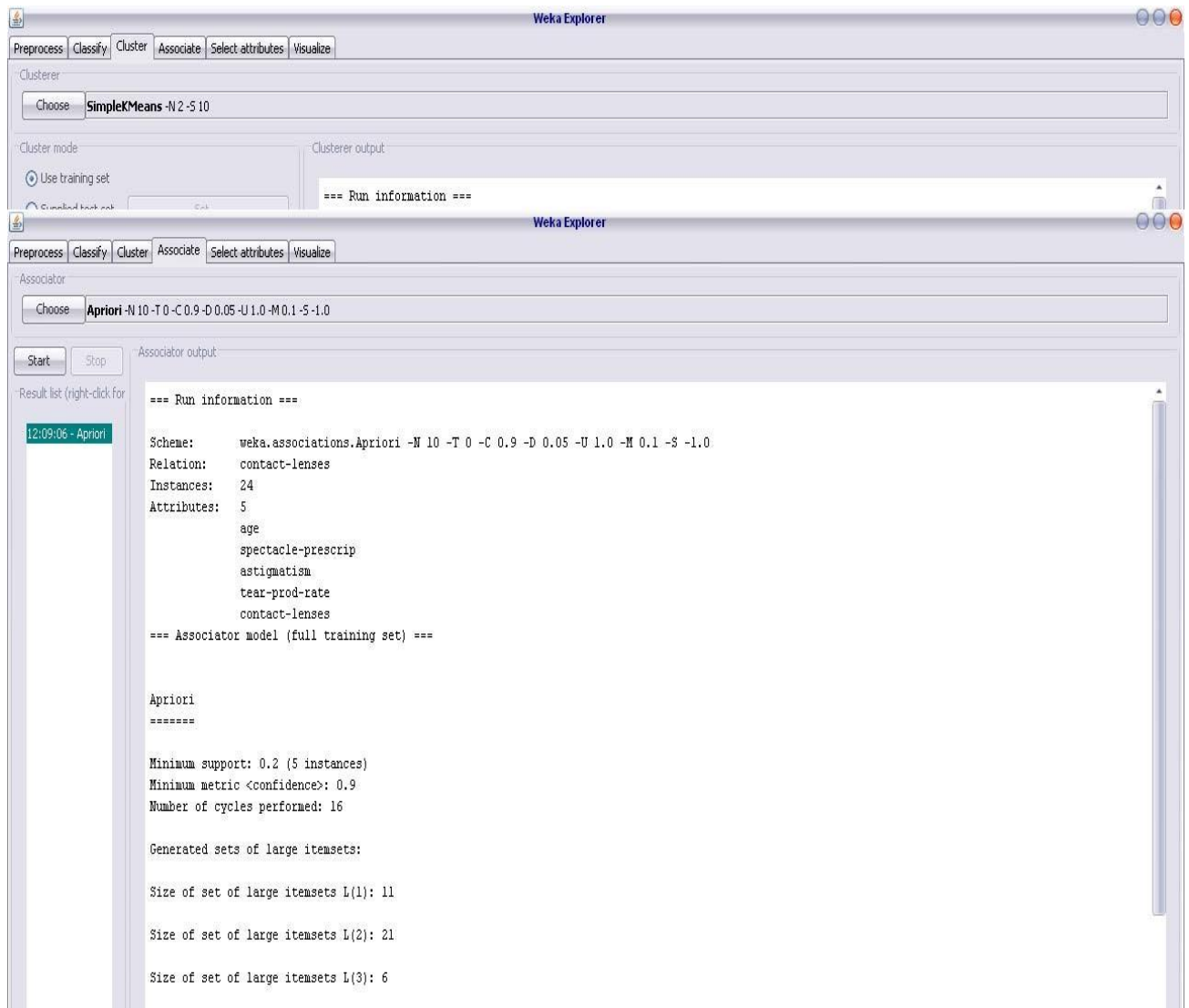
#### Dataset contactlenses.arff

The screenshot displays the Weka Explorer interface. On the left, the 'Viewer' window shows the 'contact-lenses' dataset with 24 rows and 6 columns: No., age, spectacle-prescrip, astigmatism, tear-prod-rate, and contact-lenses. The 'age' column is highlighted. The 'Associate' window is open, showing the 'age' attribute selected. The 'Selected attribute' section displays 'Name: age', 'Missing: 0 (0%)', 'Distinct: 3', and 'Type: Nominal Unique: 0 (0%)'. Below this, a table shows the distribution of 'age' values:

No.	Label	Count
1	young	8
2	pre-presbyopic	8
3	presbyopic	8

The 'Class: contact-lenses (Nom)' dropdown is set to 'contact-lenses (Nom)'. Below the dropdown, three stacked bar charts are visible, each with a total height of 8. The bars are composed of three segments: cyan (top), red (middle), and blue (bottom). The cyan segment represents the 'young' age group, the red segment represents 'pre-presbyopic', and the blue segment represents 'presbyopic'. The 'Log' button is visible at the bottom right of the 'Associate' window.

The following screenshot shows the association rules that were generated when apriori algorithm is applied on the given dataset.



The image shows two screenshots of the Weka Explorer interface. The top screenshot shows the 'Clusterer' tab with 'SimpleKMeans -N 2 -S 10' selected. The bottom screenshot shows the 'Associator' tab with 'Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0' selected. The 'Associator output' window displays the following results:

```
contact-lenses
=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.2 (5 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 11
Size of set of large itemsets L(2): 21
Size of set of large itemsets L(3): 6

Best rules found:

1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12   conf:(1)
2. astigmatism=yes tear-prod-rate=reduced 6 ==> contact-lenses=none 6   conf:(1)
3. astigmatism=no tear-prod-rate=reduced 6 ==> contact-lenses=none 6   conf:(1)
4. spectacle-prescrip=hypermetrope tear-prod-rate=reduced 6 ==> contact-lenses=none 6   conf:(1)
5. spectacle-prescrip=myope tear-prod-rate=reduced 6 ==> contact-lenses=none 6   conf:(1)
6. contact-lenses=soft 5 ==> astigmatism=no tear-prod-rate=normal 5   conf:(1)
7. astigmatism=no contact-lenses=soft 5 ==> tear-prod-rate=normal 5   conf:(1)
8. tear-prod-rate=normal contact-lenses=soft 5 ==> astigmatism=no 5   conf:(1)
9. contact-lenses=soft 5 ==> tear-prod-rate=normal 5   conf:(1)
```

#### **4. Demonstration of Association rule process on dataset test.arff using apriori algorithm**

**Aim:** This experiment illustrates some of the basic elements of association rule mining using WEKA. The sample dataset used for this example is test.arff

Step1: Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized. In this example it is age attribute.

Step2: Clicking on the associate tab will bring up the interface for association rule algorithm.

Step3: We will use apriori algorithm. This is the default algorithm.

Step4: Inorder to change the parameters for the run (example support, confidence etc) we click on the text box immediately to the right of the choose button.

#### **Dataset test.arff**

@relation test

@attribute admissionyear {2005,2006,2007,2008,2009,2010}

@attribute course {cse,mech,it,ece}

@data

%

2005, cse

2005, it

2005, cse

2006, mech

2006, it

2006, ece

2007, it

2007, cse

2008, it

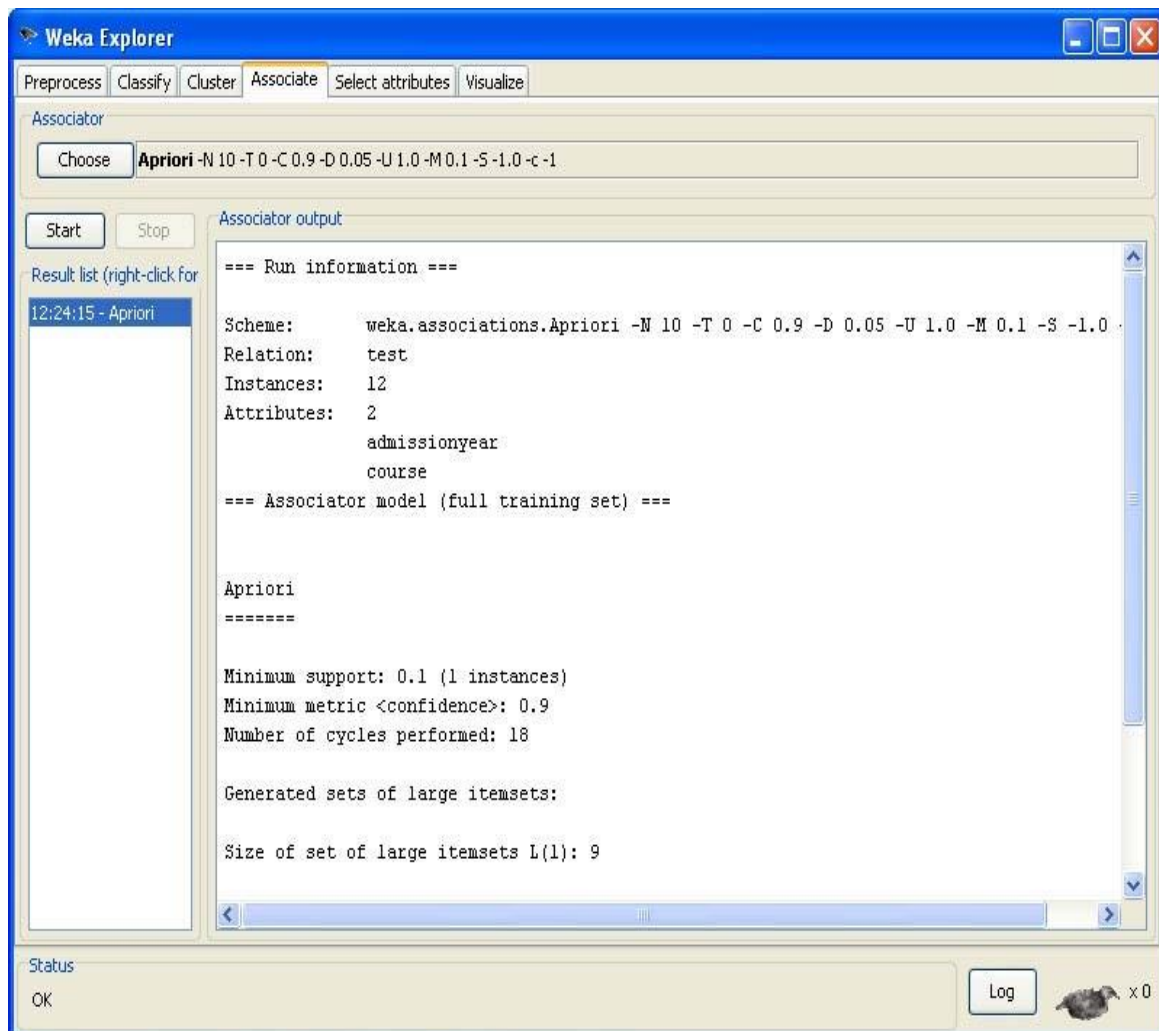
2008, cse

2009, it

2009, ece

%

The following screenshot shows the association rules that were generated when apriori algorithm is applied on the given dataset.



**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)

12:24:15 - Apriori

Associator output

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

Apriori
=====

Minimum support: 0.1 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 9


Size of set of large itemsets L(2): 11

Best rules found:

1. course=mech 1 ==> admissionyear=2006 1   conf:(1)
```

Status

OK

Log  x 0



## **5. Demonstration of classification rule process on dataset student.arff using j48 algorithm**

**Aim:** This experiment illustrates the use of j-48 classifier in weka. The sample data set used in this experiment is “student” data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment:

Step-1: We begin the experiment by loading the data (student.arff) into weka.

Step2: Next we select the “classify” tab and click “choose” button to select the “j48” classifier.

Step3: Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values. The default version does perform some pruning but does not perform error pruning.

Step4: Under the “text” options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don’t have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: We now click ”start” to generate the model .the Ascii version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: Note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Step-7: Now weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting “visualize tree” from the pop-up menu.

Step-8: We will use our model to classify the new instances.

Step-9: In the main panel under “text” options click the “supplied test set” radio button and then click the “set” button. This will pop-up a window which will allow you to open the file containing test instances.

## **Dataset student .arff**

@relation student

@attribute age {<30,30-40,>40}

@attribute income {low, medium, high}

@attribute student {yes, no}

@attribute credit-rating {fair, excellent}

@attribute buyspc {yes, no}

@data

%

<30, high, no, fair, no

<30, high, no, excellent, no

30-40, high, no, fair, yes

>40, medium, no, fair, yes

>40, low, yes, fair, yes

>40, low, yes, excellent, no

30-40, low, yes, excellent, yes

<30, medium, no, fair, no

<30, low, yes, fair, no

>40, medium, yes, fair, yes

<30, medium, yes, excellent, yes

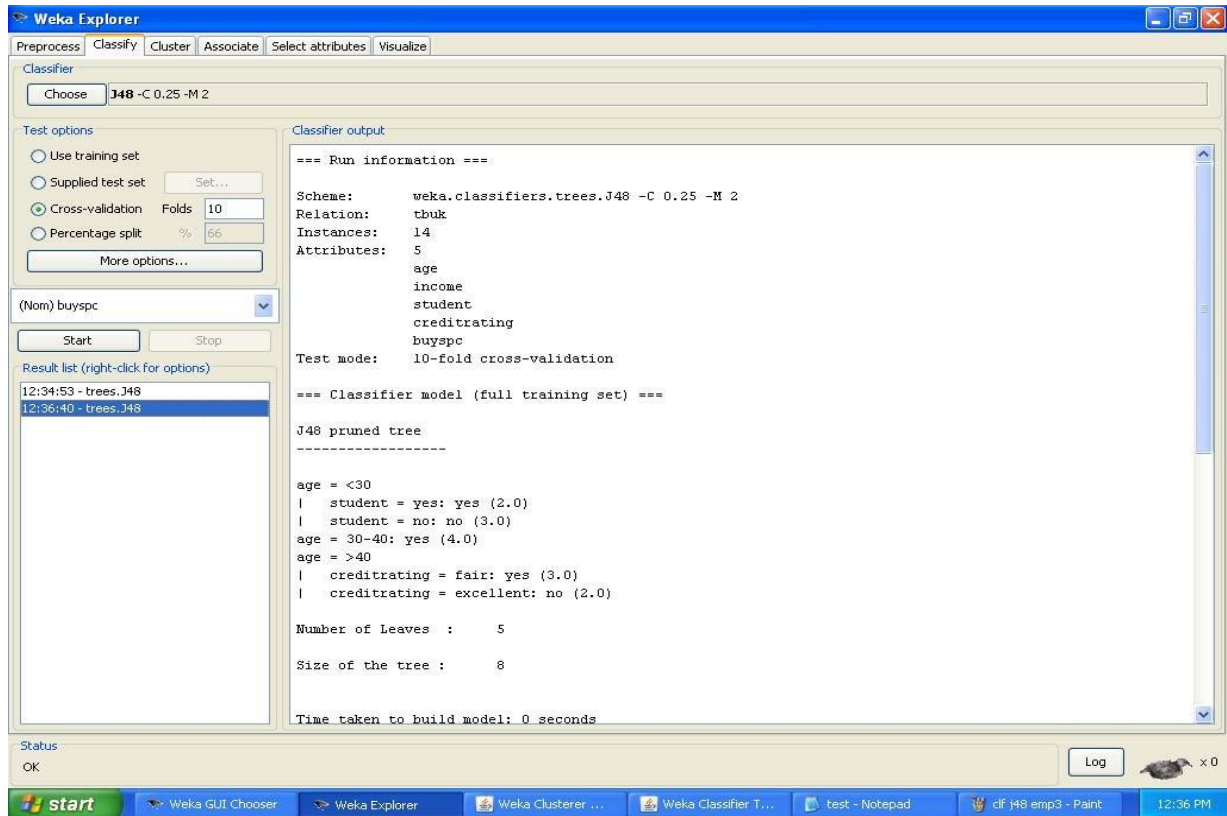
30-40, medium, no, excellent, yes

30-40, high, yes, fair, yes

>40, medium, no, excellent, no

%

The following screenshot shows the classification rules that were generated when j48 algorithm is applied on the given dataset.



**Weka Explorer**

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **J48 -C 0.25 -M 2**

Test options

Use training set  
 Supplied test set (Set...)  
 Cross-validation Folds:   
 Percentage split %   
 More options...

(Nom) buyspc

Start Stop

Result list (right-click for options)

12:34:53 - trees.J48  
 12:36:40 - trees.J48

Classifier output

Size of the tree : 8

Time taken to build model: 0 seconds

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

Correctly Classified Instances	7	50	%
Incorrectly Classified Instances	7	50	%
Kappa statistic	-0.0426		
Mean absolute error	0.4167		
Root mean squared error	0.5984		
Relative absolute error	87.5	%	
Root relative squared error	121.2987	%	
Total Number of Instances	14		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.556	0.6	0.625	0.556	0.588	0.633	yes
	0.4	0.444	0.333	0.4	0.364	0.633	no
Weighted Avg.	0.5	0.544	0.521	0.5	0.508	0.633	

=== Confusion Matrix ===

a b <-- classified as

5 4 | a = yes

3 2 | b = no

Status  
OK

Log x 0

start Weka GUI Chooser Weka Explorer Weka Clusterer ... Weka Classifier T... test - Notepad cf j48 stud1 - Paint 12:37 PM

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier: Choose J48 -C 0.25 -M 2

Test options:
 

- Use training set
- Supplied test set
- Cross-validation (Test on a user-specified dataset)
- Percentage split

Classifier output: Size of the tree : 8

Weka Classifier Tree Visualizer: 12:36:40 - trees.J48 (tbuk)

Tree View

```

    graph TD
      A((age)) -- "= <30" --> B((student))
      A -- "= 30-40" --> C[yes (4.0)]
      A -- "= >40" --> D((creditrating))
      B -- "= yes" --> E[yes (2.0)]
      B -- "= no" --> F[no (3.0)]
      D -- "= fair" --> G[yes (3.0)]
      D -- "= excellent" --> H[no (2.0)]
  
```

Result list (right-click for options):

- 12:34:53 - trees.J48
- 12:36:40 - trees.J48

Status: OK

Log

Windows taskbar: start, Weka GUI C..., Weka Explorer, Weka Cluste..., Weka Classifi..., Weka Classifi..., test - Notepad, cf j48 stud2 ..., 12:37 PM

## **6. Demonstration of classification rule process on dataset employee.arff using j48 algorithm**

**Aim:** This experiment illustrates the use of j-48 classifier in weka.the sample data set used in this experiment is “employee”data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment:

Step 1: We begin the experiment by loading the data (employee.arff) into weka.

Step2: Next we select the “classify” tab and click “choose” button to select the “j48”classifier.

Step3: Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values the default version does perform some pruning but does not perform error pruning.

Step4: Under the “text “options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don’t have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: We now click ”start” to generate the model .the ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: Note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Step-7: Now weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting “visualize tree” from the pop-up menu.

Step-8: We will use our model to classify the new instances.

Step-9: In the main panel under “text “options click the “supplied test set” radio button and then click the “set” button. This will pop-up a window which will allow you to open the file containing test instances.

**Data set employee.arff:**

@relation employee

@attribute age {25, 27, 28, 29, 30, 35, 48}

@attribute salary{10k,15k,17k,20k,25k,30k,35k,32k}

@attribute performance {good, avg, poor}

@data

%

25, 10k, poor

27, 15k, poor

27, 17k, poor

28, 17k, poor

29, 20k, avg

30, 25k, avg

29, 25k, avg

30, 20k, avg

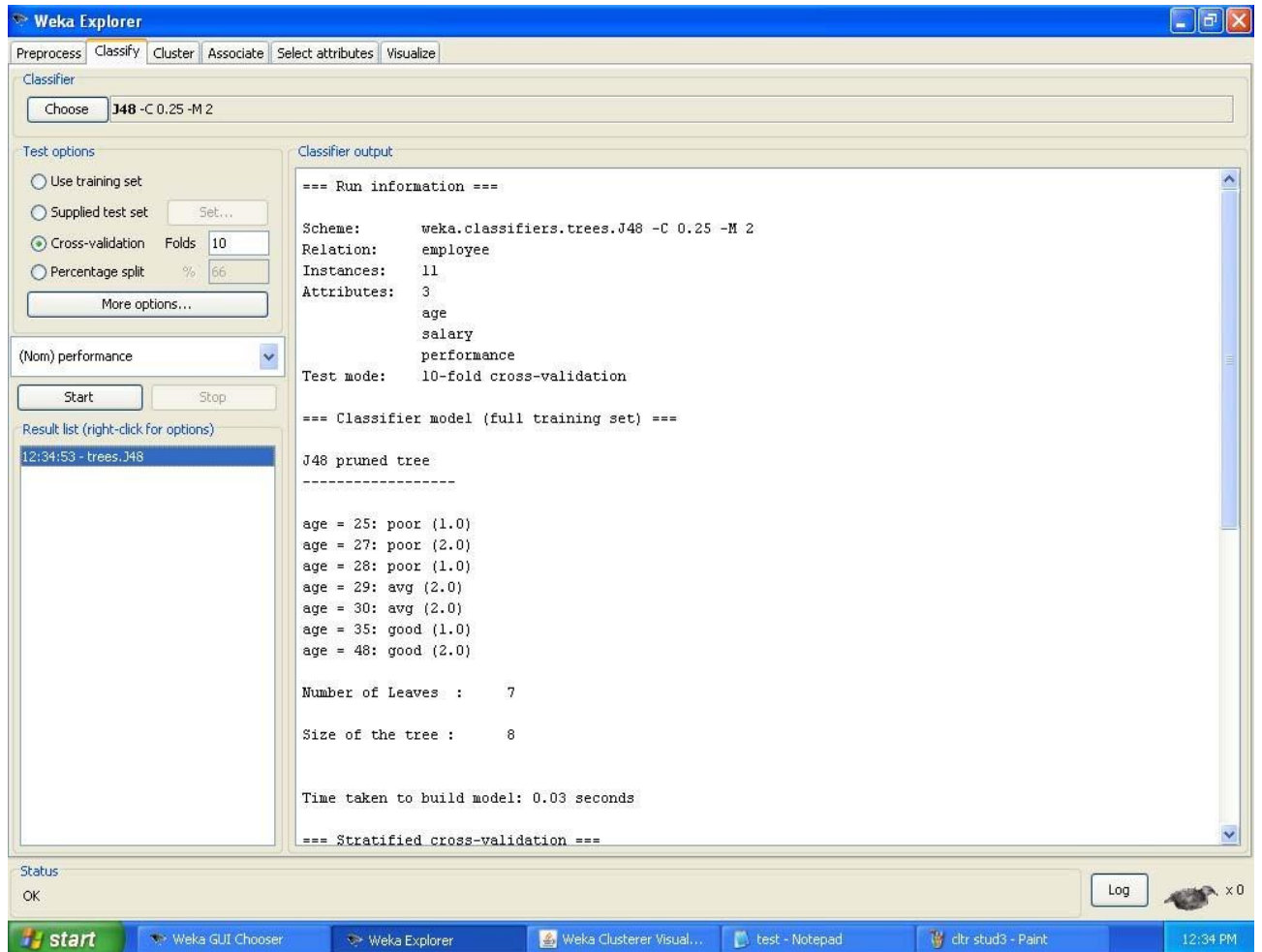
35, 32k, good

48, 34k, good

48, 32k,good

%

The following screenshot shows the classification rules that were generated when j48 algorithm is applied on the given dataset.





**Weka Explorer**

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier

Choose **J48 -C 0.25 -M 2**

Test options

Use training set  
 Supplied test set (Set...)  
 Cross-validation Folds **10**  
 Percentage split % **66**  
 More options...

(Nom) performance

Start Stop

Result list (right-click for options)

12:34:53 - trees.j48

Classifier output

Time taken to build model: 0.03 seconds

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

Correctly Classified Instances	6	54.5455 %
Incorrectly Classified Instances	5	45.4545 %
Kappa statistic	0.2949	
Mean absolute error	0.2209	
Root mean squared error	0.3501	
Relative absolute error	46.716 %	
Root relative squared error	69.5748 %	
Total Number of Instances	11	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.333	0	1	0.333	0.5	0.771	good
	1	0.714	0.444	1	0.615	1	avg
	0.25	0	1	0.25	0.4	0.804	poor
Weighted Avg.	0.545	0.26	0.798	0.545	0.506	0.866	

=== Confusion Matrix ===

```

a b c <-- classified as
1 2 0 | a = good
0 4 0 | b = avg
0 3 1 | c = poor
  
```

Status: OK

Log

Windows taskbar: start | Weka GUI Chooser | Weka Explorer | Weka Clusterer Visual... | test - Notepad | clj j48 emp - Paint | 12:35 PM

**Weka Explorer**

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier: Choose **J48 -C 0,25 -M 2**

Test options:  
 Use training set  
 Supplied test set (Set...)  
 Cross-validation Folds: **10**  
 Percentage split %: **66**  
 More options...

(Nom) performance: Start Stop

Result list (right-click for options):  
 12:34:53 - trees.J48

Classifier output:  
 Time taken to build model: 0.03 seconds  
 === Summary Statistics ===  
 Correctly classified instances: 12  
 Incorrectly classified instances: 8  
 Kappa statistic: 0.6  
 Mean absolute error: 0.26666666666666666  
 Root mean square error: 0.5165151651515151  
 Relative error: 0.4477272727272727  
 Root relative standard error: 1.0  
 Total number of instances: 20  
 === Detailed Statistics ===  
 Detailed description of the classifier follows:  
 a b c  
 1 2 0 | a = good  
 0 4 0 | b = avg  
 0 3 1 | c = poor

**Weka Classifier Tree Visualizer: 12:34:53 - trees.J48 (employ...)**

Tree View

```

graph TD
    age((age)) --<= 25 --> poor1[poor (1.0)]
    age --<= 27 --> poor2[poor (2.0)]
    age --<= 28 --> poor3[poor (1.0)]
    age --<= 29 --> avg1[avg (2.0)]
    age --<= 30 --> avg2[avg (2.0)]
    age --<= 35 --> good1[good (1.0)]
    age -->= 48 --> good2[good (2.0)]
  
```

====

Log

Windows taskbar: start | Weka GUI Chooser | Weka Explorer | Weka Clusterer ... | Weka Classifier T... | test - Notepad | clf\_j48 emp2 - Paint | 12:35 PM

## **7. Demonstration of classification rule process on dataset employee.arff using id3 algorithm**

**Aim:** This experiment illustrates the use of id3 classifier in weka. The sample data set used in this experiment is “employee”data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment:

1. We begin the experiment by loading the data (employee.arff) into weka.

Step2: next we select the “classify” tab and click “choose” button to select the “id3”classifier.

Step3: now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values his default version does perform some pruning but does not perform error pruning.

Step4: under the “text “options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don’t have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: we now click”start”to generate the model .the ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Step-7: now weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting “visualize tree” from the pop-up menu.

Step-8: we will use our model to classify the new instances.

Step-9: In the main panel under “text “options click the “supplied test set” radio button and then click the “set” button. This will show pop-up window which will allow you to open the file containing test instances.

**Data set employee.arff:**

@relation employee

@attribute age {25, 27, 28, 29, 30, 35, 48}

@attribute salary{10k,15k,17k,20k,25k,30k,35k,32k}

@attribute performance {good, avg, poor}

@data

%

25, 10k, poor

27, 15k, poor

27, 17k, poor

28, 17k, poor

29, 20k, avg

30, 25k, avg

29, 25k, avg

30, 20k, avg

35, 32k, good

48, 34k, good

48, 32k, good

%

The following screenshot shows the classification rules that were generated when id3 algorithm is applied on the given dataset.

The screenshot displays the Weka Explorer application window. The 'Classifier' dropdown is set to 'id3'. Under 'Test options', 'Cross-validation' is selected with 'Folds' set to 10. The 'Classifier output' pane shows the following information:

```
=== Run information ===  
  
Scheme:      weka.classifiers.trees.Id3  
Relation:    employee  
Instances:   11  
Attributes:  3  
             age  
             salary  
             performance  
Test mode:   10-fold cross-validation  
  
=== Classifier model (full training set) ===  
  
Id3  
  
age = 25: poor  
age = 27: poor  
age = 28: poor  
age = 29: avg  
age = 30: avg  
age = 35: good  
age = 48: good  
  
Time taken to build model: 0 seconds  
  
=== Stratified cross-validation ===  
=== Summary ===  
  
Correctly Classified Instances      8           72.7273 %  
Incorrectly Classified Instances    0           0 %  
Kappa statistic                     1
```

The 'Result list' on the left shows several entries for 'trees.J48' and 'trees.Id3', with the last entry '13:37:18 - trees.Id3' selected. The status bar at the bottom indicates 'OK'.

**Weka Explorer**

Preprocess | **Classify** | Cluster | Associate | Select attributes | Visualize

Classifier: Choose **Id3**

Test options:

- Use training set
- Supplied test set (Set...)
- Cross-validation (Folds: 10)
- Percentage split (%: 66)

More options...

(Nom) performance

Start Stop

Result list (right-click for options):

- 12:34:53 - trees.J48
- 12:36:40 - trees.J48
- 13:33:34 - trees.J48
- 13:37:18 - trees.Id3

Classifier output:

Time taken to build model: 0 seconds

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

Correctly Classified Instances	8	72.7273 %
Incorrectly Classified Instances	0	0 %
Kappa statistic	1	
Mean absolute error	0	
Root mean squared error	0	
Relative absolute error	0 %	
Root relative squared error	0 %	
Unclassified Instances	3	27.2727 %
Total Number of Instances	11	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	0.833	good
	1	0	1	1	1	1	avg
	1	0	1	1	1	0.75	poor
Weighted Avg.	1	0	1	1	1	0.896	

=== Confusion Matrix ===

```

a b c <-- classified as
2 0 0 | a = good
0 4 0 | b = avg
0 0 2 | c = poor
  
```

Status: OK

Log

Windows taskbar: start | start | Weka GUI Chooser | Weka Explorer | Weka Classifier Tree ... | My Computer | id3 emp1 - Paint

## **8.Demonstration of classification rule process on dataset employee.arff using naïve bayes algorithm**

**Aim:** This experiment illustrates the use of naïve bayes classifier in weka. The sample data set used in this experiment is “employee”data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment:

1. We begin the experiment by loading the data (employee.arff) into weka.

Step2: next we select the “classify” tab and click “choose” button to select the “id3”classifier.

Step3: now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values his default version does perform some pruning but does not perform error pruning.

Step4: under the “text “options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don’t have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: we now click”start”to generate the model .the ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Step-7: now weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting “visualize tree” from the pop-up menu.

Step-8: we will use our model to classify the new instances.

Step-9: In the main panel under “text “options click the “supplied test set” radio button and then click the “set” button. This will show pop-up window which will allow you to open the file containing test instances.

**Data set employee.arff:**

@relation employee

@attribute age {25, 27, 28, 29, 30, 35, 48}

@attribute salary {10k,15k,17k,20k,25k,30k,35k,32k}

@attribute performance {good, avg, poor}

@data

%

25, 10k, poor

27, 15k, poor

27, 17k, poor

28, 17k, poor

29, 20k, avg

30, 25k, avg

29, 25k, avg

30, 20k, avg

35, 32k, good

48, 34k, good

48, 32k, good

%



The following screenshot shows the classification rules that were generated when naive bayes algorithm is applied on the given dataset.

The screenshot shows the Weka Explorer interface with the Naive Bayes classifier selected. The Classifier output window displays the following information:

```

=== Run information ===
Scheme:      weka.classifiers.bayes.NaiveBayes
Relation:    employee
Instances:   11
Attributes:  3
              age
              salary
              performance
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

Attribute    Class    good    avg    poor
              (0.29) (0.36) (0.36)
-----
age
25           1.0    1.0    2.0
27           1.0    1.0    3.0
28           1.0    1.0    2.0
29           1.0    3.0    1.0
30           1.0    3.0    1.0
35           2.0    1.0    1.0
48           3.0    1.0    1.0
[total]     10.0   11.0   11.0

salary
10k          1.0    1.0    2.0
15k          1.0    1.0    2.0
  
```

**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) performance

Start Stop

Result list (right-click for options):

- 12:34:53 - trees.J48
- 12:36:40 - trees.J48
- 13:33:34 - trees.J48
- 13:37:18 - trees.Id3
- 13:38:55 - bayes.NaiveBayes

Status: OK

Log

---

**Classifier output**

Time taken to build model: 0 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	10	90.9091 %
Incorrectly Classified Instances	1	9.0909 %
Kappa statistic	0.8625	
Mean absolute error	0.2899	
Root mean squared error	0.3171	
Relative absolute error	61.3111 %	
Root relative squared error	63.0158 %	
Total Number of Instances	11	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	good
	1	0.143	0.8	1	0.889	1	avg
	0.75	0	1	0.75	0.857	1	poor
Weighted Avg.	0.909	0.052	0.927	0.909	0.908	1	

=== Confusion Matrix ===

```

a b c <-- classified as
3 0 0 | a = good
0 4 0 | b = avg
0 1 3 | c = poor

```

## **9. Demonstration of clustering rule process on dataset iris.arff using simple k-means**

**Aim:** This experiment illustrates the use of simple k-mean clustering with Weka explorer. The sample data set used for this example is based on the iris data available in ARFF format. This document assumes that appropriate preprocessing has been performed. This iris dataset includes 150 instances.

### **Steps involved in this Experiment**

Step 1: Run the Weka explorer and load the data file iris.arff in preprocessing interface.

Step 2: In order to perform clustering select the 'cluster' tab in the explorer and click on the choose button. This step results in a dropdown list of available clustering algorithms.

Step 3 : In this case we select 'simple k-means'.

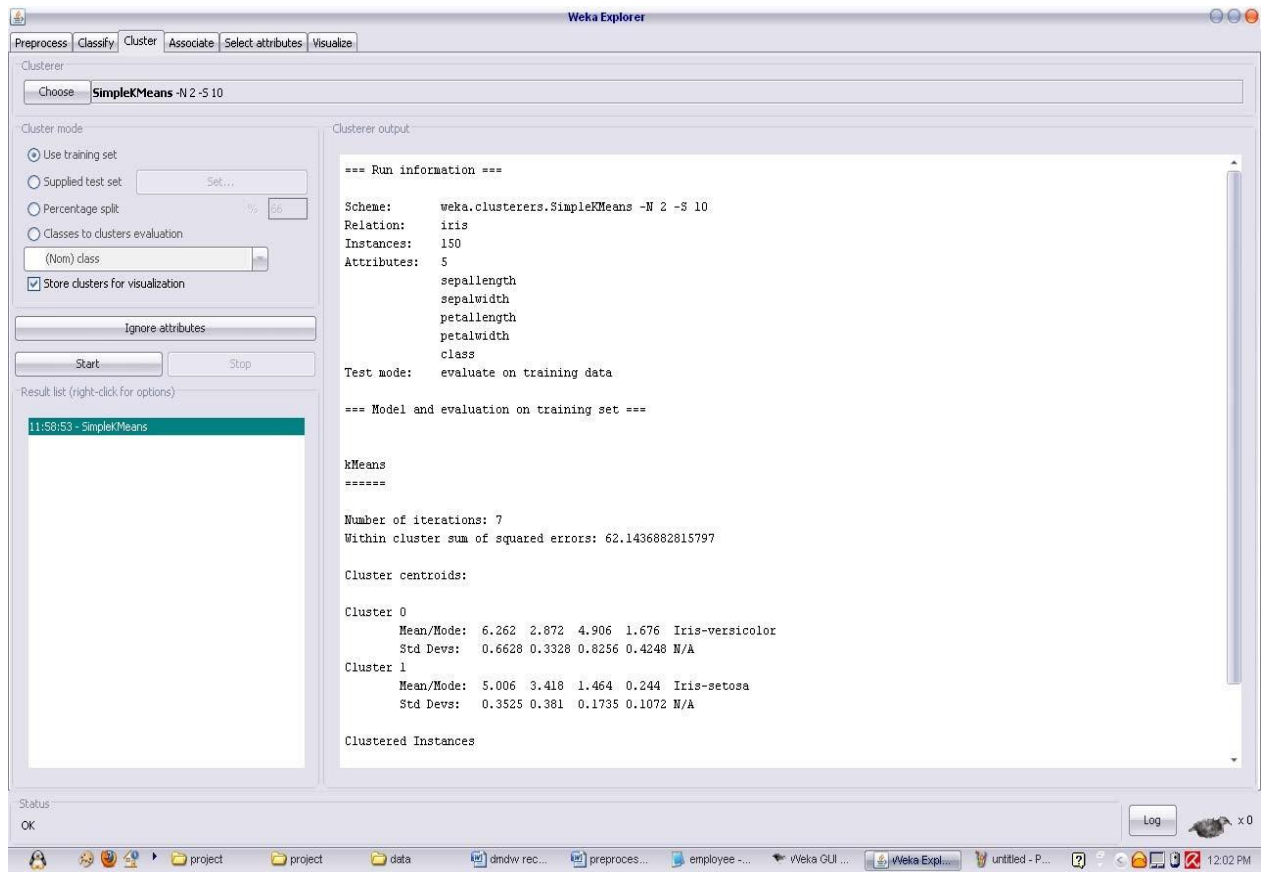
Step 4: Next click in text button to the right of the choose button to get popup window shown in the screenshots. In this window we enter six on the number of clusters and we leave the value of the seed on as it is. The seed value is used in generating a random number which is used for making the internal assignments of instances of clusters.

Step 5 : Once of the option have been specified. We run the clustering algorithm there we must make sure that they are in the 'cluster mode' panel. The use of training set option is selected and then we click 'start' button. This process and resulting window are shown in the following screenshots.

Step 6 : The result window shows the centroid of each cluster as well as statistics on the number and the percent of instances assigned to different clusters. Here clusters centroid are means vectors for each clusters. This clusters can be used to characterized the cluster. For eg, the centroid of cluster1 shows the class iris.versicolor mean value of the sepal length is 5.4706, sepal width 2.4765, petal width 1.1294, petal length 3.7941.

Step 7: Another way of understanding characteristics of each cluster through visualization ,we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments.

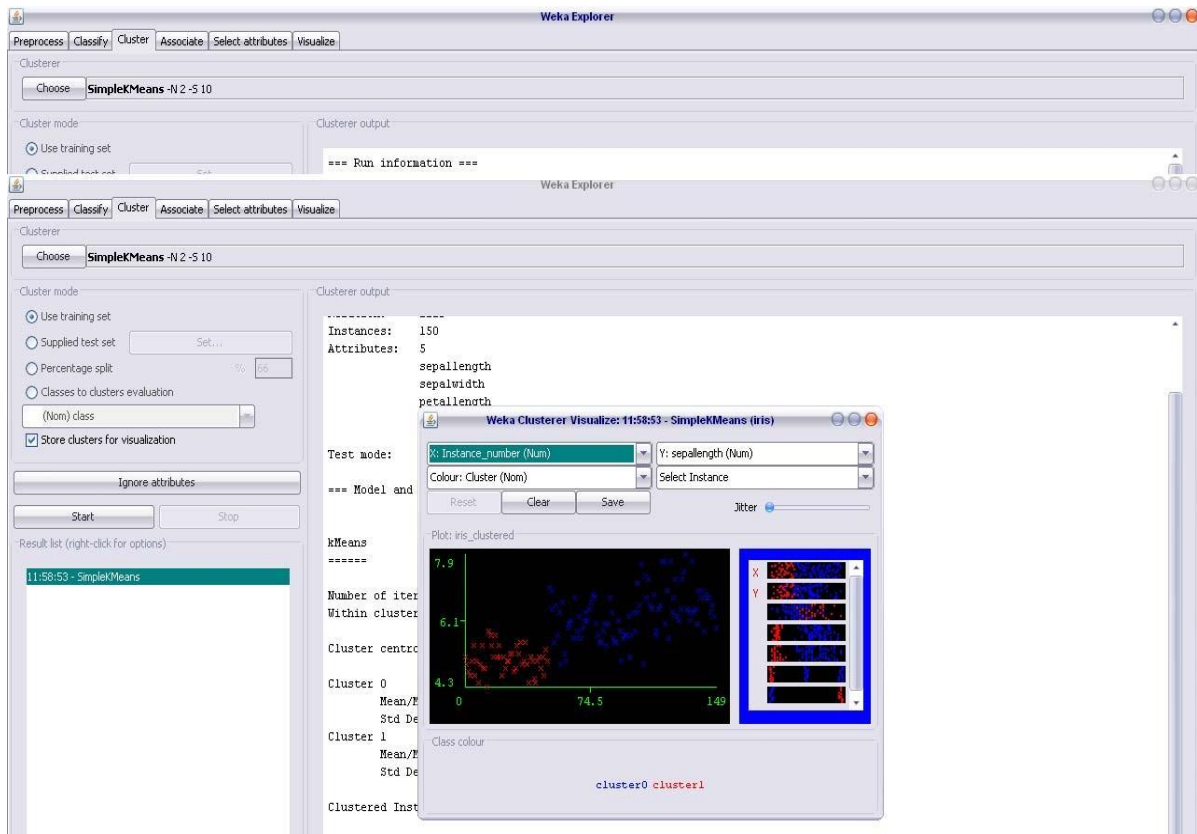
The following screenshot shows the clustering rules that were generated when simple k means algorithm is applied on the given dataset.



## Interpretation of the above visualization

From the above visualization, we can understand the distribution of sepal length and petal length in each cluster. For instance, for each cluster is dominated by petal length. In this case by changing the color dimension to other attributes we can see their distribution with in each of the cluster.

Step 8: We can assure that resulting dataset which included each instance along with its assign cluster. To do so we click the save button in the visualization window and save the result iris k-mean .The top portion of this file is shown in the following figure.



## **10. Demonstration of clustering rule process on dataset student.arff using simple k-means**

**Aim:** This experiment illustrates the use of simple k-mean clustering with Weka explorer. The sample data set used for this example is based on the student data available in ARFF format. This document assumes that appropriate preprocessing has been performed. This istudent dataset includes 14 instances.

Steps involved in this Experiment

Step 1: Run the Weka explorer and load the data file student.arff in preprocessing interface.

Step 2: Inorder to perform clustering select the 'cluster' tab in the explorer and click on the choose button. This step results in a dropdown list of available clustering algorithms.

Step 3 : In this case we select 'simple k-means'.

Step 4: Next click in text button to the right of the choose button to get popup window shown in the screenshots. In this window we enter six on the number of clusters and we leave the value of the seed on as it is. The seed value is used in generating a random number which is used for making the internal assignments of instances of clusters.

Step 5 : Once of the option have been specified. We run the clustering algorithm there we must make sure that they are in the 'cluster mode' panel. The use of training set option is selected and then we click 'start' button. This process and resulting window are shown in the following screenshots.

Step 6 : The result window shows the centroid of each cluster as well as statistics on the number and the percent of instances assigned to different clusters. Here clusters centroid are means vectors for each clusters. This clusters can be used to characterized the cluster.

Step 7: Another way of understanding characteristics of each cluster through visualization ,we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments.

### **Interpretation of the above visualization**

From the above visualization, we can understand the distribution of age and instance number in each cluster. For instance, for each cluster is dominated by age. In this case by changing the color dimension to other attributes we can see their distribution with in each of the cluster.

Step 8: We can assure that resulting dataset which included each instance along with its assign cluster. To do so we click the save button in the visualization window and save the result student k-mean .The top portion of this file is shown in the following figure.

## Dataset student .arff

@relation student

@attribute age {<30,30-40,>40}

@attribute income {low,medium,high}

@attribute student {yes,no}

@attribute credit-rating {fair,excellent}

@attribute buyspc {yes,no}

@data

%

<30, high, no, fair, no

<30, high, no, excellent, no

30-40, high, no, fair, yes

>40, medium, no, fair, yes

>40, low, yes, fair, yes

>40, low, yes, excellent, no

30-40, low, yes, excellent, yes

<30, medium, no, fair, no

<30, low, yes, fair, no

>40, medium, yes, fair, yes

<30, medium, yes, excellent, yes

30-40, medium, no, excellent, yes

30-40, high, yes, fair, yes

>40, medium, no, excellent, no

%

The following screenshot shows the clustering rules that were generated when simple k-means algorithm is applied on the given dataset.

The screenshot displays the Weka Explorer interface with the SimpleKMeans clustering algorithm applied. The 'Clusterer' window shows the command: `SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10`. The 'Cluster mode' section is set to 'Use training set' with 'Store clusters for visualization' checked. The 'Clusterer output' window shows the following run information:

```

=== Run information ===

Scheme:      weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last"
Relation:    tbuk
Instances:   14
Attributes:  5
              age
              income
              student
              creditrating
              buyspc

Test mode:   evaluate on training data

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

kMeans
=====

Number of iterations: 5
Within cluster sum of squared errors: 25.0
Missing values globally replaced with mean/mode

Cluster centroids:

Attribute      Full Data      Cluster#
                (14)          (9)          (5)
-----
age             <30            <30          30-40
income          medium         medium       low
student         yes            no           yes
  
```

The 'Result list' shows three entries for SimpleKMeans at different times: 12:26:56, 12:27:32, and 12:30:41. The status bar at the bottom indicates 'OK' and shows the system tray with a clock at 12:31 PM.



**Weka Explorer**

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Clusterer: Choose **SimpleKMeans** -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Cluster mode

- Use training set
- Supplied test set
- Percentage split %
- Classes to clusters evaluation
- (Nom) buyspc
- Store clusters for visualization

Ignore attributes

Start  Stop

Result list (right-click for options)

- 12:26:56 - SimpleKMeans
- 12:27:32 - SimpleKMeans
- 12:30:41 - SimpleKMeans

Clusterer output

```

buyspc
Test mode: evaluate on training data

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

kMeans
=====
Number of iterations: 5
Within cluster sum of squared errors: 25.0
Missing values globally replaced with mean/mode

Cluster centroids:


Attribute      Full Data      Cluster#
                (14)          (9)          (5)
-----
age             <30            <30            30-40
income          medium         medium         low
student         yes            no             yes
creditrating    fair           fair           fair
buyspc          yes            yes            yes

Clustered Instances

0      9 ( 64%)
1      5 ( 36%)

```

Status: OK

Log  x 0

start | Weka GUI Chooser | Weka Explorer | test - Notepad | ctr stud - Paint | 12:32 PM

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Clusterer

Choose **SimpleKMeans** -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Cluster mode

- Use training set
- Supplied test set Set...
- Percentage split % 66
- Classes to clusters evaluation (Nom) buyspc
- Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

- 12:26:56 - SimpleKMeans
- 12:27:32 - SimpleKMeans
- 12:30:41 - SimpleKMeans

Clusterer output

buyspc

Test mode: evaluate on training data

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

Weka Clusterer Visualize: 12:30:41 - SimpleKMeans (tbuk)

X: Instance\_number (Num) Y: age (Nom)

Colour: Cluster (Nom) Select Instance

Reset Clear Open Save Jitter

Plot: tbuk\_clustered

Class colour

cluster1 cluster2

Status OK Log x 0

start Weka GUI Chooser Weka Explorer Weka Clusterer Visual... test - Notepad cltr stud2 - Paint 12:33 PM