Nagar Yuwak Shikshan Sanstha's

YESHWANTRAO CHAVAN COLLEGE OF ENGINEERING

(An Autonomous Institution affiliated Rashtrasanth Tukadoji Maharaj Nagpur University, Nagpur)

HINGNA ROAD, WANADONGRI, NAGPUR

DEPARTMENT OF INFORMATION TECHNOLOGY

Academic Year: 2022-223

Course: Data Mining Lab **Sem.:7th**
Course Code: IT-2402 **Teaching Scheme: 2 Hrs/Week**
CA: 40 marks (4 MSPAs – 15 Marks each)
Name of Faculty: Prof. P. G. Jaiswal, Dr.Bhushan Manjre, Prof.Shweta Bokde

LIST OF PRACTICALS

Sr. No	Name of Practical	CO Mapped	PO Mapped
1	Introduction to **WEKA** – an open source data mining tool, Installing WEKA and Importing data from inbuilt datasets in WEKA Creating an .ARFF file and importing it in WEKA.	**CO1**	**PO1,PO2, PO5**
2	To implement **data preprocessing** on given dataset using **ORANGE /WEKA** machine learning tool.	**CO1,C04**	**PO1,PO2, PO5**
3	To Perform Association rule mining on dataset using **Apriori algorithm** with the help of data mining toolkit.	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
4	Demonstration of Association rule mining on dataset using **FP-Growth** algorithm	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
5	Demonstration of Classification using **Decision Tree (J48)** on dataset using the **WEKA** Explorer.	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
6	Write a program to implement **Naive Bayes Classification**.	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
7	Write a program to classify the tuples using **Linear Regression** methods.	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
8	Use WEKA to implement the following Clustering Algorithms – **K-means, Agglomerative, Divisive**	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
9	Write a program to implement **K-means clustering** algorithm	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
10	Implementation of KDD process in WEKA –Knowledge Flow	**CO2, CO3**	**PO1,PO2, PSO1,PO5 ,**
11	Case study-Implement project based on Data Pre-processing, Data analysis, Data Exploration using online dataset.	**CO2, CO3,CO4**	**PO1,PO2, PSO1,PO5 ,**

Prof. P.G.Jaiswal
Dr.Bhushan Manjre
Prof.Shweta Bokde
Course- Teacher(s)

Dr. R. C.Dharmik
HOD, IT

Lab1:

Aim : Introduction to WEKA – an open source data mining tool, Installing WEKA and Importing data from inbuilt datasets in WEKA ,Creating a .ARFF file and importing it in WEKA

Introduction to Weka - A Toolkit for data mining

Weka is open source software under the GNU General Public License.System is developed at the University of Waikato in New Zealand. "Weka" stands for the Waikato Environment for Knowledge Analysis. The software is freely available at http://www.cs.waikato.ac.nz/ml/weka

The system is written using object oriented language Java.There are several different levels at which Weka can be used . Weka provides implementations of state -of-the -art data mining and machine learning algorithms. Weka contains modules for data preprocessing, classification, clustering and association rule extraction.

Main features of Weka include:

49 data preprocessing tools
76 classification/regression algorithms
8 clustering algorithms
15 attribute/subset evaluators + 10 search algorithms for feature selection.
3 algorithms for finding association rules
3 graphical user interfaces

"The Explorer"	(exploratory data analysis)
"The Experimenter"	(experimental environment)
"The KnowledgeFlow"	(new process model inspired interface)

Weka: Download and Installation

Download Weka (the stable version) from http://www.cs.waikato.ac.nz/ml/weka/
Choose a self -extracting executable (including Java VM)
(If you are interested in modifying/extending weka there is a developer version that includes the source code)
After download is completed, run the self extracting file to install Weka, and use the default setups.

Start the Weka

From windows desktop,
-click "Start", choose "All programs",Choose "Weka 3.6" to start Weka
– Then the first interface window appears:
•Weka GUI Chooser (Fig. 1)

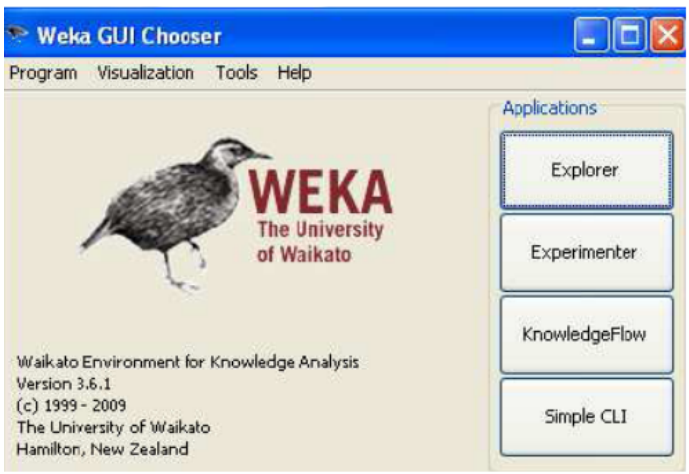


Fig. 1: Weka GUI

WEKA data formats:

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. It is a default file format for data analysis in WEKA.

ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information.

The **Header** of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
% 1. Title: Iris Plants Database
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris
@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

The **Data** of the ARFF file looks like the following:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
```

Lines that begin with a % are comments. The **@RELATION**, **@ATTRIBUTE** and **@DATA** declarations are case insensitive.

Examples

Several well-known machine learning datasets are distributed with Weka in the $WEKAHOME/data directory as ARFF files.

The ARFF Header Section

The ARFF Header section of the file contains the relation declaration and attribute declarations.

The @relation Declaration

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

```
@relation <relation-name>
```

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

The @attribute Declarations

Attribute declarations take the form of an orderd sequence of **@attribute** statements. Each attribute in the data set has its own **@attribute** statement which uniquely defines the name of that attribute and it's data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then Weka expects that all that attributes values will be found in the third comma delimited column.

The format for the **@attribute** statement is:

```
@attribute <attribute-name> <datatype>
```

where the <*attribute-name*> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <*datatype*> can be any of the four types currently (version 3.2.1) supported by Weka:

- numeric
- <nominal-specification>
- string
- date [<date-format>]

where <nominal-specification> and <date-format> are defined below. The keywords **numeric**, **string** and **date** are case insensitive.

Numeric attributes

Numeric attributes can be real or integer numbers.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}

For example, the class value of the Iris dataset can be defined as follows:

```
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Values that contain spaces must be quoted.

String attributes

String attributes allow us to create attributes containing arbitrary textual values. This is very useful in text-mining applications, as we can create datasets with string attributes, then write Weka Filters to manipulate strings (like StringToWordVectorFilter). String attributes are declared as follows:

```
@ATTRIBUTE LCC    string
```

Date attributes

Date attribute declarations take the form:

```
@attribute <name> date [<date-format>]
```

where <name> is the name for the attribute and <date-format> is an optional string specifying how date values should be parsed and printed (this is the same format used by SimpleDateFormat). The default format string accepts the ISO-8601 combined date and time format: "yyyy-MM-dd'T'HH:mm:ss".

Dates must be specified in the data section as the corresponding string representations of the date/time (see example below).

ARFF Data Section

The ARFF Data section of the file contains the data declaration line and the actual instance lines.

The @data Declaration

The **@data** declaration is a single line denoting the start of the data segment in the file. The format is:

```
    @data
```

The instance data

Each instance is represented on a single line, with carriage returns denoting the end of the instance.

Attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section (i.e. the data corresponding to the nth **@attribute** declaration is always the nth field of the attribute).

Missing values are represented by a single question mark, as in:

```
    @data
    4.4,?,1.5,?,Iris-setosa
Values of string and nominal attributes are case sensitive, and any that
contain space must be quoted, as follows:
    @relation LCCvsLCSH

    @attribute LCC string
    @attribute LCSH string

    @data
    AG5,   'Encyclopedias and dictionaries.;Twentieth century.'
    AS262, 'Science -- Soviet Union -- History.'
    AE5,   'Encyclopedias and dictionaries.'
```

Data import in WEKA

- Data can be imported from a file in various formats: ARFF, CSV, C4.5, binary
- Data can also be read from a URL or from an SQL database (using JDBC)
- Pre-processing tools in WEKA are called “filters”
- WEKA contains filters for:
    - Discretization, normalization, resampling, attribute selection, transforming and combining attributes, …

Weka Application Interfaces

•Explorer– preprocessing, attribute selection, learning, visualiation
•Experimenter– testing and evaluating machine learning algorithms
•Knowledge Flow– visual design of KDD process–Explorer
•Simple Command -line– A simple interface for typing commands

Conclusion :

Lab 2.

a. 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.

Create a Dataset student.arff as given below:

@relation student

@attribute age integer

@attribute income {low, medium, high}

@attribute student {yes, no}

@attribute credit-rating {fair, excellent}

@attribute buyspc {yes, no}

@data

28, high, no, fair, no

24, high, no, excellent, no

35, high, no, fair, yes

41, medium, no, fair, yes

42, low, yes, fair, yes

43, low, yes, excellent, no

33, low, yes, excellent, yes

22, medium, no, fair, no

23, low, yes, fair, no

44, medium, yes, fair, yes

25, medium, yes, excellent, yes

32, medium, no, excellent, yes

37, high, yes, fair, yes

45, medium, no, excellent, no

Apply following steps:

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.

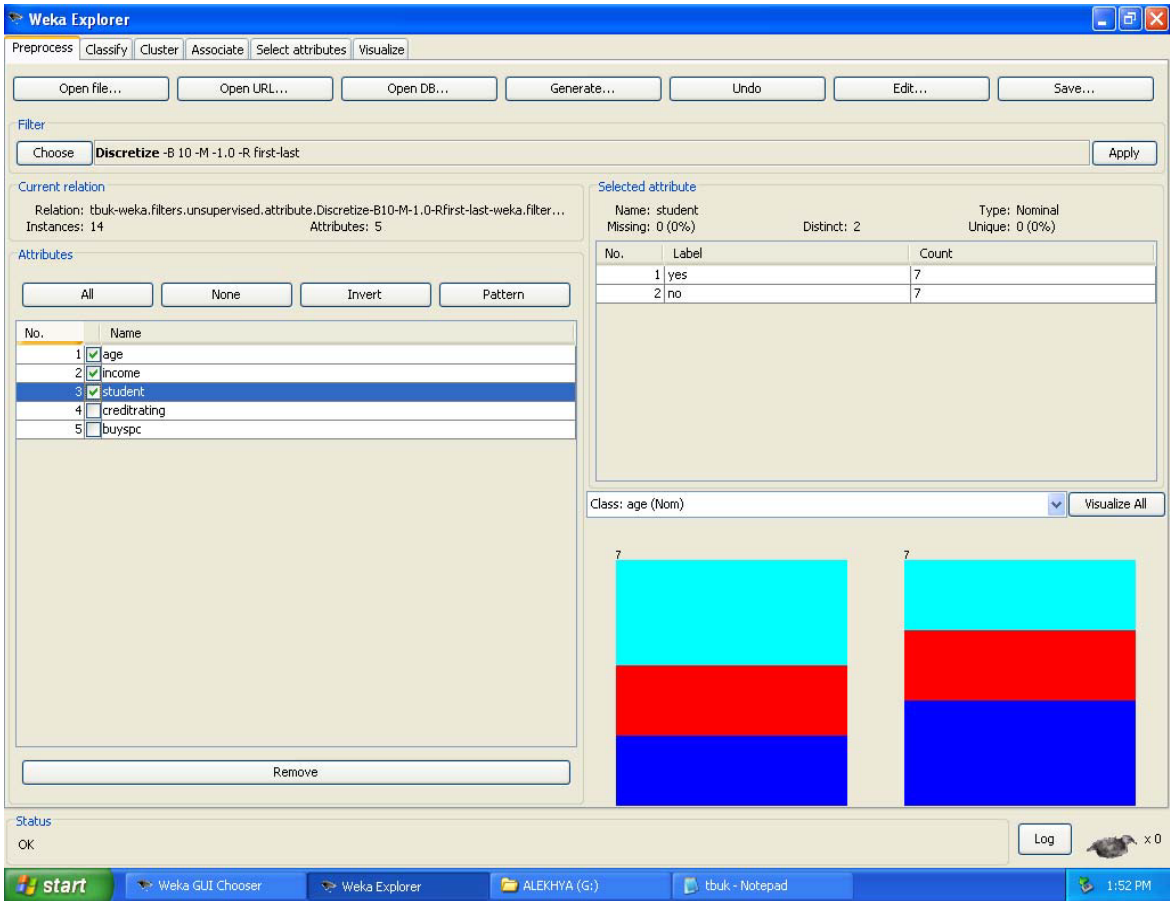
d)Save the new working relation as an arff file by clicking save button on the top(button)panel.(student.arff).

Discretization

1) Sometimes 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.

- Let us divide the values of age attribute into three bins(intervals).
- First load the dataset into weka(student.arff)
- Select the age attribute.
- Activate filter-dialog box and select "WEKA.filters.unsupervised.attribute.discretize"from the list.
- To change the defaults for the filters,click on the box immediately to the right of the choose button.
- 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.
- Enter '3' as the number of bins.Leave the remaining field values as they are.
- Click OK button.
- Click apply in the filter panel.This will result in a new working relation with the selected attribute partition into 3 bins.
- Save the new working relation in a file called student-data-discretized.arff .

The following screenshot shows the effect of discretization.



b. Demonstration of preprocessing on dataset bank.csv:

i. Create bank.csv file with following data and import it in weka and perform data preprocessing tasks.

ii. Remove id attribute.

iii. Discretize age and income attributes

id	age	gender	region	income	married	children	car	save_act	current_act	mortgage	pep
ID12101	48	FEMALE	INNER_CITY	17546	NO	1	NO	NO	NO	NO	YES
ID12102	40	MALE	TOWN	30085.1	YES	3	YES	NO	YES	YES	NO
ID12103	51	FEMALE	INNER_CITY	16575.4	YES	0	YES	YES	YES	NO	NO
ID12104	23	FEMALE	TOWN	20375.4	YES	3	NO	NO	YES	NO	NO
ID12105	57	FEMALE	RURAL	50576.3	YES	0	NO	YES	NO	NO	NO
ID12106	57	FEMALE	TOWN	37869.6	YES	2	NO	YES	YES	NO	YES
ID12107	22	MALE	RURAL	8877.07	NO	0	NO	NO	YES	NO	YES
ID12108	58	MALE	TOWN	24946.6	YES	0	YES	YES	YES	NO	NO
ID12109	37	FEMALE	SUBURBAN	25304.3	YES	2	YES	NO	NO	NO	NO
ID12110	54	MALE	TOWN	24212.1	YES	2	YES	YES	YES	NO	NO
ID12111	66	FEMALE	TOWN	59803.9	YES	0	NO	YES	YES	NO	NO
ID12112	52	FEMALE	INNER_CITY	26658.8	NO	0	YES	YES	YES	YES	NO
ID12113	44	FEMALE	TOWN	15735.8	YES	1	NO	YES	YES	YES	YES
ID12114	66	FEMALE	TOWN	55204.7	YES	1	YES	YES	YES	YES	YES
ID12115	36	MALE	RURAL	19474.6	YES	0	NO	YES	YES	YES	NO
ID12116	38	FEMALE	INNER_CITY	22342.1	YES	0	YES	YES	YES	YES	NO
ID12117	37	FEMALE	TOWN	17729.8	YES	2	NO	NO	NO	YES	NO
ID12118	46	FEMALE	SUBURBAN	41016	YES	0	NO	YES	NO	YES	NO
ID12119	62	FEMALE	INNER_CITY	26909.2	YES	0	NO	YES	NO	NO	YES
ID12120	31	MALE	TOWN	22522.8	YES	0	YES	YES	YES	NO	NO
ID12121	61	MALE	INNER_CITY	57880.7	YES	2	NO	YES	NO	NO	YES
ID12122	50	MALE	TOWN	16497.3	YES	2	NO	YES	YES	NO	NO
ID12123	54	MALE	INNER_CITY	38446.6	YES	0	NO	YES	YES	NO	NO
ID12124	27	FEMALE	TOWN	15538.8	NO	0	YES	YES	YES	YES	NO
ID12125	22	MALE	INNER_CITY	12640.3	NO	2	YES	YES	YES	NO	NO
ID12126	56	MALE	INNER_CITY	41034	YES	0	YES	YES	YES	YES	NO
ID12127	45	MALE	INNER_CITY	20809.7	YES	0	NO	YES	YES	YES	NO

Conclusion:

Lab3:
Aim: To illustrate some of the basic elements of association rule mining using WEKA.

Introduction
Developed by Agrawal and Srikant 1994 .
Innovative way to find association rules on large scale, allowing implication outcomes that consist of more than one item
Based on minimum support threshold
Limitations of Apriori algorithm
Needs several iterations of the data
Uses a minimum support threshold
Difficulties to find rarely occuring events
Alternative methods (other than apriori) can address this by using a minimum support thresold
Some competing alternative approaches focus on partition and sampling.

• Data mining can typically be used with transactional databases (for ex. in shopping cart analysis)
• Aim can be to build association rules about the shopping events
• Based on item sets, such as

{milk, cocoa powder} 2-itemset
{milk, corn flakes, bread} 3-itemset

Association rules
• Items that occur often together can be associated to each other
• These together occuring items form a **frequent itemset**
• Conclusions based on the frequent itemsets form **association rules**
• For ex. {milk, cocoa powder} can bring a rule *cocoa powder*$_e$*milk*

Support and confidence
• If confidence gets a value of 100 % the rule is an **exact rule**
• Even if confidence reaches high values the rule is not useful unless the support value is high as well
• Rules that have both high confidence and support are called **strong rules**
• Some competing alternative approaches can generate useful rules even with low support values

Generating association rules
• Usually consists of two sub problems:
1) Finding frequent itemsets whose occurrences exceed a predefined minimum support threshold
2) Deriving association rules from those frequent itemsets (with the constrains of minimum confidence threshold)
• These two sub problems are solved iteratively until new rules no more emerge
• The second sub problem is quite straight- forward and most of the research focus is on the first sub problem.

Use of apriori algorithm
• Initial information: transactional database D and user-defined numeric minimum support threshold *min_sup*
• Algorithm uses knowledge from previous iteration phase to produce frequent itemsets
• This is reflected in the Latin origin of the name that means ‖from what comes before.

Creating frequent sets
• Let‘s define:$C_k$as a candidate itemset of size kL_k as a frequent itemset of size k

• Main steps of iteration are:
1) Find frequent set L_{k-1}
2) Join step: C_k is generated by joining L_{k-1} with itself (Cartesian product Lk-1 x Lk-1)
3) Prune step (apriori property): Any (k − 1) size itemset that is not frequent cannot be a subset of a frequent k size itemset, hence should be removed
4) Frequent set L_k has been achieved.

Algorithm uses breadth-first search and a hash tree structure to make candidate itemsets efficiently Then occurrence frequency for each candidate itemset is counted .Those candidate itemsets that have higher frequency than minimum support threshold are qualified to be frequent itemsets.

APRIORI ALGORITHM IN PSEUDOCODE
L_1= {frequent items};
for(k= 2; L_{k-1}!=∅; k++) **do begin**
C_k= candidates generated from L_{k-1} (that is: Cartesian product L_{k-1} x L_{k-1} and eliminating any k-1 size itemset that is not frequent); **for each** transaction t in database **do**
increment the count of all candidates in C_k that are contained in t
L_k = candidates in C_k with *min_sup*
end
return

Apriori algorithm using WEKA :

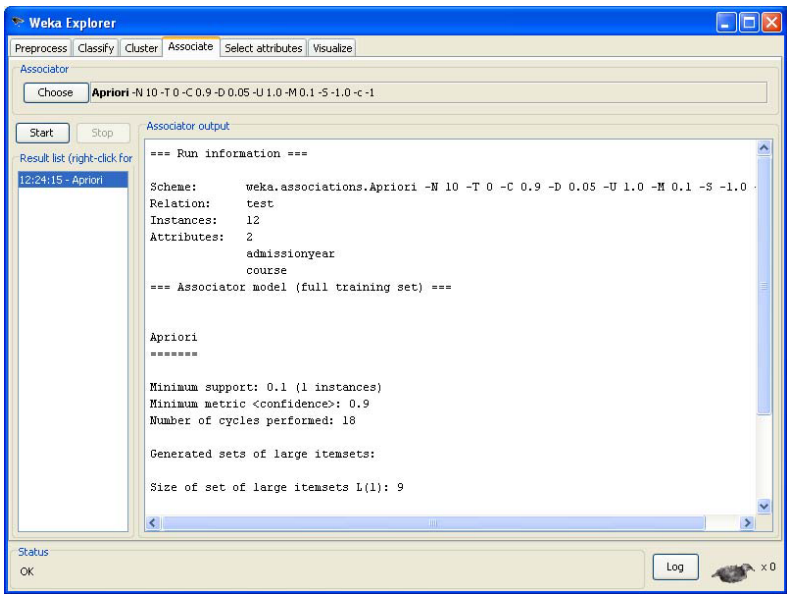
Step1: Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized. In this example it is age attribute.

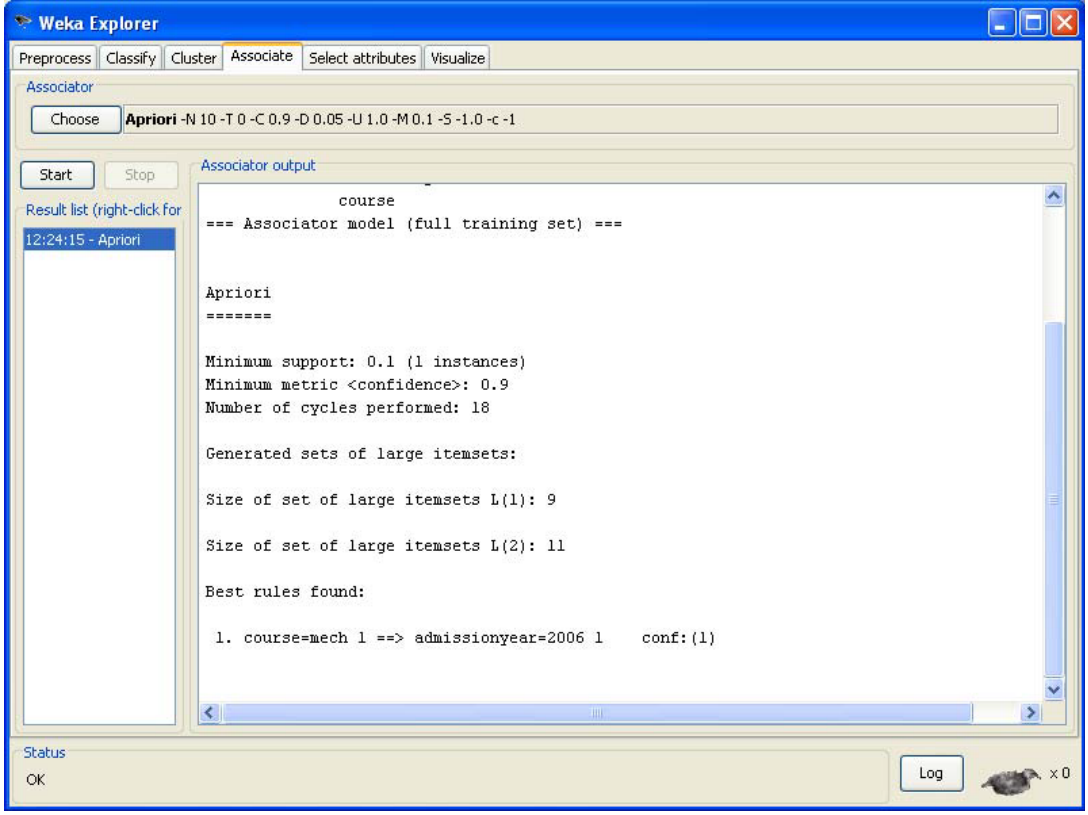
Step2: Clicking on the associate tab will bring up the interface for association rule algorithm.

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.

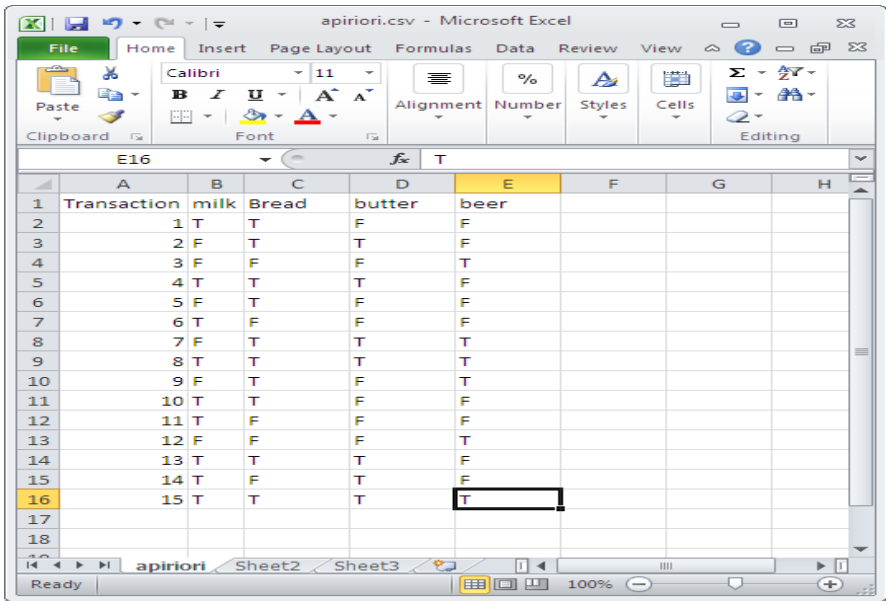
The following screenshot shows the association rules that were generated when apriori algorithm is applied on the given dataset.





Task:

1. Open **Excel** and prepare following dataset and save as **test.csv.** Apply preprocessing steps(if required) and then apply Apriori algorithm on it to find frequent association rules and **interpret your results.**



	A	B	C	D	E	F	G	H
1	Transaction	milk	Bread	butter	beer			
2	1	T	T	F	F			
3	2	F	T	T	F			
4	3	F	F	F	T			
5	4	T	T	T	F			
6	5	F	T	F	F			
7	6	T	F	F	F			
8	7	F	T	T	T			
9	8	T	T	T	T			
10	9	F	T	F	T			
11	10	T	T	F	F			
12	11	T	F	F	F			
13	12	F	F	F	T			
14	13	T	T	T	F			
15	14	T	F	T	F			
16	15	T	T	T	T			
17								
18								

2. Apply Apriori on **supermarket dataset**(available in weka) , discover association rules and **interpret** your results

Conclusion :

Lab4:

Aim:
To illustrate the association mining using FP-Growth in weka.

Introduction
Apriori: uses a generate-and-test approach – generates candidate itemsets and tests if they are frequent.
– Generation of candidate itemsets is expensive (in both space and time)
– Support counting is expensive
• Subset checking (computationally expensive)
• Multiple Database scans (I/O)

FP-Growth: allows frequent itemset discovery without candidate itemset generation. Two step approach:

– Step 1: Build a compact data structure called the FP-tree.
Built using 2 passes over the data-set.
– Step 2: Extracts frequent item sets directly from the FP-tree

Step 1: fp-tree construction
FP-Tree is constructed using 2 passes over the data-set: Pass 1:
Scan data and find support for each item.
Discard infrequent items.
Sort frequent items in decreasing order based on their support. Pass 2:

Nodes correspond to items and have a counter
1. FP-Growth reads 1 transaction at a time and maps it to a path
– Fixed order is used, so paths can overlap when transactions share items. In this case, counters are incremented
2. Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines)
– The more paths that overlap, the higher the compression. FP-tree may fit in memory.
3. Frequent itemsets extracted from the FP-Tree.

Procedure:
1. Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized.
2. Clicking on the associate tab will bring up the interface for association rule algorithm.
3. We will use FP-Growth algorithm.
4. In order 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.

Task:

1. Open **Excel** and prepare following dataset and save as **test.csv (lab3 table).** Apply preprocessing steps(if required) and then apply FP-GROWTH algorithm on it to find frequent association rules and **interpret your results.**
2. Apply FP-GROWTH on **supermarket dataset**(available in weka) , discover association rules and **interpret** your results

Conclusion :

Lab5:

Aim:
To illustrate the Classification using decision tree in weka.

Introduction

Task1:

Open the data/iris.arff Dataset
Click the "**Open file…**" button to open a data set and double click on the "**data**" directory. Weka provides a number of small common machine learning datasets that you can use to practice on.Select the "**iris.arff**" file to load the Iris dataset. The Iris Flower dataset is a famous dataset from statistics and is heavily borrowed by researchers in machine learning. It contains 150 instances (rows) and 4 attributes (columns) and a class attribute for the species of iris flower (one of setosa, versicolor, and virginica).

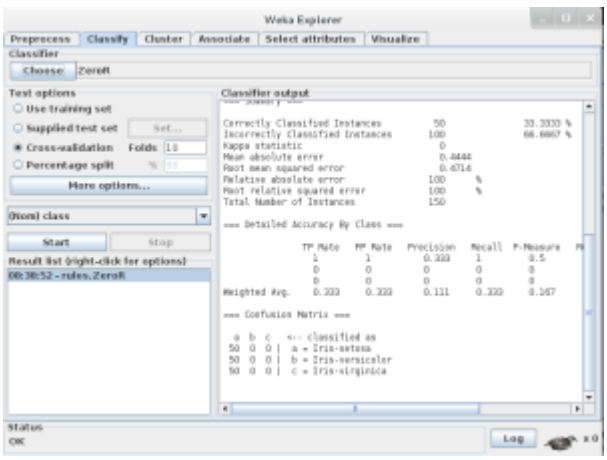
Select and Run an Algorithm

Once loaded a dataset, it's time to choose a machine learning algorithm to model the problem and make predictions.

Click the "**Classify**" tab. This is the area for running algorithms against a loaded dataset in Weka.

You will note that the "**ZeroR**" algorithm is selected by default.

Click the "**Start**" button to run this algorithm.



Weka Results for the ZeroR algorithm on the Iris flower dataset

The ZeroR algorithm selects the majority class in the dataset (all three species of iris are equally present in the data, so it picks the first one: setosa) and uses that to make all predictions. This is the baseline for the dataset and the measure by which all algorithms can be compared. The result is 33%, as expected (3 classes, each equally represented, assigning one of the three to each prediction results in 33% classification accuracy).

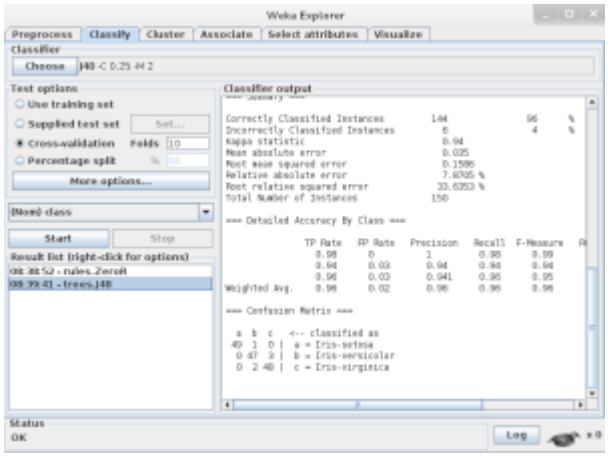
You will also note that the test options selects **Cross Validation** by default with **10 folds**. This means that the dataset is split into 10 parts: the first 9 are used to train the algorithm, and the 10th is used to assess the algorithm. This process is repeated, allowing each of the 10 parts of the split dataset a chance to be the held-out test set. You can read more about cross validation here.

The J48 algorithm:

Click the “Choose” button in the “Classifier” section and click on “trees” and click on the “J48” algorithm.

This is an implementation of the C4.8 algorithm in Java (“J” for Java, 48 for C4.8, hence the J48 name) and is a minor extension to the famous C4.5 algorithm. You can read more about the C4.5 algorithm here.

Click the “**Start**” button to run the algorithm.



Weka J48 algorithm results on the Iris flower dataset

5. Review Results

After running the J48 algorithm, you can note the results in the “Classifier output” section.

The algorithm was run with 10-fold cross-validation: this means it was given an opportunity to make a prediction for each instance of the dataset (with different training folds) and the presented result is a summary of those predictions.

```
Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   R
                 0.98      0          1         0.98      0.99
                 0.94      0.03       0.94      0.94      0.94
                 0.96      0.03       0.941     0.96      0.95
Weighted Avg.    0.96      0.02       0.96      0.96      0.96

=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Just the results of the J48 algorithm on the Iris flower dataset in Weka

Firstly, note the Classification Accuracy. You can see that the model achieved a result of 144/150 correct or 96%, which seems a lot better than the baseline of 33%.

Secondly, look at the Confusion Matrix. You can see a table of actual classes compared to predicted classes and you can see that there was 1 error where an Iris-setosa was classified as an Iris-versicolor, 2 cases where Iris-virginica was classified as an Iris-versicolor, and 3 cases where

an Iris-versicolor was classified as an Iris-setosa (a total of 6 errors). This table can help to explain the accuracy achieved by the algorithm.

Task2 : apply J48 algorithm on student dataset (refer class example- age, income, student, credit rating, buy_pc) . Show and interpret the results . show also the decision tree.

Conclusion

In this lab , loaded first the dataset and ran first machine learning algorithm (an implementation of the C4.8 algorithm) in Weka. The ZeroR algorithm doesn't really count: it's just a useful baseline.