

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		
11.	Design a decision tree by pruning the nodes on your own. Convert the decision trees into “if- then-else rules”. The decision tree must consists of 2-3 levels and convert it into a set of rules.		

12. Generate Association rules for the following transactional database using Apriori algorithm.

TID	List of Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5

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 with the Discretize filter applied to the 'student' attribute. The filter is set to 'Discretize -B 10 -M -1.0 -R first-last'. The 'student' attribute is selected, and its distribution is visualized as a stacked bar chart with three categories: 'yes', 'no', and a third unlabeled category (likely 'unknown' or 'other').

Attributes List:

No.	Name
1	age
2	income
3	student
4	creditrating
5	buyspc

Selected attribute details:

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

Visualization Data:

No.	Label	Count
1	yes	7
2	no	7

The visualization shows two stacked bar charts. The left chart shows the distribution of the 'student' attribute (7 'yes', 7 'no'). The right chart shows the distribution of the 'age' attribute (7 'young', 7 'middle-aged', 7 'old').

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 partition 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	educ...
1	1.0		5.0				40.0			2.0
2	2.0		4.5				35.0	ret_allw		yes
3							38.0	empl_c...		5.0
4	3.0		3.7			5.0	tc			yes
5	3.0		4.5			5.0				
6	2.0		2.0							6.0
7	3.0		4.0			5.0	tc			
8	3.0		6.9			2.3				
9	2.0		3.0						12.0	25.0
10	1.0		5.7				40.0	empl_c...		4.0
11	3.0		3.5			4.6	none			3.0
12	2.0		6.4							4.0
13	2.0		3.5				40.0			2.0
14	3.0		3.5			5.1	tcf			4.0
15	1.0		3.0				36.0			10.0
16	2.0		4.5			4.0	none			
17	1.0		2.8							2.0
18	1.0		2.1				40.0	ret_allw	2.0	3.0
19	1.0		2.0				38.0	none		yes
20	2.0		4.0			5.0	tcf			5.0
21	2.0		4.3				4.4			4.0
22	2.0		2.5				3.0			
23	3.0		3.5			4.0	4.6	tcf		
24	2.0		4.5							4.0
25	1.0		6.0							3.0
26	3.0		2.0			2.0	2.0	none		
27	2.0		4.5					tcf		yes
28	2.0		3.0				3.0	none		yes
29	2.0		5.0					none		5.0
30	3.0		2.0				2.5	none		no
31	3.0		4.5			5.0	none			no
32	3.0		3.0			2.5	tc			5.0
33	2.0		2.5				2.5	empl_c...		
34	2.0		4.0				5.0	none		3.0
35	3.0		2.0			2.5	2.1	tc		1.0
36	2.0		2.0				2.0	none		no
37	1.0		2.0					tc		4.0
38	1.0		2.8					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 displays the Weka Explorer interface with the 'Discretize' filter applied to the 'duration' attribute. The 'Selected attribute' table shows the resulting nominal bins and their counts.

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 visualization shows a stacked bar chart for the 'class' attribute. The x-axis represents the discretized bins, and the y-axis represents the count. The bars are colored blue and red, with the total count for each bin displayed above it.

Bin Label	Count
{-inf-1.2]}	10
{1.2-1.4]}	0
{1.4-1.6]}	0
{1.6-1.8]}	0
{1.8-2]}	27
{2-2.2]}	0
{2.2-2.4]}	0
{2.4-2.6]}	0
{2.6-2.8]}	0
{2.8-inf]}	19

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. The 'Viewer' window shows the 'contact-lenses' dataset with the following data:

No.	age	spectacle-prescrip	astigmatism	tear-prod-rate	contact-lenses
	Nominal	Nominal	Nominal	Nominal	Nominal
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-pr...	myope	no	reduced	none
10	pre-pr...	myope	no	normal	soft
11	pre-pr...	myope	yes	reduced	none
12	pre-pr...	myope	yes	normal	hard
13	pre-pr...	hypermetrope	no	reduced	none
14	pre-pr...	hypermetrope	no	normal	soft
15	pre-pr...	hypermetrope	yes	reduced	none
16	pre-pr...	hypermetrope	yes	normal	none
17	presb...	myope	no	reduced	none
18	presb...	myope	no	normal	none
19	presb...	myope	yes	reduced	none
20	presb...	myope	yes	normal	hard
21	presb...	hypermetrope	no	reduced	none
22	presb...	hypermetrope	no	normal	soft
23	presb...	hypermetrope	yes	reduced	none
24	presb...	hypermetrope	yes	normal	none

The 'Associate' window shows the 'age' attribute selected. The 'Selected attribute' section displays: Name: age, Missing: 0 (0%), Distinct: 3, Type: Nominal, Unique: 0 (0%). Below this is a table with 3 rows:

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

The 'Class: contact-lenses (Nom)' section shows three stacked bar charts, each with a total count of 8. The bars are composed of three segments: blue (bottom), red (middle), and cyan (top).

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

The screenshot displays two windows from the Weka Explorer application. The top window is the 'Clusterer' interface, showing 'SimpleKMeans -N 2 -S 10' selected. The bottom window is the 'Associator' interface, showing 'Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0' selected. The 'Associator output' pane in the bottom window contains the following text:

```
=== 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  
Relation:    contact-lenses  
Instances:   24  
Attributes:  5  
             age  
             spectacle-prescrip  
             astigmatism  
             tear-prod-rate  
             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
```

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -N 2 -S 10

Cluster mode

Use training set

Clusterer output

```
=== Run information ===
```

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

Start Stop

Associator output:

```
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=normal contact-lenses=soft 5 => astigmatism=no 5 conf:(1)
2. astigmatism=yes tear-prod-rate=normal 5 => contact-lenses=soft 5 conf:(1)
3. astigmatism=no tear-prod-rate=normal 5 => contact-lenses=soft 5 conf:(1)
4. astigmatism=no contact-lenses=soft 5 => tear-prod-rate=normal 5 conf:(1)
5. tear-prod-rate=normal contact-lenses=soft 5 => astigmatism=no 5 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 => astigmatism=no tear-prod-rate=normal 5 conf:(1)
10. astigmatism=yes tear-prod-rate=normal 5 => contact-lenses=soft 5 conf:(1)
11. astigmatism=no tear-prod-rate=normal 5 => contact-lenses=soft 5 conf:(1)
12. tear-prod-rate=normal 12 => contact-lenses=none 12 conf:(1)
13. astigmatism=yes tear-prod-rate=reduced 6 => contact-lenses=none 6 conf:(1)
14. astigmatism=no tear-prod-rate=reduced 6 => contact-lenses=none 6 conf:(1)
15. spectacle-prescrip=hypermetrope tear-prod-rate=reduced 6 => contact-lenses=none 6 conf:(1)
16. spectacle-prescrip=myope tear-prod-rate=reduced 6 => contact-lenses=none 6 conf:(1)
17. spectacle-prescrip=normal tear-prod-rate=reduced 6 => contact-lenses=none 6 conf:(1)
18. tear-prod-rate=reduced 6 => contact-lenses=none 6 conf:(1)
```

Result list (right-click for)

12:09:06 - Apriori

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

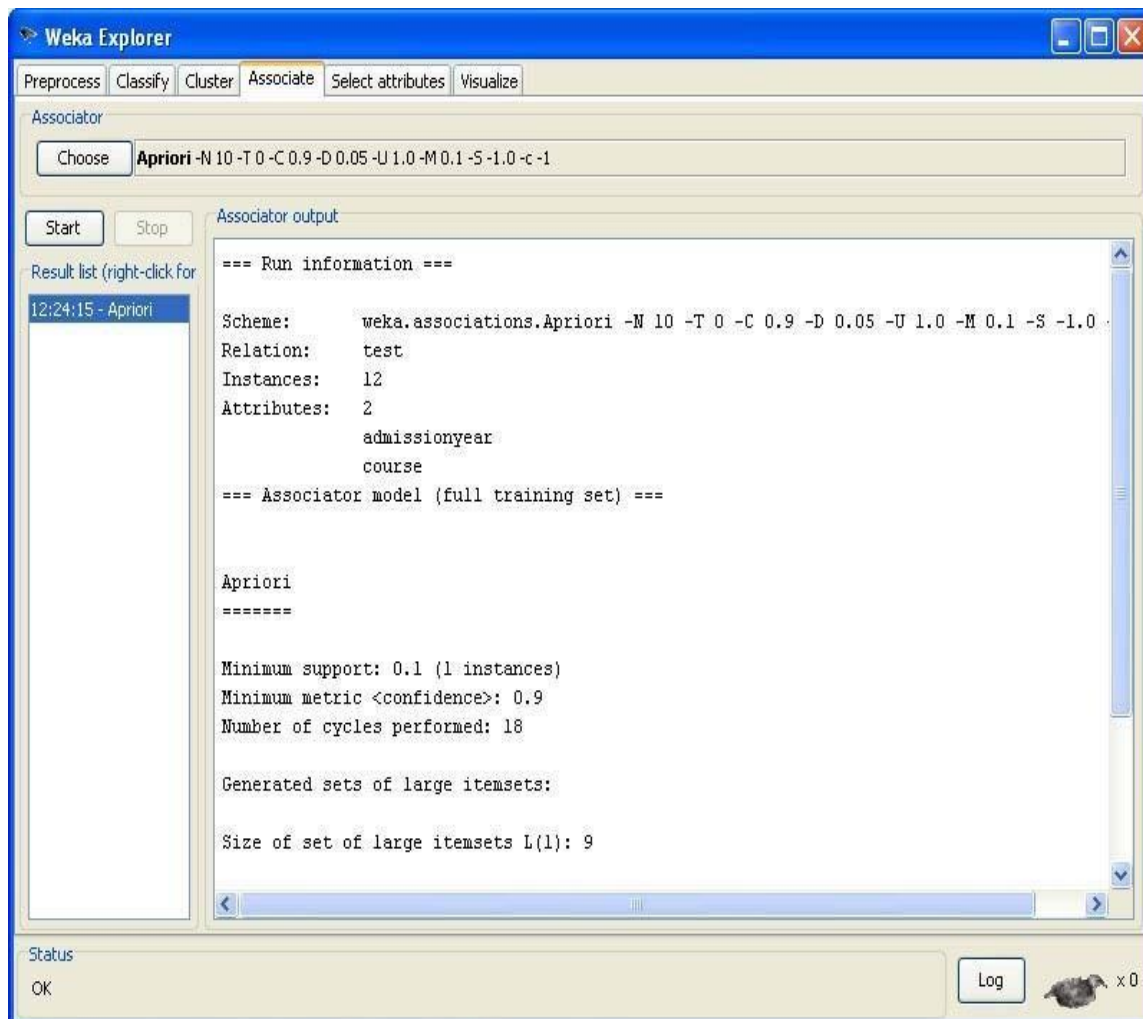
2008, cse

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 j48 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 choose 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 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

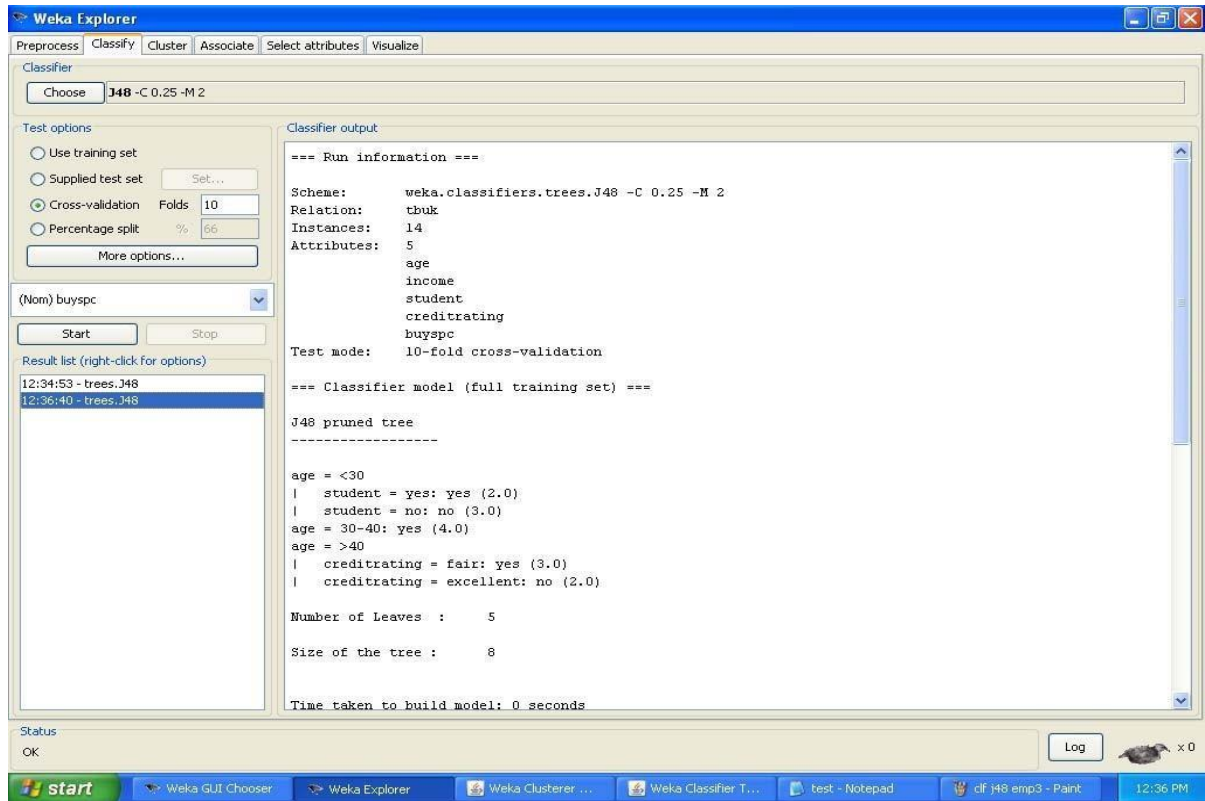
30-40, medium, no, excellent, yes

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: **10**
- Percentage split %: **66**

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

Windows taskbar: start | Weka GUI Chooser | Weka Explorer | Weka Clusterer... | Weka Classifier T... | test - Notepad | df 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 % 66

 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

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)]
  
```

Class: Yes no

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 wills 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

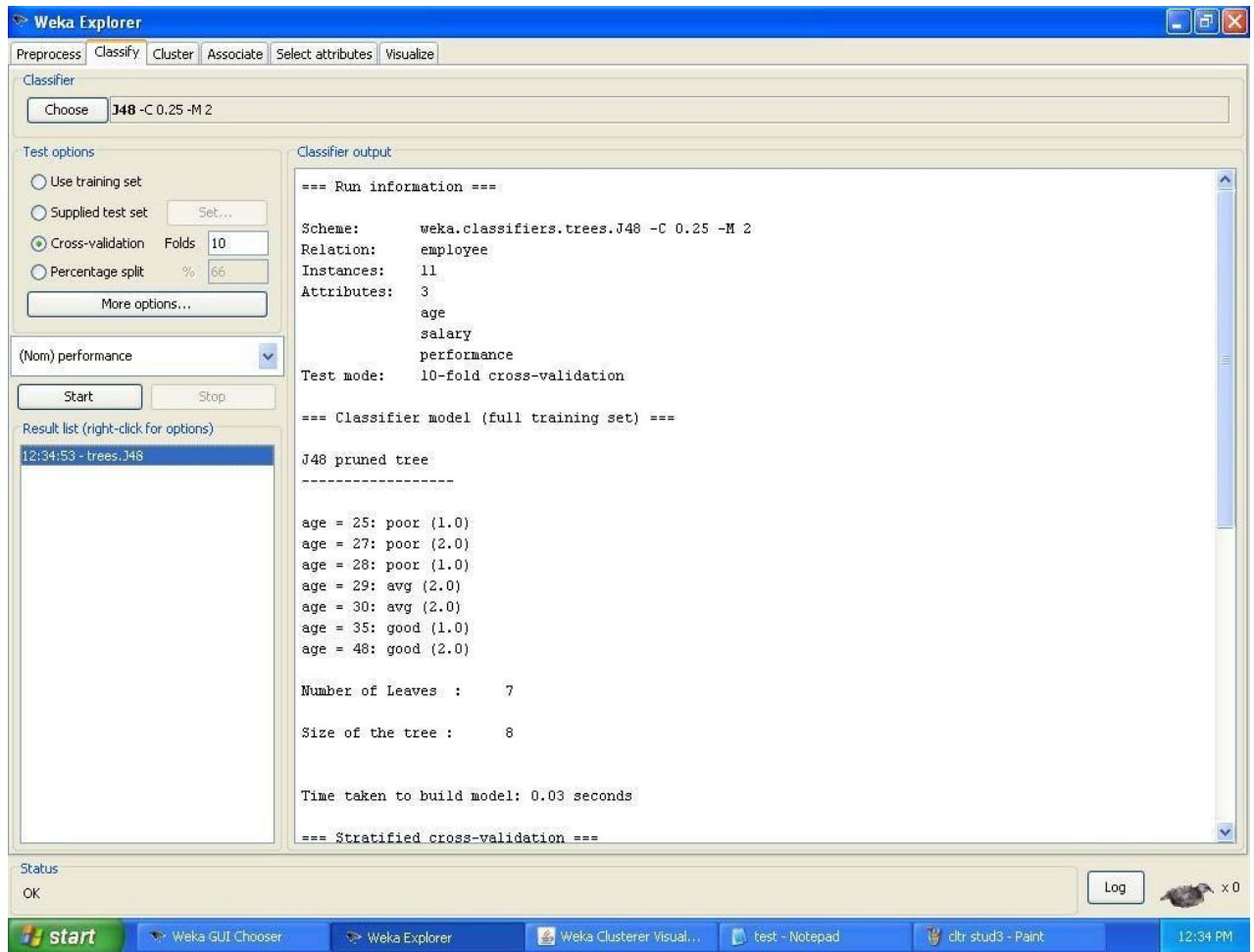
35, 32k, good

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  x 0

Windows taskbar: start | Weka GUI Chooser | Weka Explorer | Weka Clusterer Visual... | test - Notepad | df 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

```

=== Sum
=== Sum
Correct
Incorre
Kappa s
Mean ab
Root me
Relativ
Root re
Total M
=== Det

```

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

Tree View

```

graph TD
    age((age)) -->|<= 25| poor1[poor (1.0)]
    age -->|> 25| node1(( ))
    node1 -->|<= 27| poor2[poor (2.0)]
    node1 -->|> 27| node2(( ))
    node2 -->|<= 28| poor3[poor (1.0)]
    node2 -->|> 28| node3(( ))
    node3 -->|<= 29| avg1[avg (2.0)]
    node3 -->|> 29| node4(( ))
    node4 -->|<= 30| avg2[avg (2.0)]
    node4 -->|> 30| node5(( ))
    node5 -->|<= 35| good1[good (1.0)]
    node5 -->|> 35| node6(( ))
    node6 -->|<= 48| good2[good (2.0)]
    node6 -->|> 48| poor4[poor (1.0)]

```

Weights

```

=== Cor
a b c
1 2 0 | a = good
0 4 0 | b = avg
0 3 1 | c = poor

```

class
good
avg
poor

Status: OK

Log

Windows taskbar: start, Weka GUI Chooser, Weka Explorer, Weka Clusterer..., Weka Classifier T..., test - Notepad, cf 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

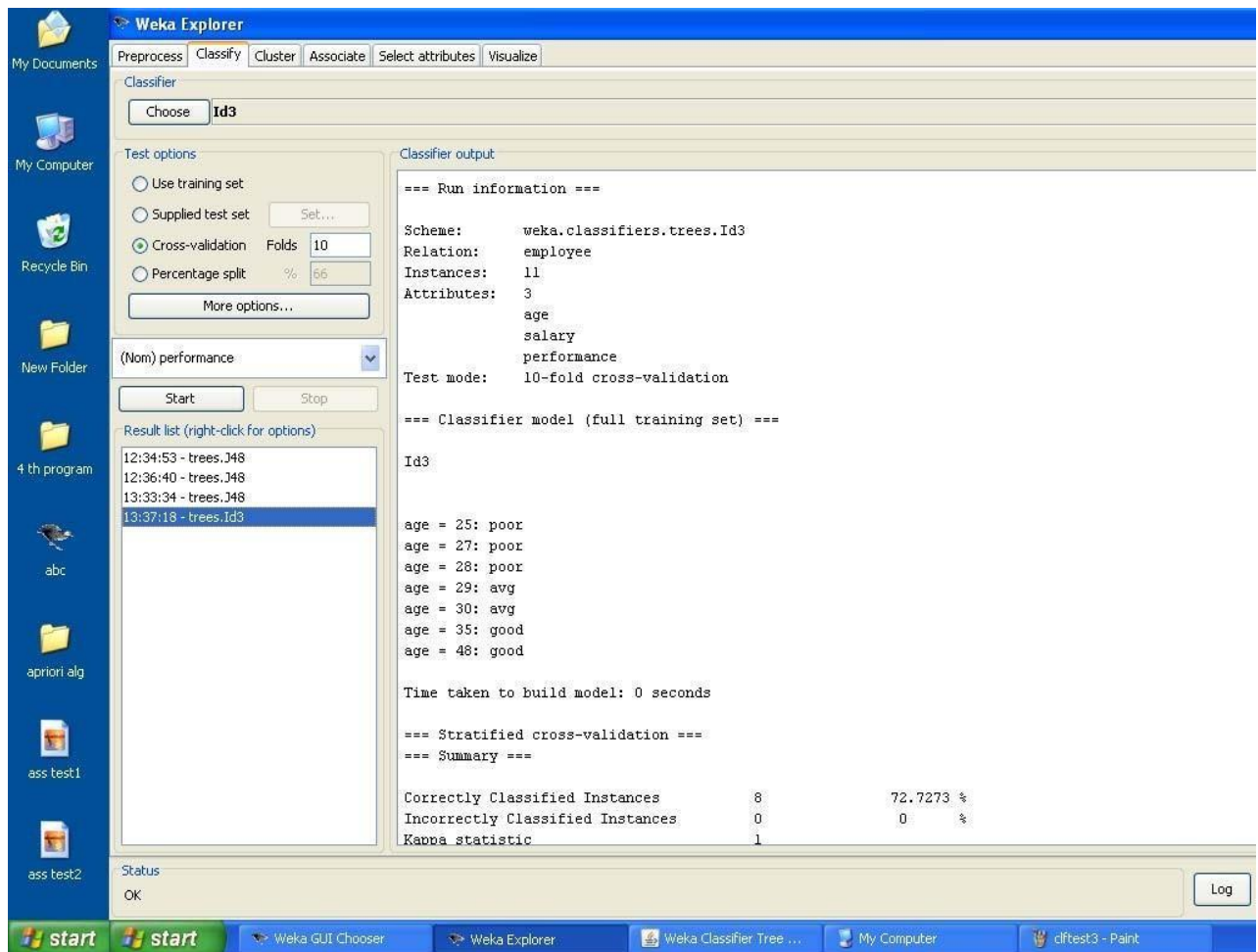
35, 32k, good

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.



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 (%: **55**)

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 NaiveBayes classifier selected. The main output pane 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
-----
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
  
```

The confusion matrix for the 'performance' attribute is as follows:

Attribute	good	avg	poor
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 %: 56

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

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

```

Log

start start Weka GUI Chooser Weka Explorer Weka Classifier Tree ... My Computer naive emp1 - Paint

2. 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.

The screenshot displays the Weka Explorer interface with the SimpleKMeans algorithm applied to the Iris dataset. The 'Clusterer' tab is active, showing the 'SimpleKMeans -N 2 -S 10' configuration. The 'Cluster mode' section is set to 'Use training set'. The 'Clusterer output' pane shows the following results:

```
=== Run information ===  
  
Scheme: weka.clusterers.SimpleKMeans -N 2 -S 10  
Relation: iris  
Instances: 150  
Attributes: 5  
    sepallength  
    sepalwidth  
    petallength  
    petalwidth  
    class  
Test mode: evaluate on training data  
  
=== Model and evaluation on training set ===  
  
kMeans  
=====
```

Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797

Cluster centroids:

Cluster	Mean/Mode	Std Devs	Class
Cluster 0	6.262 2.872 4.906 1.676	0.6628 0.3328 0.8256 0.4248	Iris-versicolor
Cluster 1	5.006 3.418 1.464 0.244	0.3525 0.381 0.1735 0.1072	Iris-setosa

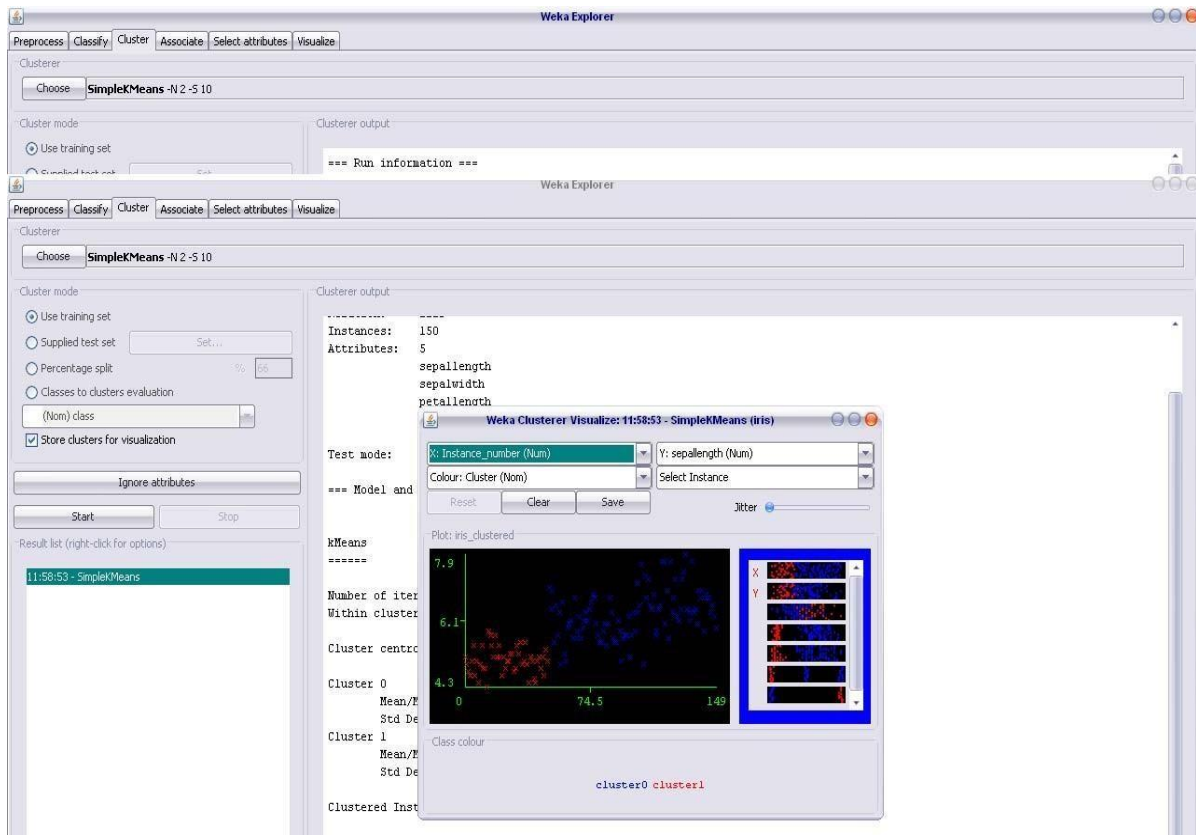
Clustered Instances

The status bar at the bottom shows 'OK' and a 'Log' button. The Windows taskbar at the very bottom shows the time as 12:02 PM.

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 characterstics 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

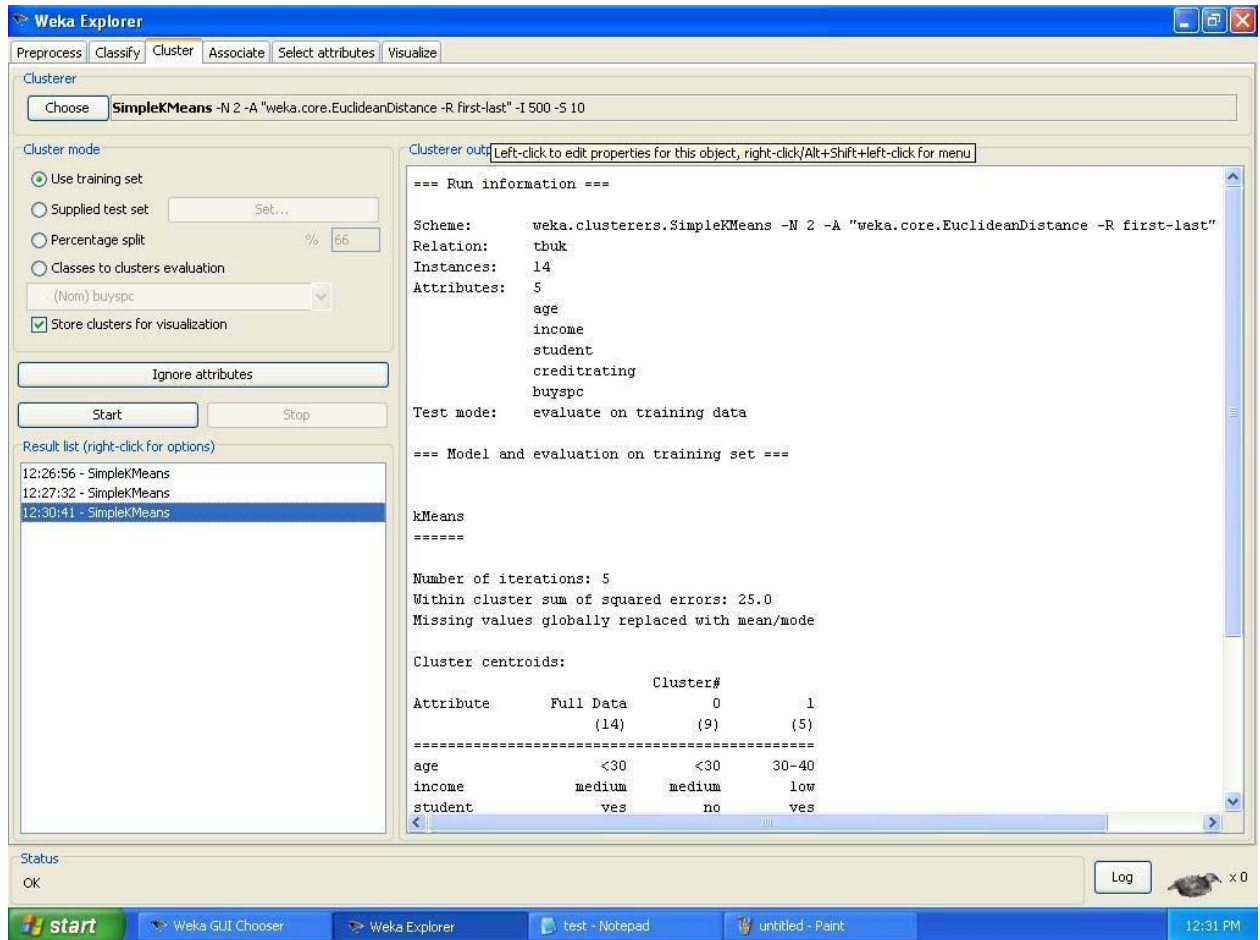
30-40, medium, no, excellent, yes

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.



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
- 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# 0	Cluster# 1	
	(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: **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

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

1. Generate Association rules for the following transactional database using Apriori algorithm.

TID	List of Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5