GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.

Department of Computer Science and Engineering

III Year II Semester

2019 - 2020

DATA MINING-LAB
FACULTY MANUAL

Prepared by

Dr. G.V.S.N.R.V.Prasad
Professor & Vice Prinicpal(Academics)

Dr.A.Jagadeswara Rao
Associate Professor

Mr. K. Bala Brahmeswara
Assistant Professor

Mrs. Y. Aditya
Assistant Professor

GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institution with Permanent Affiliation to JNTUK, Kakinada)

Seshadri Rao Knowledge Village, Gudlavalleru – 521356

DEPARTMENT OF COMPUTER SCIENCE AND ENGIEERING

INSTITUTE VISION & MISSION

Institute Vision:

To be a leading institution of engineering education and research, preparing students for

leadership in their fields in a caring and challenging learning environment.

Institute Mission:

- To produce quality engineers by providing state-of-the-art engineering education.
- To attract and retain knowledgeable, creative, motivated and highly skilled individuals whose leadership and contributions uphold the college tenets of education, creativity, research and responsible public service.
- To develop faculty and resources to impart and disseminate knowledge and information to students and also to society that will enhance educational level, which in turn, will contribute to social and economic betterment of society.
- To provide an environment that values and encourages knowledge acquisition and academic freedom, making this a preferred institution for knowledge seekers.
- To provide quality assurance.
- To partner and collaborate with industry, government, and R&D institutes to develop new knowledge and sustainable technologies and serve as an engine for facilitating the nation's economic development.
- To impart personality development skills to students that will help them to succeed and lead.
- To instil in students the attitude, values and vision that will prepare them to lead lives of personal integrity and civic responsibility.
- To promote a campus environment that welcomes and makes students of all races, cultures and civilizations feel at home.
- Putting students face to face with industrial, governmental and societal challenges.

DEPARTMENT VISION & MISSION

Vision

To be a Centre of Excellence in Computer Science and Engineering education and training to meet the challenging needs of the industry and society.

Mission

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Identify, analyze, formulate and solve computing problems both independently and in a team environment using appropriate modern tools.

PEO 2: Develop software systems with significant technical, legal, ethical, social, environmental and economic considerations.

PEO 3: Exhibit commitment in lifelong learning, professional development and leadership and communicate effectively with professional clients and the public.

PROGRAM SPECIFIC OUTCOMES (PSOs)

1. Design, develop, test and maintain reliable software systems and intelligent systems.

2. Design and develop websites, web apps and mobile apps.

Program Outcomes (Pos)

(B.Tech in Computer Science and Engineering)

Engineering students will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental

contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Department of Computer Science and Engineering

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab. c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

COURSE NAME: **DATA MINING LAB**

COURSE CODE: **CS2514**

COURSE OBJECTIVES:

To exercise the data mining techniques such as classification, clustering, pattern mining etc with different datasets and dynamic parameters using WEKA tool.

COURSE OUTCOMES:

Upon successful completion of the course, the students will be able to

- Learn to execute data mining tasks using a data mining toolkit (such as WEKA) and visualize the results.
- Demonstrate the working of algorithms for data mining tasks such association rule mining, classification, clustering and regression.

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES:

COURSE OUTCOMES	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	P10	P11	P12
CO1	√	√		√	√							√
CO2	√	√			√							√

DATAMINING LAB INDEX

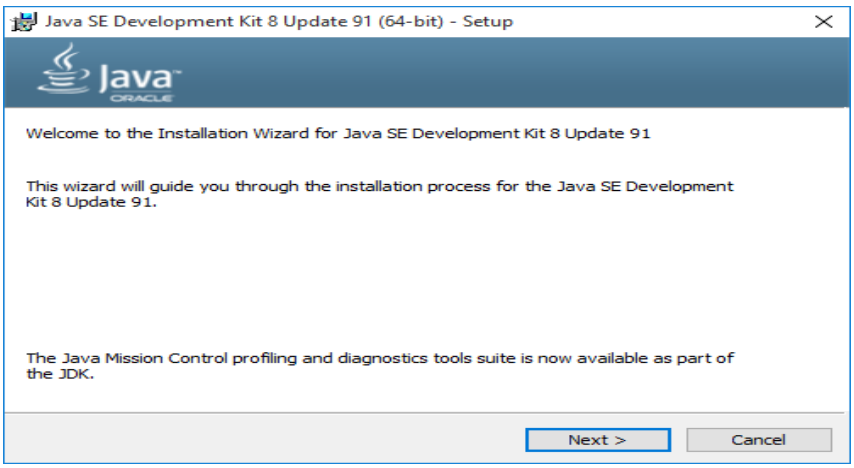
ADDITIONAL –EXPERIMENT LIST

AIM:

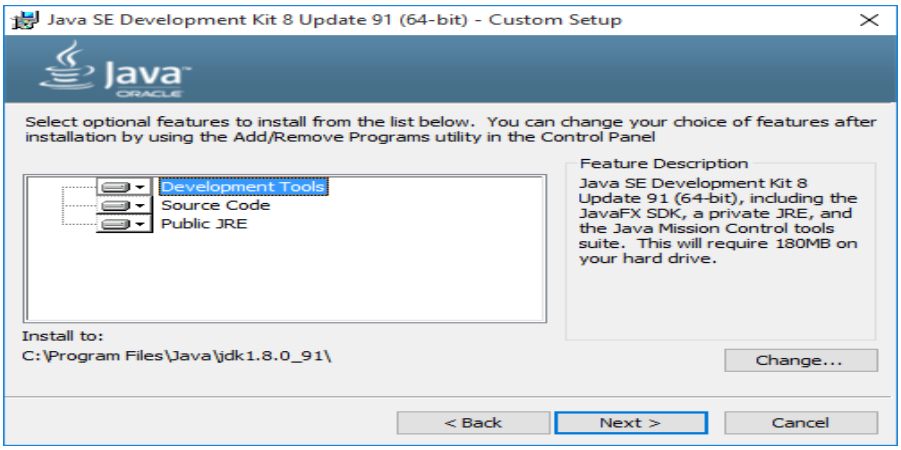
i) Downloading and/or installation of WEKA data mining toolkit

Note: if weka installation package don't have java included in it then compatible version of the java has to be installed initially as given below.

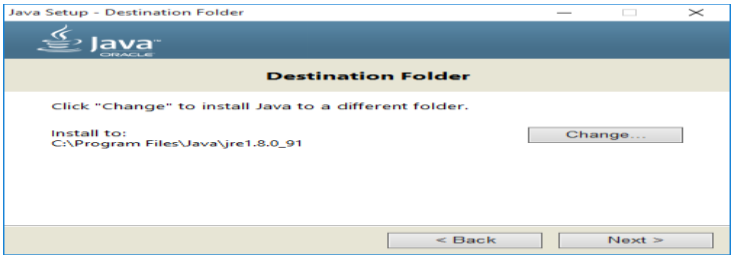
Step 1: Select appropriate JAVA version (Here we are using:jdk1.8). Operate the executable file of welcome window to the installation wizard of Development kit ,click next button



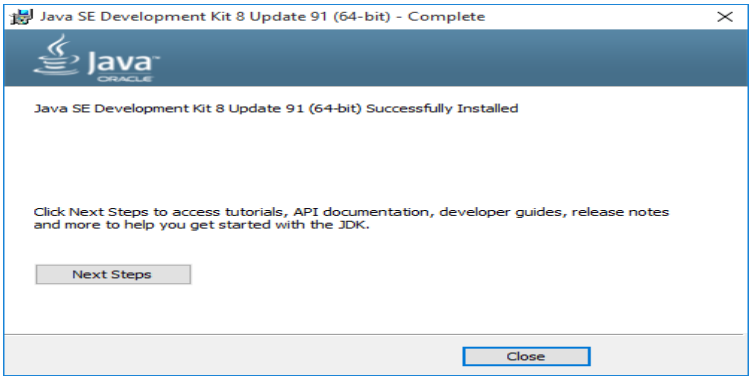
Step 2: Custom setup window to select path and click next.



Step 3: In the destination folder window we check java file stored location and click next.



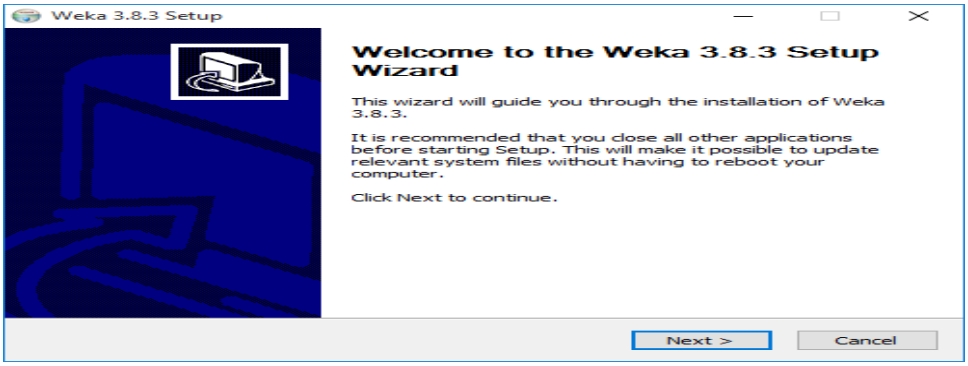
Step 4: Installation process completion window is displayed and click close.



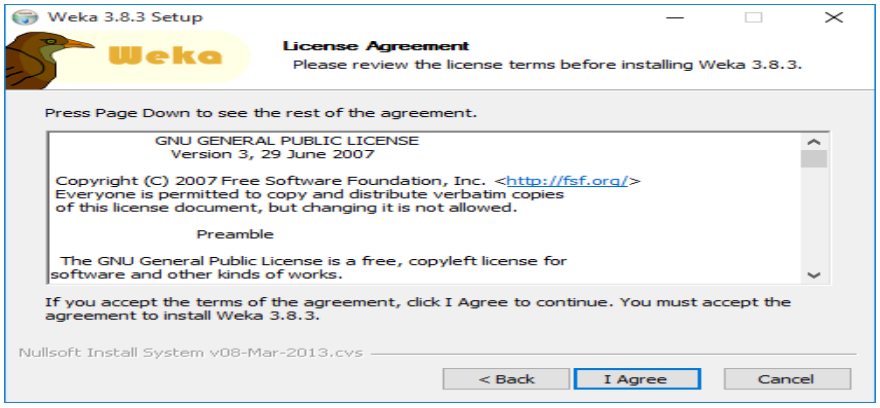
Weka –Installation:

Go to the Weka website, http://www.cs.waikato.ac.nz/ml/weka/, and download the software.On the left hand side, click on the link that says download. Select the appropriate link corresponding to the version of the software based on your operating system and whether or not you already have Java VM running on your machine.Save the self extracting executable to the disk and double-click on it to install weka.

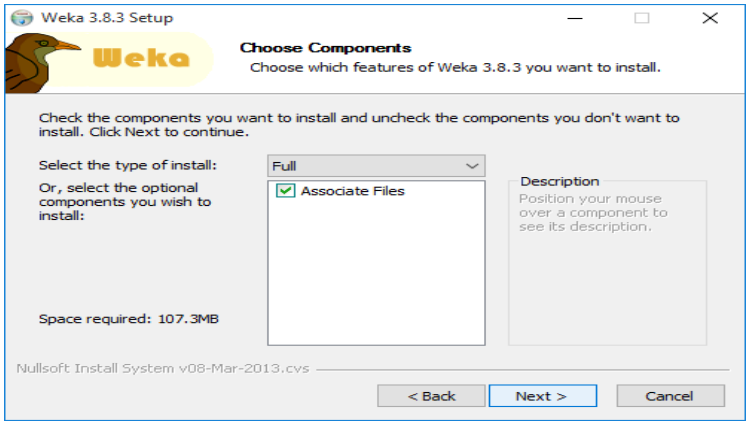
Step 1: Welcome setup wizard window is viewed where next button has to be clicked



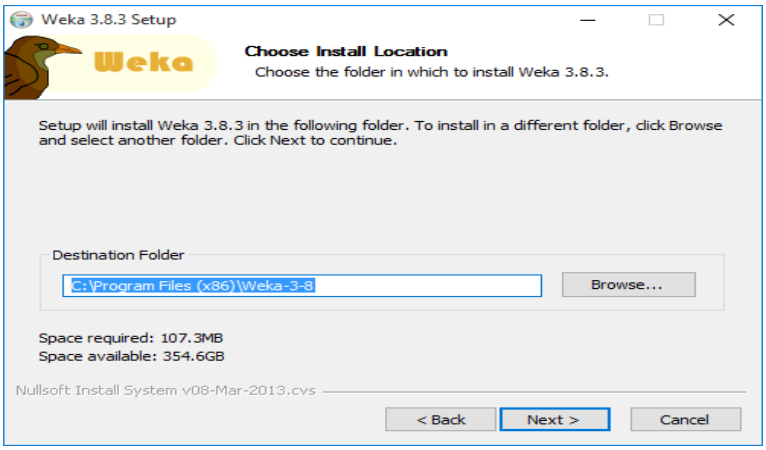
Step2: In the license agreement window accept the agreement with “I Agree” button.



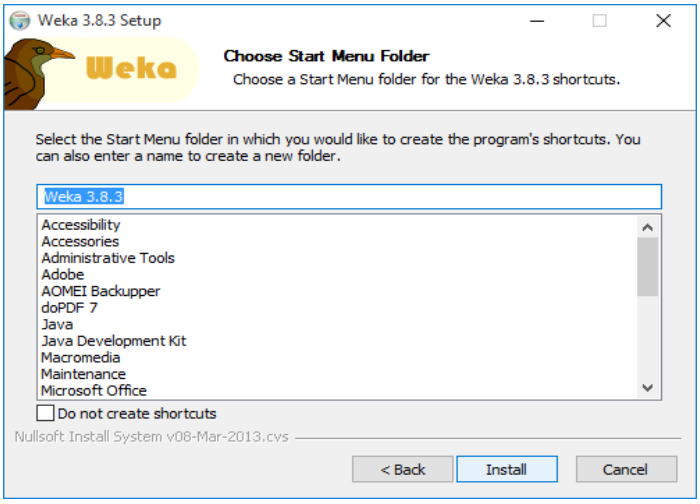
Step 3: The component window is displayed where the type of installation must be “Full” and click on next.



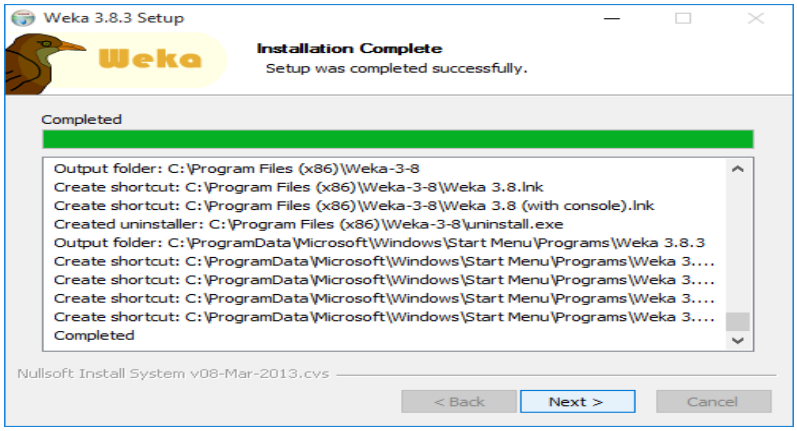
Step 4: In the installation window to select the path where the software loads can be selected through Browse button to the destination folder and click next button.



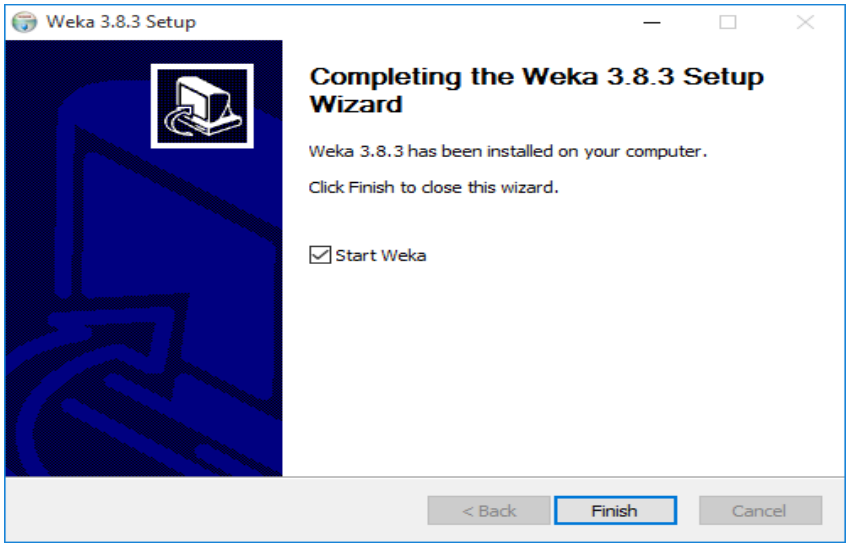
Step 5: To add to the list of programs from the start menu check the create shortcut and press the Install button



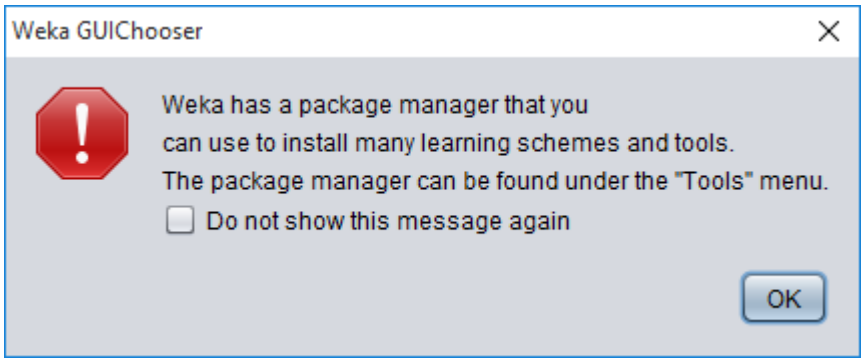
Step 6: completion of installation is indicated in the window where the next button is selected.



Step 7: The setup wizard with the start up of Weka is chosen and finish the process.



Step 8: An warning window tells about the feature of package manager, click ok.

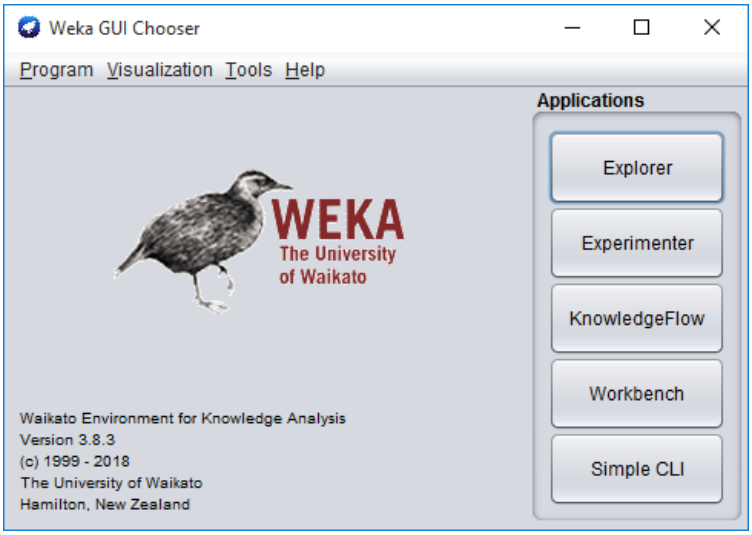


Step 9: Weka home page is viewed.

Step 10: The installation process is completed fully.

2. Understand the feature of Weka toolkit

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting point for launching Weka's main GUI applications and supporting tools. If one prefers a MDI ("multiple document interface") appearance, then this is provided by an alternative launcher called "Main" (class weka.gui.Main). The GUI Chooser consists of four buttons—one for each of the four major Weka applications—and four menus.



The buttons can be used to start the following applications:

• **Explorer** An environment for exploring data with WEKA (the rest of this documentation deals with this application in more detail).

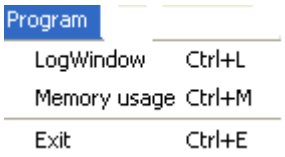
• **Experimenter** An environment for performing experiments and conduct ing statistical tests between learning schemes.

• **KnowledgeFlow** This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.

• **SimpleCLI** Provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface.

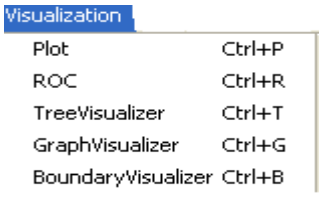
The menu consists of four sections:

1. **Program**



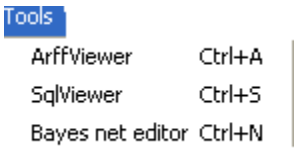
- **LogWindow** Opens a log window that captures all that is printed to stdout or stderr. Useful for environments like MS Windows, where WEKA is normally not started from a terminal.
- **Exit** Closes WEKA.

2. **Visualization**



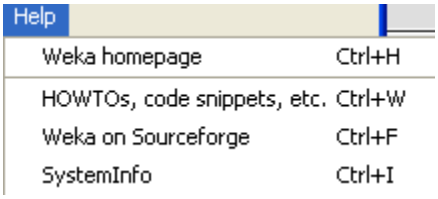
- **Plot** For plotting a 2D plot of a dataset.
- **ROC** Displays a previously saved ROC curve.
- **TreeVisualizer** For displaying directed graphs, e.g., a decision tree.
- **GraphVisualizer** Visualizes XML BIF or DOT format graphs, e.g., for Bayesian networks.
- **BoundaryVisualizer** Allows the visualization of classifier decision boundaries in two dimensions.

3. Tools



- **ArffViewer** An MDI application for viewing ARFF files in spread- sheet format.
- **SqlViewer** Represents an SQL worksheet, for querying databases via JDBC.
- **Bayes net editor** An application for editing, visualizing and learn- ing Bayes nets.

4. Help



- **Weka homepage** Opens a browser window with WEKA's home-page.
- **HOWTOs, code snippets, etc.** The general WekaWiki [2], con-taining lots of examples and HOWTOs around the development and use of WEKA.
- **Weka on Sourceforge**WEKA's project homepage on Sourceforge.net.
- **SystemInfo** Lists some internals about the Java/WEKA environment, e.g., the CLASSPATH.

3. Navigate the options available in the WEKA (ex. Select attributes panel, preprocess panel, classify panel, cluster panel, associate panel and visualize panel).

Explorer

1. The user interface

1.1 Section Tabs

At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.
The tabs are as follows:

1. **Preprocess**. Choose and modify the data being acted on.
2. **Classify**. Train and test learning schemes that classify or perform regression.
3. **Cluster.** Learn clusters for the data.
4. **Associate**. Learn association rules for the data.
5. **Select attributes**. Select the most relevant attributes in the data.
6. **Visualize**. View an interactive 2D plot of the data.

1.2 Status Box

The status box appears at the very bottom of the window. It displays messages that keep you informed about what's going on. For example, if the Explorer is busy loading a file, the status box will say that.

- **Memory information**. Display in the log box the amount of memory available to WEKA.
- **Run garbage collector**. Force the Java garbage collector to search for memory that is no longer needed and free it up, allowing more memory for new tasks. Note that the garbage collector is constantly running as a background task anyway.

1.3 Log Button

Clicking on this button brings up a separate window containing a scrollable text field. Each line of text is stamped with the time it was entered into the log. As you perform actions in WEKA, the log keeps a record of what has happened.

1.4 WEKA Status Icon

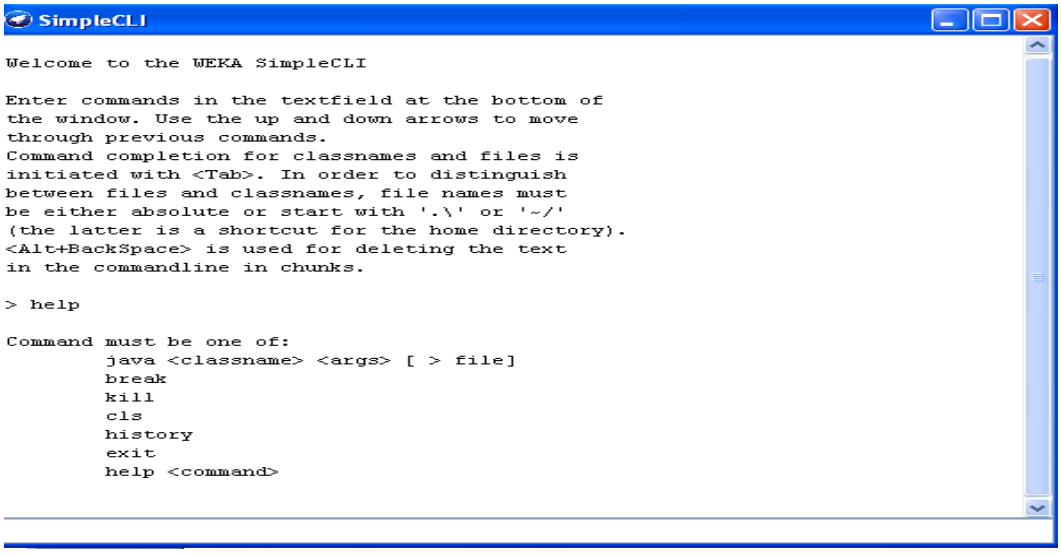
To the right of the status box is the WEKA status icon. When no processes are running, the bird sits down and takes a nap. The number beside the × symbol gives the number of concurrent processes running.

1.5 Graphical output

Most graphical displays in WEKA, e.g., the Graph Visualizer or the Tree Visualizer, support saving the output to a file. A dialog for saving the output can be brought up with Alt+Shift+left-click. Supported formats are currently Windows Bitmap, JPEG, PNG and EPS (encapsulated Postscript). The dialog also allows you to specify the dimensions of the generated image.

Simple CLI

The Simple CLI provides full access to all Weka classes, i.e., classifiers, filters, clusterers, etc., but without the hassle of the CLASSPATH (it facilitates the one, with which Weka was started). It offers a simple Weka shell with separated command line and output.



Commands

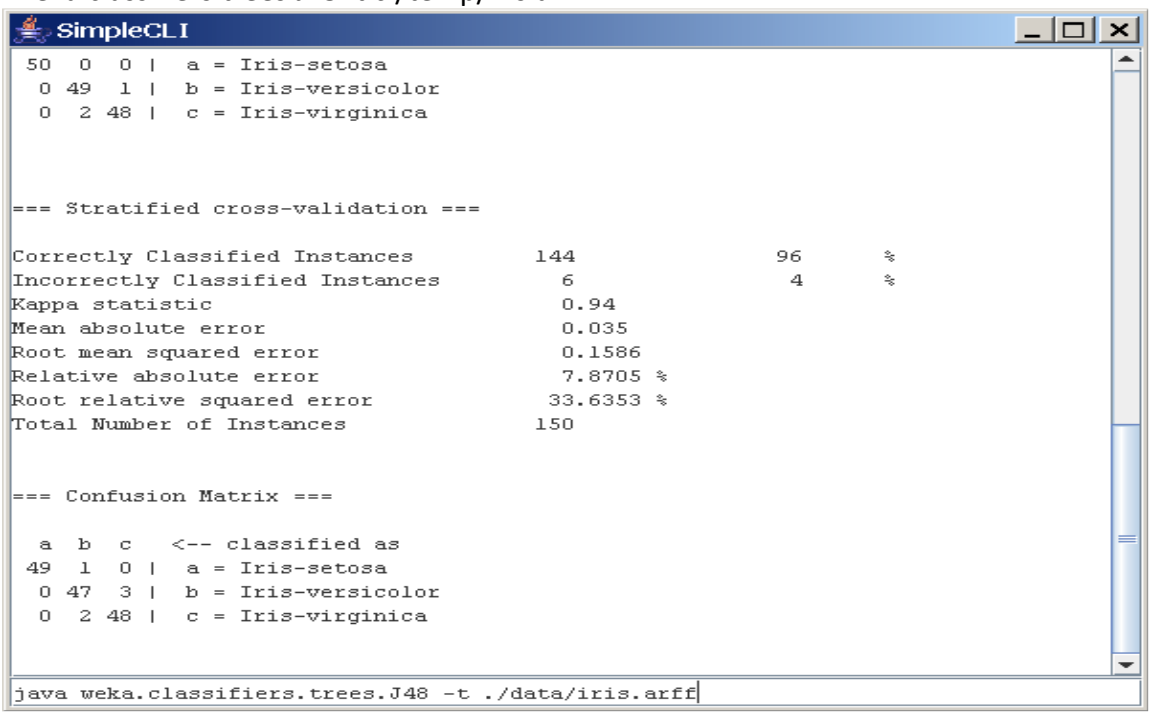
The following commands are available in the Simple CLI:

- **java** <classname> [<args>]
  invokes a java class with the given arguments (if any)

- **break** stops the current thread, e.g., a running classifier, in a friendly manner.
- **Kill** stops the current thread in an unfriendly fashion
- **cls** clears the output area
- **exit** exits the Simple CLI
- **help** [<command>] provides an overview of the available commands if without a command name as argument, otherwise more help on the specified command.

Invocation

In order to invoke a Weka class, one has only to prefix the class with ”java”. This command tells the Simple CLI to load a class and execute it with any given parameters. E.g., the J48 classifier can be invoked on the iris dataset with the following command:
java weka.classifiers.trees.J48 -t c:/temp/iris.arff



Command redirection

Starting with this version of Weka one can perform a basic redirection:
java weka.classifiers.trees.J48 -t test.arff > j48.txt
Note: the > must be preceded and followed by a space, otherwise it is not recognized as redirection, but part of another parameter.

WEEK 2:

iv) Study the arff file format
v) Explore the available data sets in WEKA.
vi) Load a data set (ex. Weather dataset, Iris dataset, etc.)
vii) Load each dataset and observe the following:

a) List the attribute names and they types
b) Number of records in each dataset
c) Identify the class attribute (if any)
d) Plot histogram
e) Determine the number of records for each class.
f) Visualize the data in various dimensions

iv) Study the arff file format

Attribute-Relation File Format (ARFF)

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software. This document descibes the version of ARFF used with Weka versions 3.2 to 3.3; this is an extension of the ARFF format as described in the data mining book written by Ian H. Witten and Eibe Frank (the new additions are string attributes, date attributes, and sparse instances).

This explanation was cobbled together by Gordon Paynter (gordon.paynter at ucr.edu) from the Weka 2.1 ARFF description, email from Len Trigg (lenbok at myrealbox.com) and Eibe Frank (eibe at cs.waikato.ac.nz), and some datasets. It has been edited by Richard Kirkby (rkirkby at cs.waikato.ac.nz). Contact Len if you're interested in seeing the ARFF 3 proposal.

Overview

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
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,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.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Phases.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Tables.'
```

Dates must be specified in the data section using the string representation specified in the attribute declaration. For example:

```
@RELATION Timestamps

@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"

@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
```

Sparse ARFF files

Sparse ARFF files are very similar to ARFF files, but data with value 0 are not be explicitly represented.

Sparse ARFF files have the same header (i.e **@relation** and **@attribute** tags) but the data section is different. Instead of representing each value in order, like this:

```
@data
0, X, 0, Y, "class A"
```

0, 0, W, 0, "class B"

the non-zero attributes are explicitly identified by attribute number and their value stated, like this:

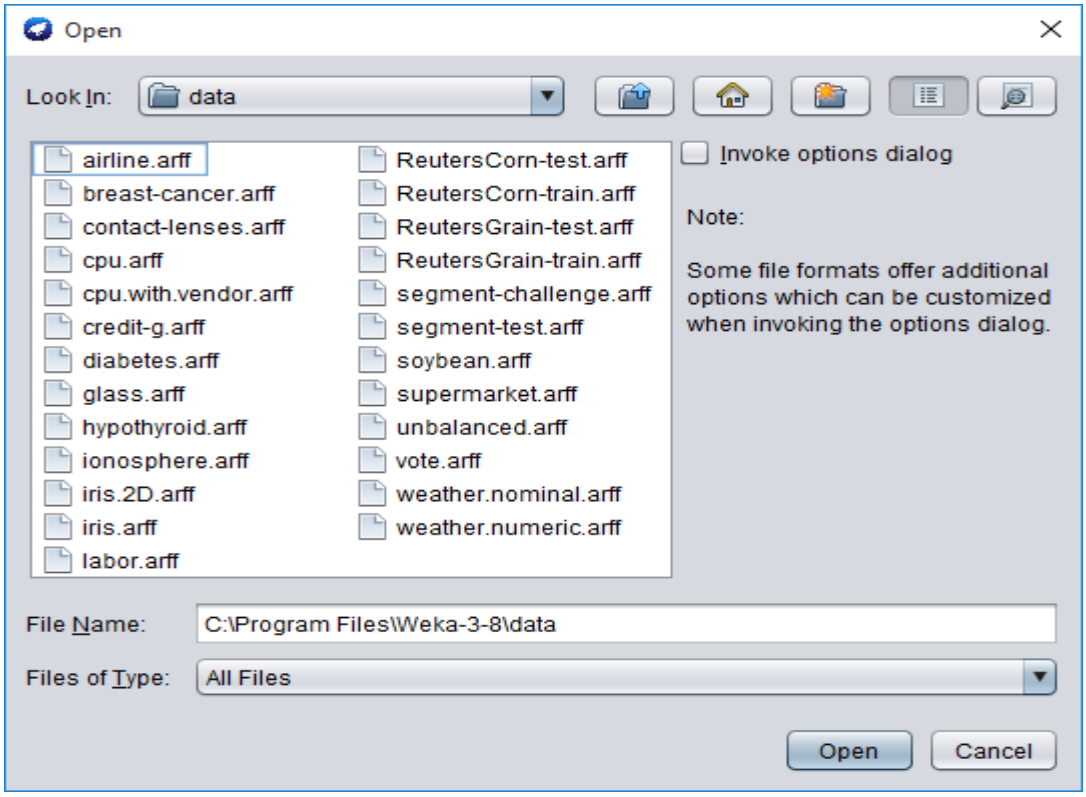
@data
{1 X, 3 Y, 4 "class A"}
{2 W, 4 "class B"}

Each instance is surrounded by curly braces, and the format for each entry is: <index> <space> <value> where index is the attribute index (starting from 0).

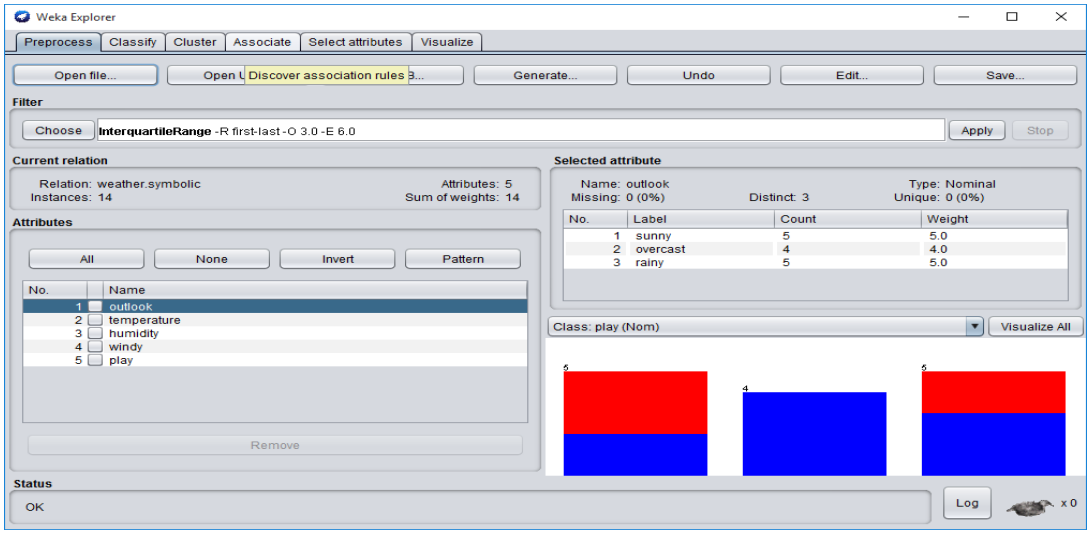
Note that the omitted values in a sparse instance are **0**, they are not "missing" values! If a value is unknown, you must explicitly represent it with a question mark (?).

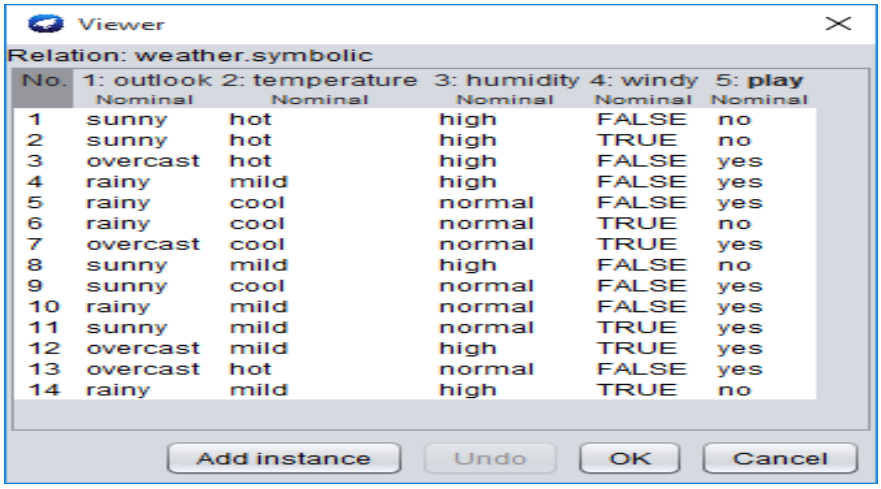
Warning: There is a known problem saving SparseInstance objects from datasets that have string attributes. In Weka, string and nominal data values are stored as numbers; these numbers act as indexes into an array of possible attribute values (this is very efficient). However, the first string value is assigned index 0: this means that, internally, this value is stored as a 0. When a SparseInstance is written, string instances with internal value 0 are not output, so their string value is lost (and when the arff file is read again, the default value 0 is the index of a different string value, so the attribute value appears to change). To get around this problem, add a dummy string value at index 0 that is never used whenever you declare string attributes that are likely to be used in SparseInstance objects and saved as Sparse ARFF files.

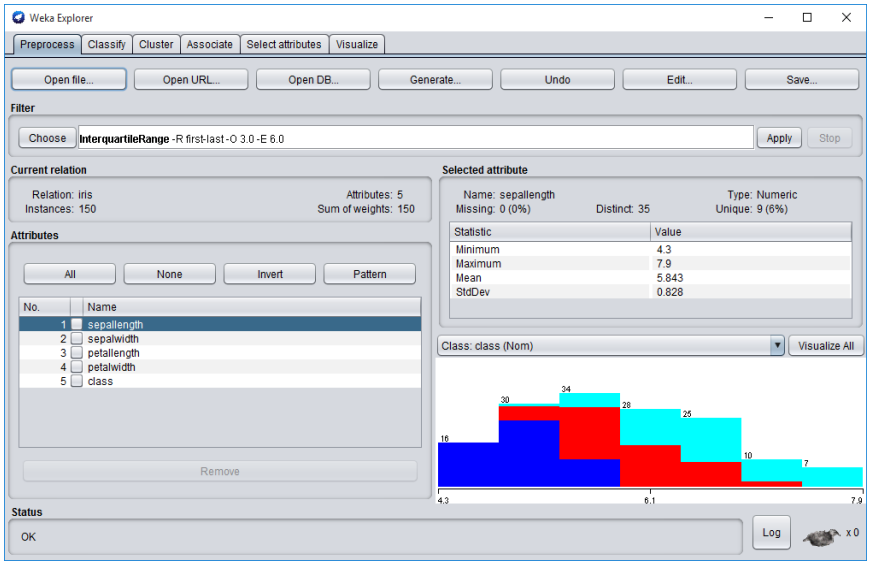
v) Explore the available data sets in WEKA.

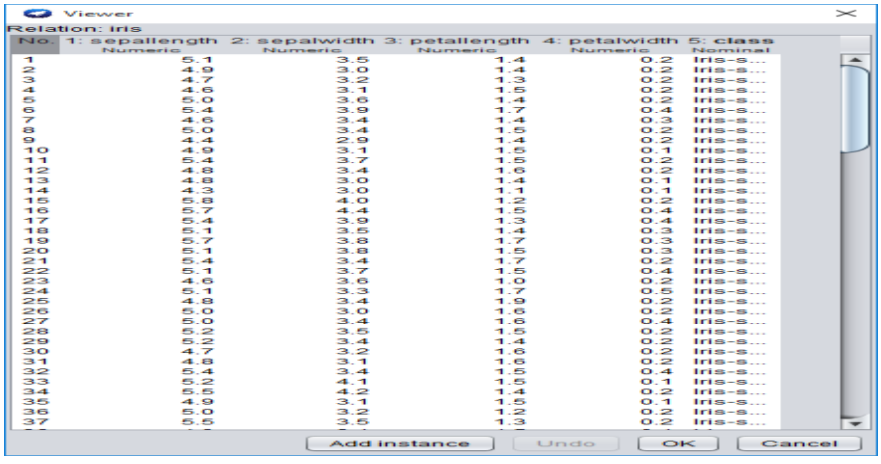


vi) Load a data set (ex. Weather dataset, Iris dataset, etc.)




Viewer
Relation: weather.symbolic
No. 1: outlook Nominal 2: temperature Nominal 3: humidity Nominal 4: windy Nominal 5: play Nominal
1 sunny hot high FALSE no
2 sunny hot high TRUE no
3 overcast hot high FALSE yes
4 rainy mild high FALSE yes
5 rainy cool normal FALSE yes
6 rainy cool normal TRUE no
7 overcast cool normal TRUE yes
8 sunny mild high FALSE no
9 sunny cool normal FALSE yes
10 rainy mild normal FALSE yes
11 sunny mild normal TRUE yes
12 overcast mild high TRUE yes
13 overcast hot normal FALSE yes
14 rainy mild high TRUE no
Add instance
Undo
OK
Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
InterquartileRange -R first-last -O 3.0 -E 6.0
Apply
Stop
Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150
Selected attribute
Name: sepallength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Class: class (Nom)
Visualize All
Status
OK
Log
x 0


Viewer
Relation: iris
No. 1: sepallength Numeric 2: sepalwidth Numeric 3: petallength Numeric 4: petalwidth Numeric 5: class Nominal
1 5.1 3.5 1.4 0.2 Iris-s...
2 4.9 3.0 1.4 0.2 Iris-s...
3 4.7 3.2 1.3 0.2 Iris-s...
4 4.6 3.1 1.5 0.2 Iris-s...
5 5.0 3.6 1.4 0.2 Iris-s...
6 5.4 3.9 1.7 0.4 Iris-s...
7 4.6 3.4 1.4 0.3 Iris-s...
8 5.0 3.4 1.5 0.2 Iris-s...
9 4.4 2.9 1.4 0.2 Iris-s...
10 4.9 3.1 1.5 0.1 Iris-s...
11 5.4 3.7 1.5 0.2 Iris-s...
12 4.8 3.4 1.6 0.2 Iris-s...
13 4.8 3.0 1.4 0.1 Iris-s...
14 4.3 3.0 1.1 0.1 Iris-s...
15 5.8 4.0 1.2 0.2 Iris-s...
16 5.7 4.4 1.5 0.4 Iris-s...
17 5.4 3.9 1.3 0.4 Iris-s...
18 5.1 3.5 1.4 0.3 Iris-s...
19 5.7 3.8 1.7 0.3 Iris-s...
20 5.1 3.8 1.5 0.3 Iris-s...
21 5.4 3.4 1.7 0.2 Iris-s...
22 5.1 3.7 1.5 0.4 Iris-s...
23 4.6 3.6 1.0 0.2 Iris-s...
24 5.1 3.3 1.7 0.5 Iris-s...
25 4.8 3.4 1.9 0.2 Iris-s...
26 5.0 3.0 1.6 0.2 Iris-s...
27 5.0 3.4 1.6 0.4 Iris-s...
28 5.2 3.5 1.5 0.2 Iris-s...
29 5.2 3.4 1.4 0.2 Iris-s...
30 4.7 3.2 1.6 0.2 Iris-s...
31 4.8 3.1 1.6 0.2 Iris-s...
32 5.4 3.4 1.5 0.4 Iris-s...
33 5.2 4.1 1.5 0.1 Iris-s...
34 5.5 4.2 1.4 0.2 Iris-s...
35 4.9 3.1 1.5 0.1 Iris-s...
36 5.0 3.2 1.2 0.2 Iris-s...
Add instance
Undo
OK
Cancel

vii) Load each dataset and observe the following:

a) List the attribute names and they types

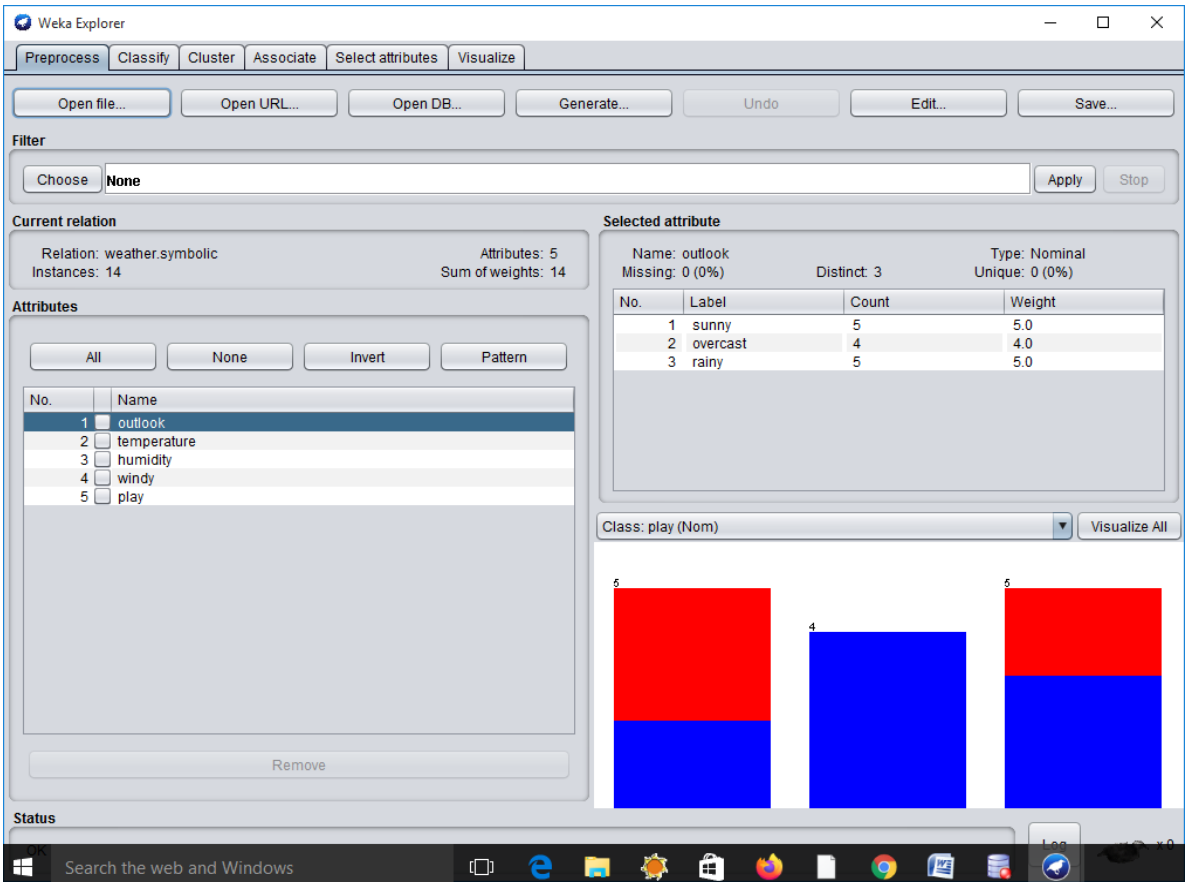
b) Number of records in each dataset

c) Identify the class attribute (if any)

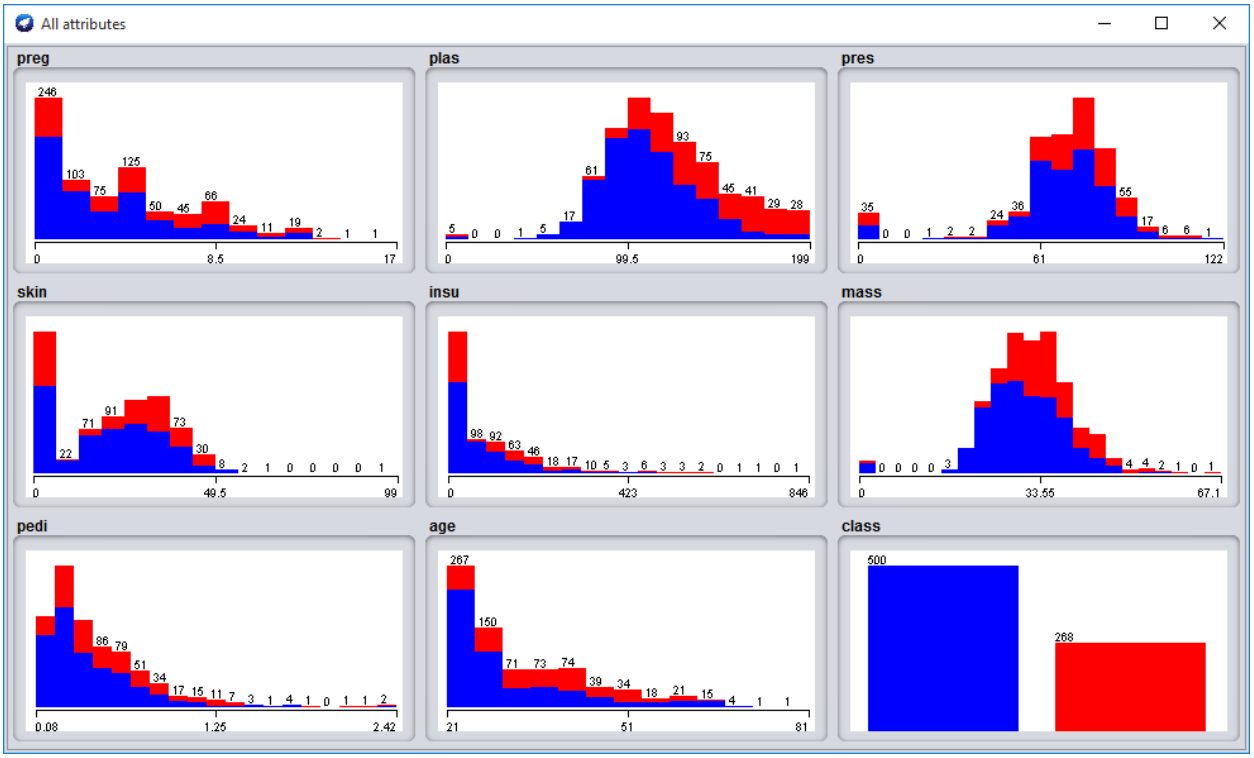
d) Plot histogram

e) Determine the number of records for each class.

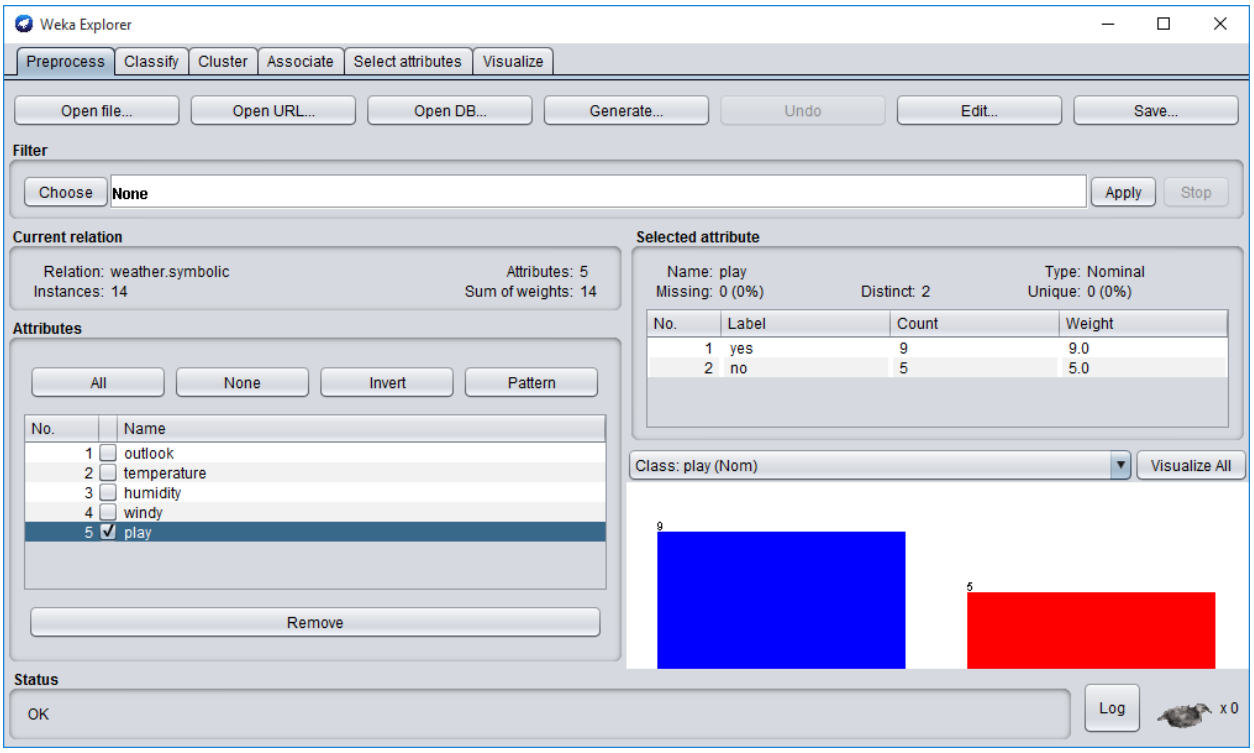
f) Visualize the data in various dimensions



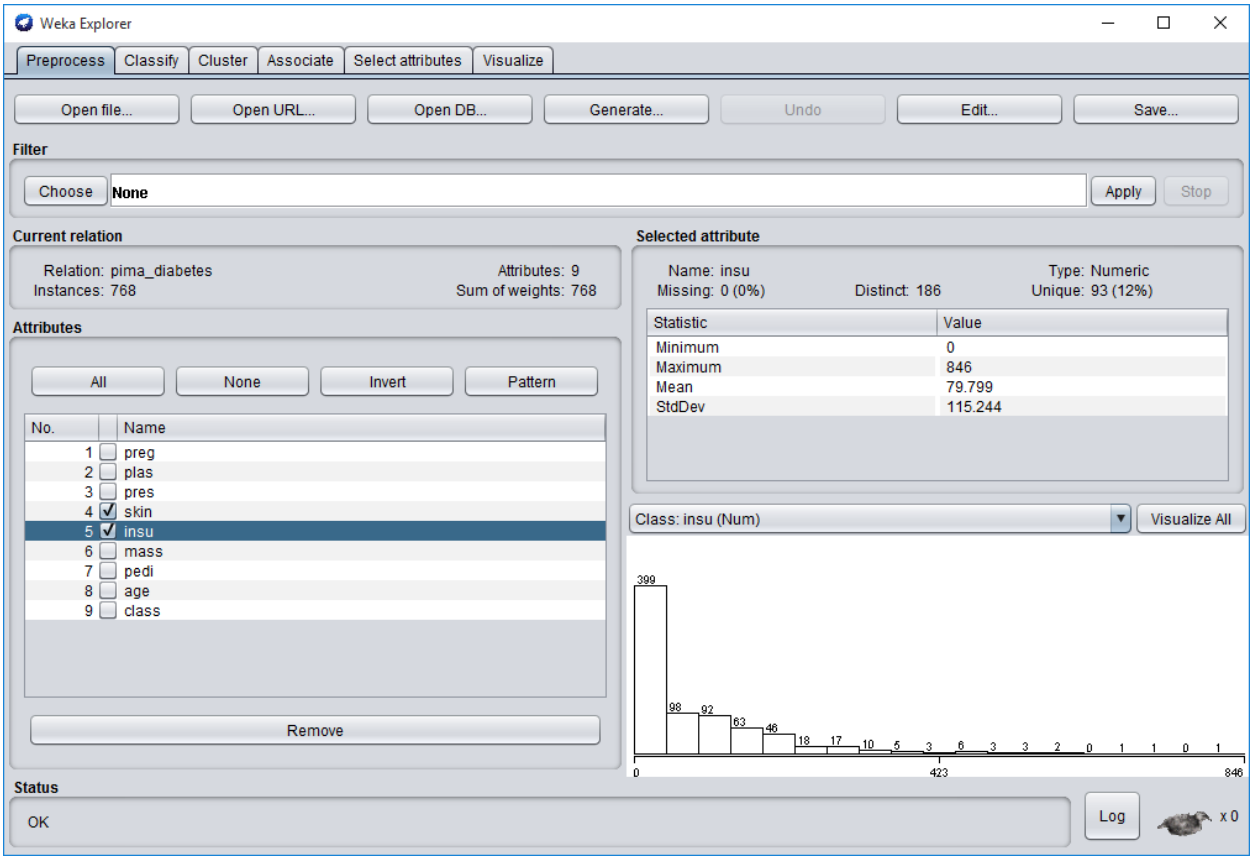
d) Plot histogram:



e) Determine the number of records for each class.



f) Visualize the data in various dimensions



VIVA-QUESTIONS

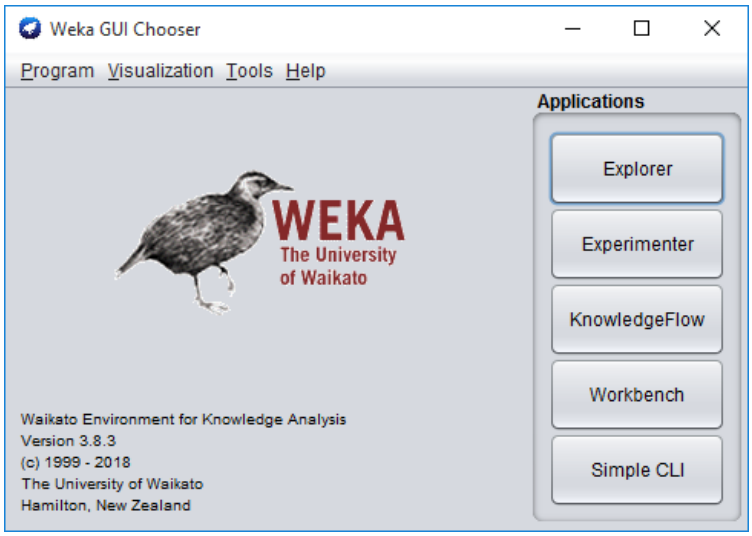
1. WEKA Stands for
2. ARFF Stands for
3. CSV Stands for
4. Present Version of WEKA Software?
5. Which of the following is not a DATA Mining Software ?
   A. WEKA B. SPSS C. Orange D. Cogno's
6. Which of the following is not a Data Type in Weka ?
   A. Numeric B. String C. Date D. Real
7. Which of the following is not a support files in Weka ?
   A. ARFF B. CSV C..cpp
8. The full form of KDD is
9. Data Mining helps in
10. Extreme values that occur infrequently are called as

EXP2: Perform data preprocessing tasks on

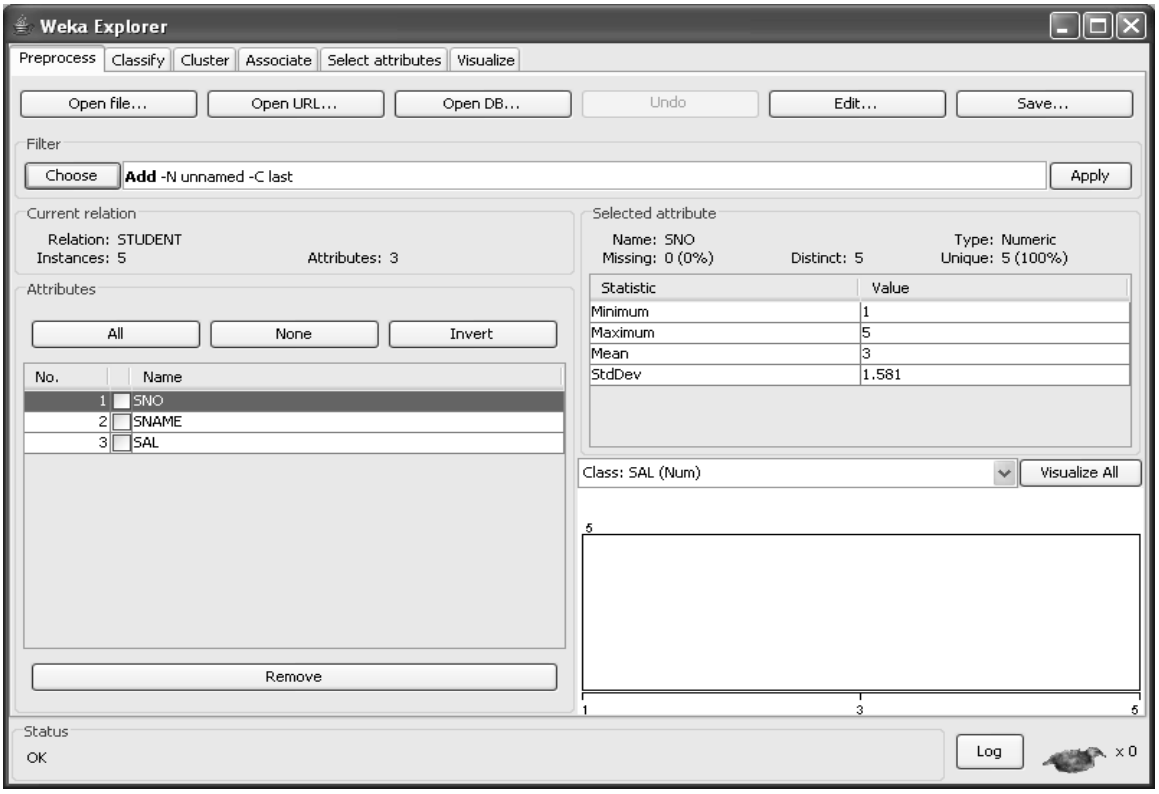
i.Add attribute ii. Add expression iii. Copy attribute iv. Remove attribute

Filters

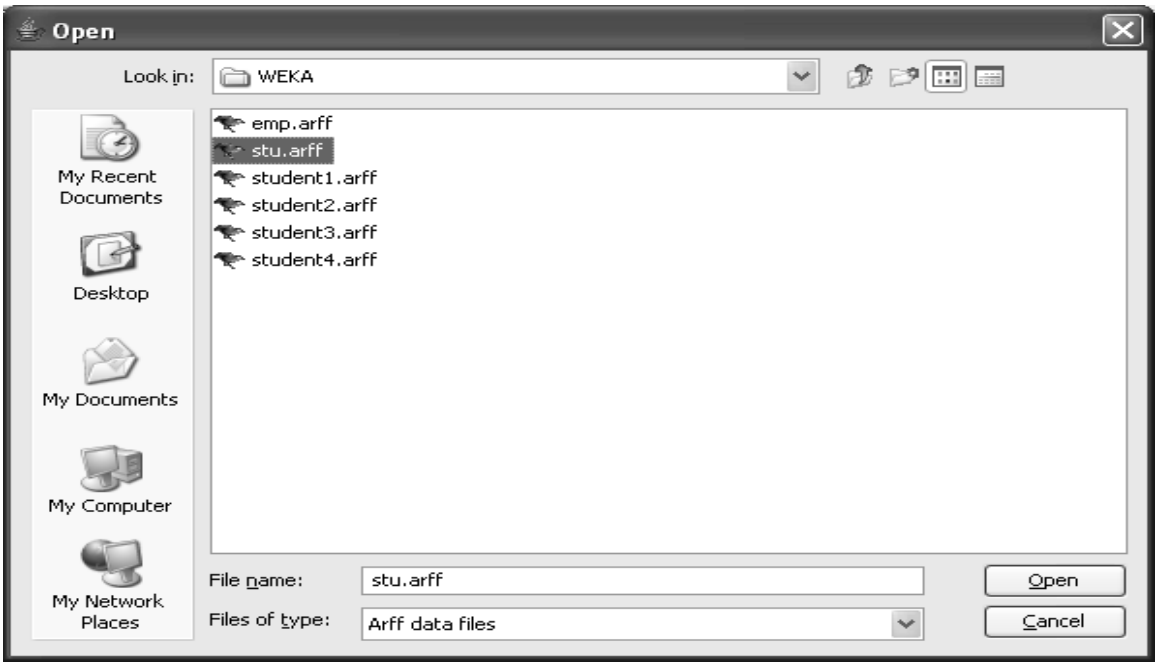
Step-1: - Go to start button then select All Programs and then select weka 3.8.3



Click on Explorer

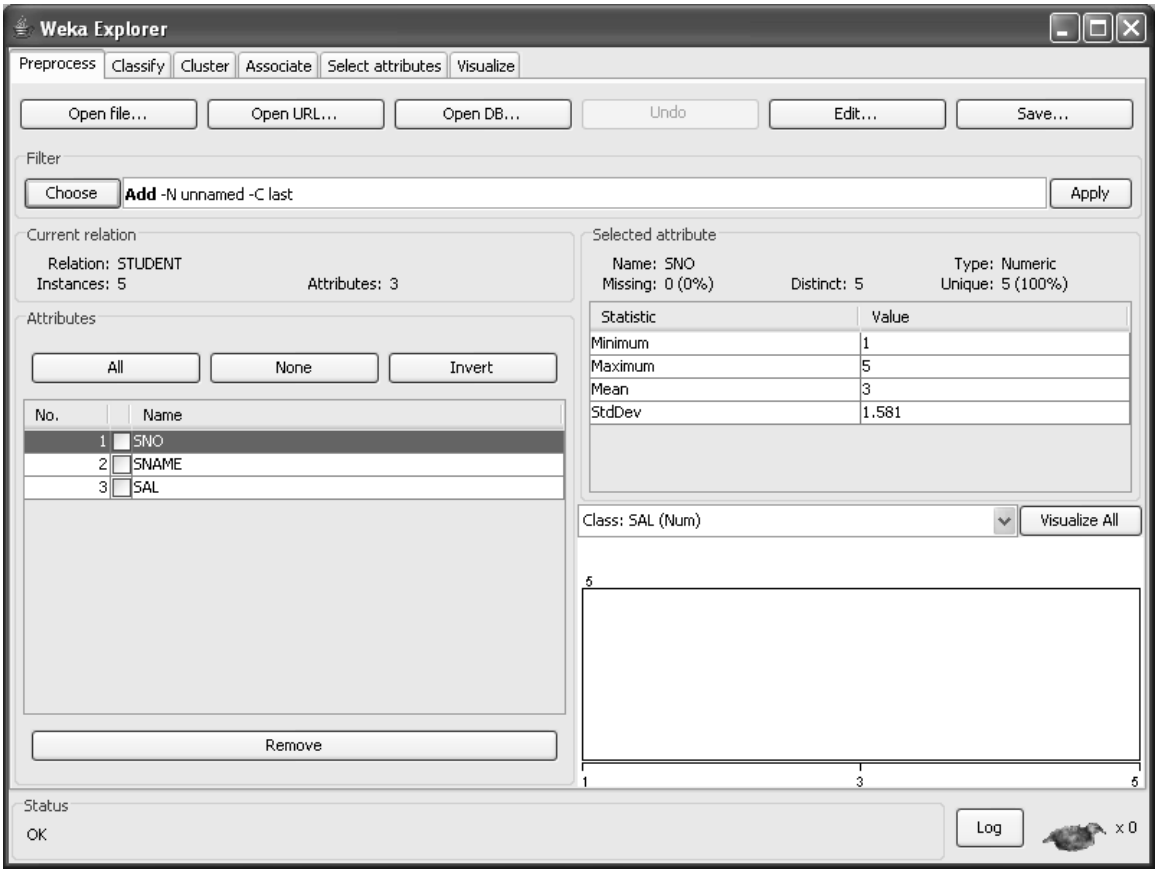


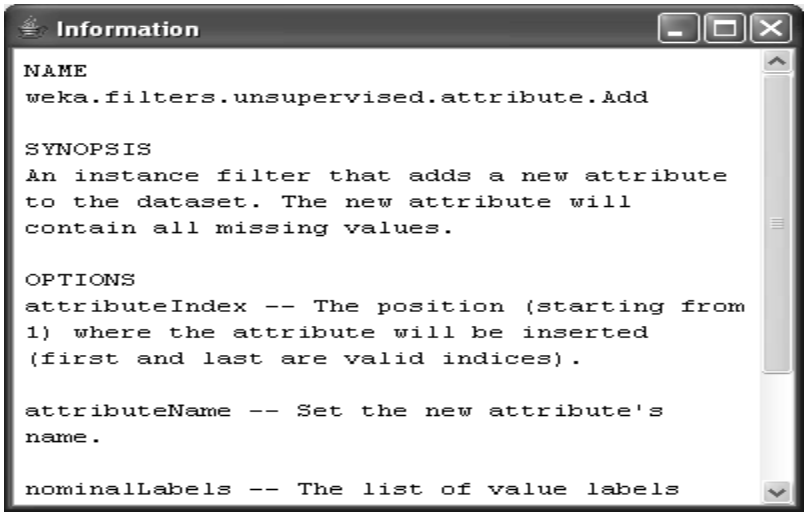
To open the required file



ADD ATTRIBUTE

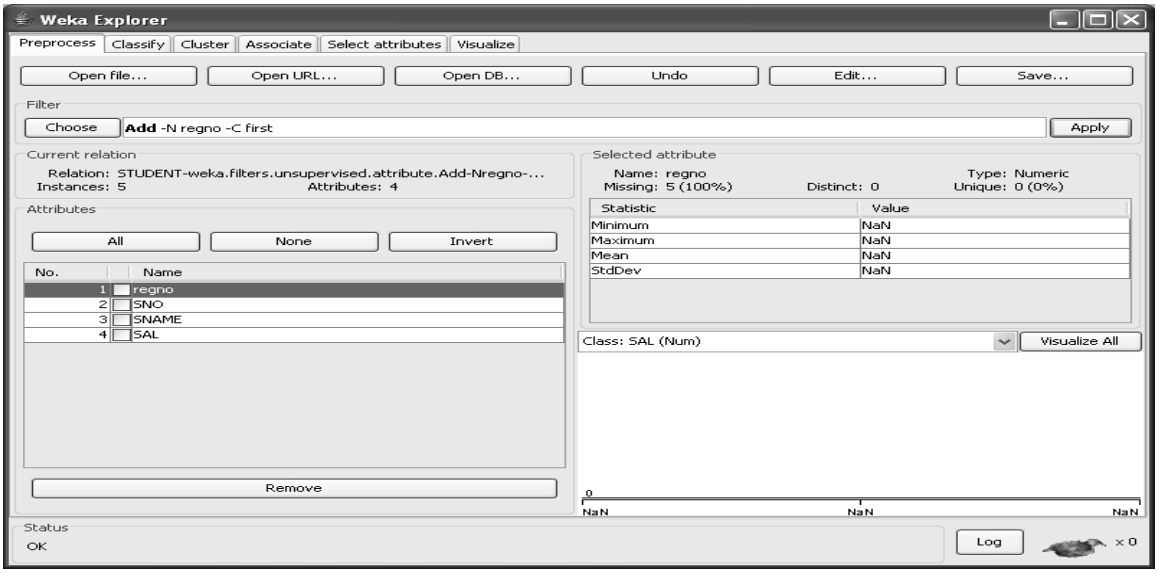
Step-2: - Go to preprocess menu and choose then select attribute of Add option.



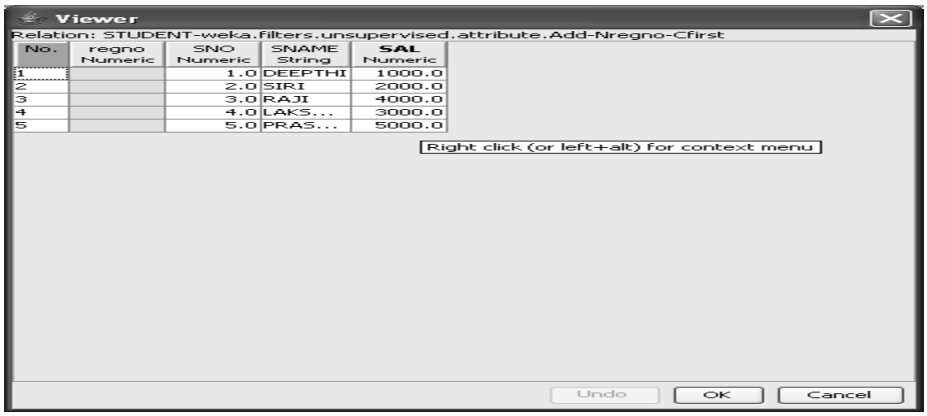


Step-3: -To enter the required fields is added to table.

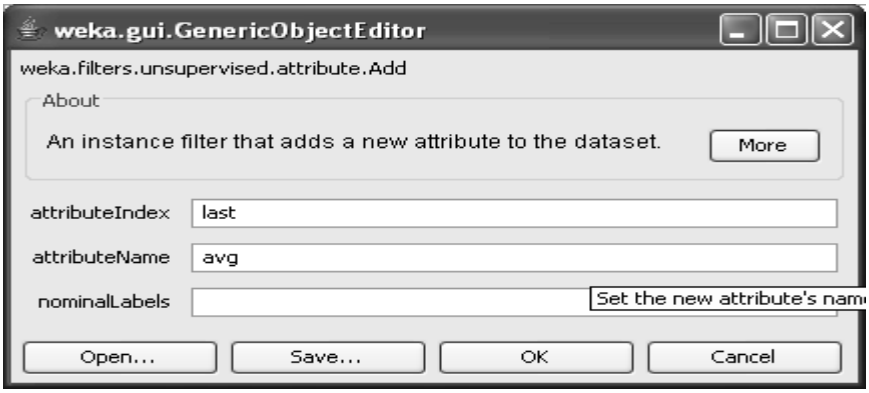
Click on ok button



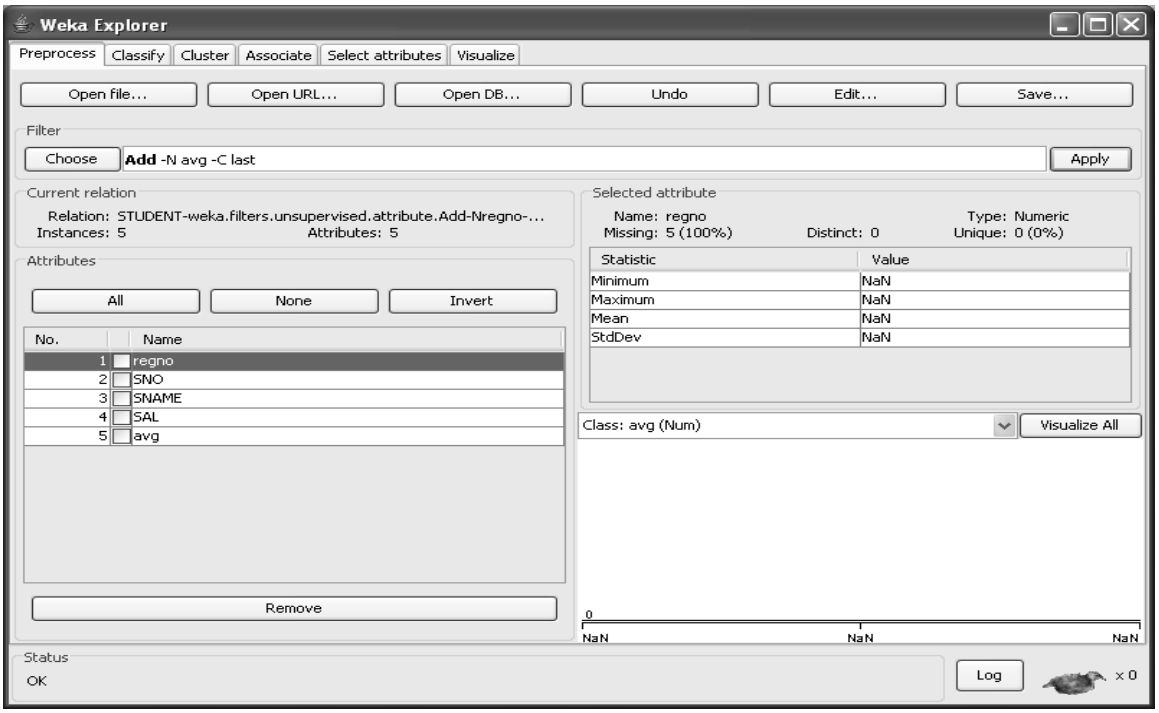
Click on Apply button.



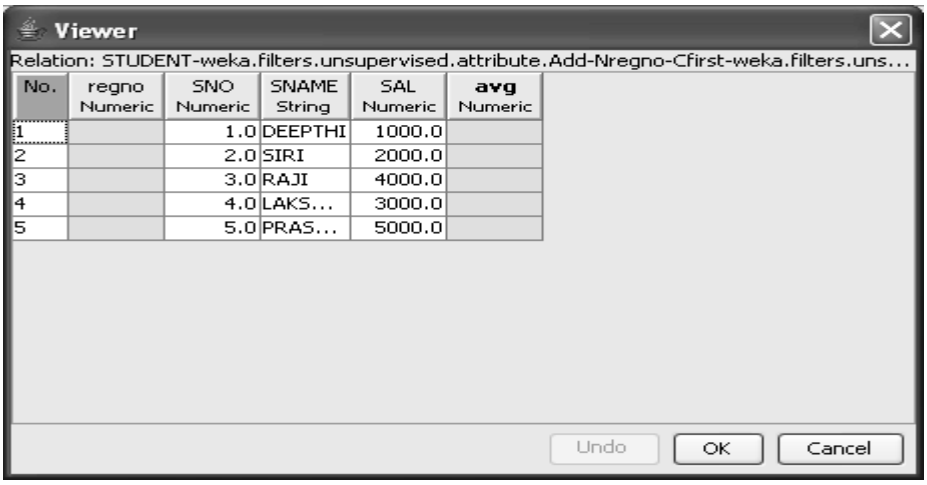
Step-4: - if the index position is last to enter the fields



Click on Ok button.

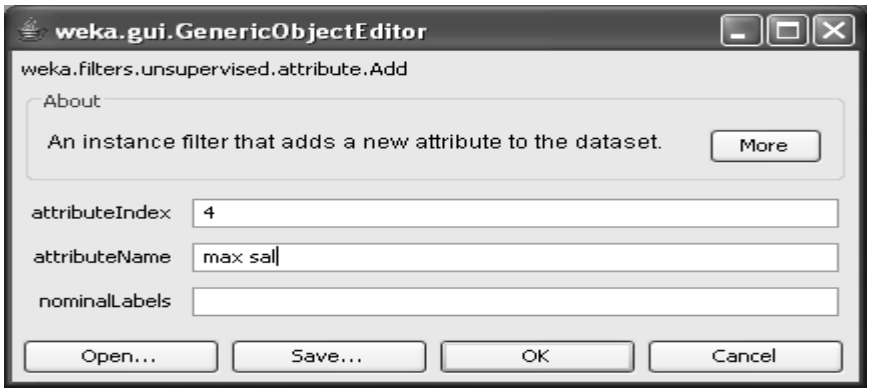


Click on Apply button.

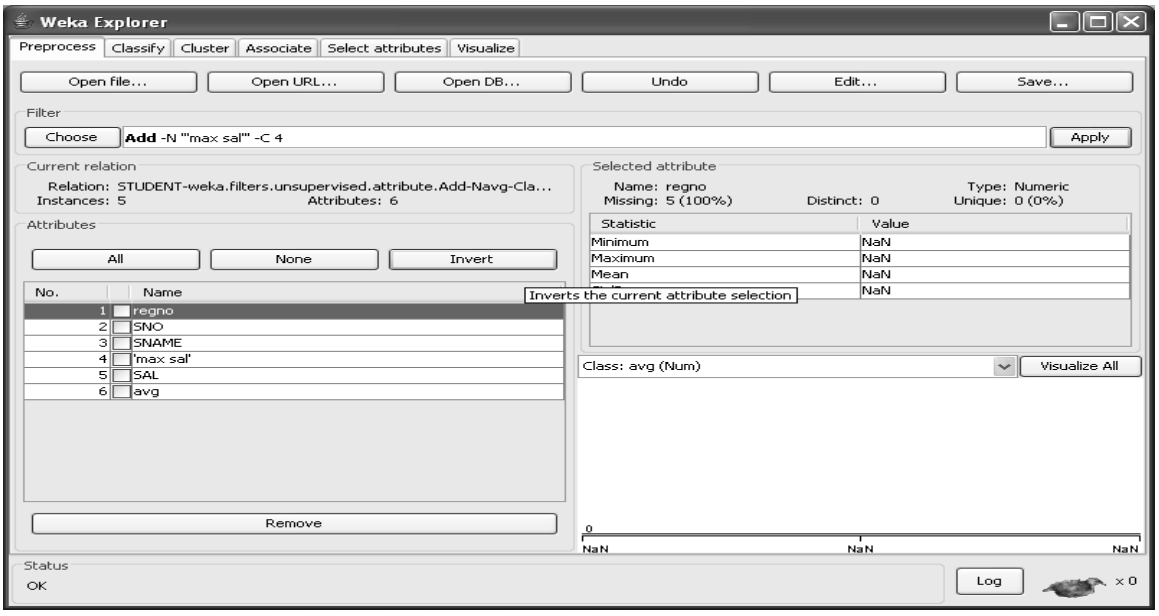


Click on ok button.

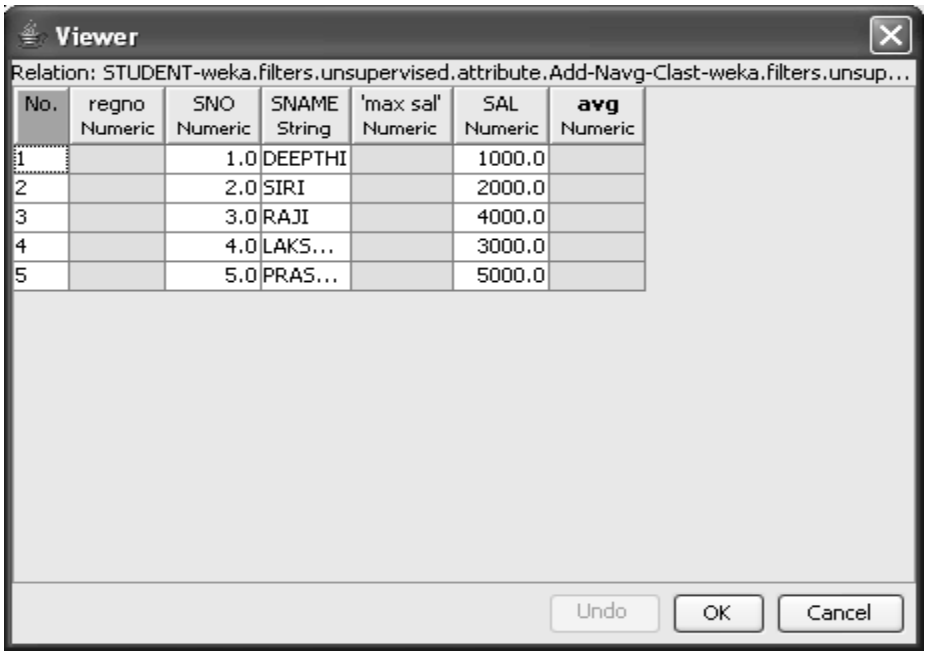
Step-5: -if the index position is middle



Click on ok button.



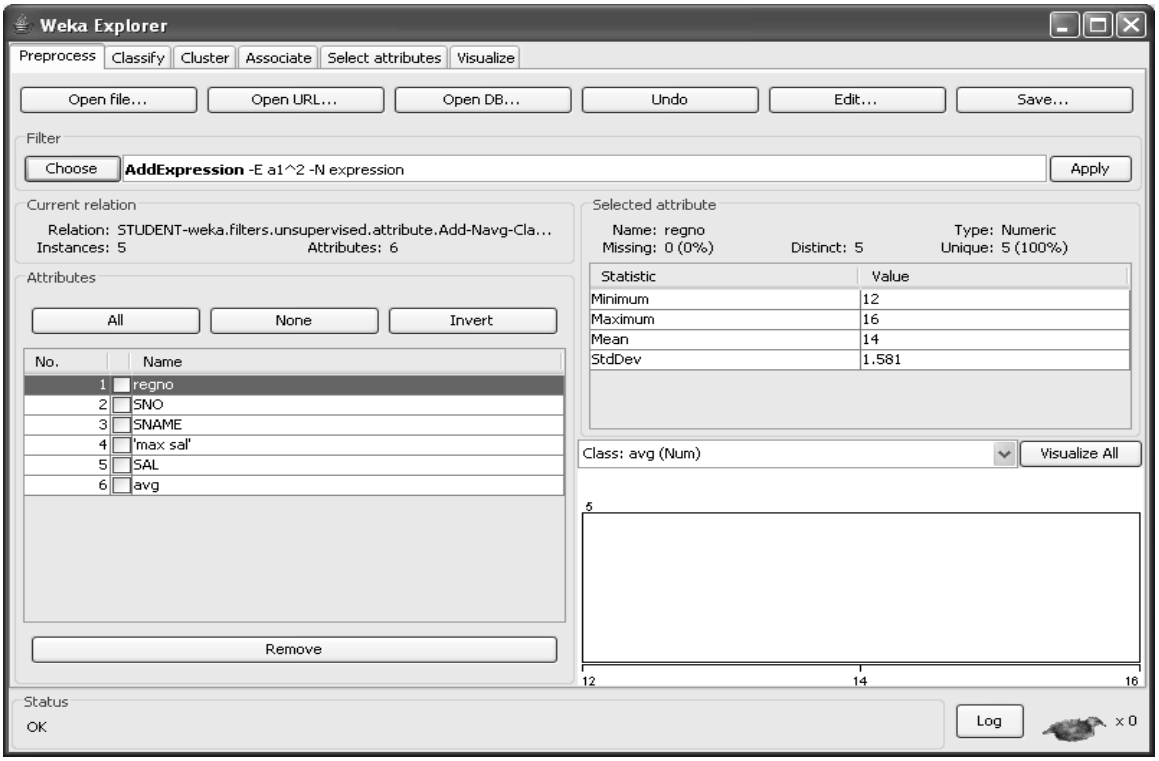
Click on Apply button.

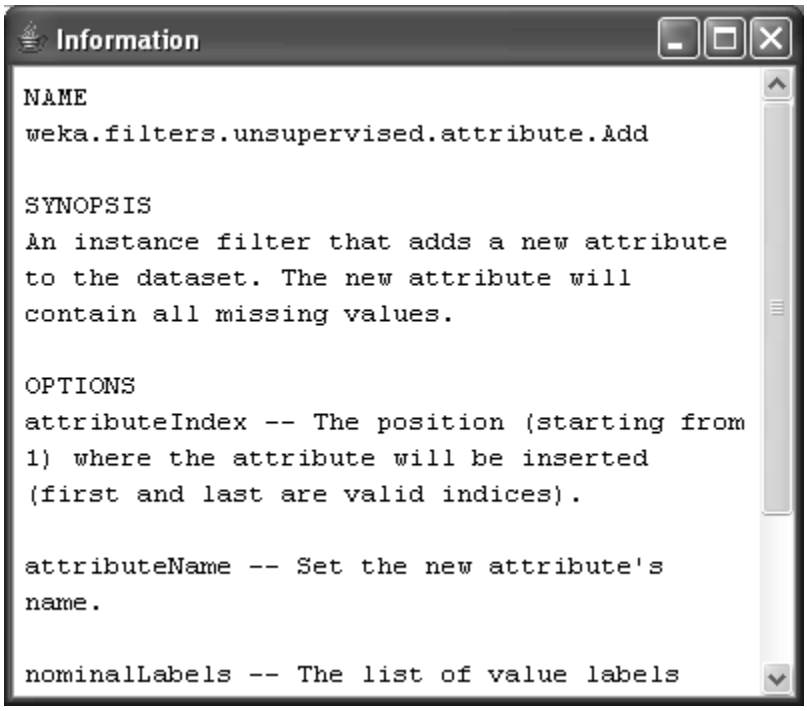


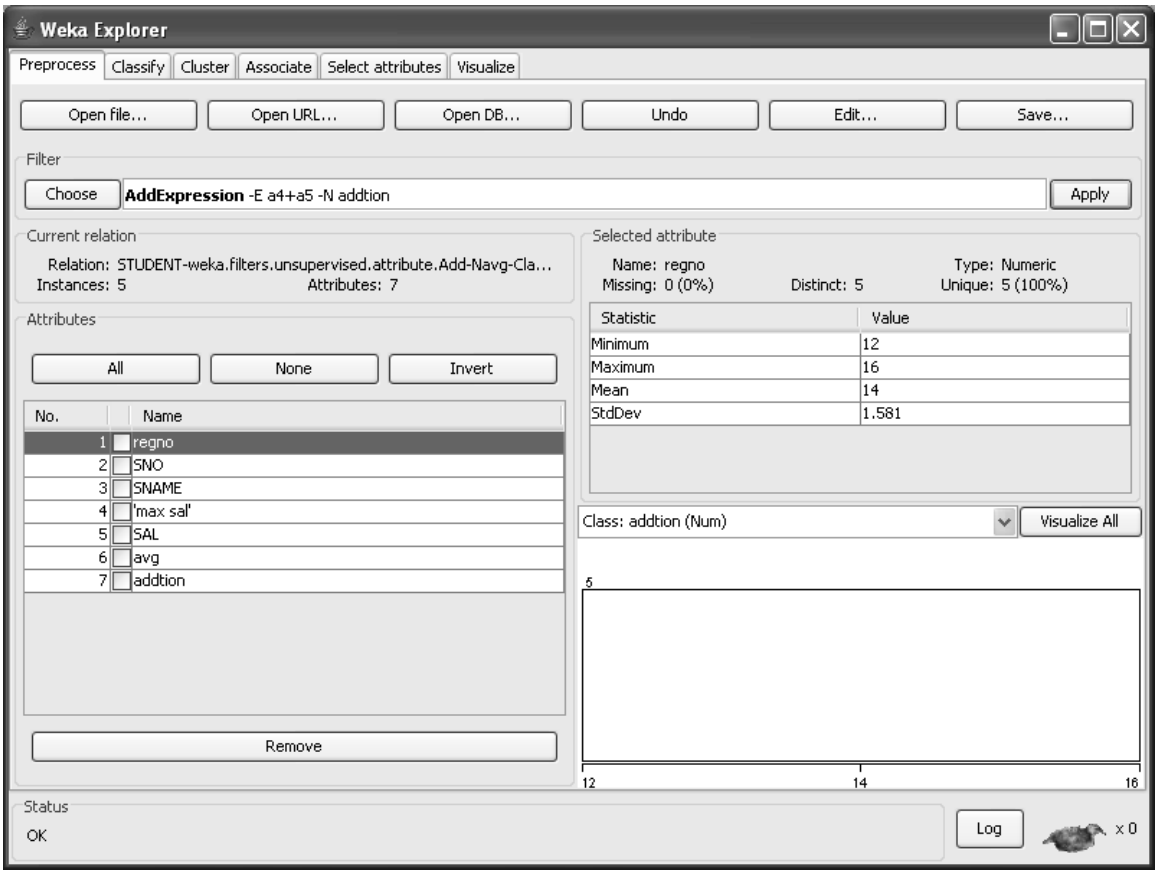
Click on ok button.

Add Expression

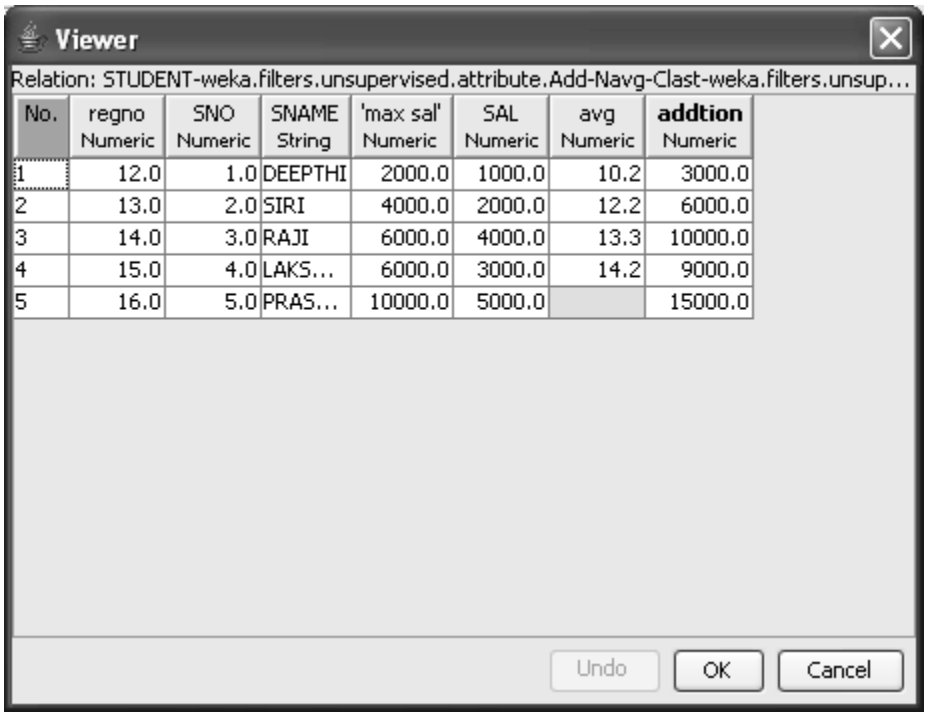
Step-6: - click on choose button and then select AddExpression option.



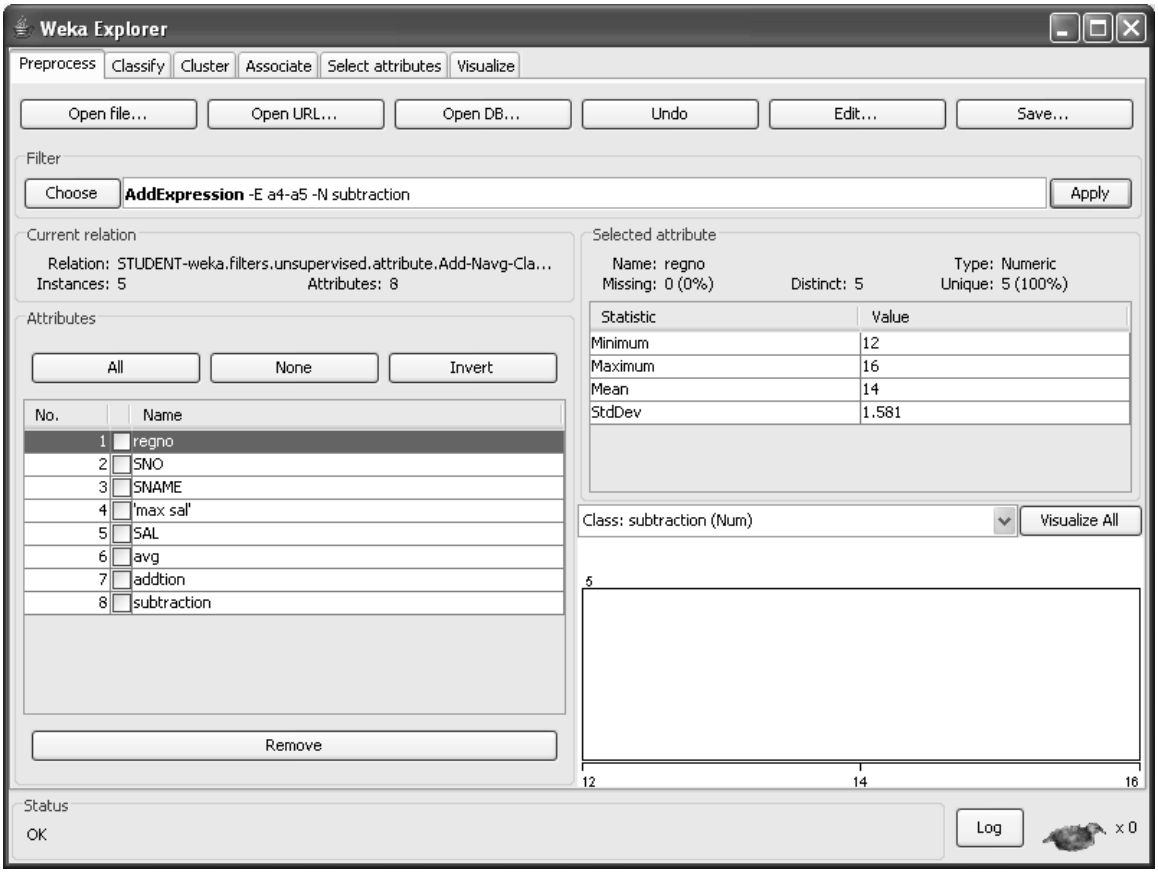




Click on Apply option.

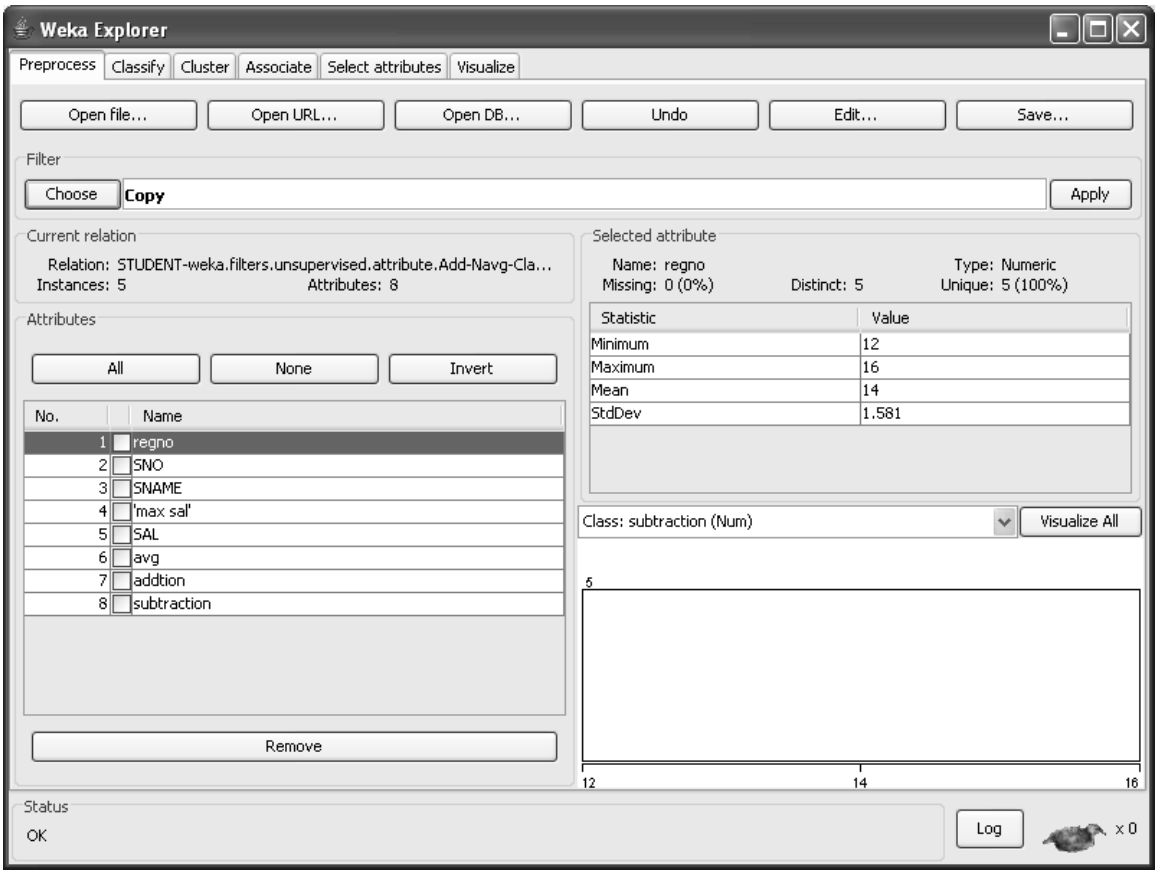


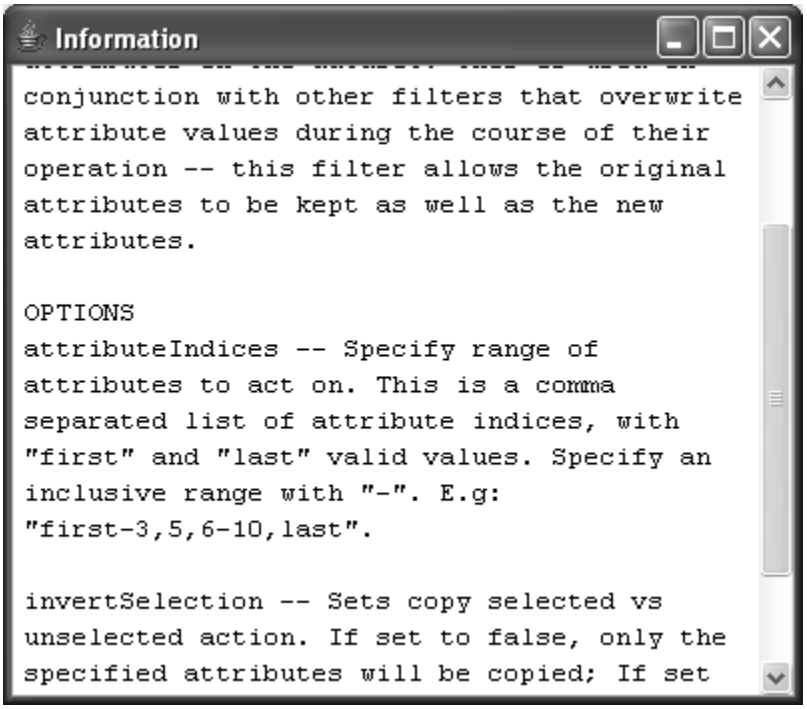
Click on ok button.



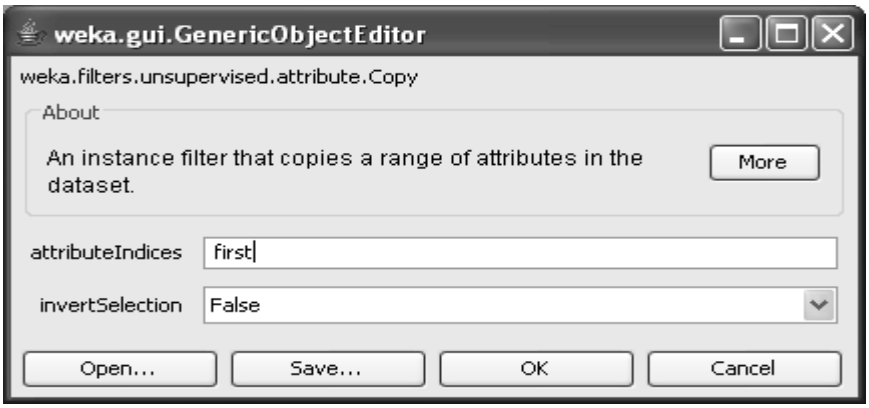
COPY

Step-7: - Click on choose button and then select copy option.

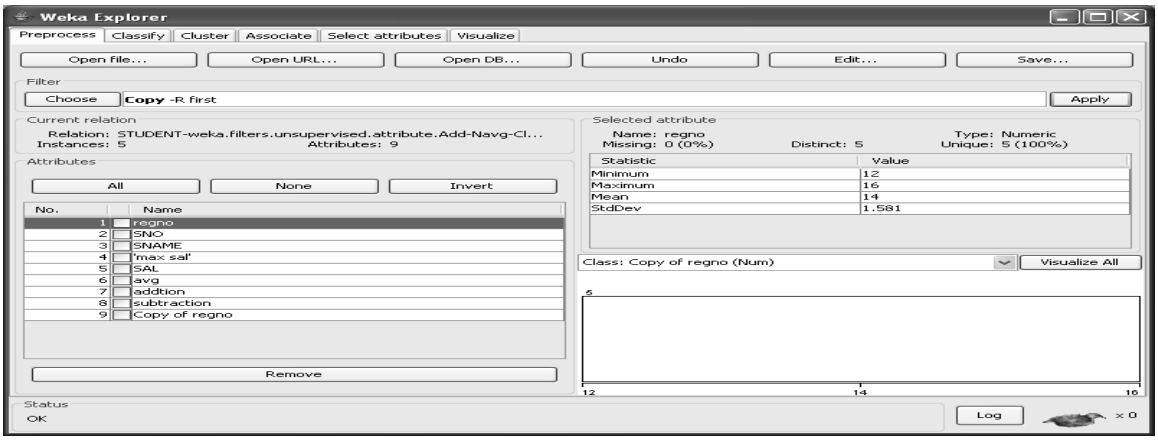




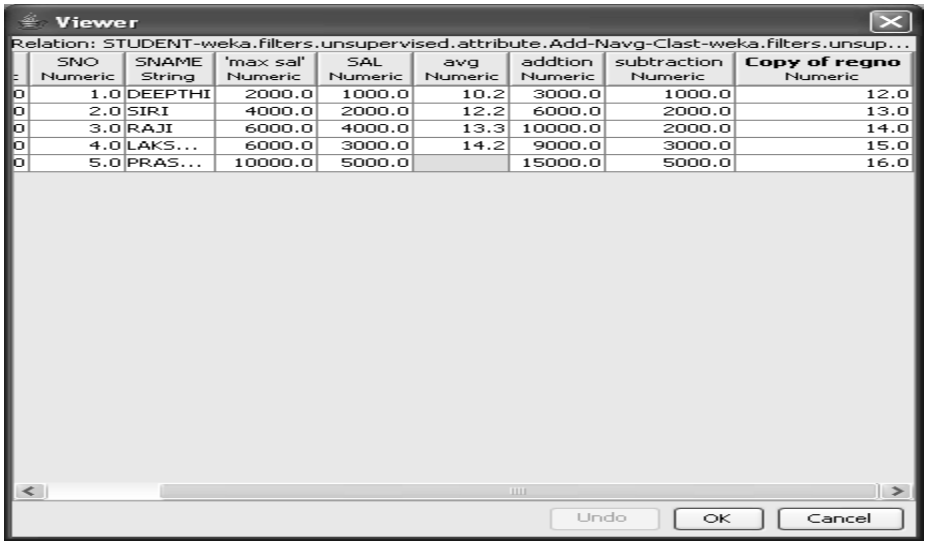
If the index place is any



Click on ok button.

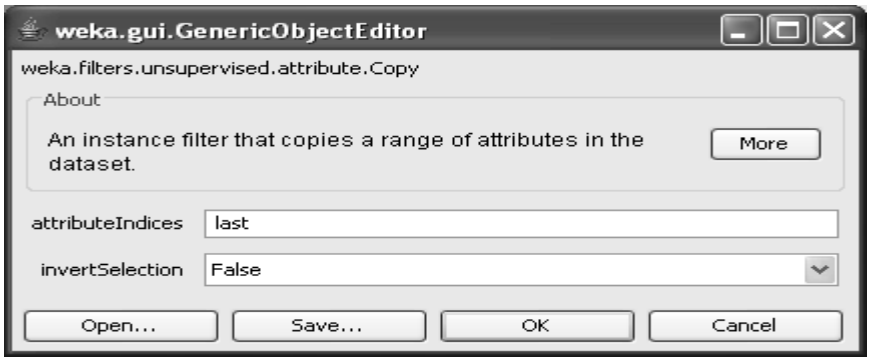


Click On ok button.

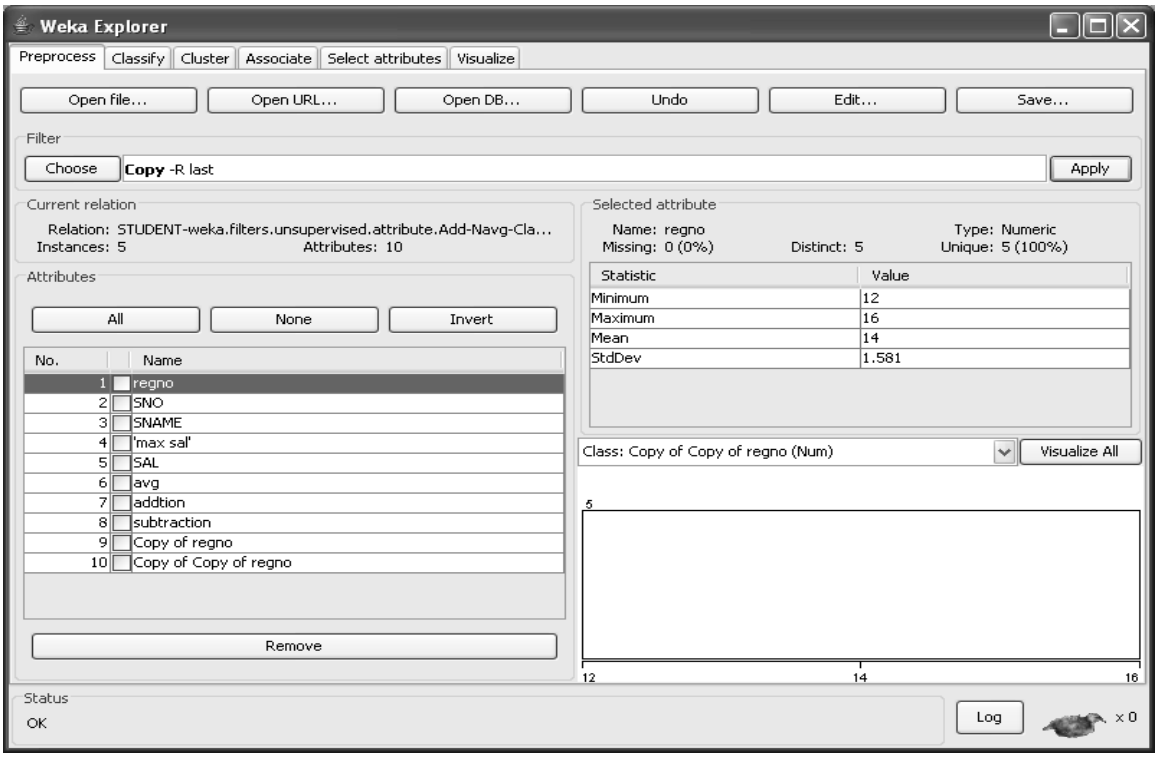


Click on ok button.

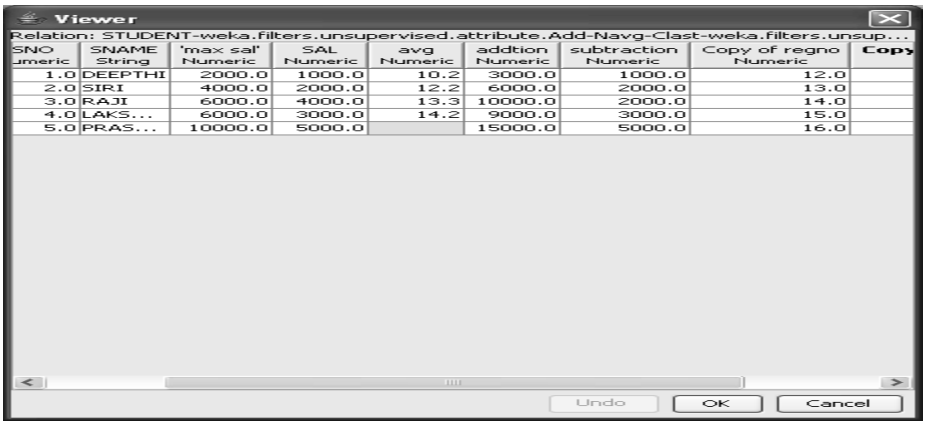
If the last position.



Click on ok button.

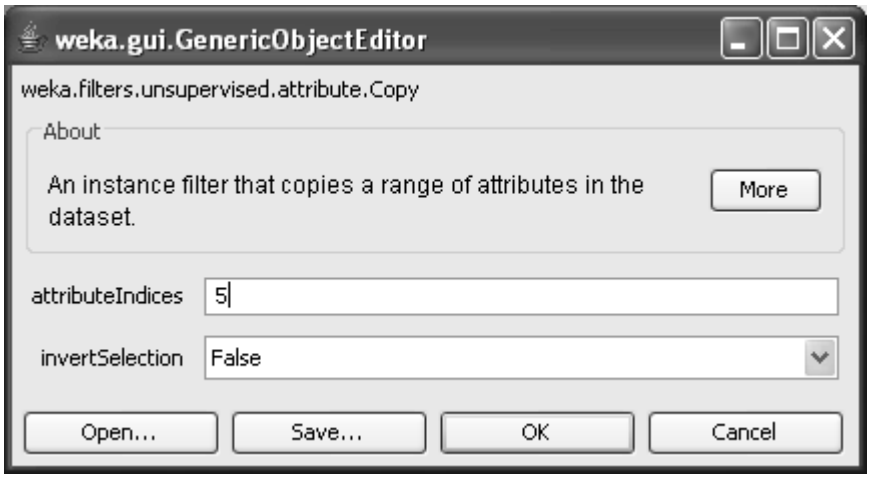


Click on Apply option.

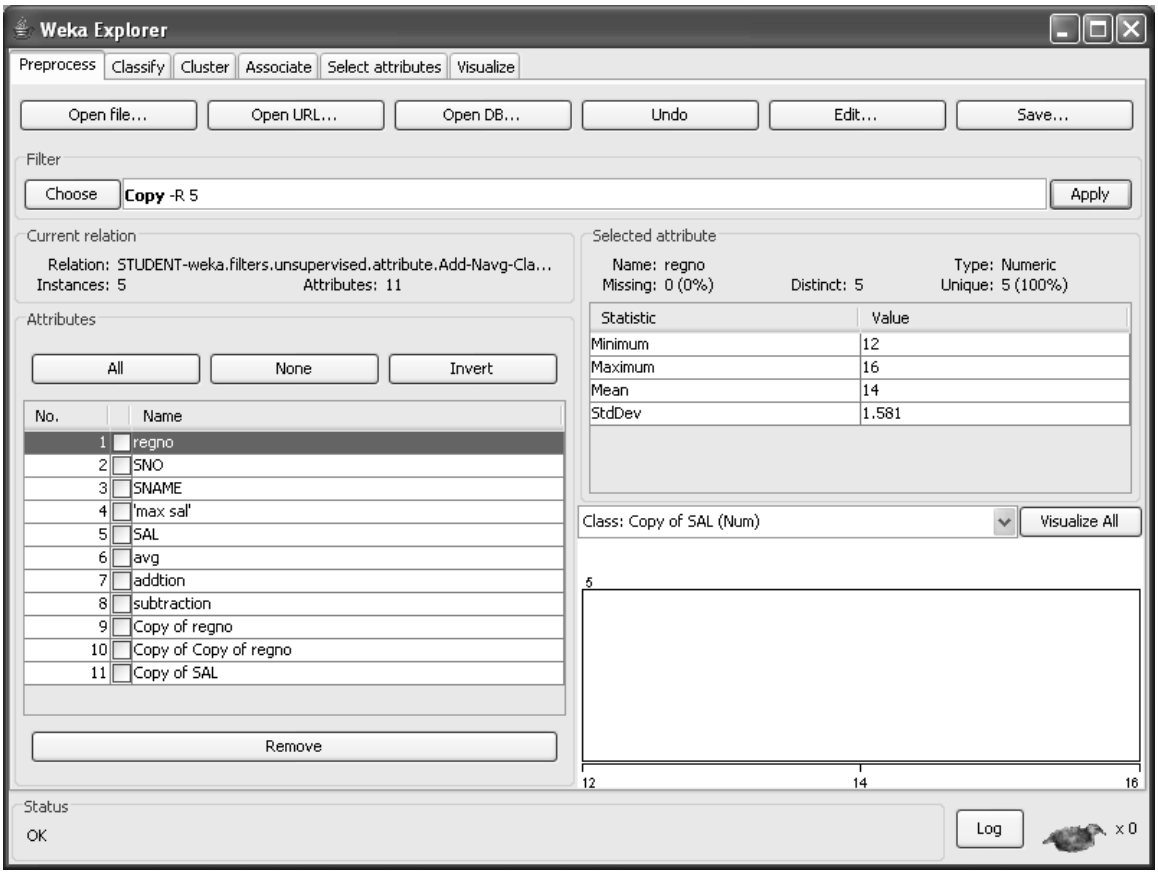


Click on ok button.

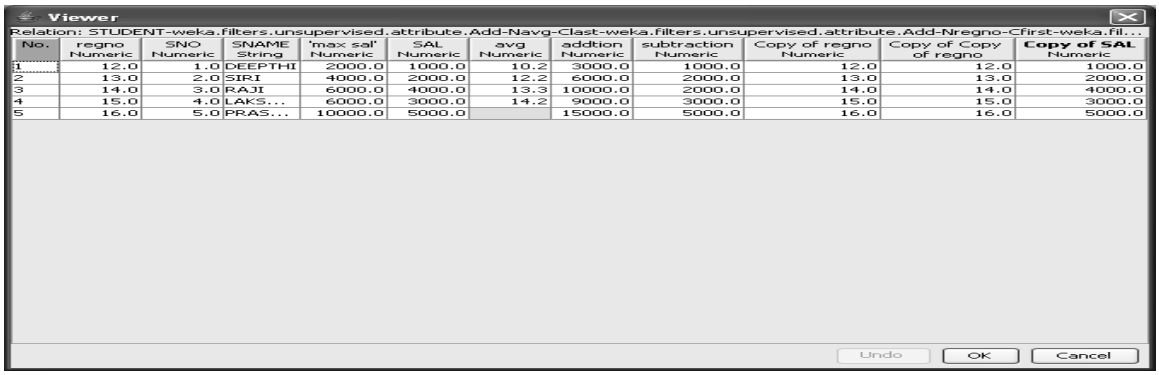
If the index position is specified.



Click on ok button.

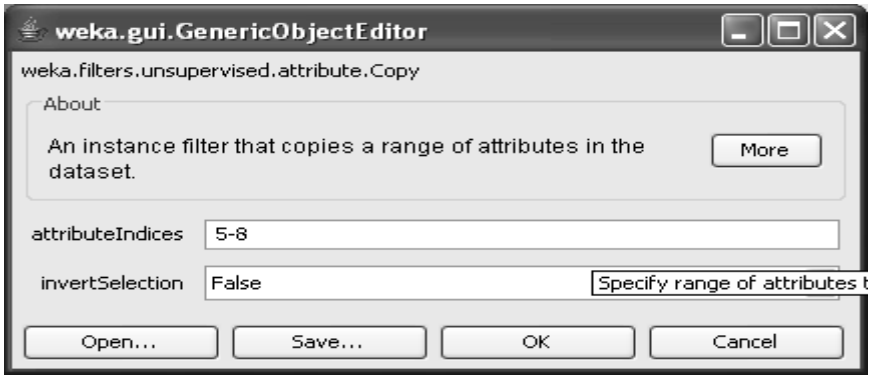


Click on Apply option.

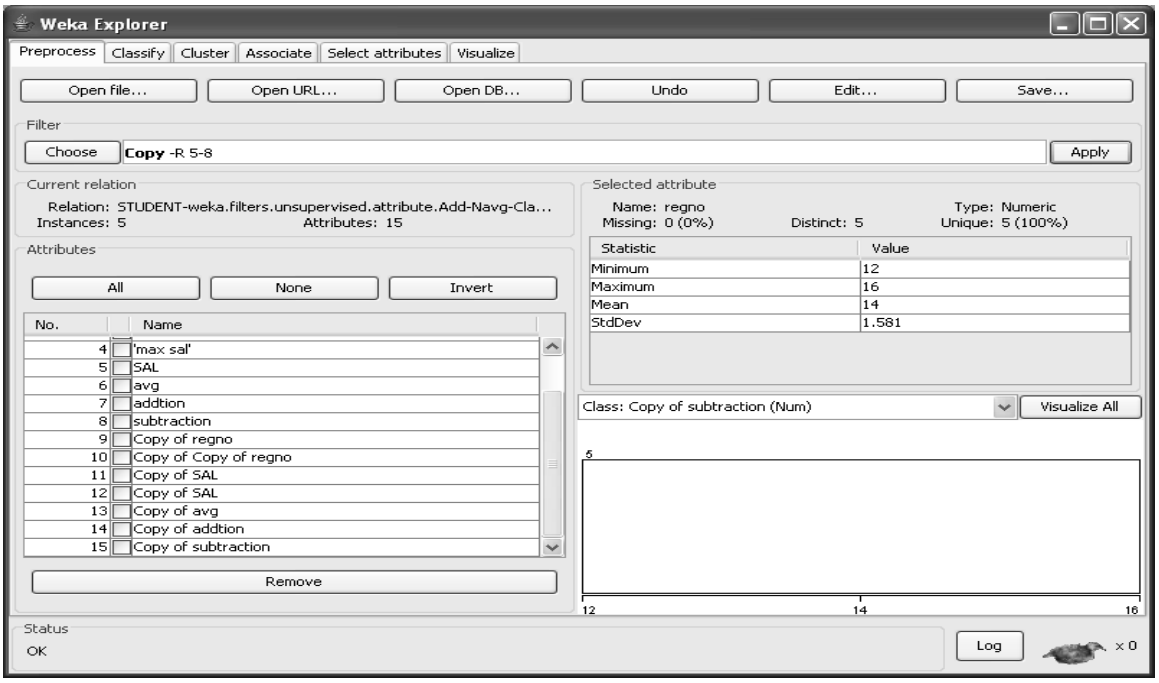


Click on ok button.

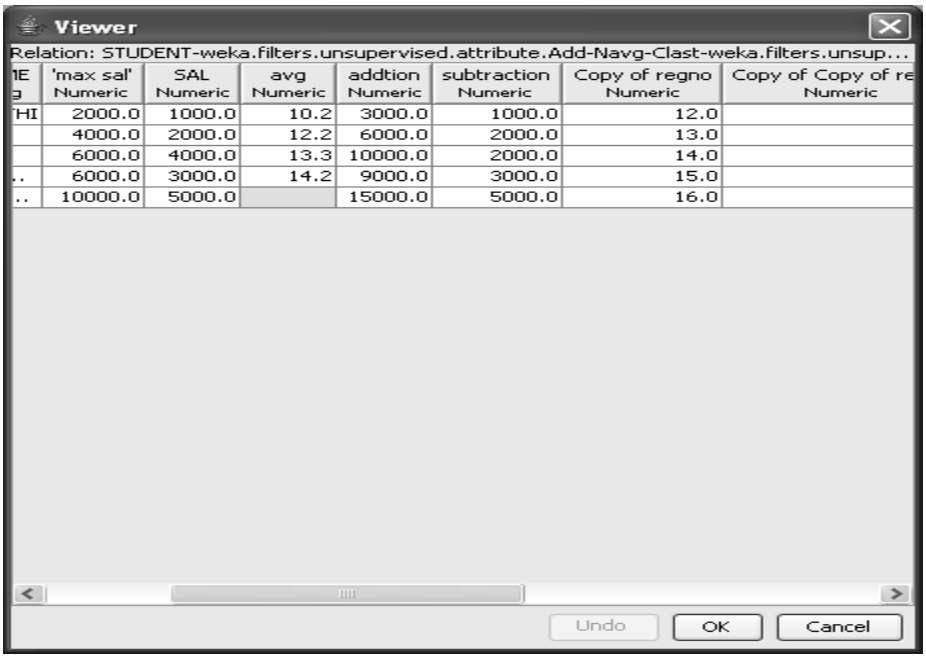
If the index position range.



Click on ok button.



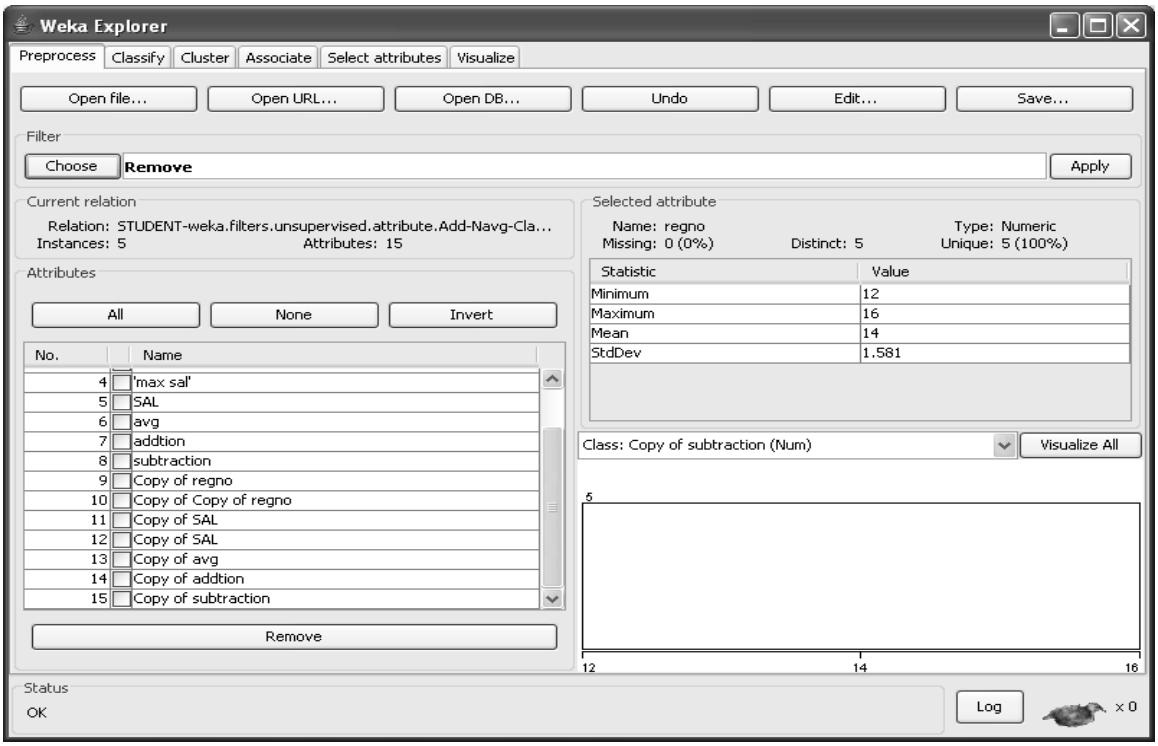
Click on Apply option.

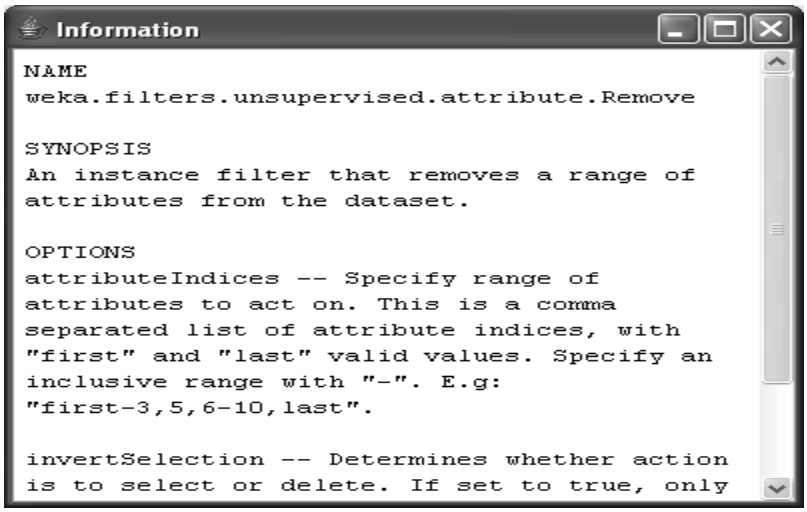


Click on ok button.

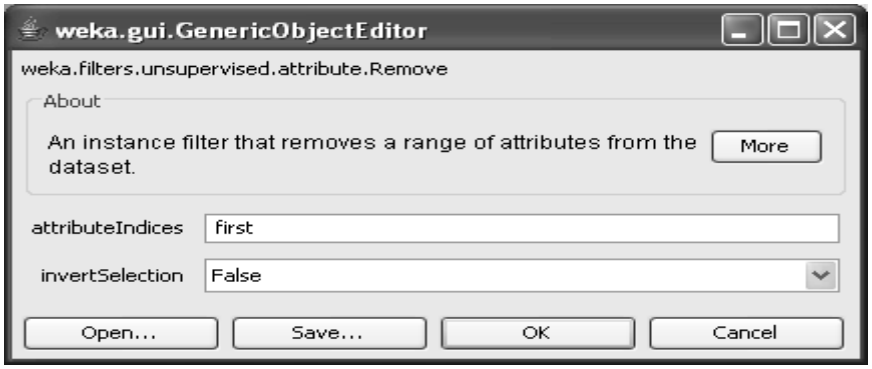
REMOVE

Step-8:- Click on choose button and then select Remove option

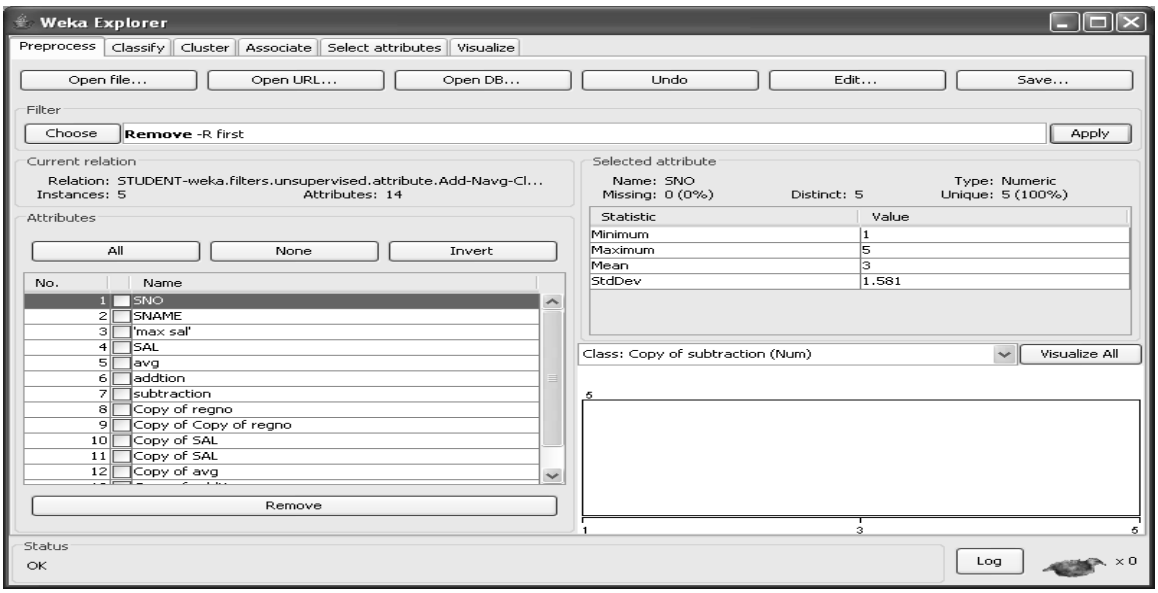




If the index position is first.

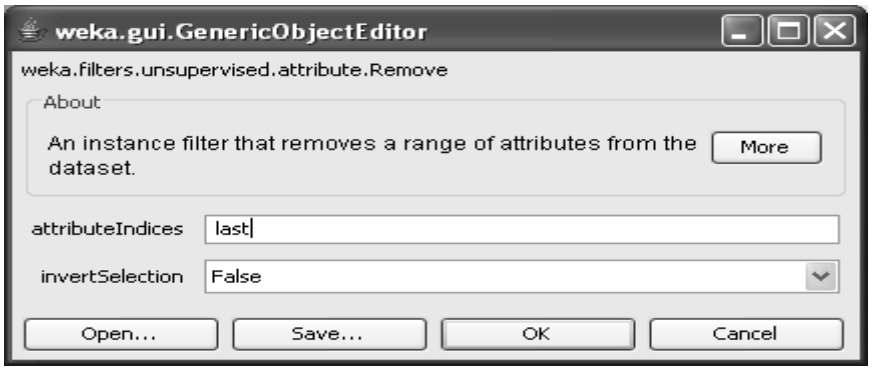


Click on ok button.

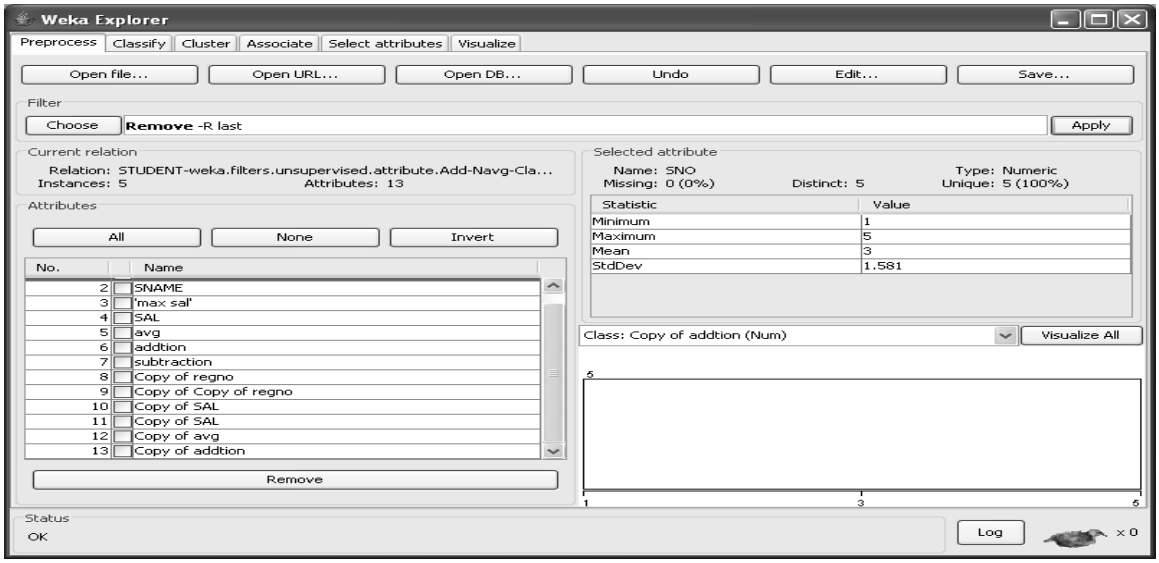


Click on Apply option.

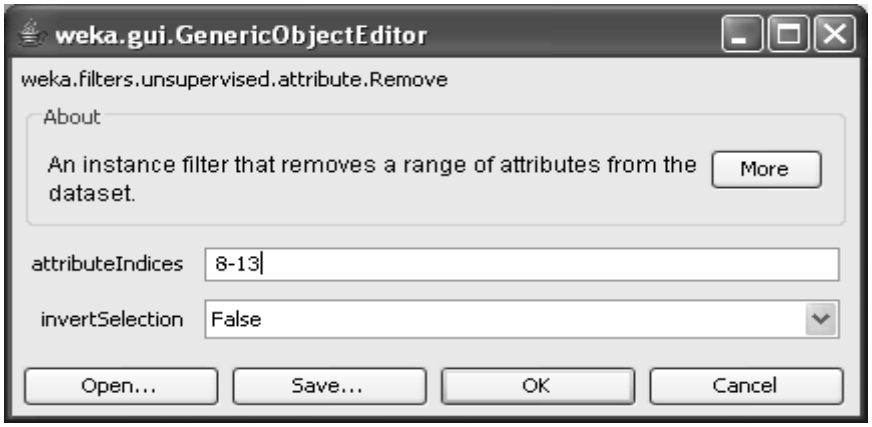
If the index position is last.



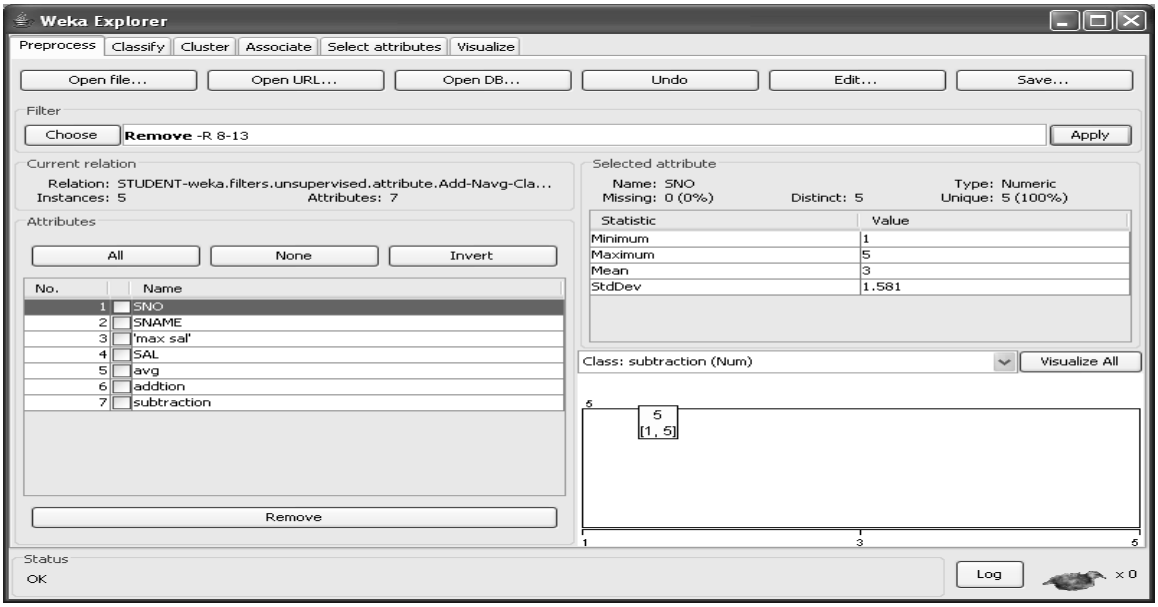
Click on ok button.



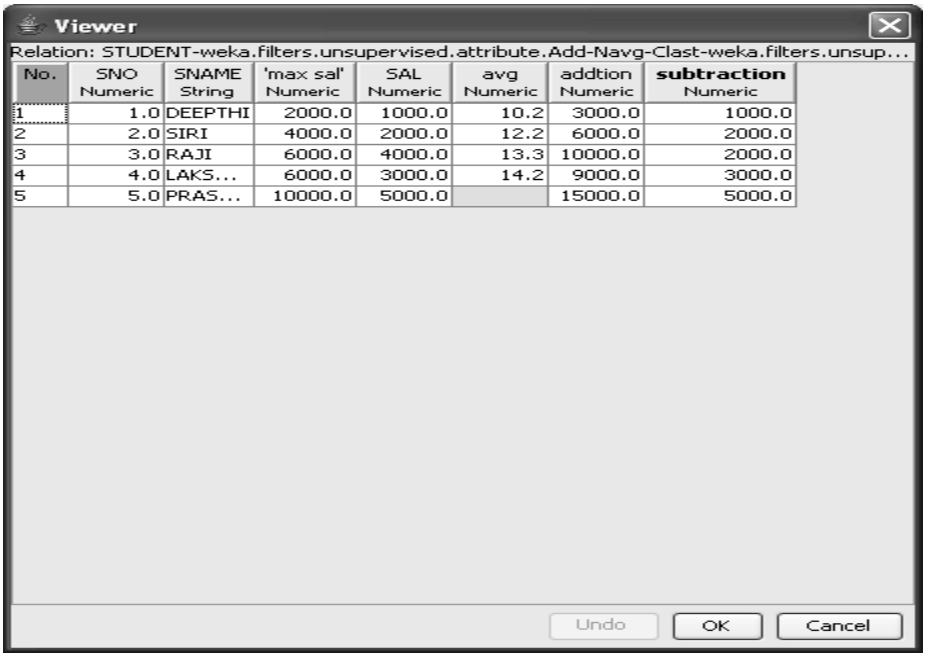
If the range .



Click on ok button.



Click on Apply option.



Click on ok button.

VIVA – QUESTIONS

1. **When to apply the data preprocessing techniques for mining the data**
2. **Which of the following is not a data preprocessing methods:**

   A. Data Visualization B. Data Discretization

   C. Data Cleaning D. Data Reduction
3. **Use the attribute mean to fill the missing value of data 1,2,3,4,5,6,__,7,8,9,10.**
4. **Data for Attendance : 50,55,60,65,70,75,80,85,90,95 Partition the above attendance data into equidepth bins of depth 5.**
5. **Data for Attendance : 4,8,15 Smoot by bin boundaries**
6. **Incorrect or invalid data is known as_____**
7. **PCA stands for**
8. **The Minimum and maximum values for the attribute income are $12,000and $98,000 , respectively. We would like to map income to the range[0.0,1.0].By min-max normalization, a value of $73,600 for income is transformed to**
9. **The mean and standard deviation of the values for the attribute income are $54,000 and $16,000 respectively. With Z-Score normalization a value of $73,600 for income is transformed to**
10. **_____ reduces the data set size by removing irrelevant or redundant attributes(dimensions)**

Experiment No. 3: Demonstrate performing classification on data sets

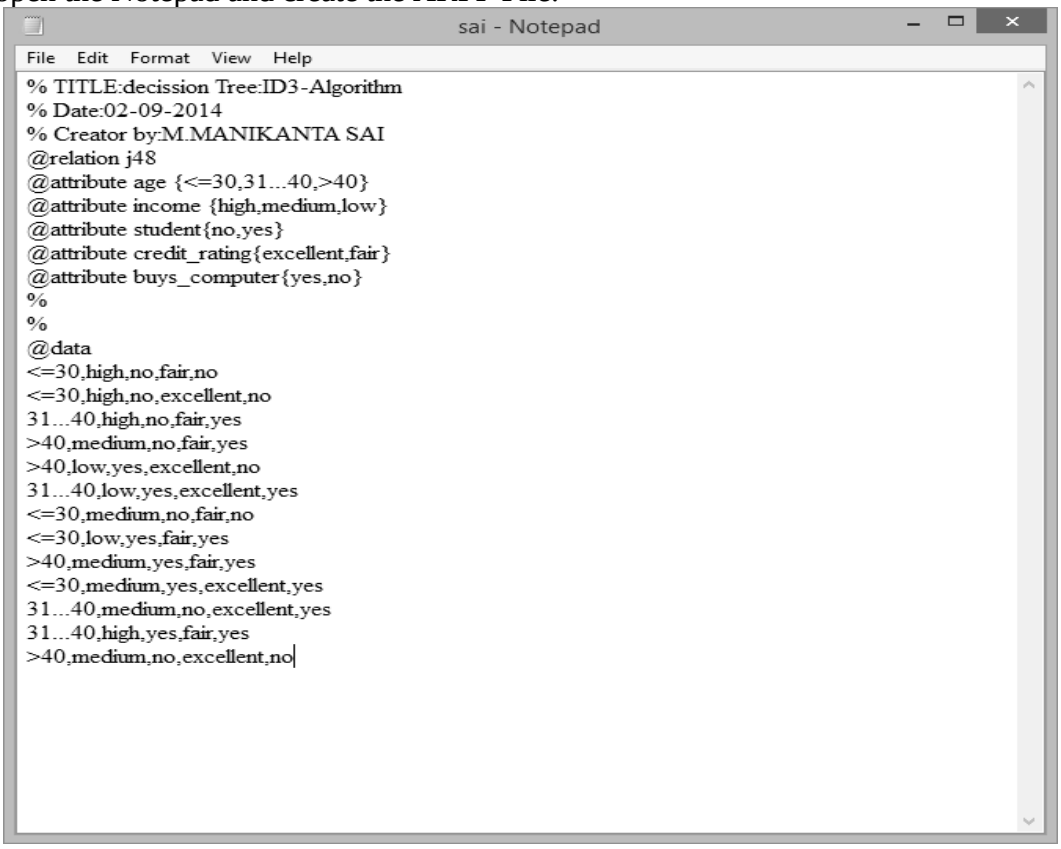
A.

i. Load weather dataset into WEKA and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values, kappa statistic.

ii. Extract if-then rules from the decision tree generated by the classifier, observe the confusion matrix and derive Accuracy, F-measure, TPrate, FPrate, precision and recall values. Apply cross-validation strategy with various fold levels and compare the accuracy results

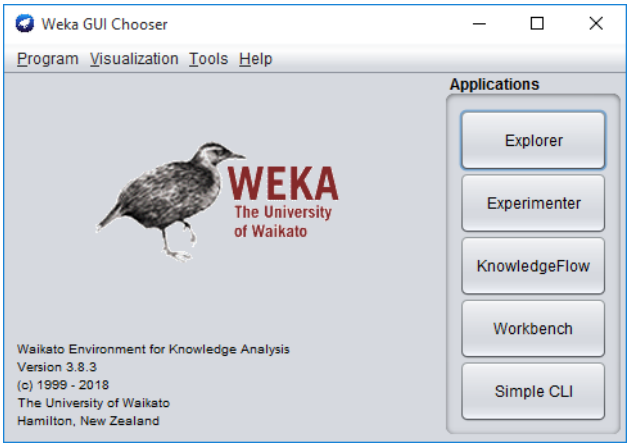
ID3 Algorithm

Step1:open the Notepad and create the ARFF File.



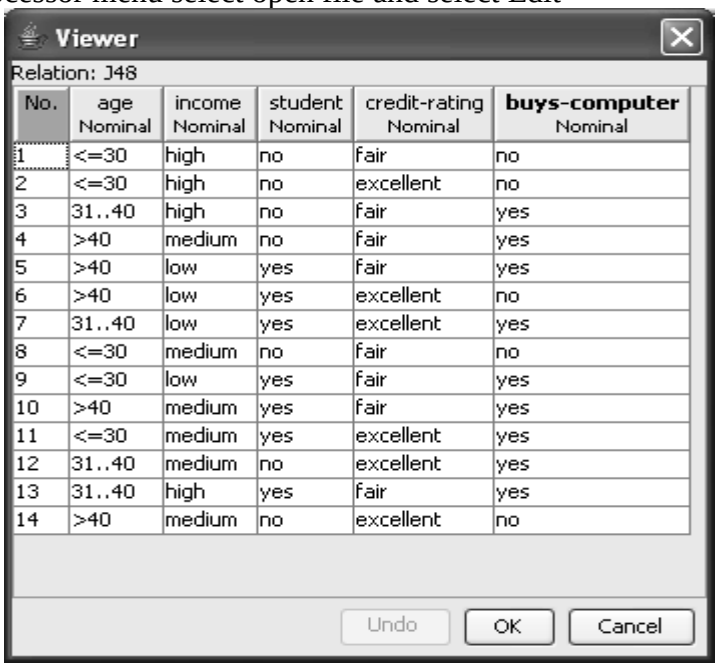
Step 2: To open All Programms⟶ weka3.8.3 ⟶ weka3.8(with console)

STEP 3: Click on **EXPLORER .**

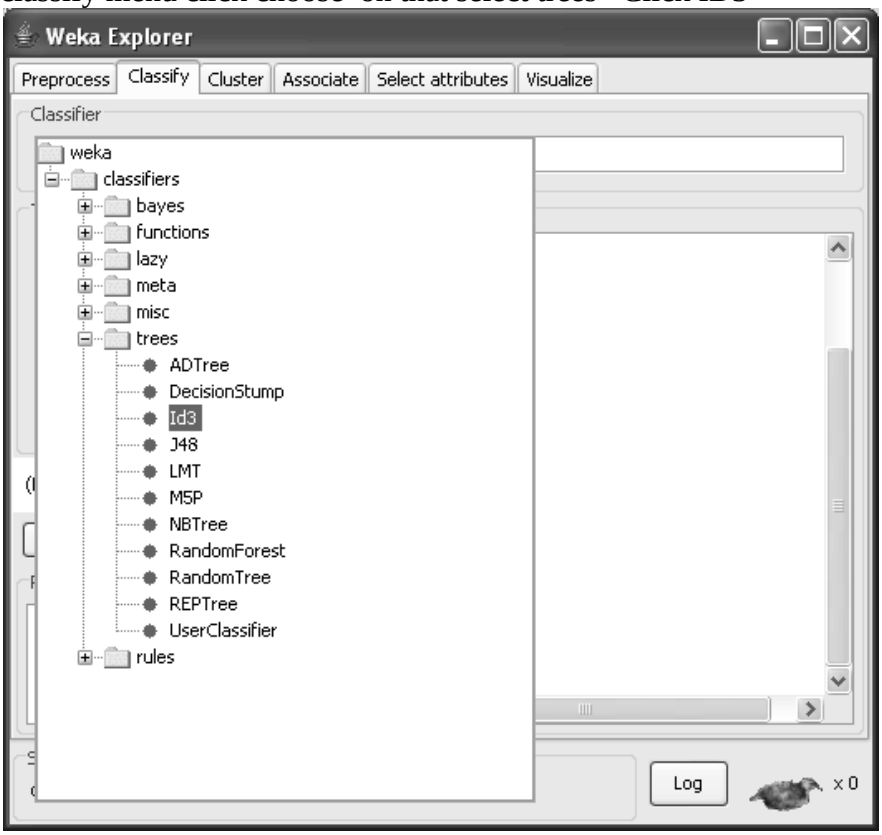


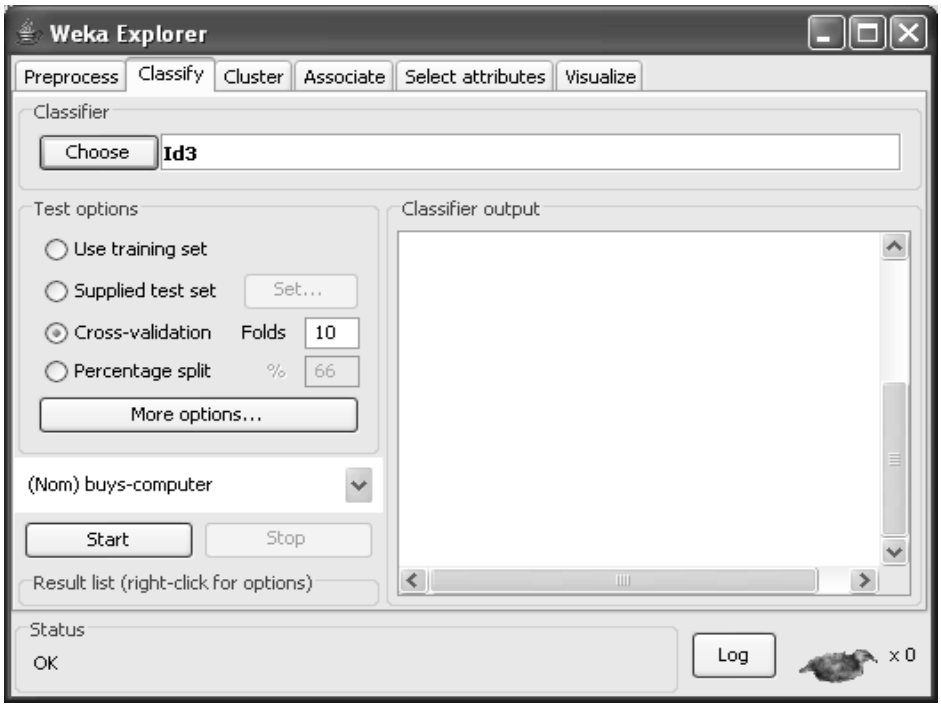
Step 4:open the ARFF file from the folder and click open.

Step 5: In the Preprocessor menu select open file and select Edit

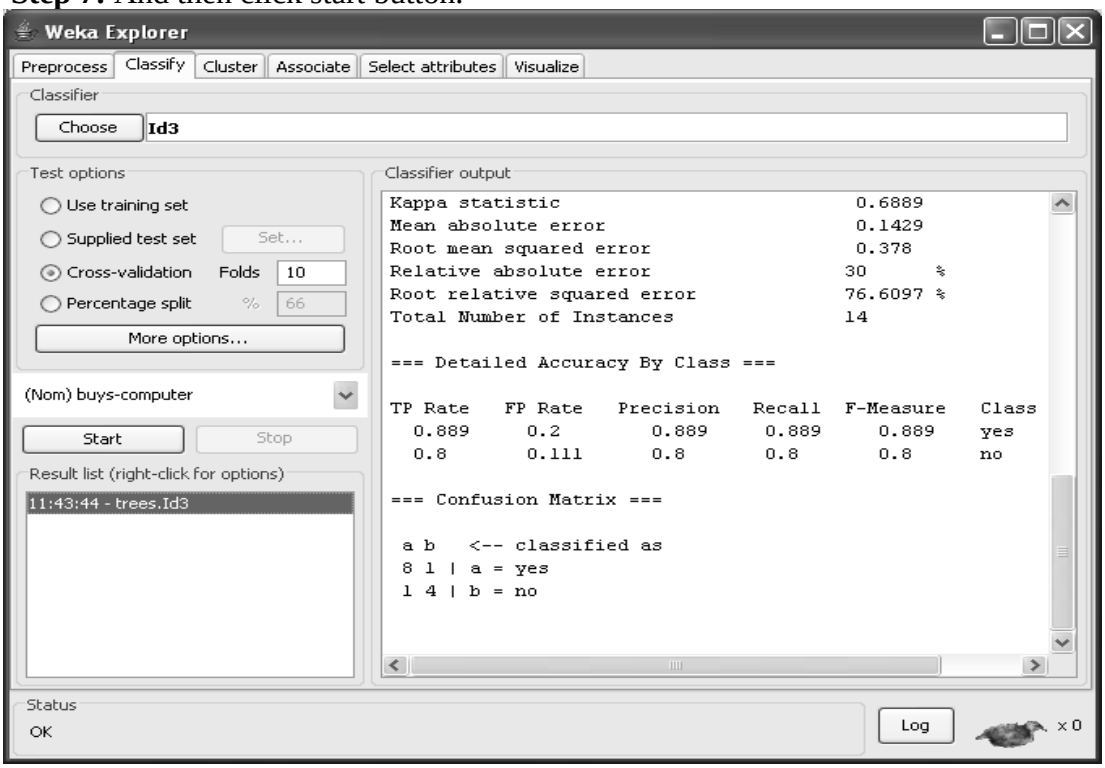


Step 6: Go to classify menu click choose on that select trees Click ID3



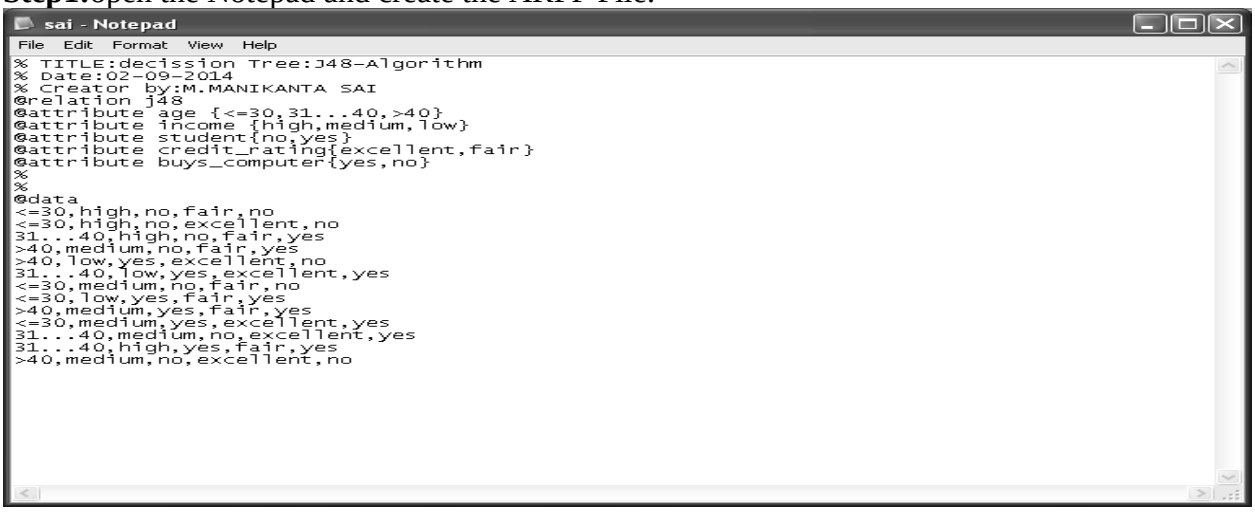


Step 7: And then click start button.

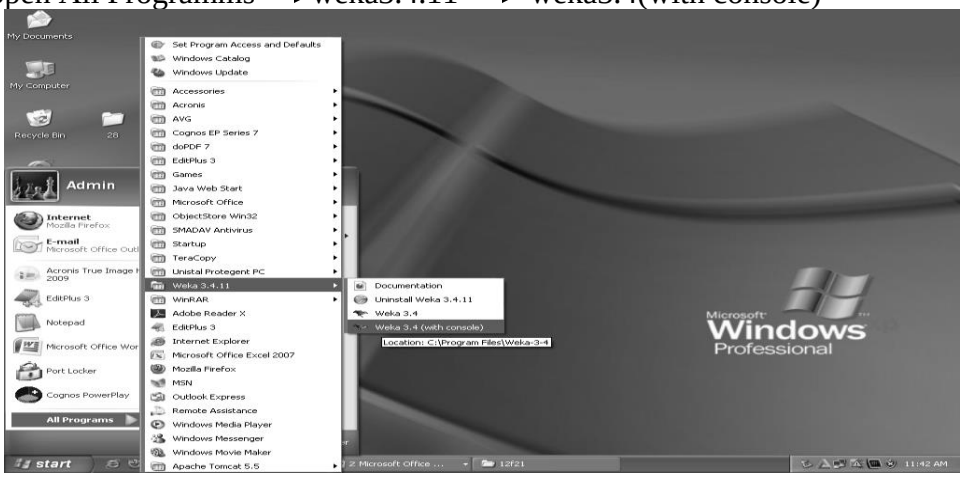


J48

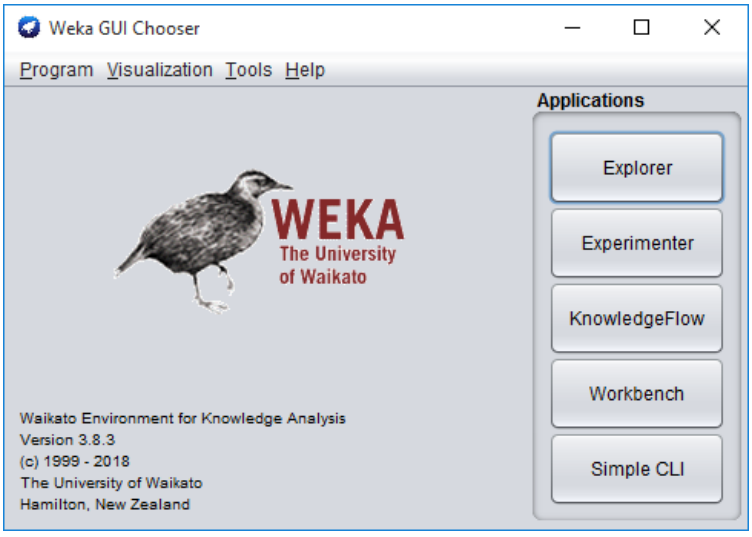
Step1:open the Notepad and create the ARFF File.



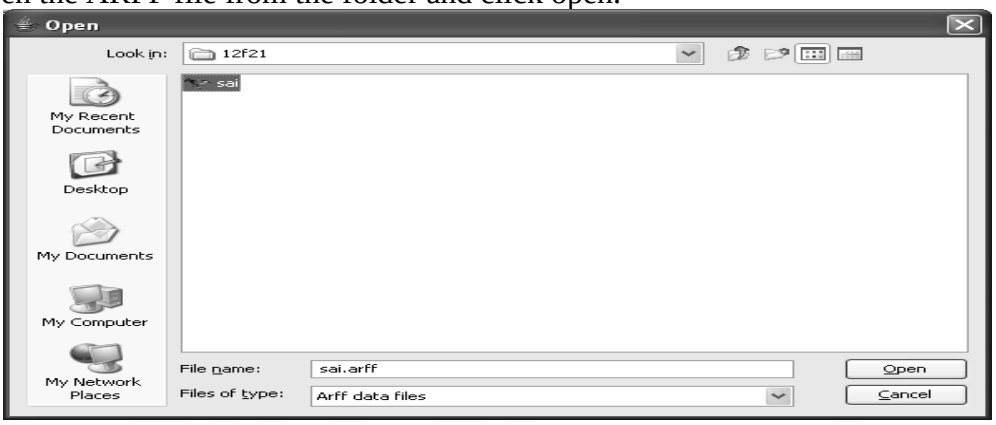
Step2: To open All Programms⟶weka3.4.11 ⟶ weka3.4(with console)

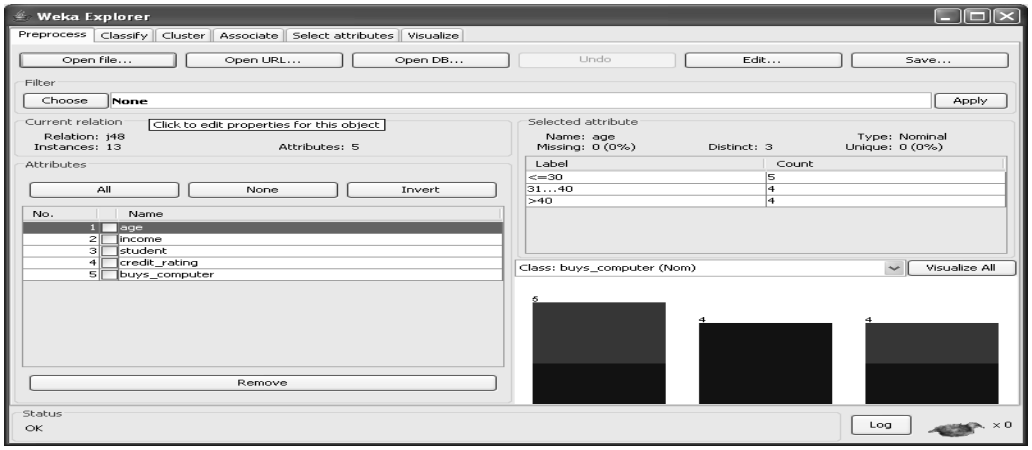


STEP 3: Click on **EXPLORER .**



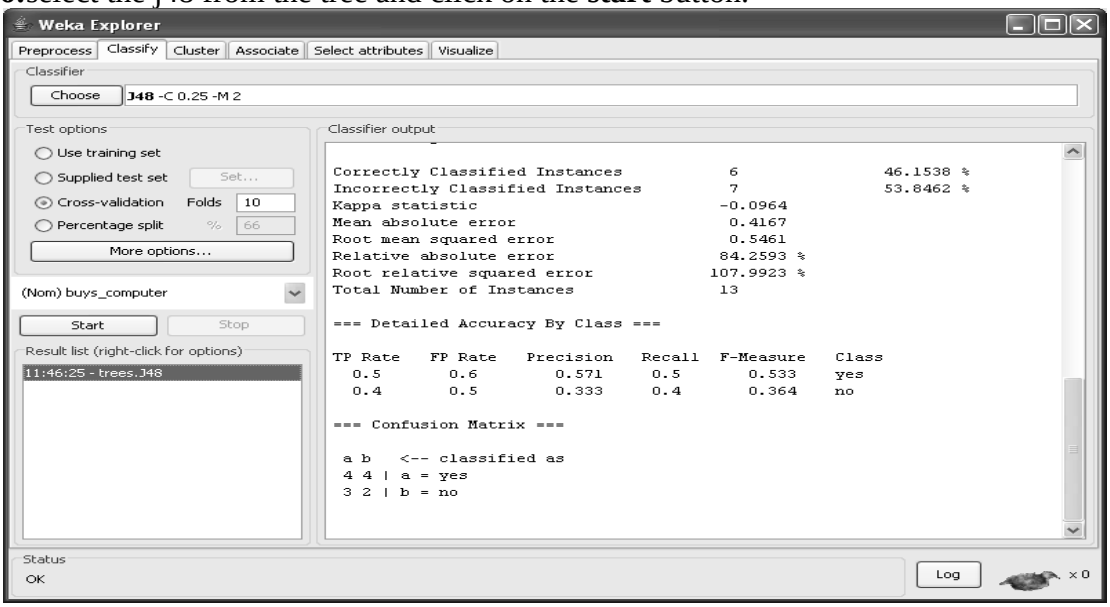
Step 4:open the ARFF file from the folder and click open.



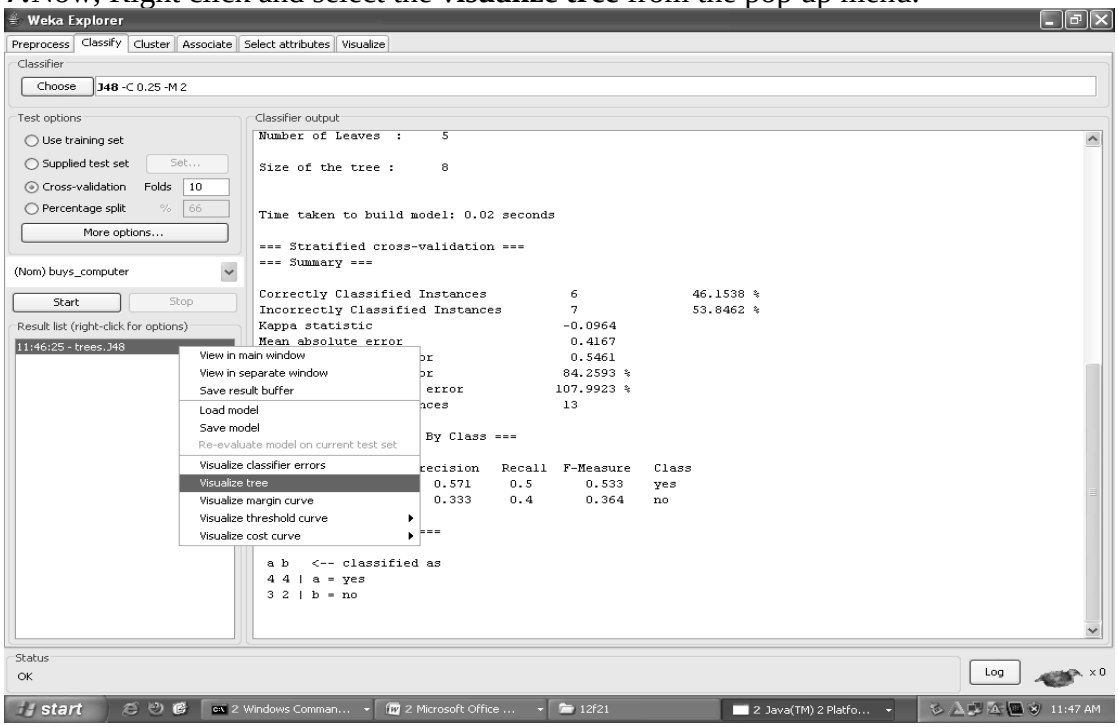


Step 5:click on the **classify** from the menu and click on the **choose.**

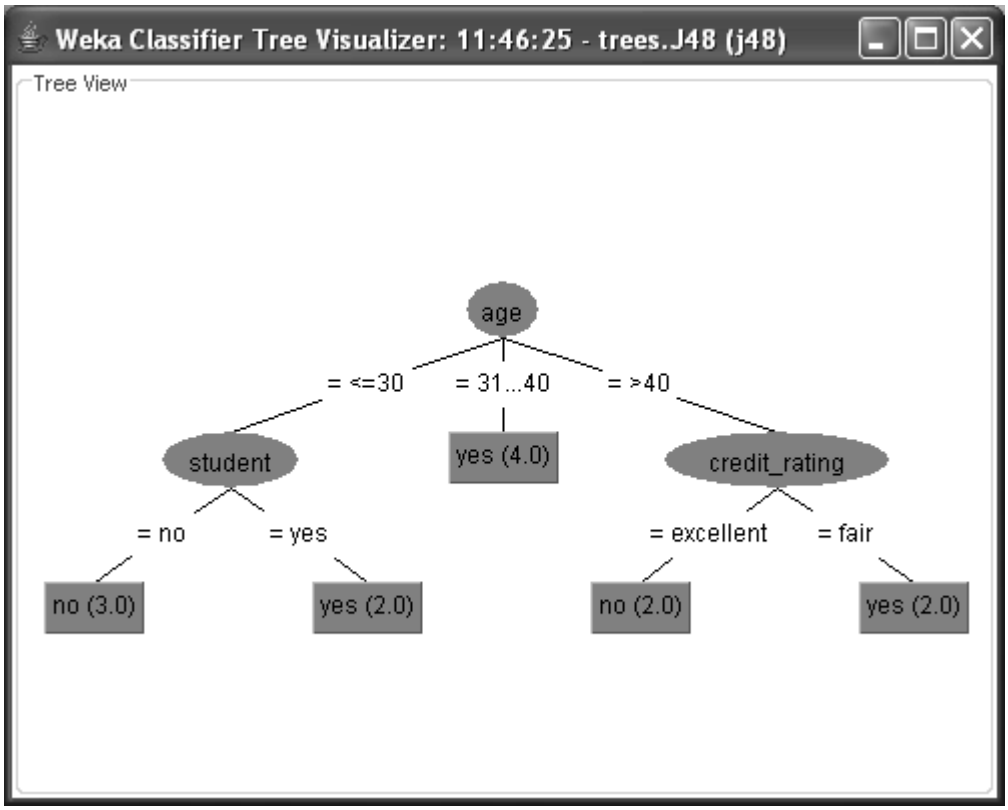
Step 6:select the j48 from the tree and click on the **start** button.



Step 7: Now, Right click and select the **visualize tree** from the pop up menu.



Output:



B.

i. Load weather dataset into WEKA and perform Naive-Bayes classification

ii. K-nearest neighbour classification. Interpret the results obtained.

Naïve Bayesian Classification

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, $\boldsymbol{X}$ = ($x1$, $x2$, : : : , xn), depicting n measurements made on the tuple from n attributes, respectively, $A1$, $A2$, : : : , An.

2. Suppose that there are m classes, $C1$, $C2$, : : : , Cm. Given a tuple, $\boldsymbol{X}$, the classifier will predict that $\boldsymbol{X}$ belongs to the class having the highest posterior probability, conditioned on $\boldsymbol{X}$. That is, the naïve Bayesian classifier predicts that tuple $\boldsymbol{X}$ belongs to the class Ci if and only if

$$P(C_i|\boldsymbol{X}) > P(C_j|\boldsymbol{X}) \quad \text{for } 1 \le j \le m, j \ne i.$$

$$P(C_i|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})}.$$

3. As $P(\boldsymbol{X})$ is constant for all classes, only $P(\boldsymbol{X}\text{j}Ci)P(Ci)$ need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are

$$\begin{aligned} P(\boldsymbol{X}|C_i) &= \prod_{k=1}^{n} P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \end{aligned}$$

4. Given data sets with many attributes, it would be extremely computationally expensive to compute $P(\boldsymbol{X}\text{j}Ci)$. In order to reduce computation in evaluating $P(\boldsymbol{X}\text{j}Ci)$, the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

We wish to predict the class

label of a tuple using naïve Bayesian classification, given the same training data as in Example weather data set for decision tree induction. The training data are in weather data set. The data tuples are described by the attributes *age*, *income*, *student*, and *credit rating*. The class label attribute, *buys computer*, has two distinct values (namely, *yes*, *no*). Let $C1$ correspond to the

class *buys computer = yes* and *C*2 correspond to *buys computer = no*. The tuple we wish to classify is

X = (*age = youth, income = medium, student = yes, credit rating = fair*)

P(*buys computer = yes*) = 9=14 = 0:643

P(*buys computer = no*) = 5=14 = 0:357

To compute *P****X***j*Ci*), for *i* = 1, 2, we compute the following conditional probabilities:

P(*age = youth*|*buys computer = yes*) = 2=9 = 0:222

P(*age = youth*| *buys computer = no*) = 3=5 = 0:600

P(*income = medium*| *buys computer = yes*) = 4=9 = 0:444

P(*income = medium*| *buys computer = no*) = 2=5 = 0:400

P(*student = yes* | *buys computer = yes*) = 6=9 = 0:667

P(*student = yes* | *buys computer = no*) = 1=5 = 0:200

P(*credit rating = fair* | *buys computer = yes*) = 6=9 = 0:667

P(*credit rating = fair* | *buys computer = no*) = 2=5 = 0:400

We need to maximize *P*(***X***j*Ci*)*P*(*Ci*), for *i* = 1, 2. *P*(*Ci*), the prior probability of each class, can be computed based on the training tuples:

Using the above probabilities, we obtain

P(***X***|*buys computer = yes*) = *P*(*age = youth* j *buys computer = yes*) _

P(*income = mediu*| *buys computer = yes*) _

P(*student = yes* | *buys computer = yes*) _

P(*credit rating = fair* | *buys computer = yes*)

= 0:222_0:444_0:667_0:667 = 0:044.

Similarly,

P(***X***|*buys computer = no*) = 0:600_0:400_0:200_0:400 = 0:019.

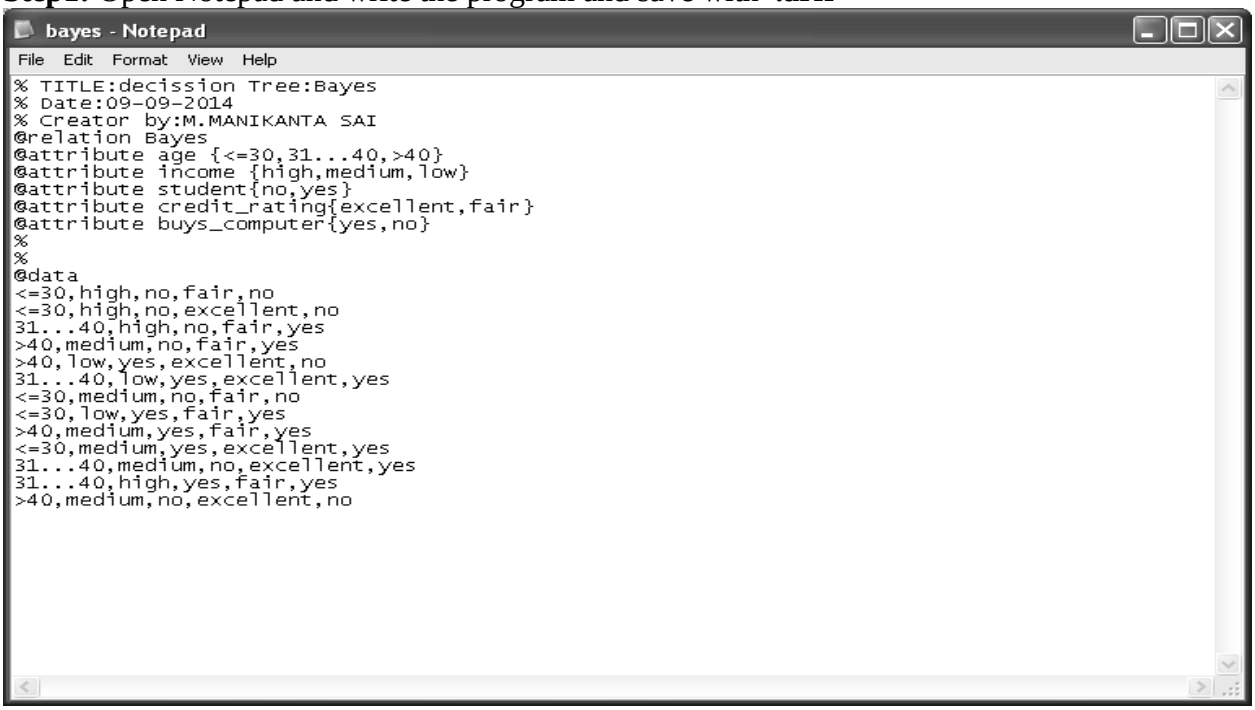
To find the class, *Ci*, that maximizes *P*(***X***j*Ci*)*P*(*Ci*), we compute

P(***X***|*buys computer = yes*)*P*(*buys computer = yes*) = 0:044_0:643 = 0:028

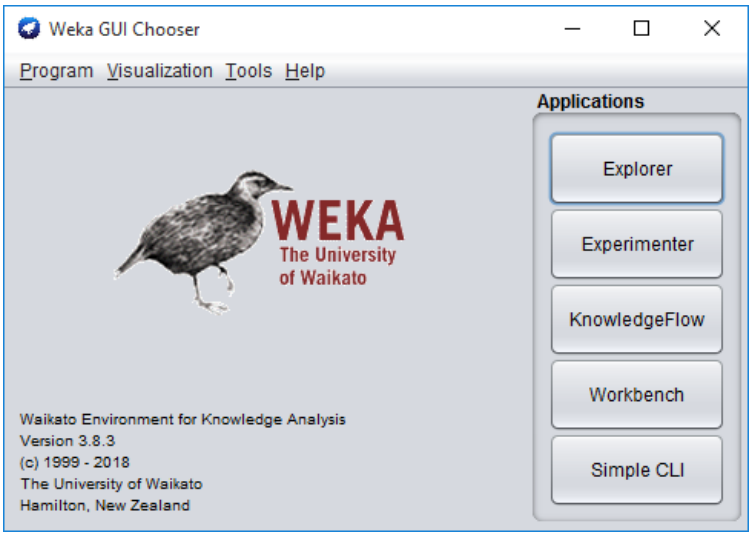
P(***X***|*buys computer = no*)*P*(*buys computer = no*) = 0:019_0:357 = 0:007

Therefore, the naïve Bayesian classifier predicts *buys computer = yes* for tuple ***X***.

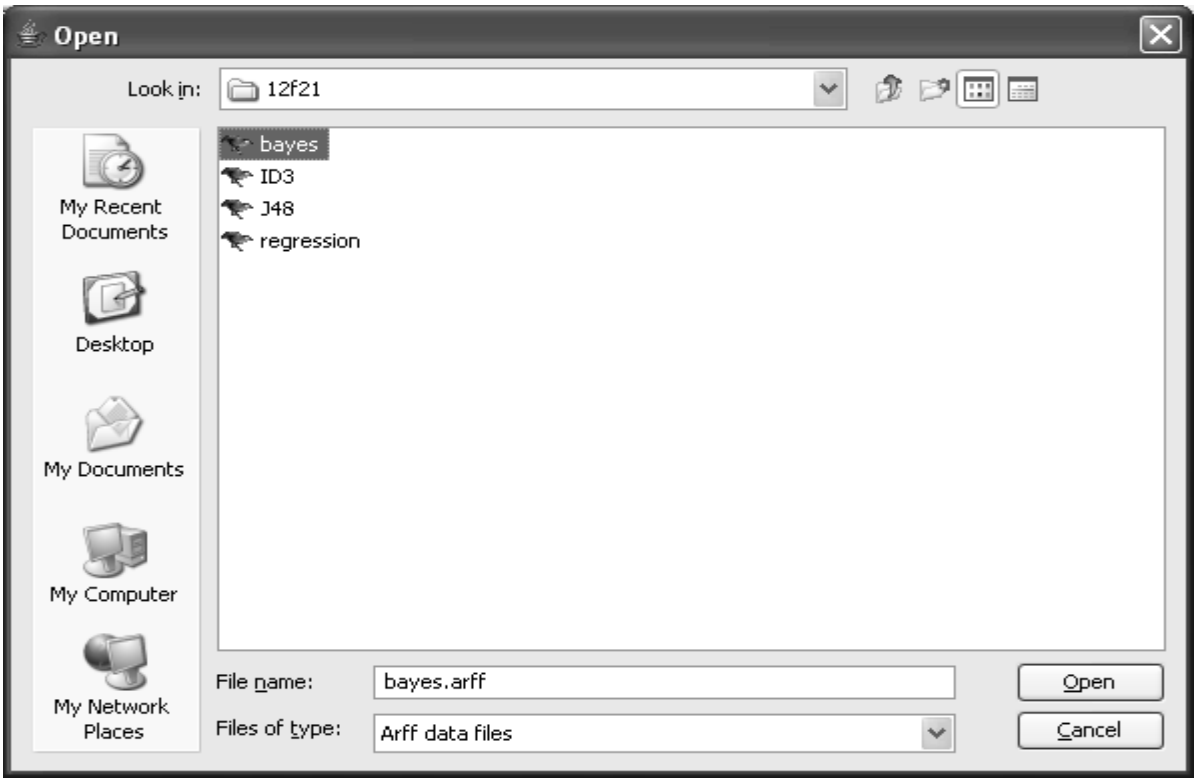
Step1: Open Notepad and write the program and save with **.arff**



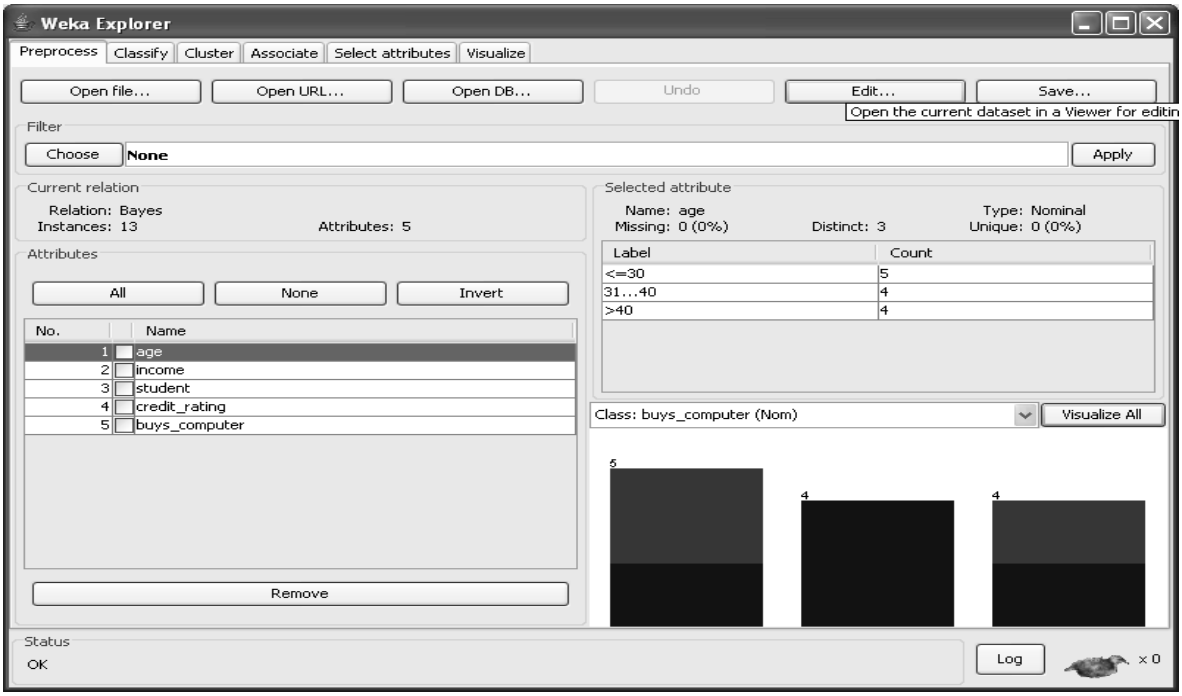
Step2: Go to Start menu and select Weka 3.8.3, on that select Weka 3.8(with console).

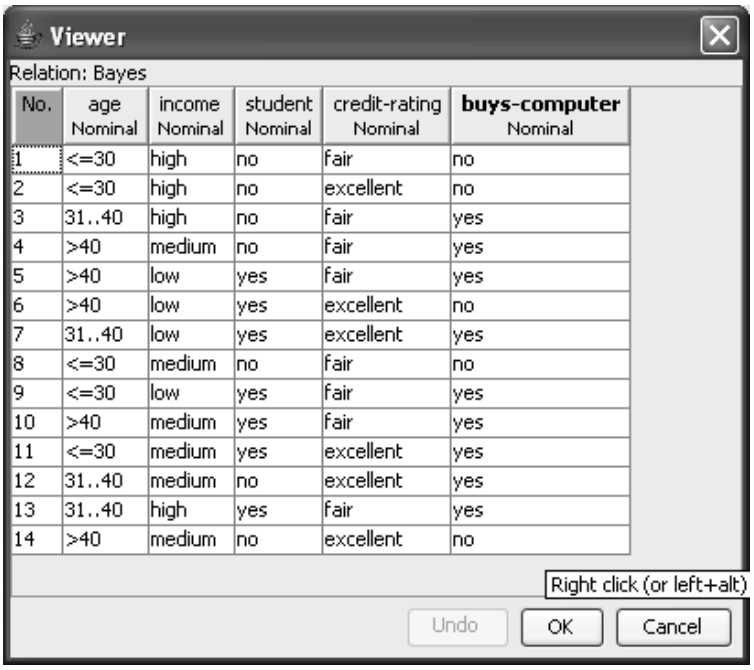


Step3:Go to Preprocessor and select the open file.



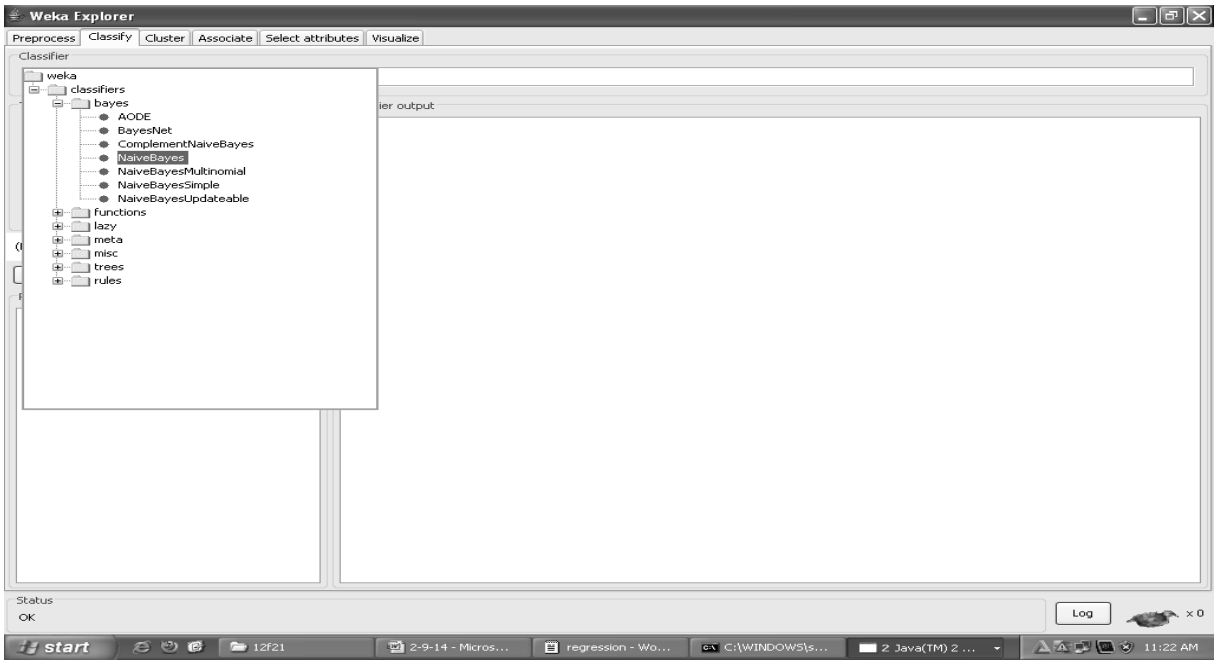
Step4: In the Preprocessor menu selectEdit.



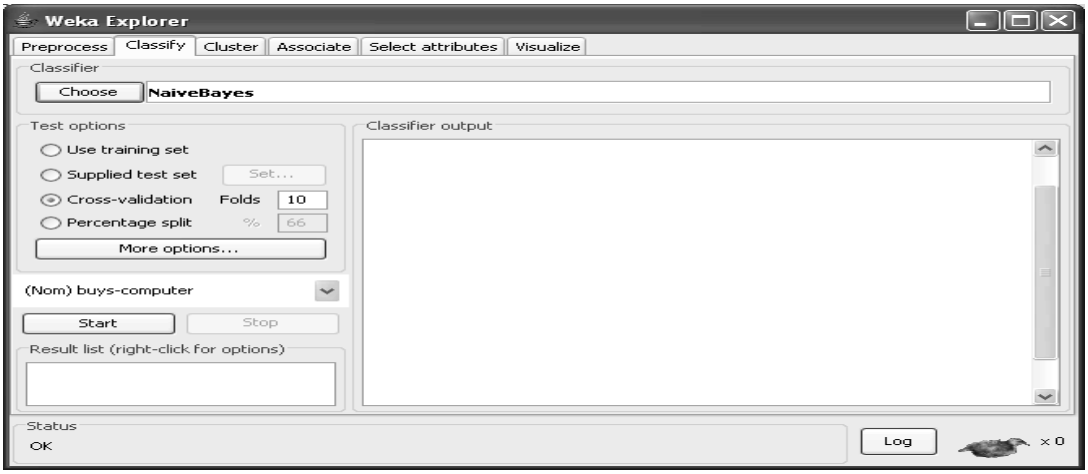


No.	age Nominal	income Nominal	student Nominal	credit-rating Nominal	**buys-computer** Nominal
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31..40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31..40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31..40	medium	no	excellent	yes
13	31..40	high	yes	fair	yes
14	>40	medium	no	excellent	no

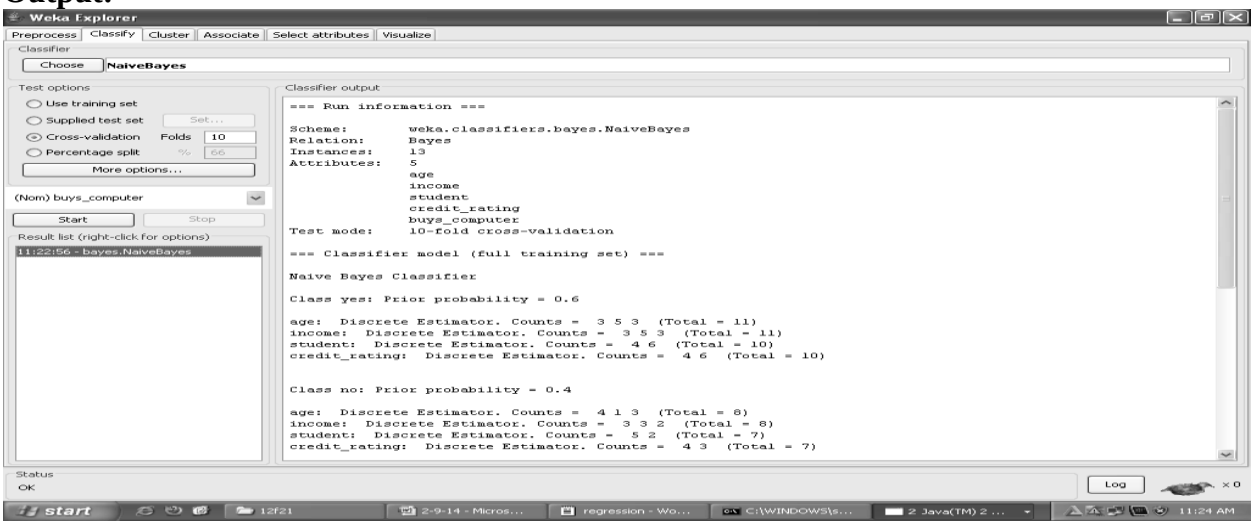
Step5: Go to **classify** tab menu ,click no **choose** button and then select **Trees** from the list Click on the **NaiveBayes** option**.**

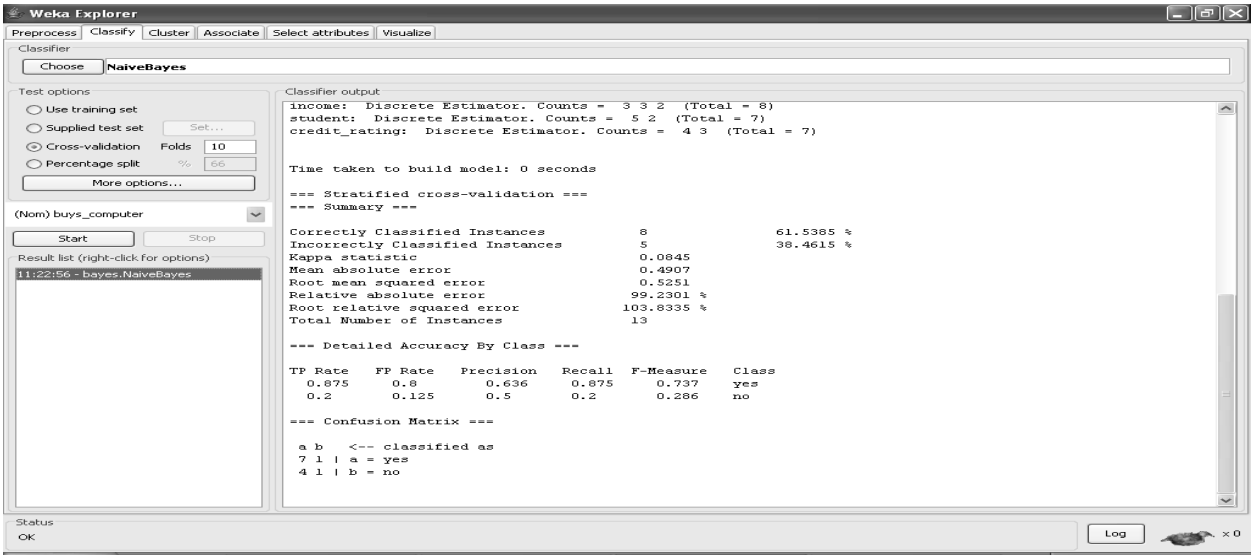


Step 6:click on the **start** button to view the result.


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...
Classifier output
(Nom) buys-computer
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0

Output:


Weka Explorer
Classifier output
=== Run information ===
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: Bayes
Instances: 13
Attributes: 5
age
income
student
credit_rating
buys_computer
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
Naive Bayes Classifier
Class yes: Prior probability = 0.6
age: Discrete Estimator. Counts = 3 5 3 (Total = 11)
income: Discrete Estimator. Counts = 3 5 3 (Total = 11)
student: Discrete Estimator. Counts = 4 6 (Total = 10)
credit_rating: Discrete Estimator. Counts = 4 6 (Total = 10)
Class no: Prior probability = 0.4
age: Discrete Estimator. Counts = 4 1 3 (Total = 8)
income: Discrete Estimator. Counts = 3 3 2 (Total = 8)
student: Discrete Estimator. Counts = 5 2 (Total = 7)
credit_rating: Discrete Estimator. Counts = 4 3 (Total = 7)


Weka Explorer
Classifier output
income: Discrete Estimator. Counts = 3 3 2 (Total = 8)
student: Discrete Estimator. Counts = 5 2 (Total = 7)
credit_rating: Discrete Estimator. Counts = 4 3 (Total = 7)
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 8 61.5385 %
Incorrectly Classified Instances 5 38.4615 %
Kappa statistic 0.0845
Mean absolute error 0.4907
Root mean squared error 0.5251
Relative absolute error 99.2301 %
Root relative squared error 103.8335 %
Total Number of Instances 13
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure Class
0.875 0.8 0.636 0.875 0.737 yes
0.2 0.125 0.5 0.2 0.286 no
=== Confusion Matrix ===
a b <-- classified as
7 1 | a = yes
4 1 | b = no

VIVA-QUESTIONS

1. Formulae for Information gain=_____
2. Formulae for GainRatio=_____
3. Compute entropy value of(9,5) = _____
4. Classification is supervised learning.
5. _____ measure is used to select the test attribute at each node in the decision tree.
6. Posterior probability can be calculated by _____ theorem.
7. Decision tree is a type of _____ algorithm.
8. KNN stand for
9. How does KNN calculate distance?
10. ID3 stands for

Experiment No. 4: Demonstrate performing association rule mining on data sets

i. Load transaction dataset into WEKA and run Aprori algorithm with different support and confidence values.
ii. The Apriori algorithm uses a generate and count strategy for deriving frequent items sets and generate association rules.

The Apriori Algorithm

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties, as we shall see following. Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are usedtoexplore (*k*+1)-itemsets. First, the setof frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted *L*1.Next, *L*1 is used to find *L*2, the set of frequent 2-itemsets, which is used to find *L*3, and so on, until no more frequent *k*-itemsets can be found. The finding of each *Lk* requires one full scan of the database.

Apriori property: *All nonempty subsets of a frequent itemset must also be frequent.* TheApriori property is based on the following observation. By definition, if an itemset *I* does not satisfy the minimum support threshold, *min sup*, then *I* is not frequent; that is, *P*(*I*) < *min sup*. If an item *A* is added to the itemset *I*, then the resulting itemset (i.e., *I* [*A*) cannot occur more frequently than *I*. Therefore, *I* [*A* is not frequent either; that is, *P*(*I* [*A*) < *min sup*

The join step: To find *Lk*, a set of candidate *k*-itemsets is generated by joining *Lk*□1 with itself. This set of candidates is denoted *Ck*. Let *l*1 and *l*2 be itemsets in *Lk*□1. The notation *li*[*j*] refers to the *j*th item in *li* (e.g., *l*1[*k*□2] refers to the second to the last item in *l*1). For the (*k*□1)-itemset, *li*, this means that the items are sorted such that *li*[1] < *li*[2] < : : : < *li*[*k*□1]. The join, *Lk*□1 on *Lk*□1, is performed, where members of *Lk*□1 are joinable if their first (*k*□2) items are in common. That is, members *l*1 and *l*2 of *Lk*□1 are joined if (*l*1[1] = *l*2[1]) ^ (*l*1[2] = *l*2[2]) ^: : :^(*l*1[*k*□2] = *l*2[*k*□2]) ^(*l*1[*k*□1] < *l*2[*k*□1]). The condition *l*1[*k*□1] < *l*2[*k*□1] simply ensures that no duplicates are generated. The resulting itemset formed by joining *l*1 and *l*2 is *l*1[1], *l*1[2], : : : , *l*1[*k*□2], *l*1[*k*□1], *l*2[*k*□1].

The prune step:*Ck* is a superset of *Lk*, that is, its members may or may not be frequent, but all of the frequent *k*-itemsets are included in*Ck*.Ascan of the database to determine the count of each

candidate in *Ck* would result in the determination of *Lk* (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to *Lk*).

Transactional data for an *AllElectronics* branch.

TID	List of item_IDs
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
T900	I1, I2, I3

In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, *C*1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

2. Suppose that the minimum support count required is 2, that is, *min sup* = 2. (Here, we are referring to *absolute* support because we are using a support count. The corresponding relative support is 2/9 = 22%). The set of frequent 1-itemsets, *L*1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support. In our example, all of the candidates in *C*1 satisfy minimum support.

3. To discover the set of frequent 2-itemsets, *L*2, the algorithm uses the join *L*1 on *L*1 to generate a candidate set of 2-itemsets, *C*2.8 *C*2 consists of □j*L*1j 2 2-itemsets. Note that no candidates are removed from*C*2 during the prune step because each subset of the candidates is also frequent.

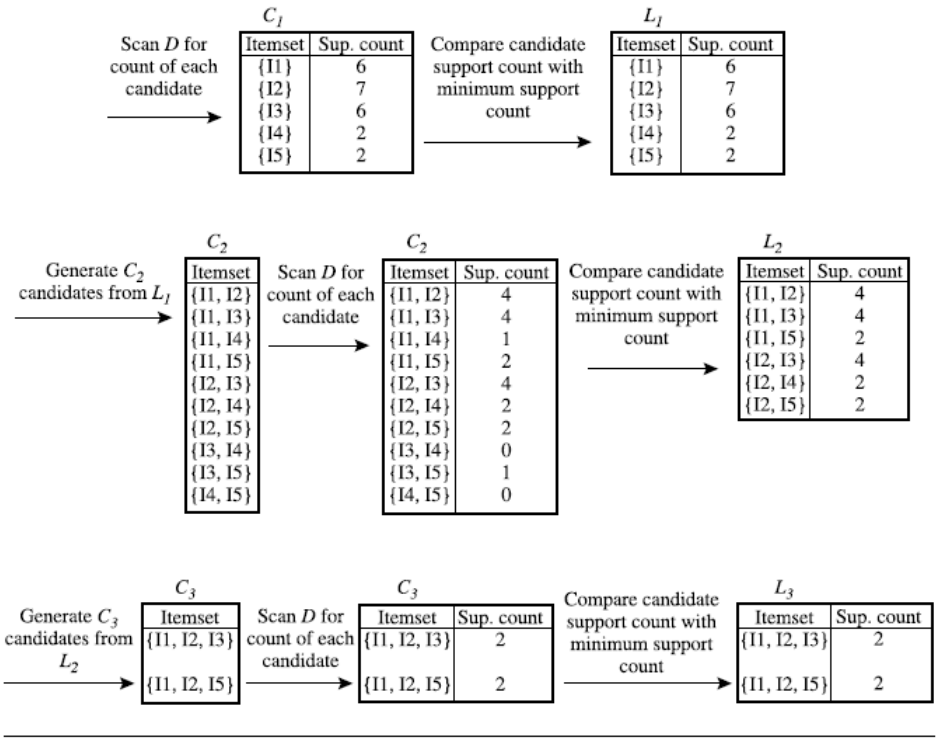
4.Next, the transactions in*D*are scanned and the support count of each candidate itemset in*C*2 is accumulated.

5. The set of frequent 2-itemsets, *L*2, is then determined, consisting of those candidate 2-itemsets in *C*2 having minimum support.

6. The generation of the set of candidate 3-itemsets,*C*3,. From the join step, we first get*C*3 =*L*2 on*L*2 = ffI1, I2, I3g, fI1, I2, I5g, fI1, I3, I5g, fI2, I3, I4g, fI2, I3, I5g, fI2, I4, I5gg. Based on the Apriori property that all subsets of a frequent itemsetmust also be frequent, we can determine that the four latter candidates cannot possibly be frequent.We therefore remove them from*C*3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of*D*to determine *L*3. Note thatwhen given a candidate *k*-itemset,we only need to check if its (*k*□1)-subsets are frequent since theApriori algorithm uses a level-wise search strategy.

7. The transactions in *D* are scanned in order to determine *L*3, consisting of those candidate 3-itemsets in *C*3.

8. The algorithm uses *L*3 on *L*3 to generate a candidate set of 4-itemsets, *C*4. Although the join results in I1, I2, I3, I5gg, this itemset is pruned because its subset I2, I3, I5 is not frequent. Thus, *C*4 = f, and the algorithm terminates, having found all of the frequent itemsets.



.

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ ϕ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // join step: generate candidates
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // prune step: remove unfruitful candidate
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

Association rule mining using the Apriori-Algorithm

Step-1: To Create an ARFF File.

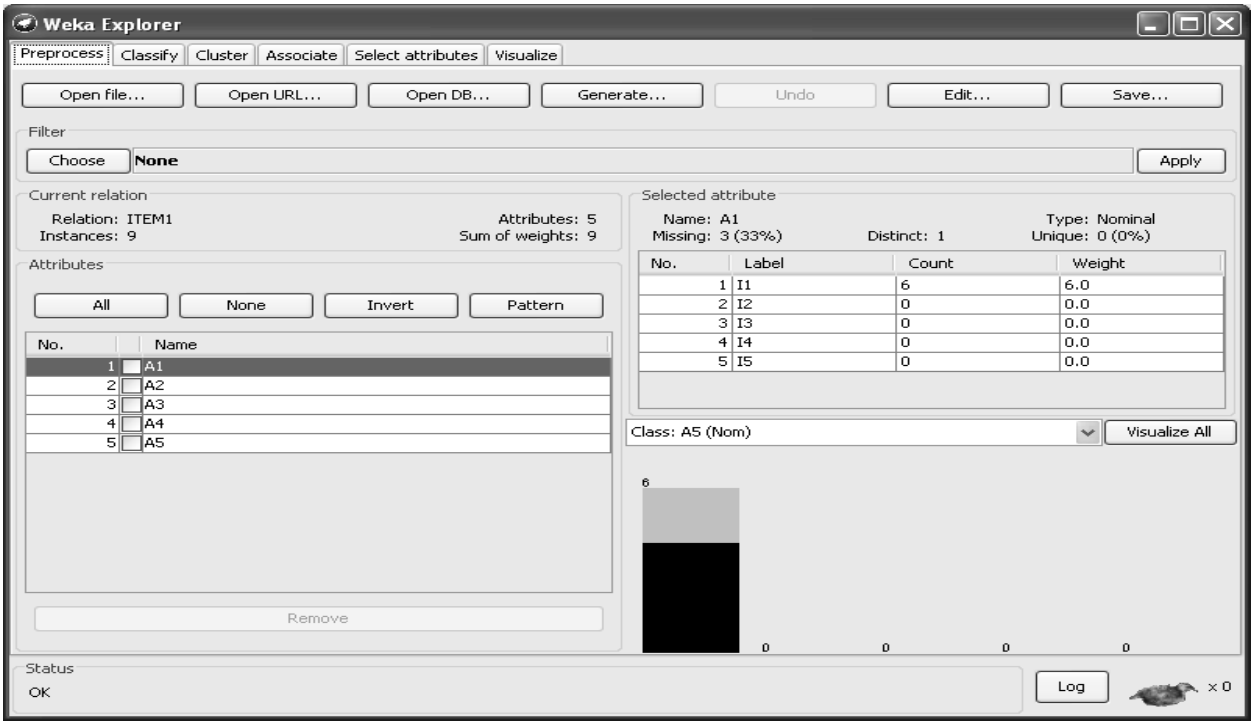
ITEM1 - Notepad

File Edit Format View Help

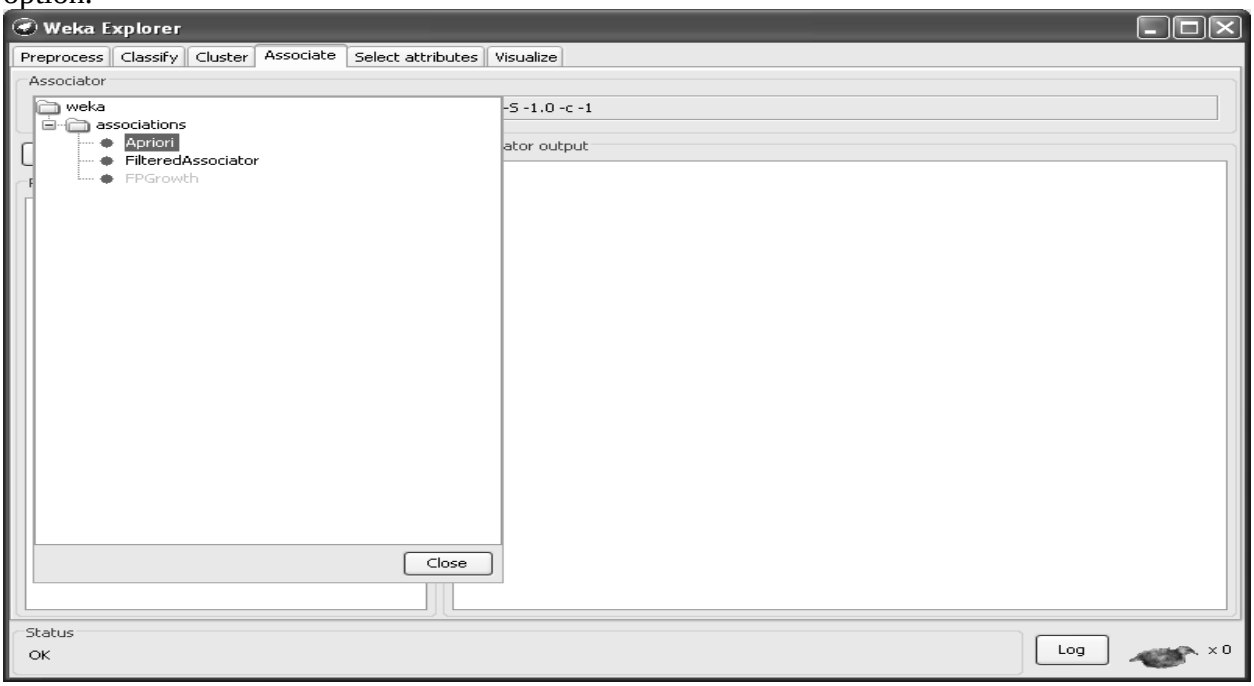
```
%TITLE : TRANSACTION -ITEM
%
%DATE :05-08-2014
%
@RELATION ITEM1
@ATTRIBUTE A1{I1,I2,I3,I4,I5}
@ATTRIBUTE A2{I1,I2,I3,I4,I5}
@ATTRIBUTE A3{I1,I2,I3,I4,I5}
@ATTRIBUTE A4{I1,I2,I3,I4,I5}
@ATTRIBUTE A5{I1,I2,I3,I4,I5}
%
%
@DATA
I1,I2,?,?,I5
?,I2,?,I4,?
?,I2,I3,?,?
I1,I2,?,I4,?
I1,?,I3,?,?
?,I2,I3,?,?
I1,?,I3,?,?
```

Step-2: -Save as the file .ARFF file format

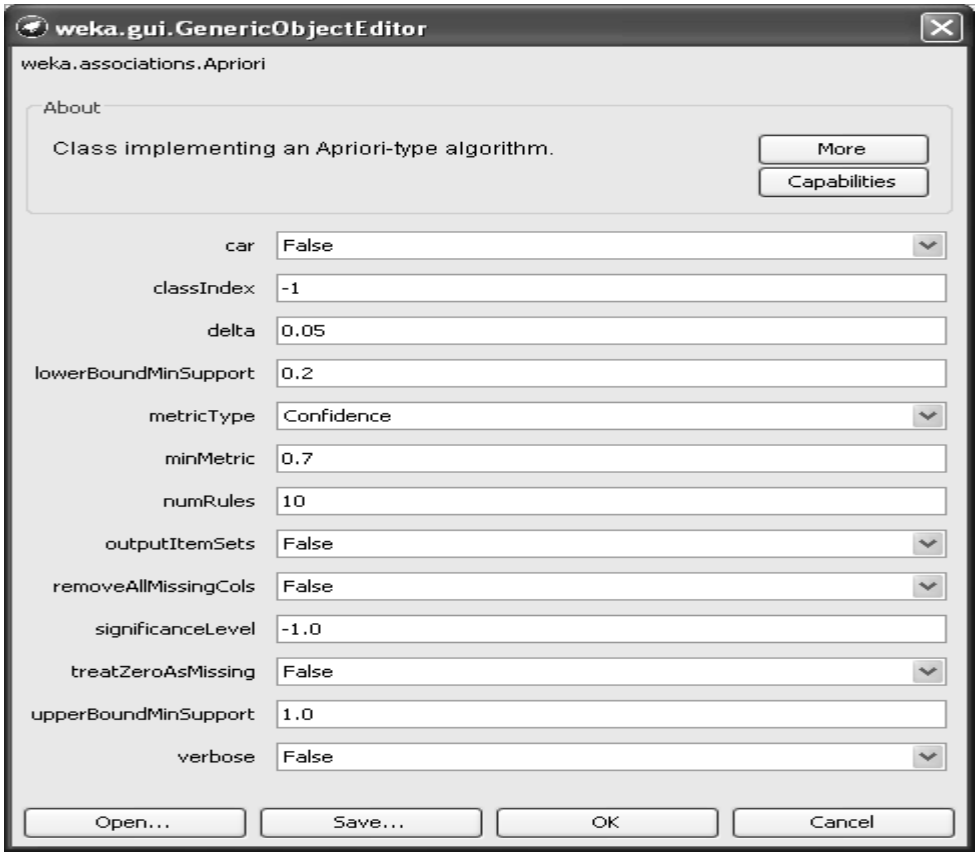
Step-3: - Go to All Programs in the start menu bar and then choose weeka 3.8.3 in the weeka 3.8(with console).
Step-4: In the weeka window to choose the Explorer and then select the Preprocess in the weeka explorer window.



Step-5: Go to Association option in the menu and then select choose button to select the Apriori option.

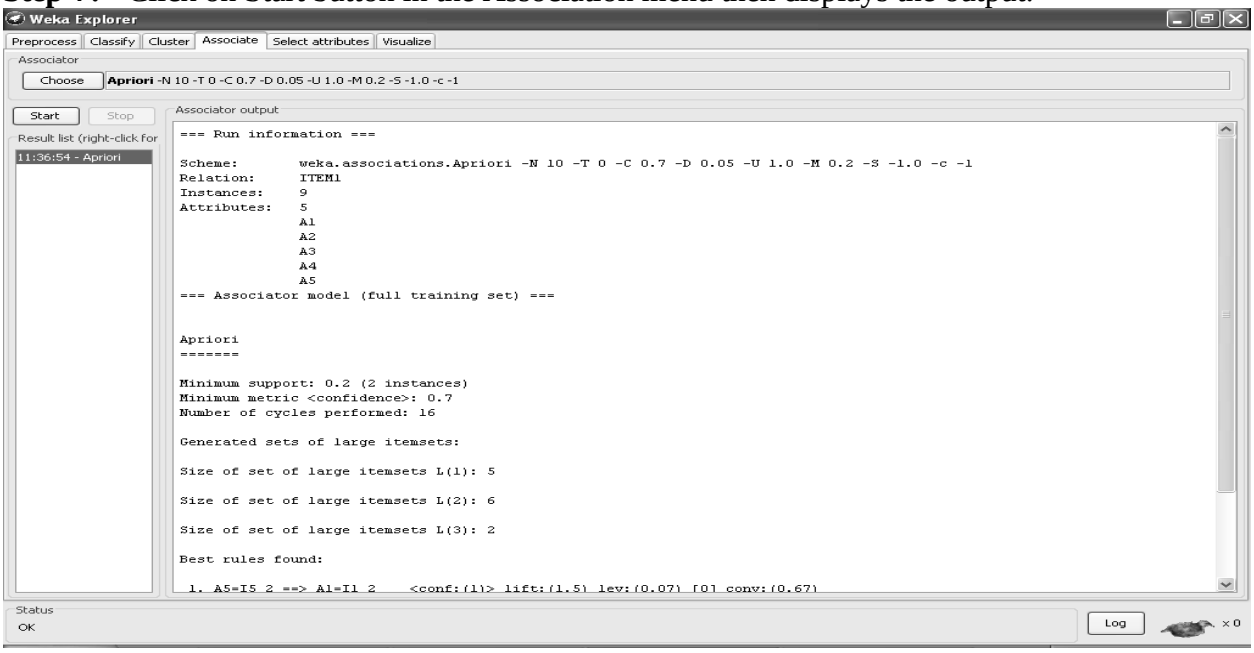


Step-6: - Click on Apriori field to configure the various options.

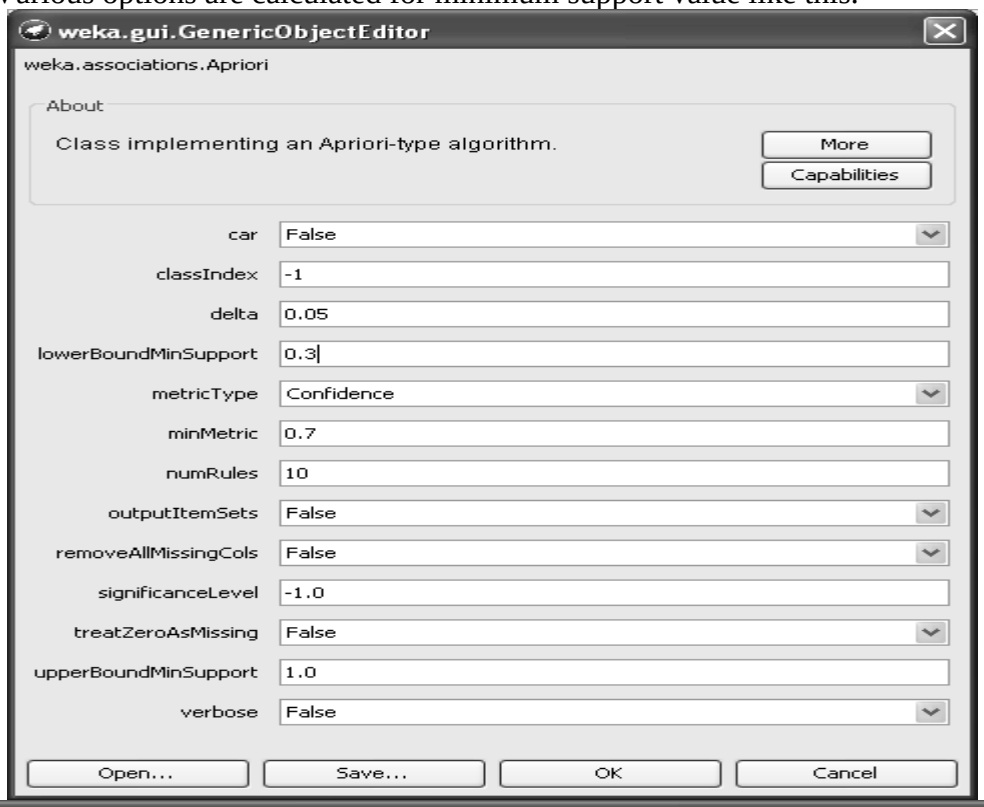


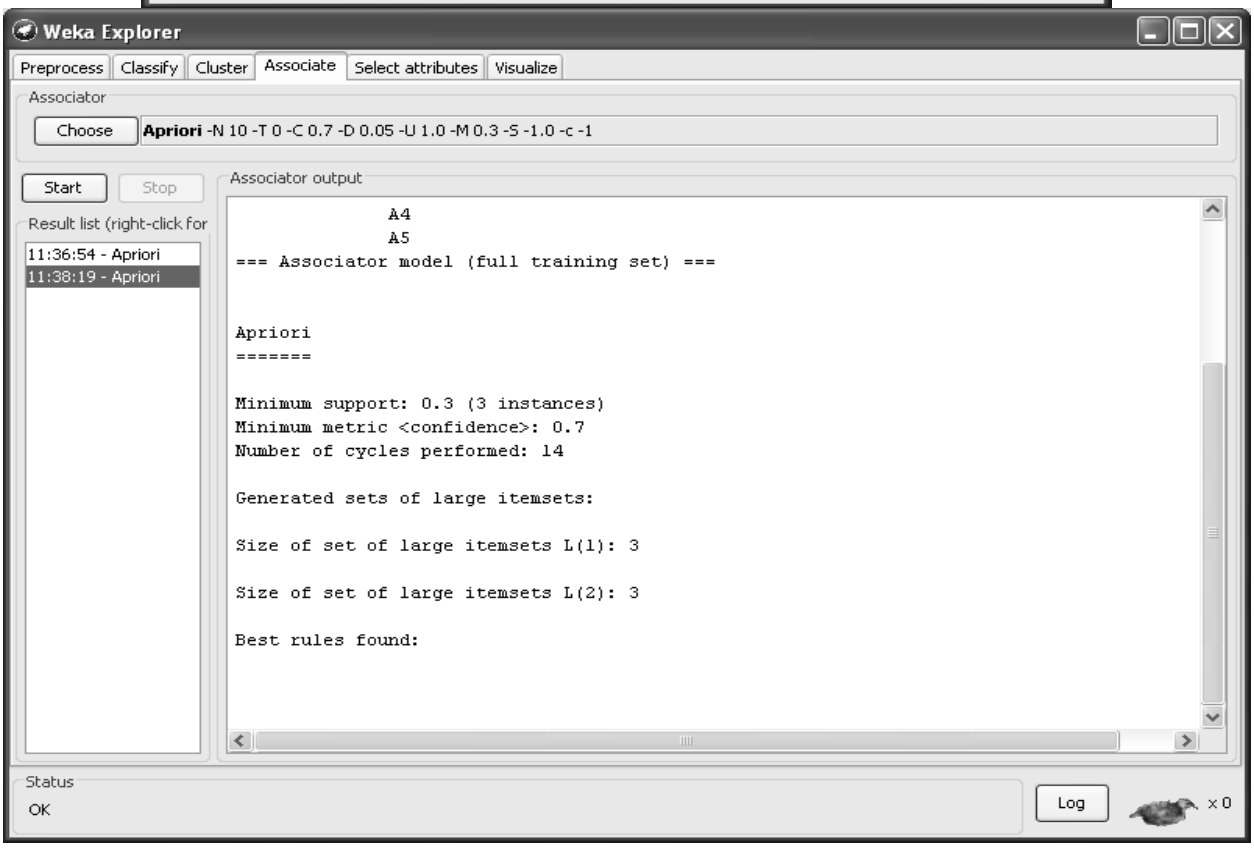
Click on OK button.

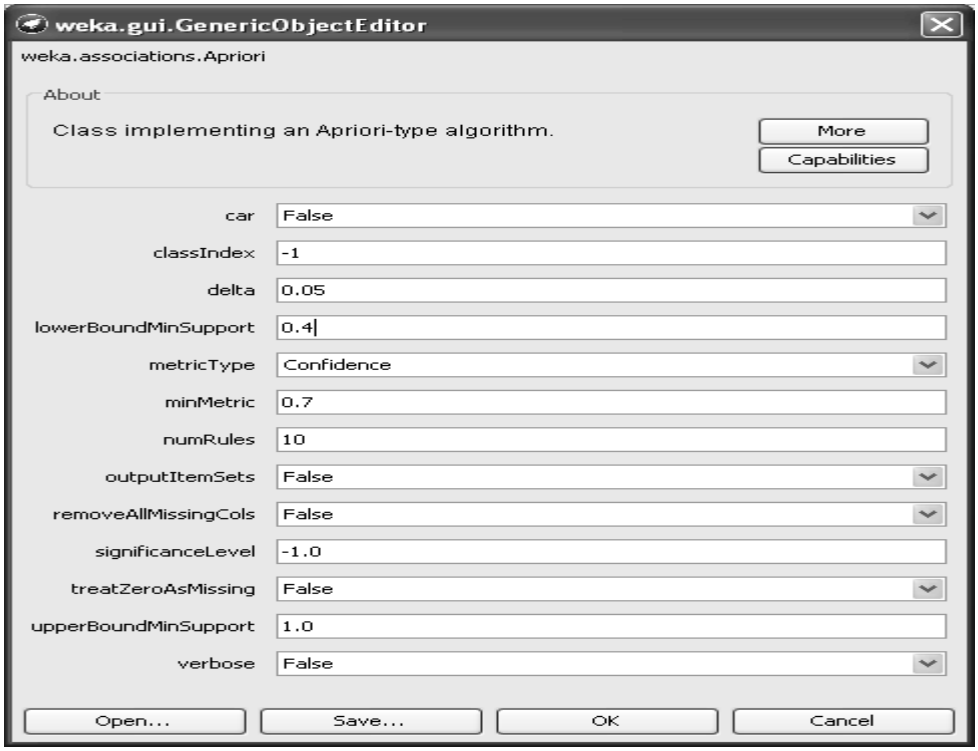
Step-7: - Click on Start button in the Association menu then displays the output.

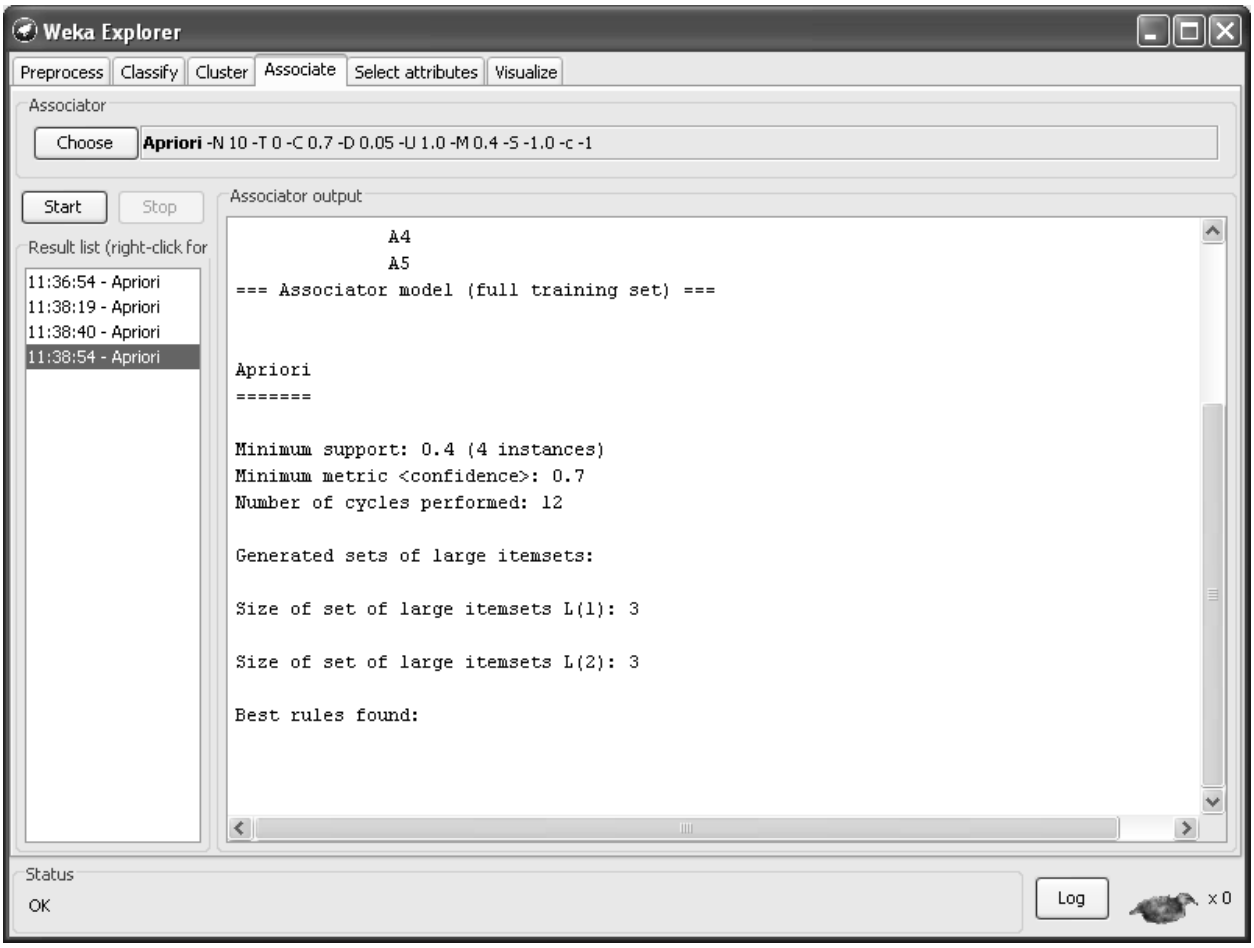


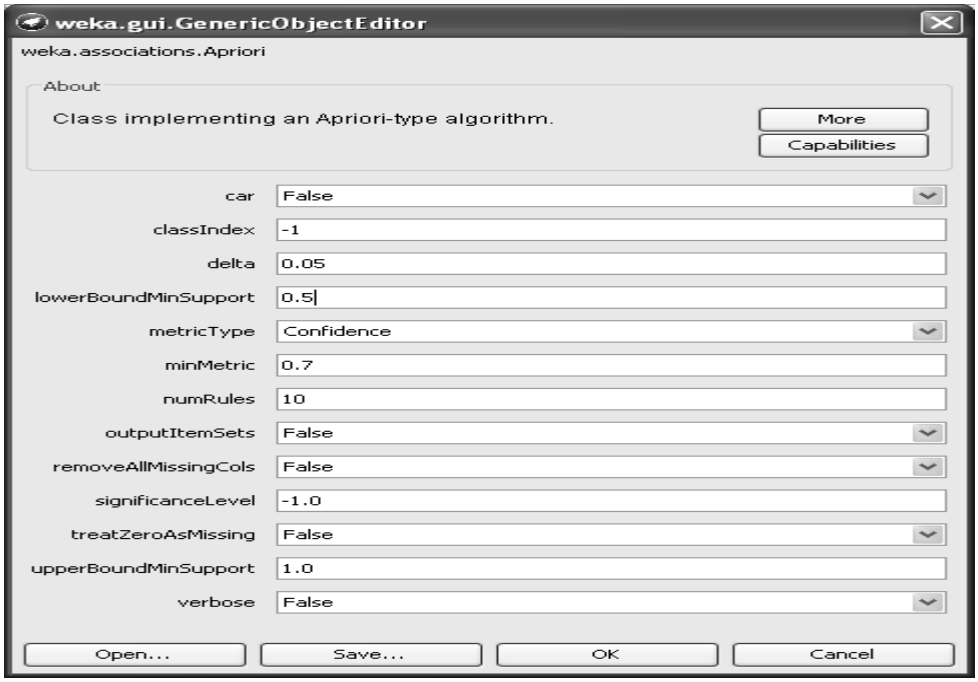
Step-8: - Various options are calculated for minimum support value like this.

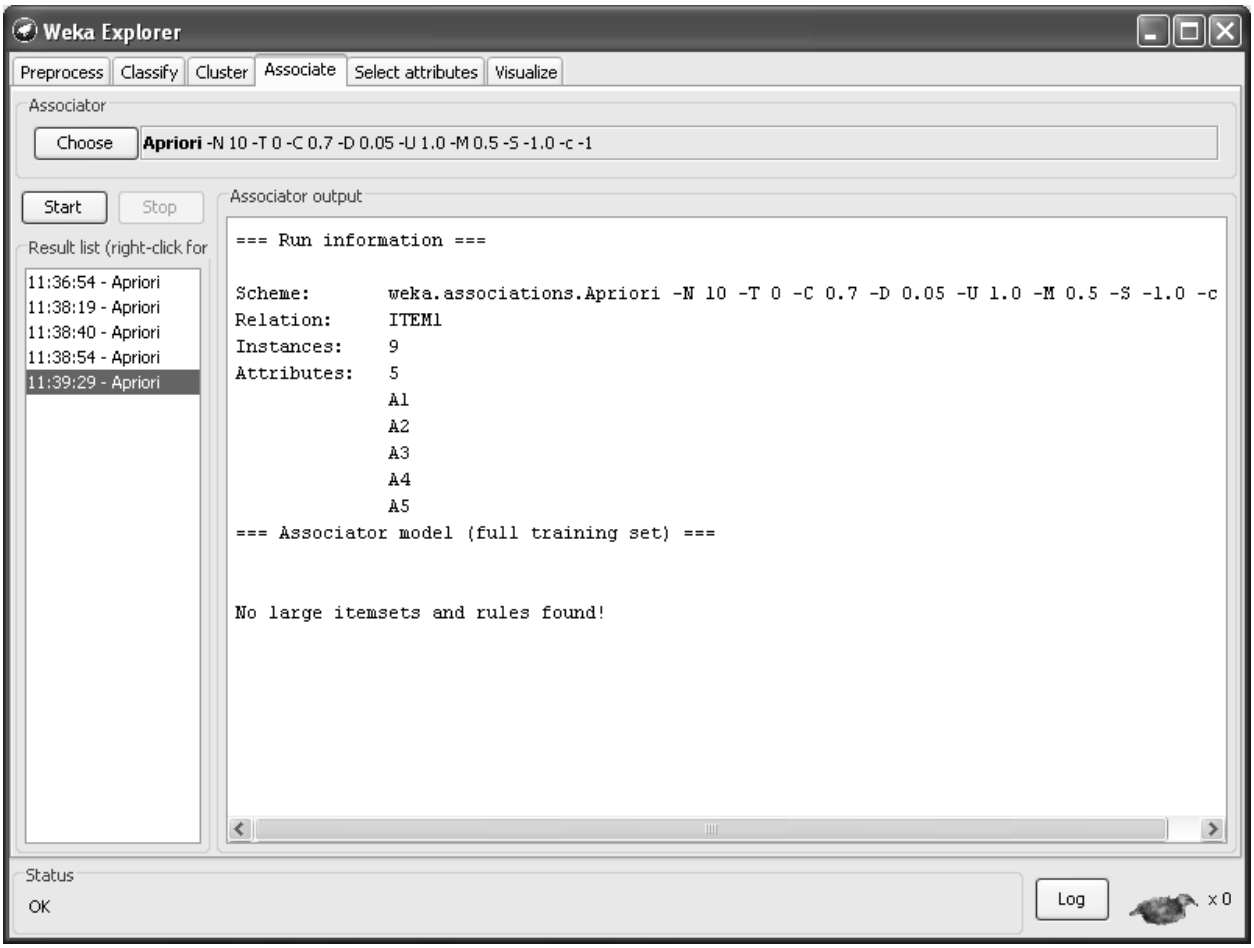





weka.gui.GenericObjectEditor
weka.associations.Apriori
About
Class implementing an Apriori-type algorithm.
More
Capabilities
car
False
classIndex
-1
delta
0.05
lowerBoundMinSupport
0.4
metricType
Confidence
minMetric
0.7
numRules
10
outputItemSets
False
removeAllMissingCols
False
significanceLevel
-1.0
treatZeroAsMissing
False
upperBoundMinSupport
1.0
verbose
False
Open...
Save...
OK
Cancel

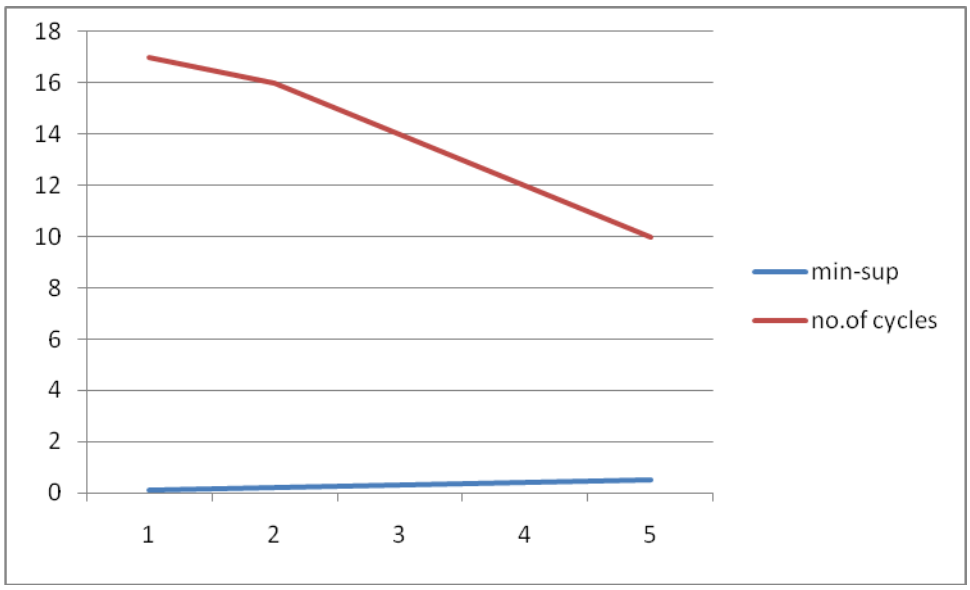

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.7 -D 0.05 -U 1.0 -M 0.4 -S -1.0 -c -1
Start
Stop
Result list (right-click for
11:36:54 - Apriori
11:38:19 - Apriori
11:38:40 - Apriori
11:38:54 - Apriori
Associator output
A4
A5
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.4 (4 instances)
Minimum metric <confidence>: 0.7
Number of cycles performed: 12
Generated sets of large itemsets:
Size of set of large itemsets L(1): 3
Size of set of large itemsets L(2): 3
Best rules found:
Status
OK
Log
x 0


weka.gui.GenericObjectEditor
weka.associations.Apriori
About
Class implementing an Apriori-type algorithm.
More
Capabilities
car
False
classIndex
-1
delta
0.05
lowerBoundMinSupport
0.5
metricType
Confidence
minMetric
0.7
numRules
10
outputItemSets
False
removeAllMissingCols
False
significanceLevel
-1.0
treatZeroAsMissing
False
upperBoundMinSupport
1.0
verbose
False
Open...
Save...
OK
Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.7 -D 0.05 -U 1.0 -M 0.5 -S -1.0 -c -1
Start
Stop
Result list (right-click for
11:36:54 - Apriori
11:38:19 - Apriori
11:38:40 - Apriori
11:38:54 - Apriori
11:39:29 - Apriori
Associator output
=== Run information ===
Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.7 -D 0.05 -U 1.0 -M 0.5 -S -1.0 -c
Relation: ITEM1
Instances: 9
Attributes: 5
A1
A2
A3
A4
A5
=== Associator model (full training set) ===
No large itemsets and rules found!
Status
OK
Log
x 0

Graph for Apriori Algorithm: -

min-sup	no.of cycles
0.1	17
0.2	16
0.3	14
0.4	12
0.5	10

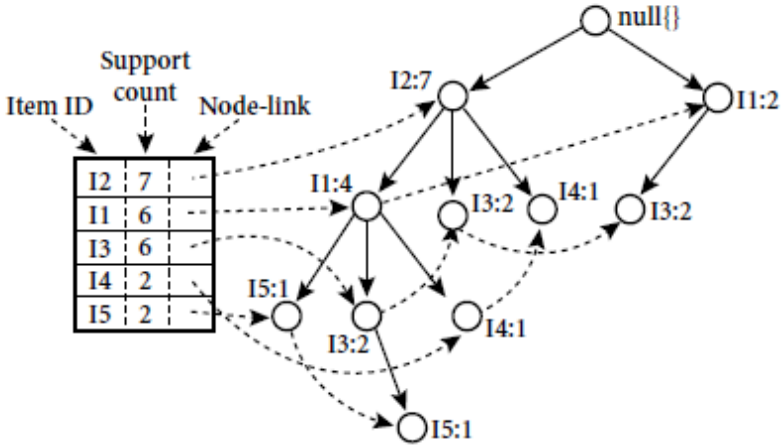


FP-growth.

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set or *list* is denoted *L*. Thus, we have *L* =I2:7 , I1: 6, I3: 6, I4: 2, I5: 2.

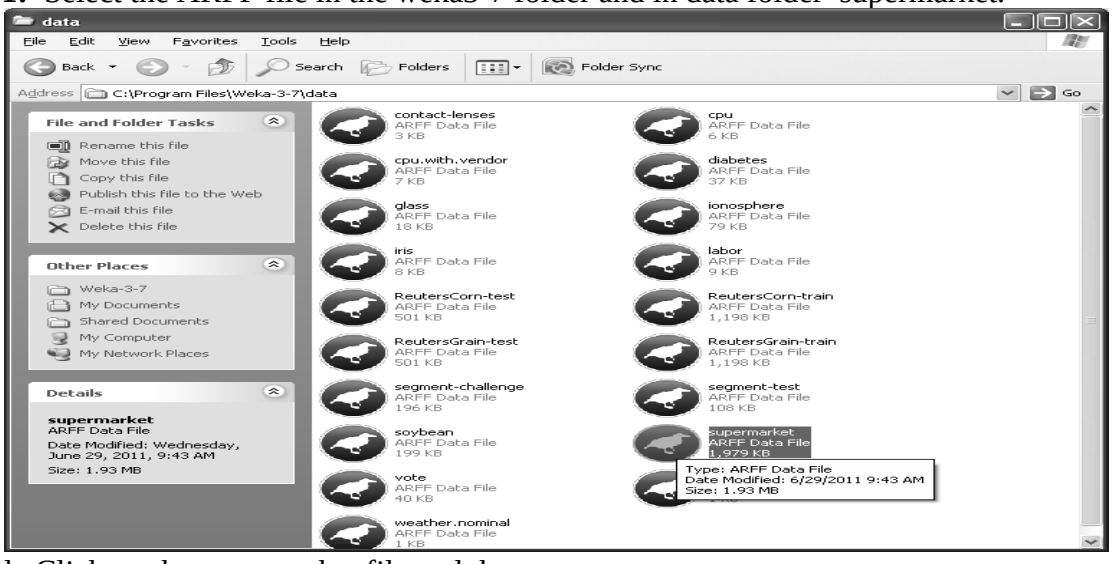
An FP-tree is then constructed as follows. First, create the root of the tree, labeled with "null." Scan database *D* a second time. The items in each transaction are processed in *L* order (i.e., sorted according to descending support count), and a branch is created for each transaction. For example, the scan of the first transaction, "T100: I1, I2, I5," which contains three items (I2, I1, I5 in *L* order), leads to the construction of the first branch of the tree with three nodes, I2: 1, I1:1, and I5: 1, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1. The second transaction, T200, contains the items I2 and I4 in *L* order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common prefix, I2, with the existing path for T100. Therefore,we instead increment the count of the I2

node by 1, and create a newnode, I4: 1,which is linked as a child of I2: 2. In general,when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

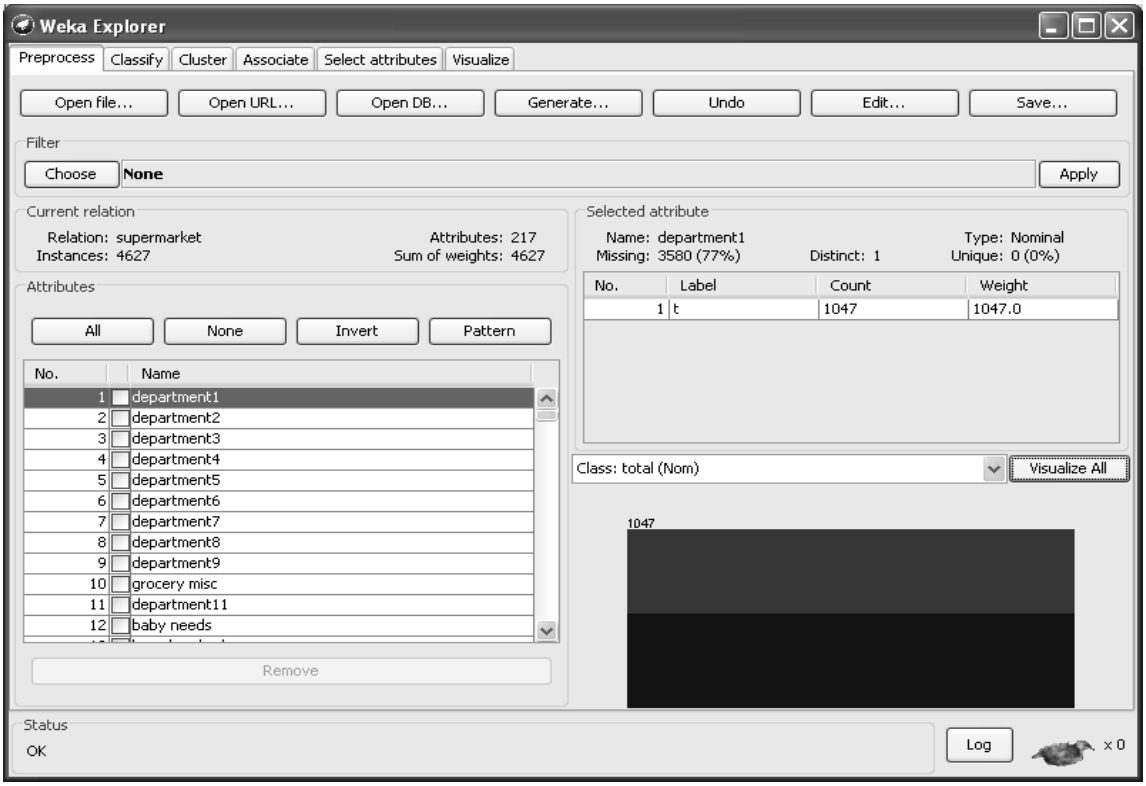


Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns Generated
I5	{{I2, I1: 1}, {I2, I1, I3: 1}}	⟨I2: 2, I1: 2⟩	{I2, I5: 2}, {I1, I5: 2}, {I2, I1, I5: 2}
I4	{{I2, I1: 1}, {I2: 1}}	⟨I2: 2⟩	{I2, I4: 2}
I3	{{I2, I1: 2}, {I2: 2}, {I1: 2}}	⟨I2: 4, I1: 2⟩, ⟨I1: 2⟩	{I2, I3: 4}, {I1, I3: 4}, {I2, I1, I3: 2}
I1	{{I2: 4}}	⟨I2: 4⟩	{I2, I1: 4}

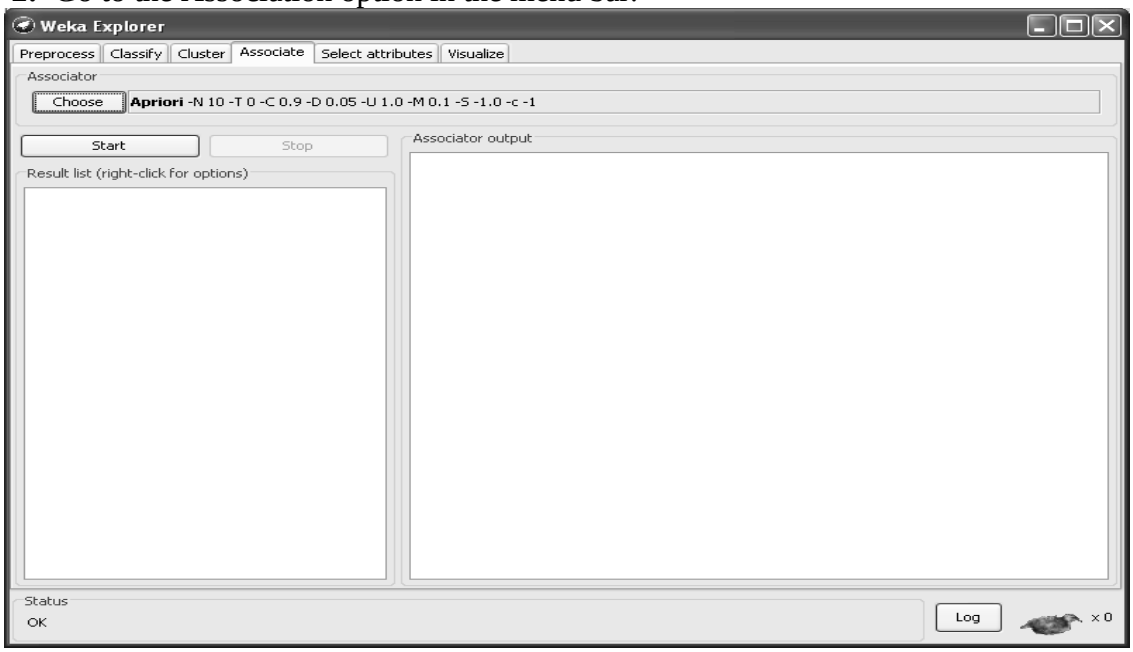
Step 1: Select the ARFF file in the weka3-7 folder and in data folder supermarket.



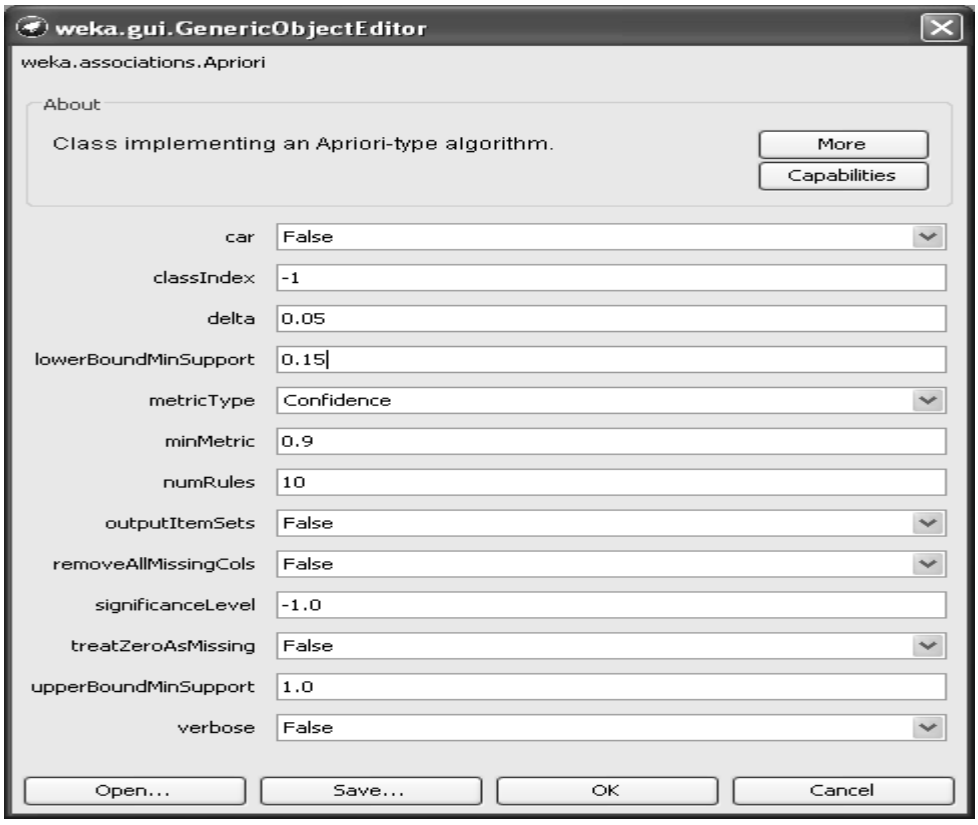
Double Click on the supermarket file and then open.



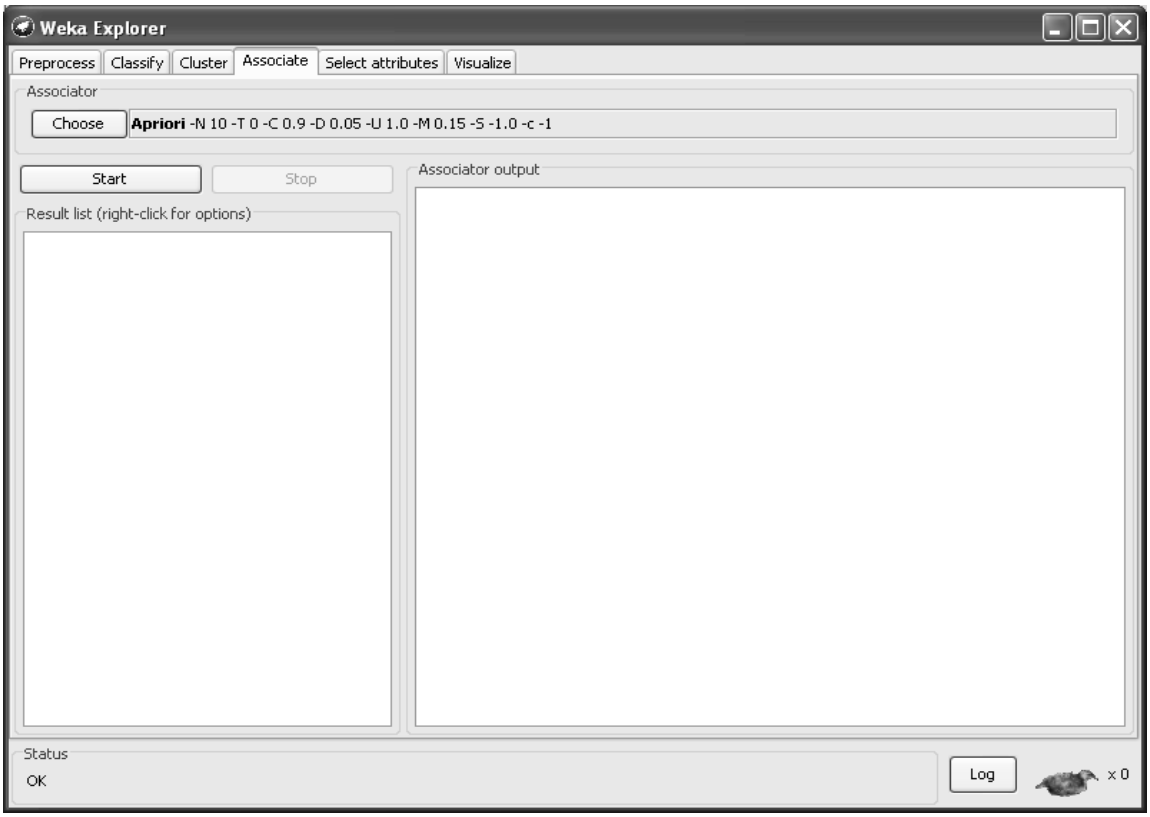
Step 2: Go to the Association option in the menu bar.

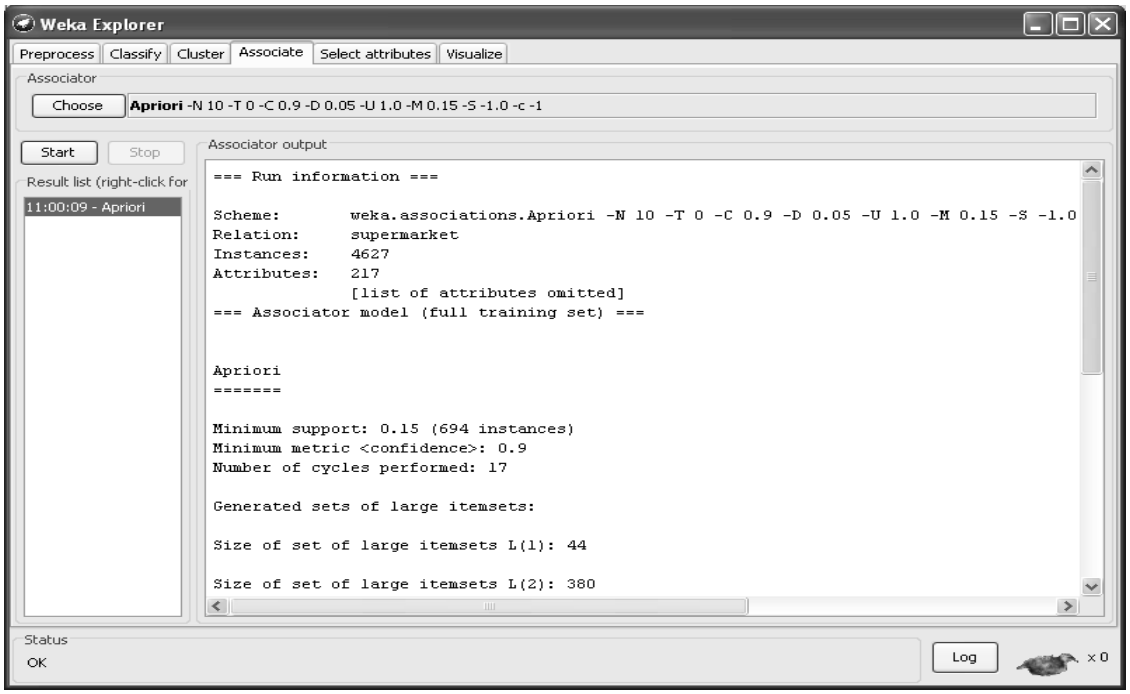


Step 3: Click on choose button and configure the various options.

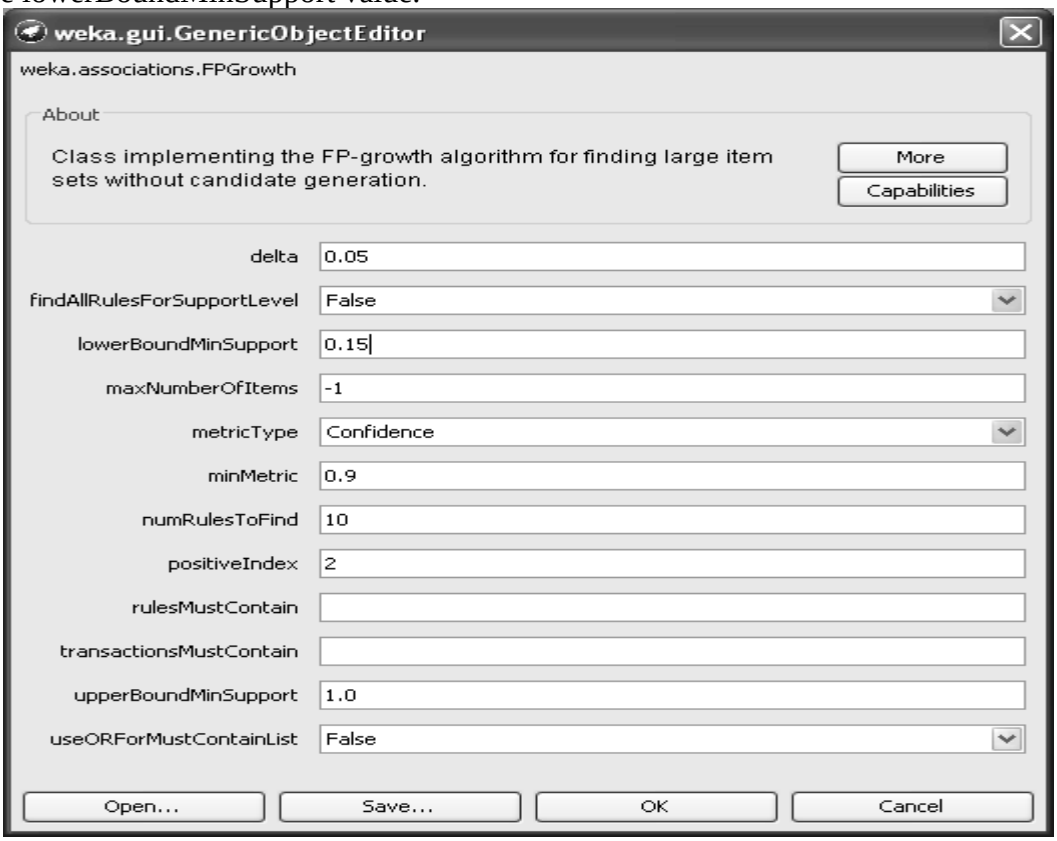


Click on OK button.

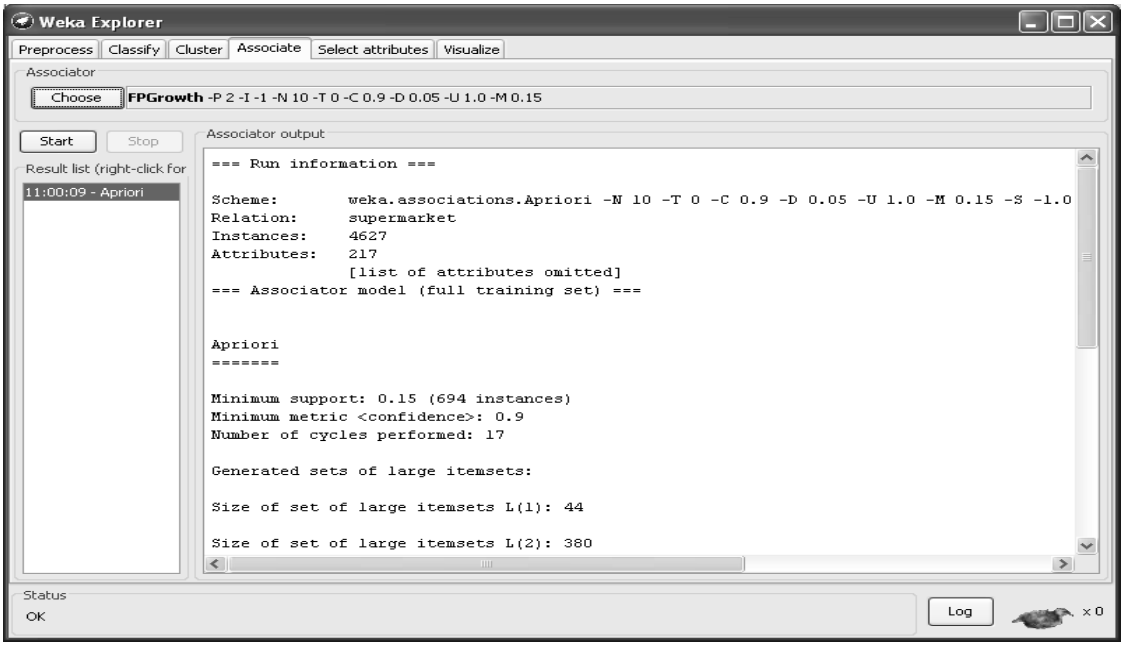




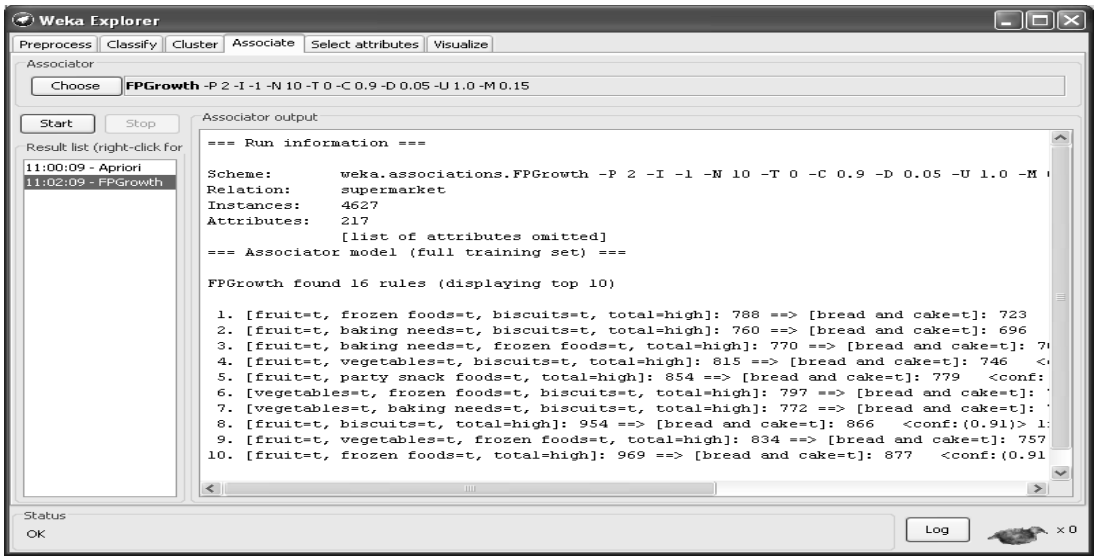
Give the lowerBoundMinSupport value.



Step 4: Click on choose button and then select the FP-growth Algorithm and configure to various minimum support values.



Click on start



min-sup	apriori	fp-growth
0.15	0.44	0.2
0.2	0.3	0.1
0.3	0.2	0.1
0.4	0.15	0.1
0.5	0.11	0.1
0.6	0.8	0.1
0.7	0.6	0.1
0.8	0.5	0.1
0.9	0.3	0.1

VIVA-QUESTIONS

1. Market-basket problem was formulated by_____.
2. The Market Basket Analysis is a typical example of_____
3. An itemset satisfying the support criterion is known as:
4. In Apriori algorithm, for generating k-item sets, we use:
5. Mathematically, Support({X} -->{Y}) is
6. All nonempty subset of a frequent itemset must also be frequent is _____ property
7. FP-Growth algorithm is based on _____.
8. The formula for Support(A ⇒ B)=
9. The formula for Confidence (A ⇒ B)=
10. Apriori method mines the frequent itemsets without candidate generation

Experiment No. 5: Demonstrate performing regression on data sets

i. Load numeric dataset into WEKA and build linear regression model.

ii. Explore simple linear regression technique that only looks at one variable

Linear Regression

Straight-line regression analysis involves a response variable, *y*, and a single predictor variable, *x*. It is the simplest form of regression, and models *y* as a linear function of *x*. That is,

$$y = b+wx;$$

where the variance of *y* is assumed to be constant, and *b* and *w* are regression coefficients specifying the Y-intercept and slope of the line, respectively. The regression coefficients, *w* and *b*, can also be thought of as weights, so that we can equivalently write,

$$y = w0+w1x$$

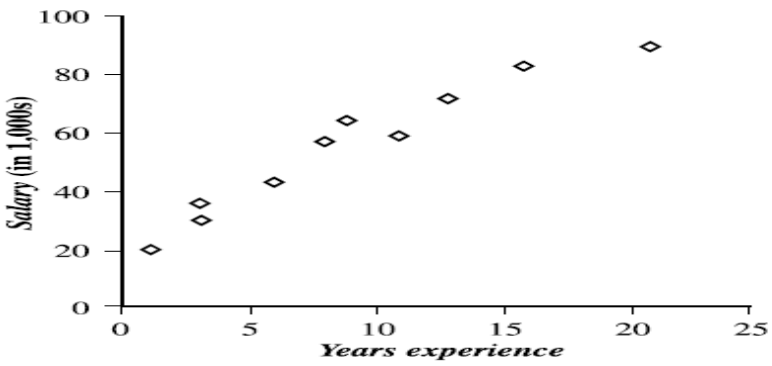
These coefficients can be solved for by the method of least squares, which estimates the best-fitting straight line as the one that minimizes the error between the actual data and the estimate of the line.

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2}$$

$$w_0 = \bar{y} - w_1\bar{x}$$

where *x* is the mean value of *x*1, *x*2, : : : , *x*j*D*j, and *y* is the mean value of *y*1, *y*2, : : : , *y*j*D*j. The coefficients *w*0 and *w*1 often provide good approximations to otherwise complicated regression equations.

x years experience	*y* salary (in $1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83



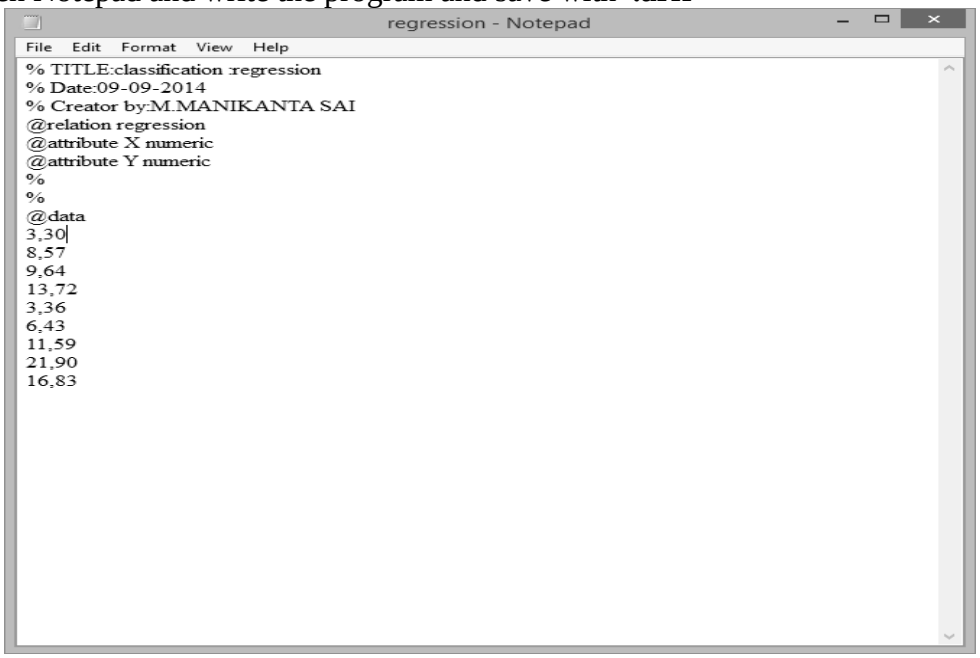
Given the above data, we compute x = 9:1 and y = 55:4. Substituting these values into Equations (6.50) and (6.51), we get

$w1$ = 3:5

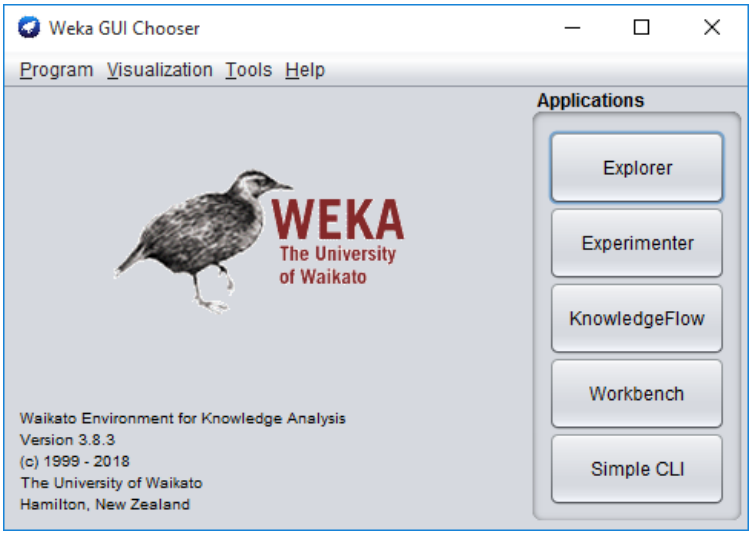
$w0$ = 23:6

Thus, the equation of the least squares line is estimated by y = 23:6+3:5x. Using this equation, we can predict that the salary of a college graduate with, say, 10 years of experience is $58,600.

Step1: Open Notepad and write the program and save with **.arff**

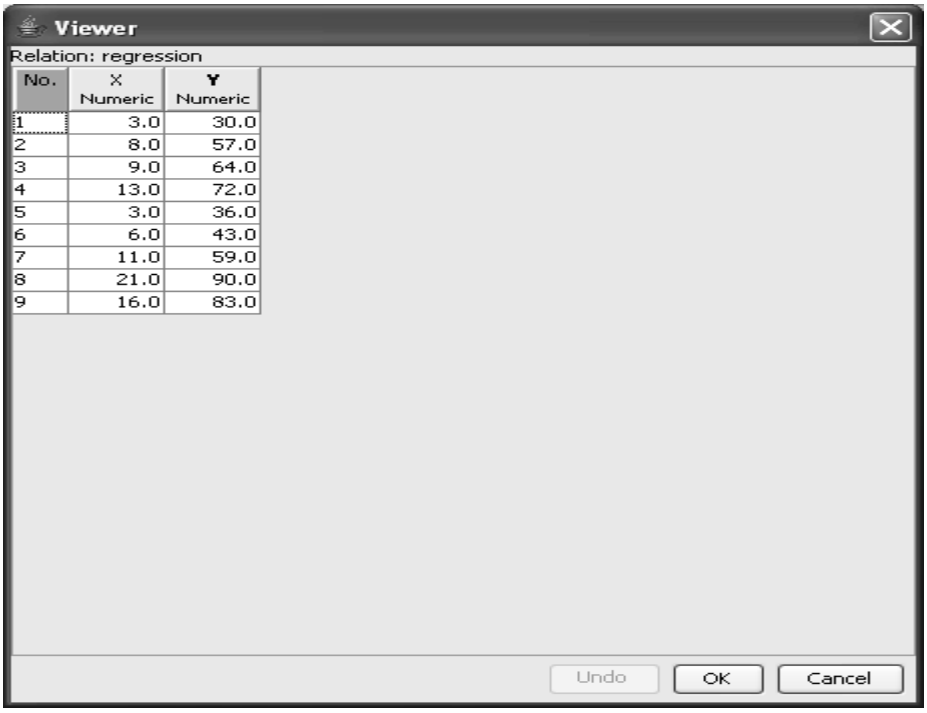


Step2: Go to Start menu and select Weka 3.8.3, on that select Weka 3.8(with console).select Knowledge Flow.

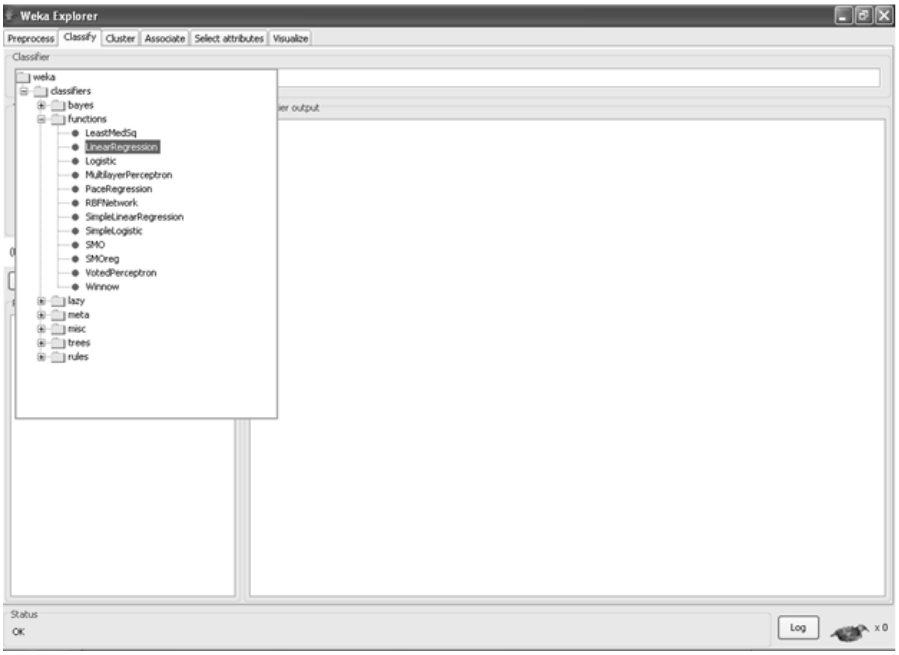


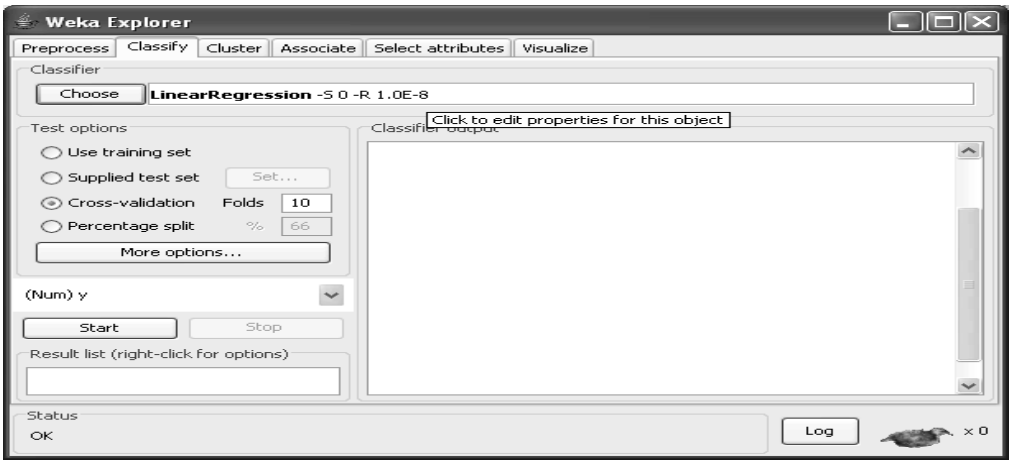
Step 3: In Weka Select Explorer window select **preprocessor** tab and select the **open file** for selection of desired file.

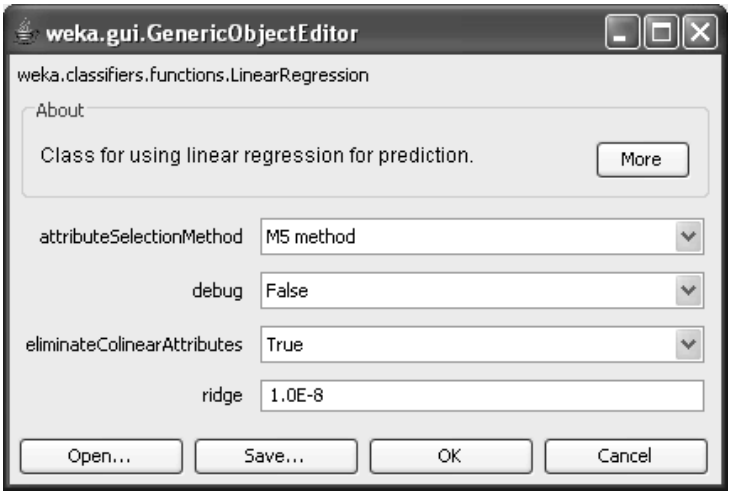
Step 4: In the Preprocessor menu select open file and select Edit



Step 5: Go to classify menu click choose classifiers on that functions and that Linear Regression.

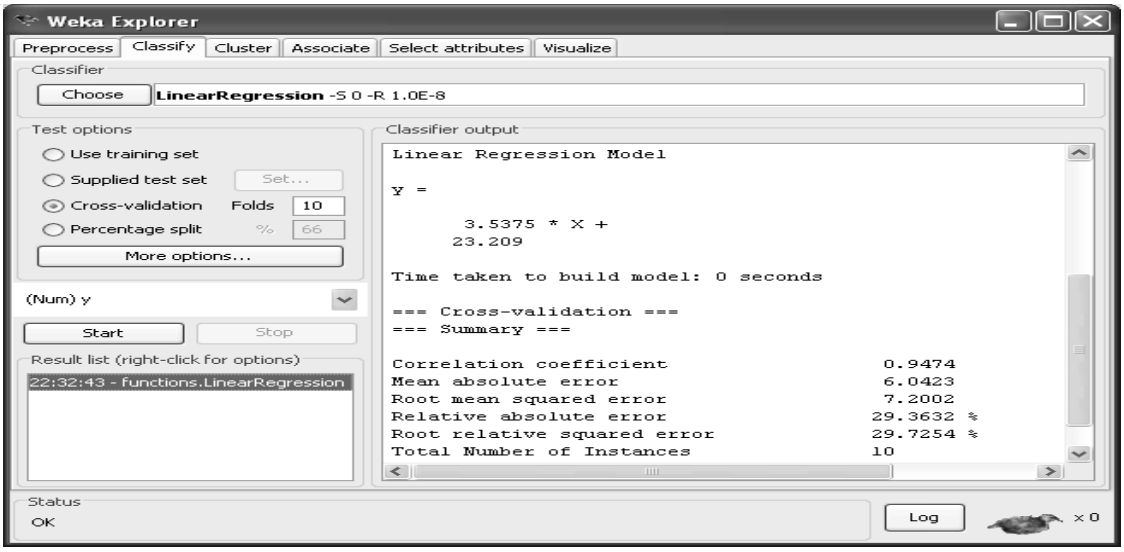






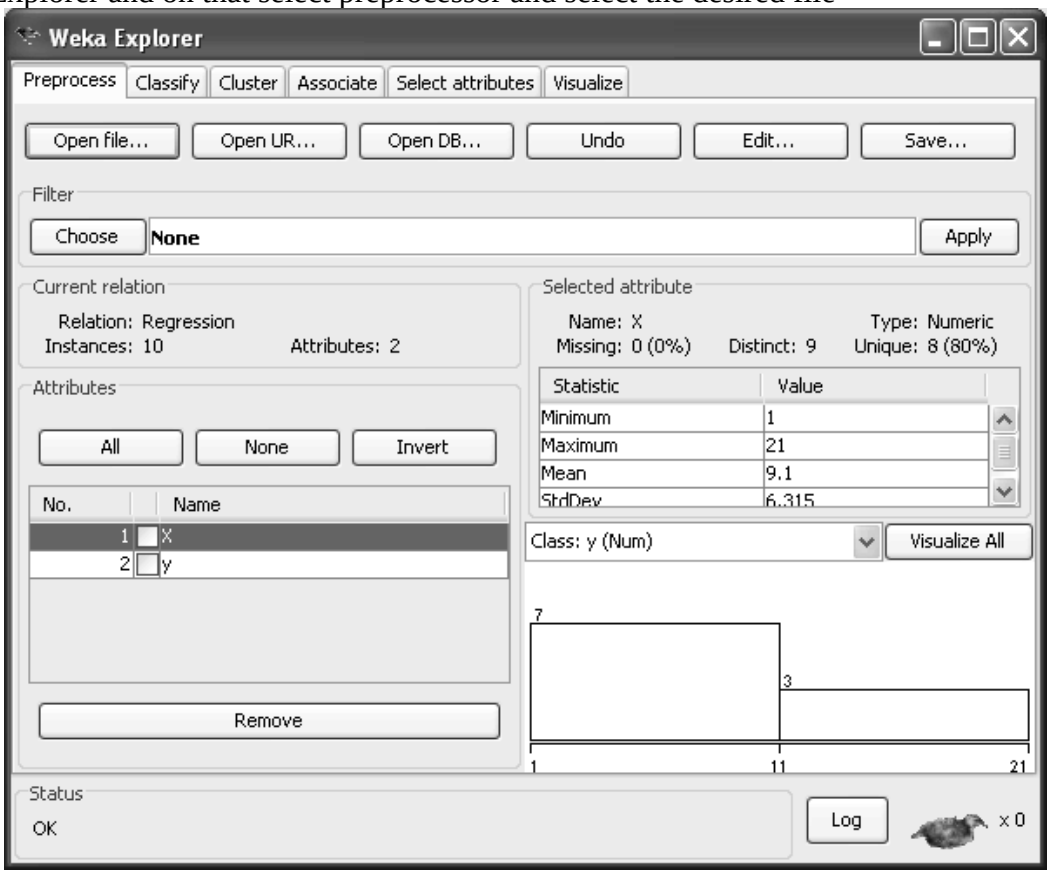
When we click on the Linear Regression option on the above the dialog box appears then click OK,and click on start button.

Output:



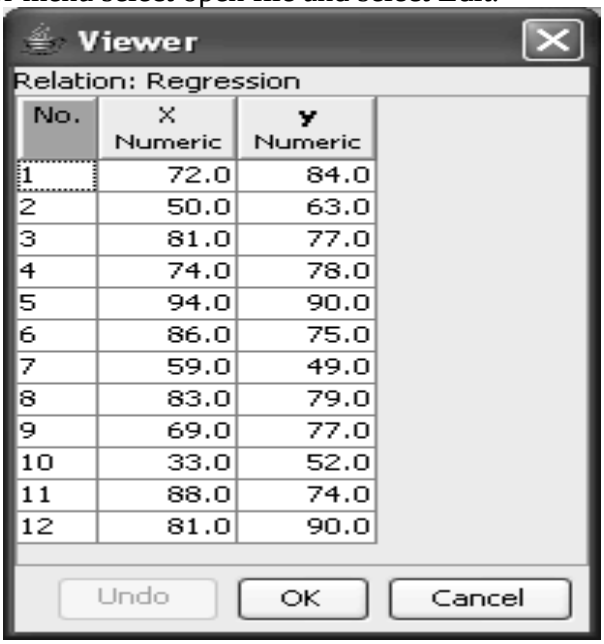
Step1: Open Notepad and write the program and save with **.arff**

Step2: Go to Start menu and select Weka 3.4.11, on that select Weka 3.4(with console) In Weka Select Explorer and on that select preprocessor and select the desired file

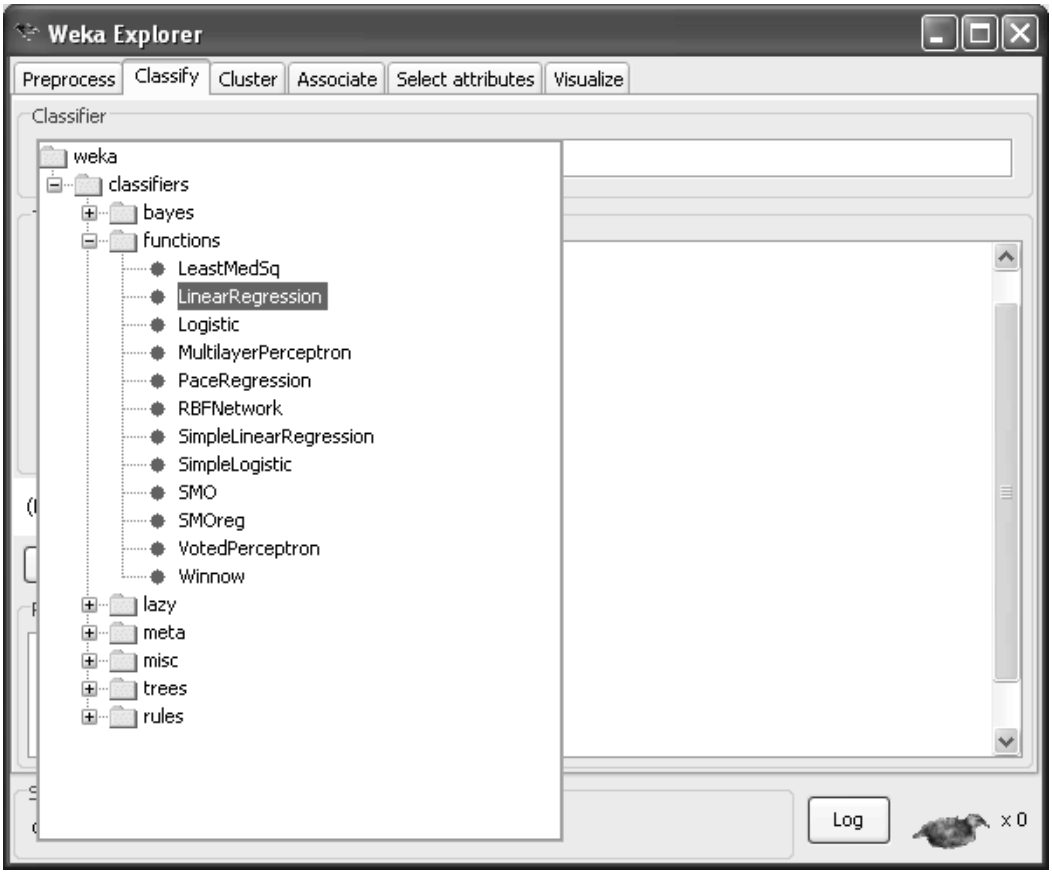


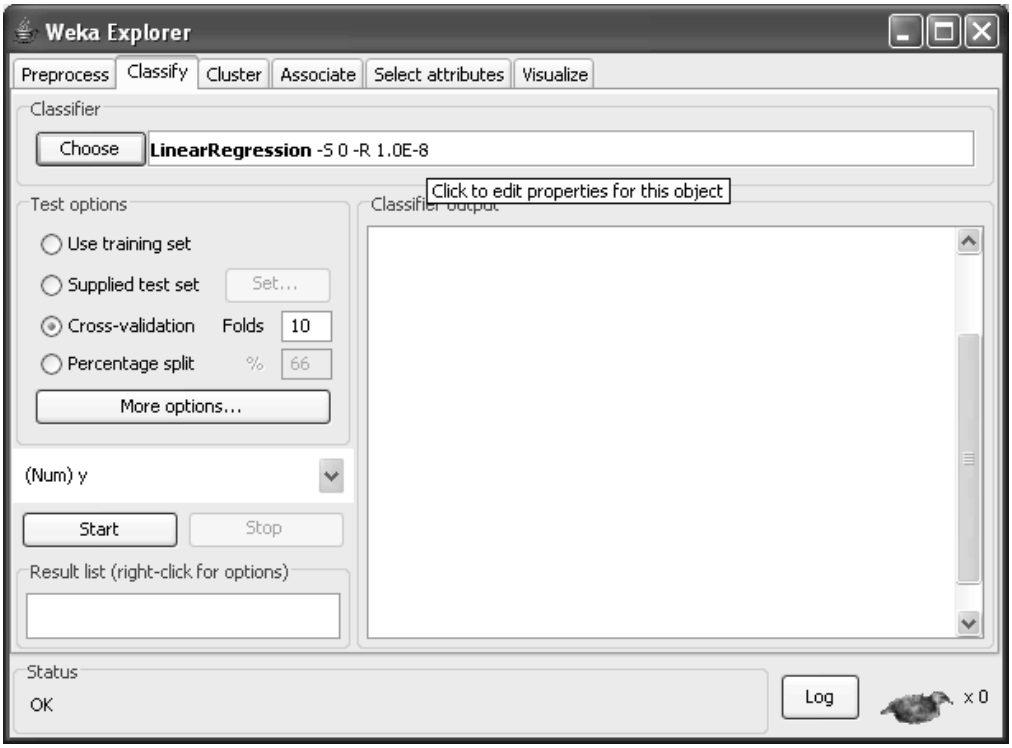
In the Preprocess open file is present on that select the desired file

Step3: In the Preprocessor menu select open file and select Edit.

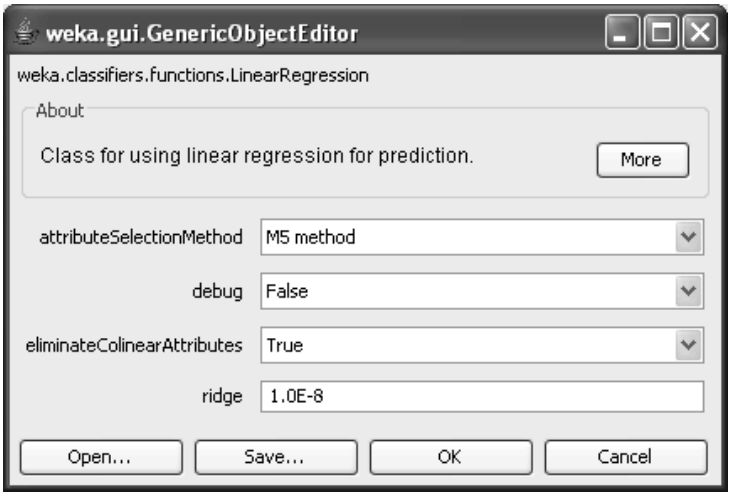


Step4: Go to classify menu click choose classifiers on that functions and that Linear Regression



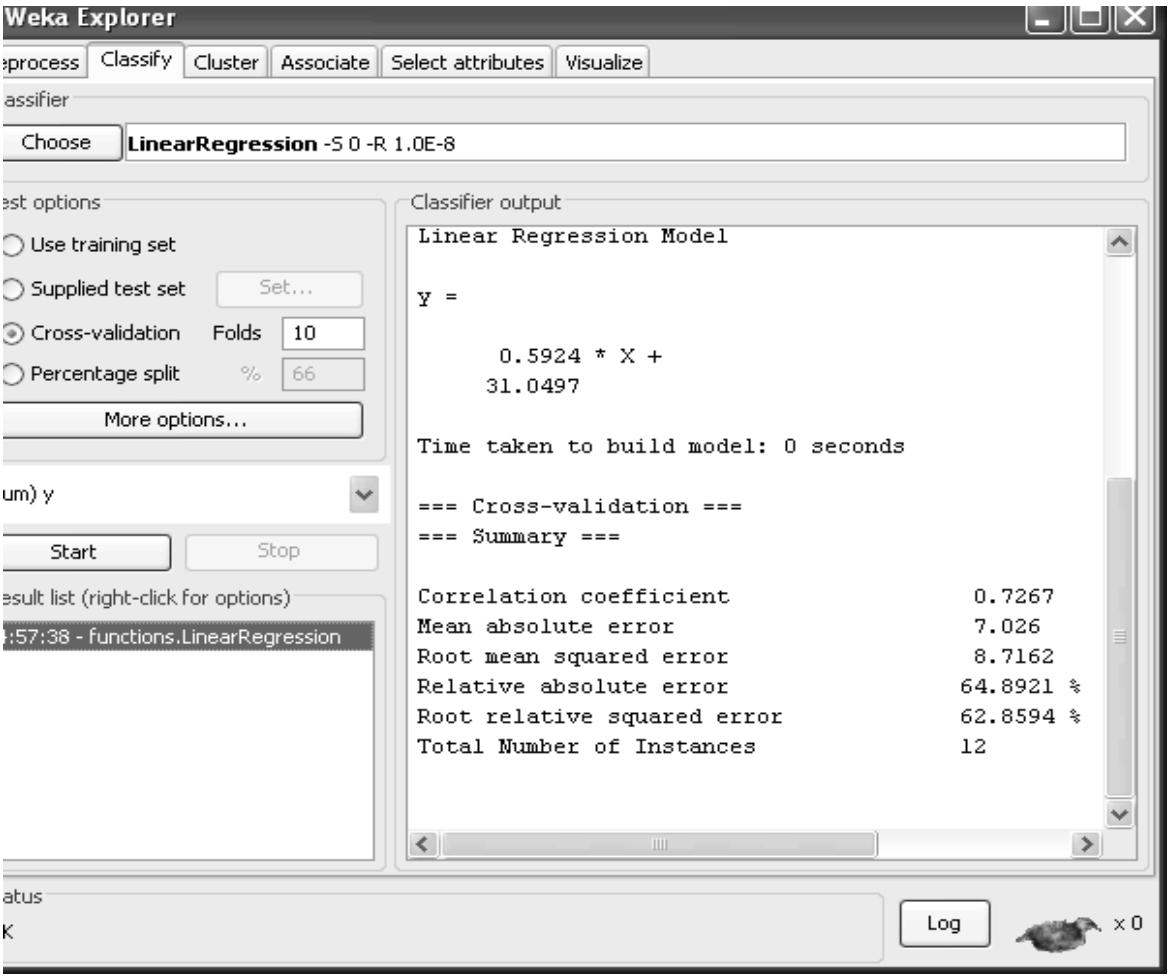


Step 5:When we click on the Linear Regression option on the above the dialog box appears then click O.K



Step 6: And then click OK. button, and then click start in the classify menu present in Weka Explorer window.

Output:



VIVA-QUESTIONS

1. The process of constructing a mathematical model or function that can be used to predict or determine one variable by another variable is called.
2. In the regression equation Y = 12 - 21X, the slope is
3. In the regression equation Y = 55.65 + 0.50X, the intercept is
4. Which of the following methods do we use to find the best fit line for data in Linear Regression
5. In the mathematical Equation of Linear Regression $Y = \beta1 + \beta2X + \epsilon$, ($\beta1$, $\beta2$) refers to
6. A regression between foot length (response variable in cm) and height (explanatory variable in inches) for 33 students resulted in the following regression equation.
   $\hat{y} = 10.9 + 0.23\ x$
   One student in the sample was 100 inches tall with a foot length of 29 cm. What is the predicted foot length for this student?
7. Considerx1, x2 to be the independent variables and y the dependent variable, which of the following
   represents a linear regression model?
8. The sum of the squared residuals is called as:
9. The value of the coefficient of determination ranges between
10. How many coefficients do you need to estimate in a simple linear regression model (One independent variable)?

Experiment No. 6: Demonstrate performing SVM classification on data sets

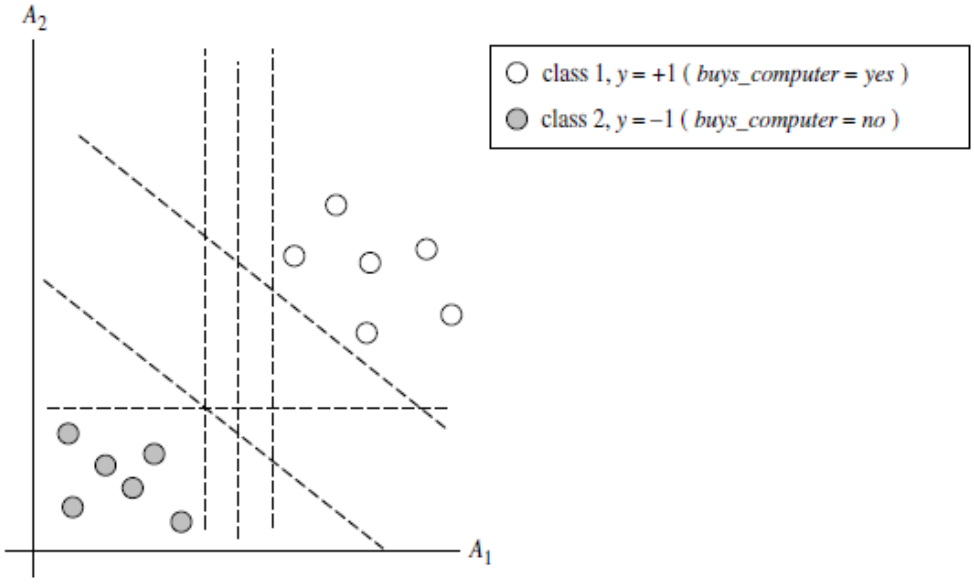
i. Load each dataset into WEKA and run proximal SVM to find a classifiers.
ii. Load each dataset into WEKA and run liner classifier using linear Programming.

SVM

Support Vector Machines, a promising new method for the classification of both linear and nonlinear data. In a nutshell, a support vector machine (or SVM) is an algorithm that works as follows. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (that is, a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane.

The Case When the Data Are Linearly Separable

Let the data set *D* be given as (***X***1, *y*1), (***X***2, *y*2), : : : , (***X***j*D*j, *y*j*D*j),where***X****i* is the setof training tupleswith associatedclass labels, *yi*. Each *yi* can take one of two values, either+1 or□1 (i.e., *yi* 2 f+1, □1g), corresponding to the classes *buys computer = yes* and *buys computer = no*, respectively.

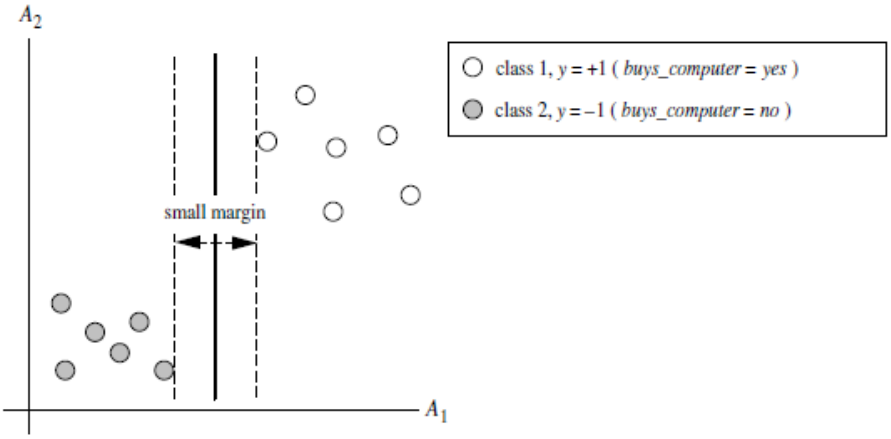


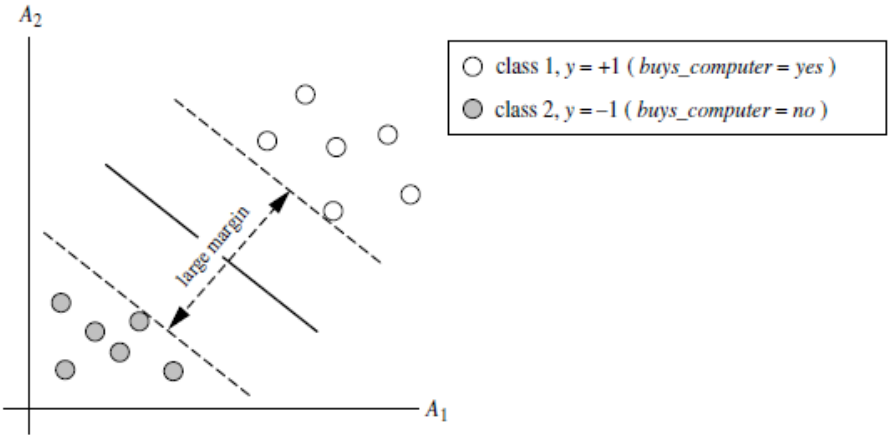
Before we get into the definition of margins, let's take an intuitive look at this figure. Both hyperplanes can correctly classify all of the given data tuples. Intuitively, however, we expect the hyperplane with the larger margin to be more accurate at classifying future data tuples than

the hyperplane with the smaller margin. This is why (during the learning or training phase), the SVM searches for the hyperplane with the largest margin, that is, the *maximum marginal hyperplane* (MMH). The associated margin gives the largest separation between classes. Getting to an informal definition of margin, we can say that the shortest distance from a hyperplane to one side of its margin is equal to the shortest distance from the hyperplane to the other side of its margin, where the "sides" of the margin are parallel to the hyperplane. When dealing with the MMH, this distance is, in fact, the shortest distance from the MMH to the closest training tuple of either class.

$$\boldsymbol{W} \cdot \boldsymbol{X} + b = 0,$$

$$w_0 + w_1 x_1 + w_2 x_2 = 0.$$



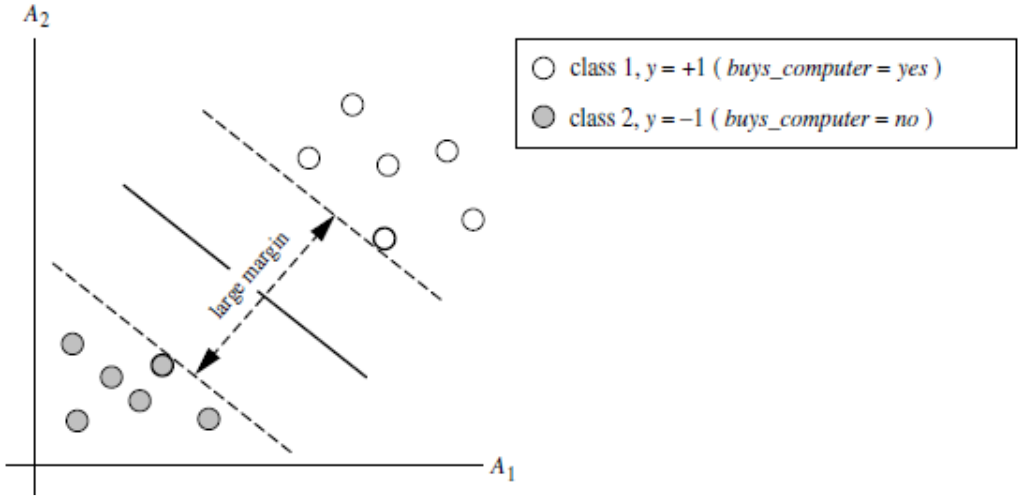


Thus, any point that lies above the separating hyperplane satisfies

$$w_0 + w_1 x_1 + w_2 x_2 > 0.$$

Similarly, any point that lies below the separating hyperplane satisfies

$$w_0 + w_1 x_1 + w_2 x_2 < 0.$$



The weights can be adjusted so that the hyperplanes defining the "sides" of the margin can be written as

$$H_1 : w_0 + w_1 x_1 + w_2 x_2 \geq 1 \text{ for } y_i = +1, \text{ and}$$
$$H_2 : w_0 + w_1 x_1 + w_2 x_2 \leq -1 \text{ for } y_i = -1.$$

[9] If $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$ then $\sqrt{\mathbf{W} \cdot \mathbf{W}} = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$.

Algorithm

 Define an optimal hyper plane: maximize margin

 Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications.

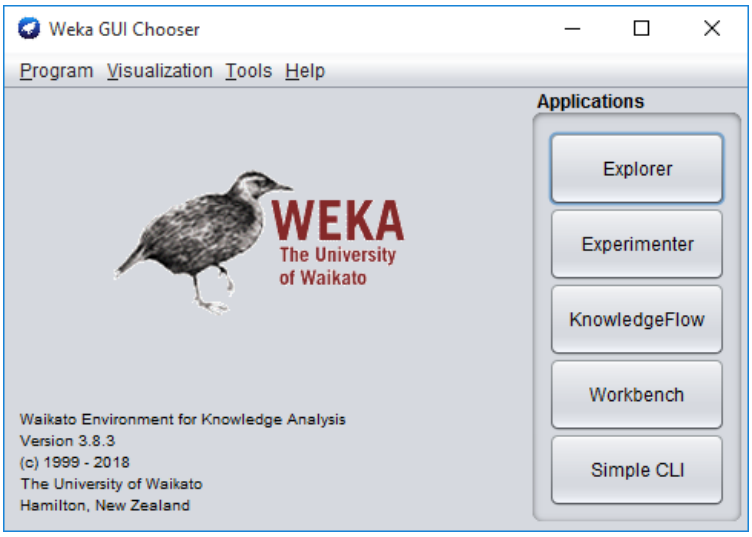
 Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space.

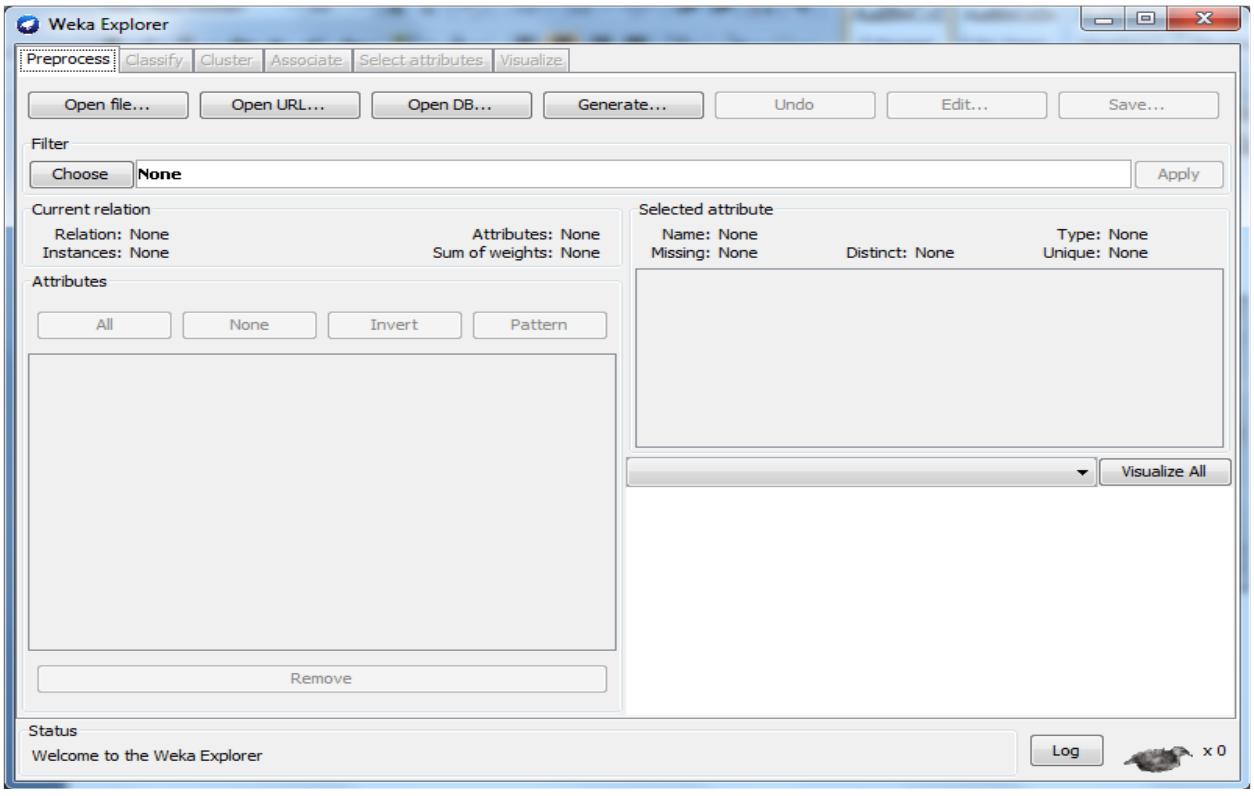
PROCEDURE:

1. We begin the experiment by loading the data (vote.arff) into weka.
2. Next we select the classify tab and click choose function button to select the Support vector machine (SMO).
3. Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button.
4. 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.

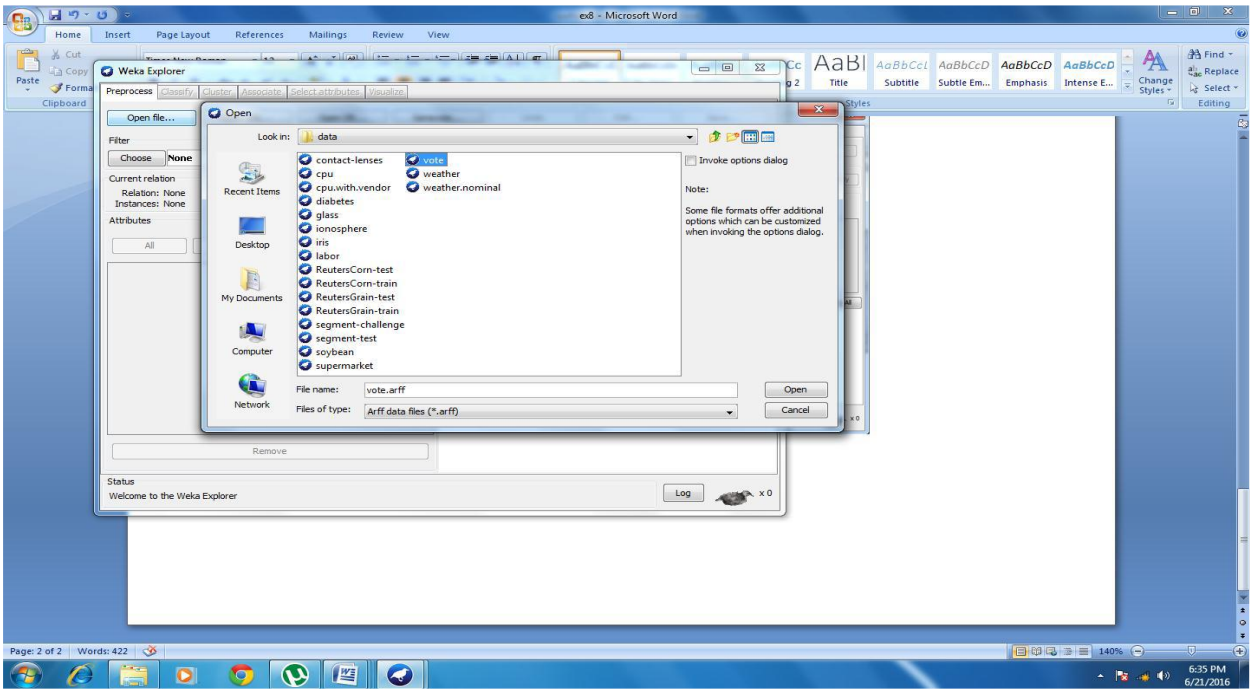
5. We now click ”start” to generate the model .the ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.
6. Note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)
7. The run information of the support vector classifier will be displayed with the correctly and incorrectly classified instances.

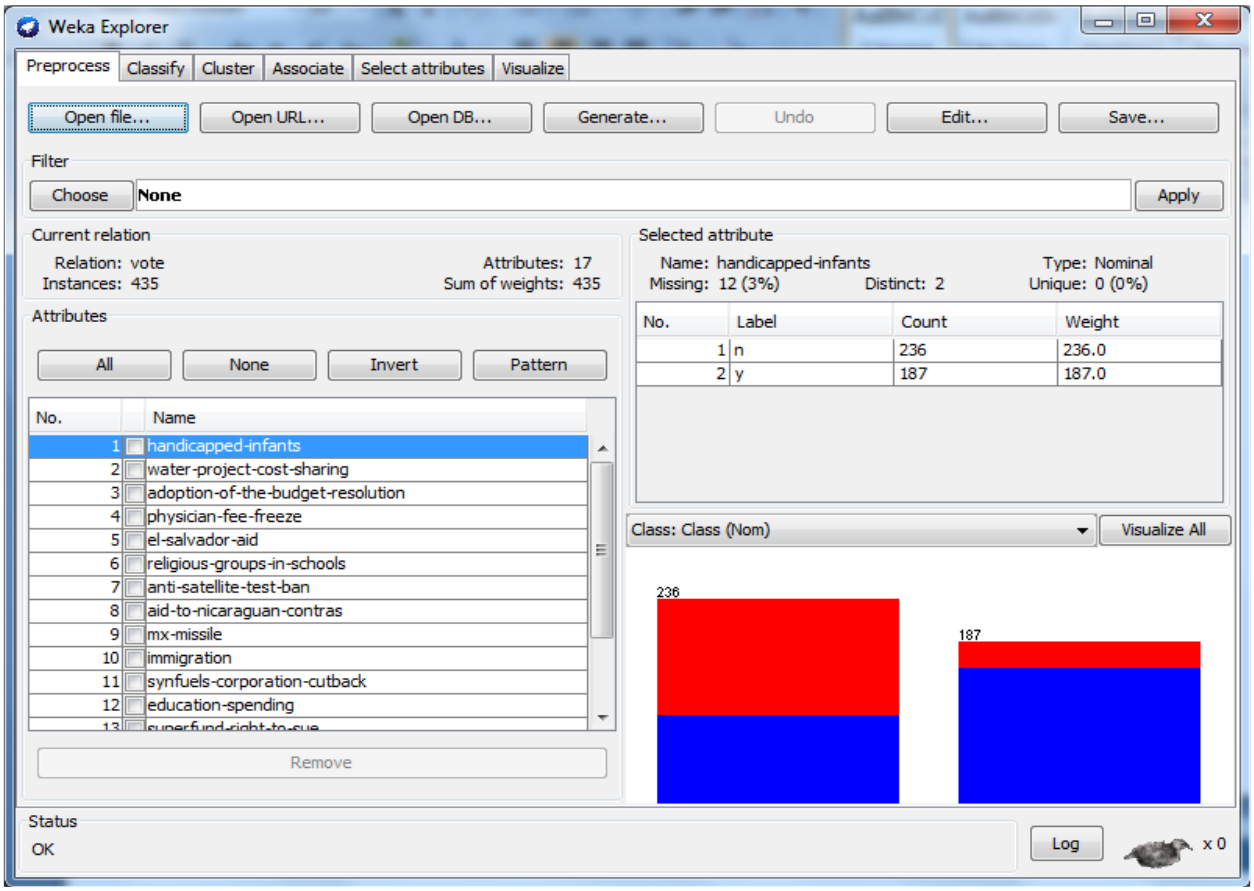
STEPS:
Open **Weka** tool – click **Explorer** in **Weka GUI Chooser**.



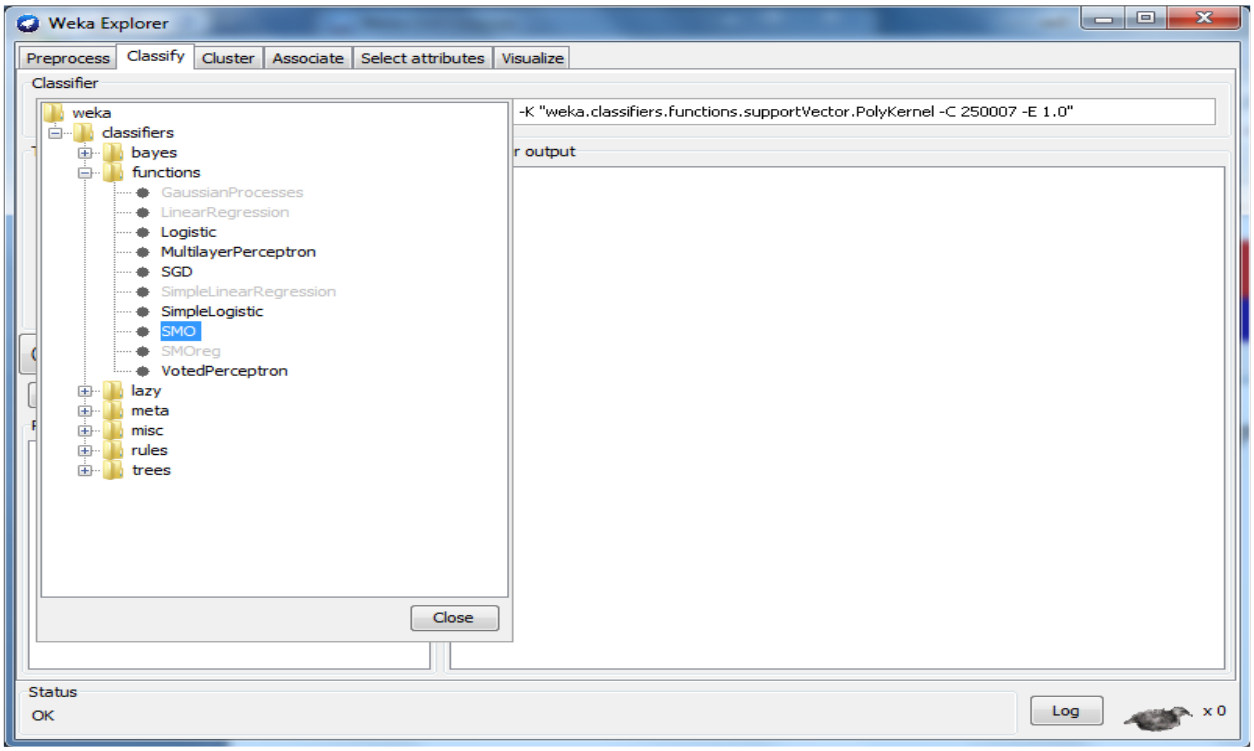


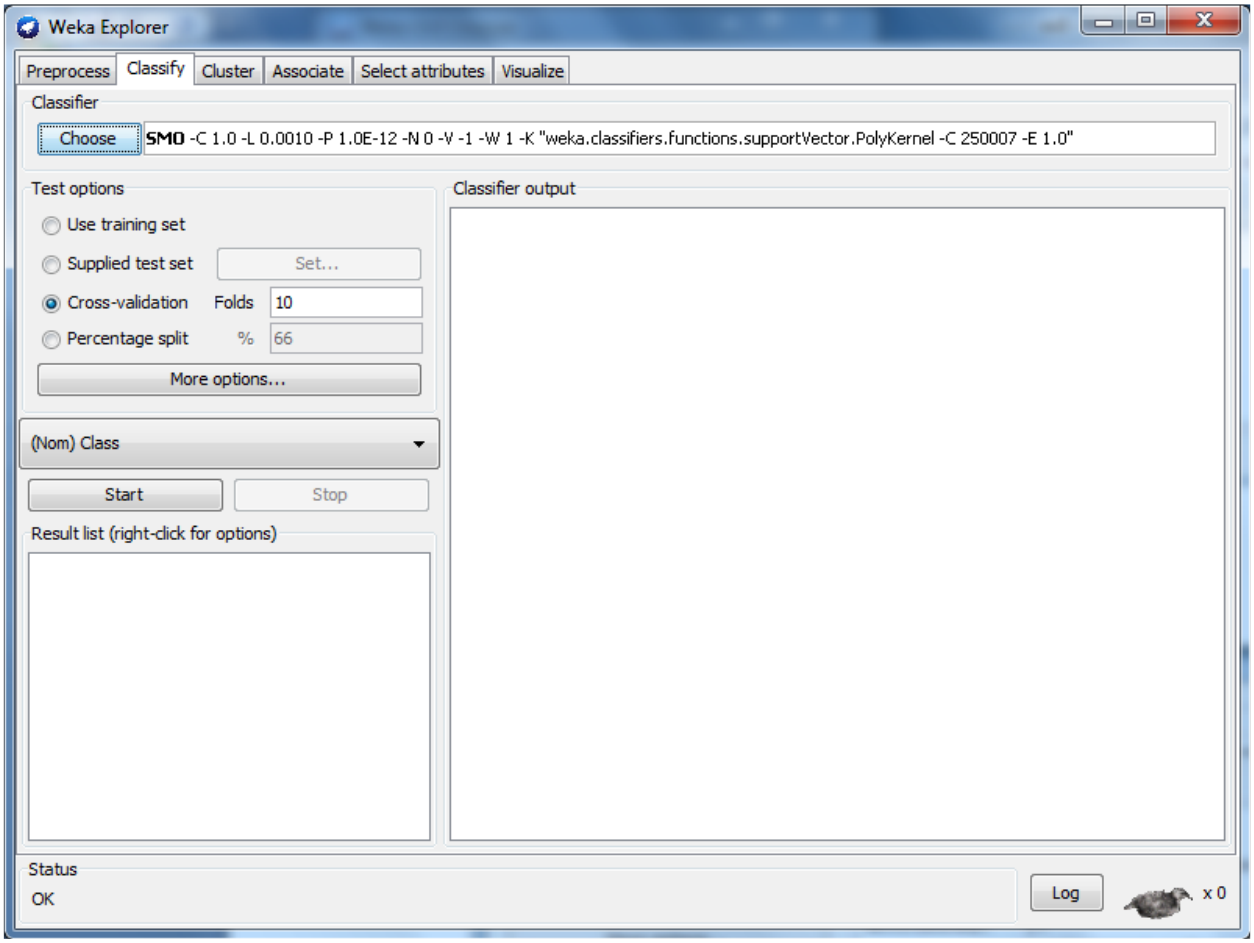
In **preprocess** tab click **Open file.. –** choose **file name** (vote.arff).



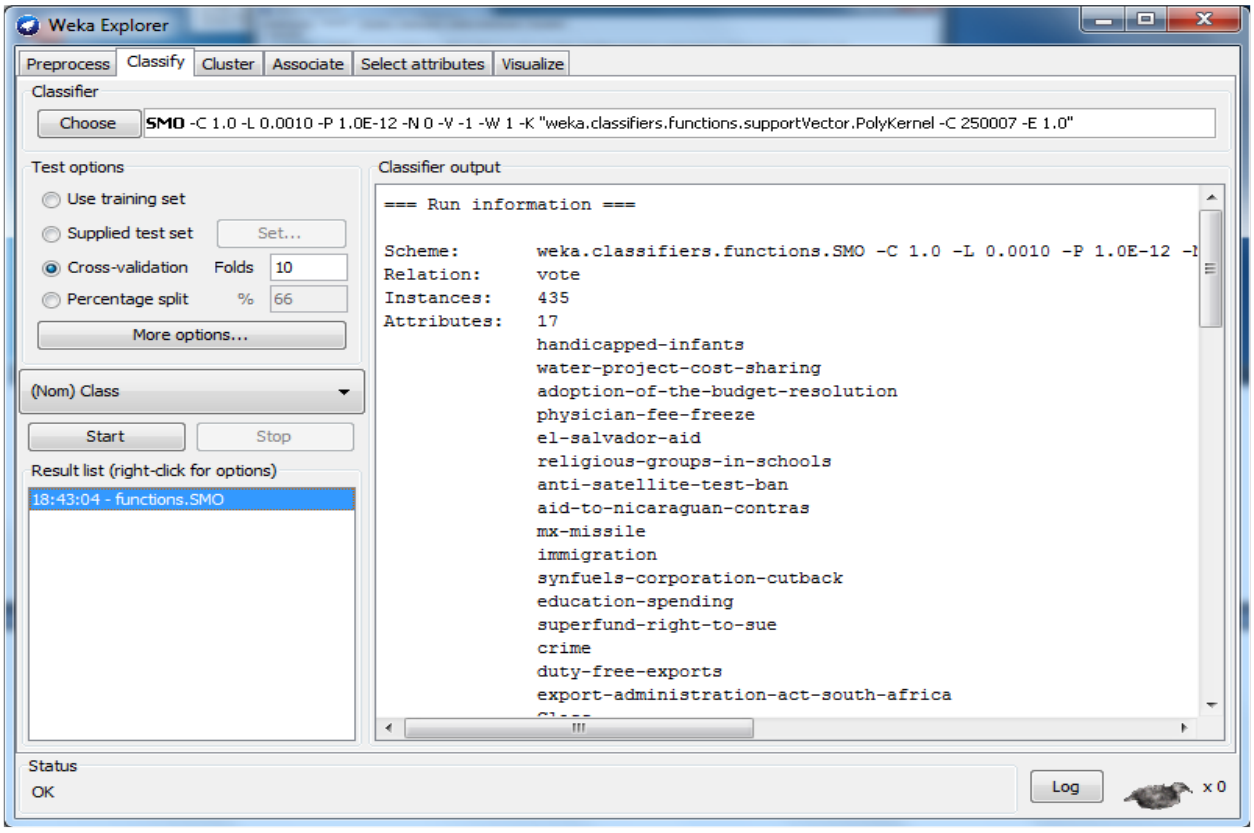


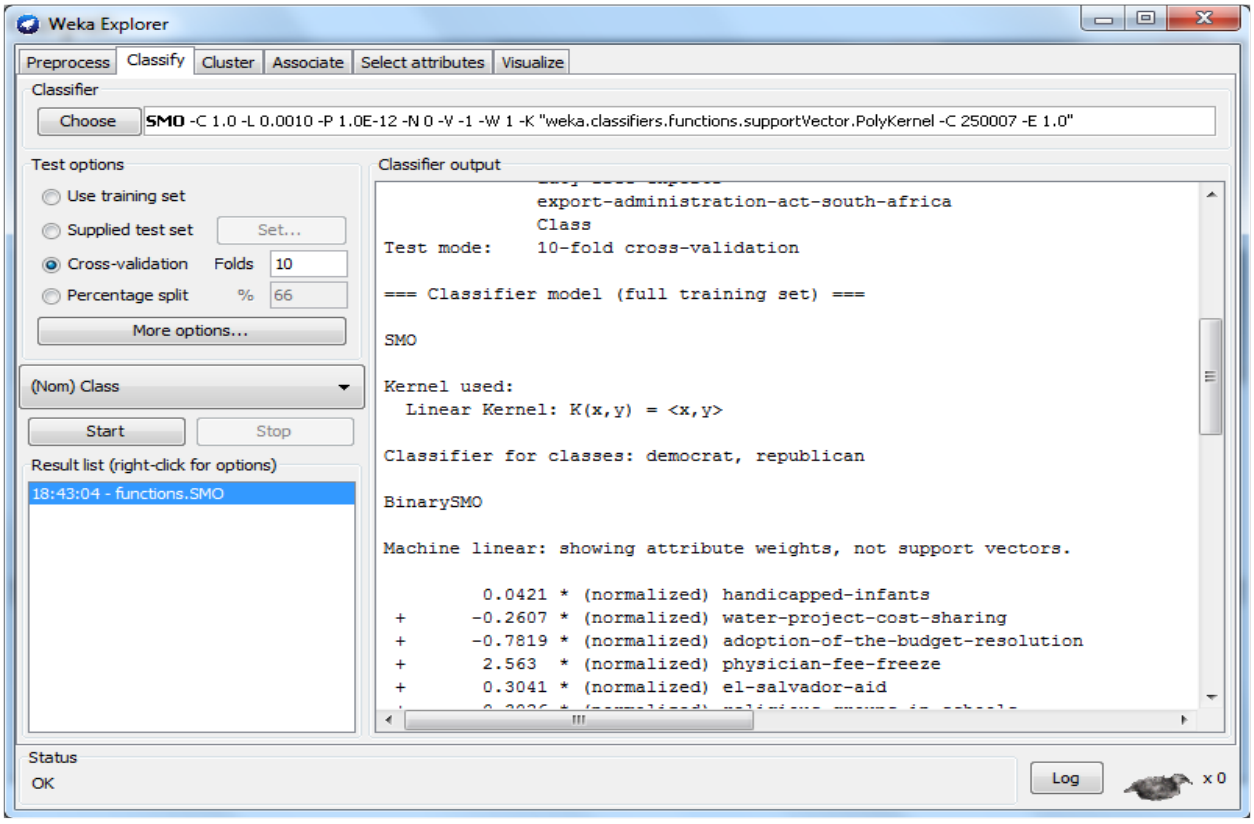
Goto **Classify** tab - click **Choose** button – select **SMO** option.

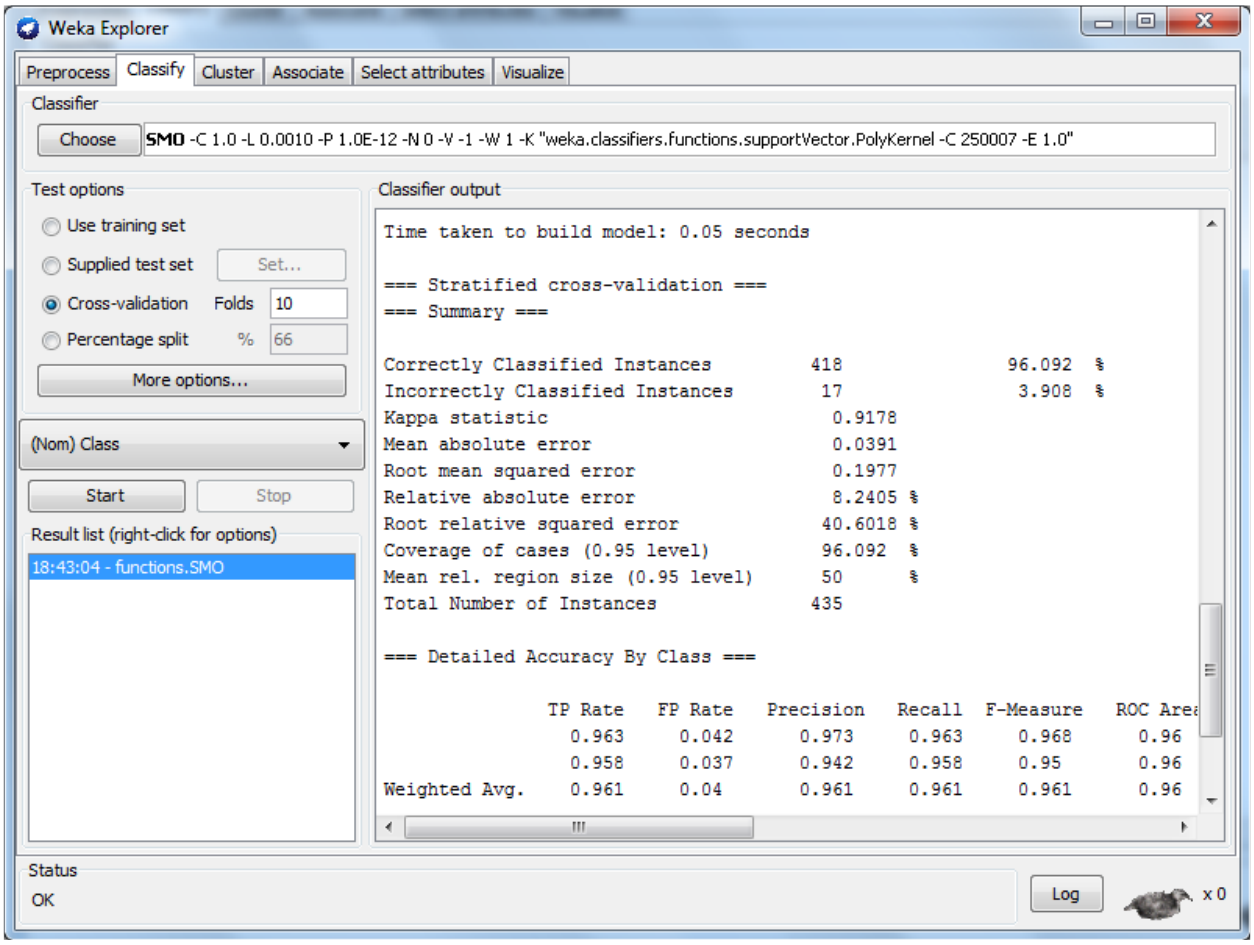

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
GaussianProcesses
LinearRegression
Logistic
MultilayerPerceptron
SGD
SimpleLinearRegression
SimpleLogistic
SMO
SMOreg
VotedPerceptron
lazy
meta
misc
rules
trees
-K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
r output
Close
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
(Nom) Class
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0

Click **Start** button.







VIVA-QUESTIONS

1. Support vector machine may be termed as:
2. What do you mean by generalization error in terms of the SVM?
3. Margin of a hyperplane is defined as:
4. In a hard margin support vector machine
5. The effectiveness of an SVM depends upon:
6. If the hyper plane WTX+b=0 coorectly classifies all the training points (Xi,Yi), where Yi={+1,-1}, then
7. Support vectors are the data points that lie closest to the decision surface
8. Consider a two class problem, whose training points are distributed in the figure below. One possible separating hyperplane.
9. For a two-class classification problem, we use an SVM classifier and obtain the following separating hyperplane. We have marked 4 instances of the training data. Identify the point which will have the most impact on the shape of the boundary on it's removal.

Experiment No. 7: Demonstrate performing clustering on data sets

i. Load each dataset into WEKA and run simple k-means clustering algorithm with different values of k (number of desired clusters).
ii. Explore visualization features of WEKA to visualize the clusters. Derive interesting insights and explain.

CLUSTERING

Cluster analysis or clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics.

Types of clustering

Data clustering algorithms can be hierarchical. Hierarchical algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("bottom-up") or divisive ("top-down"). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

Partitional algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.

Density-based clustering algorithms are devised to discover arbitrary-shaped clusters. In this approach, a cluster is regarded as a region in which the density of data objects exceeds a threshold. DBSCAN and OPTICS are two typical algorithms of this kind.

Two-way clustering, co-clustering or bi-clustering are clustering methods where not only the objects are clustered but also the features of the objects, i.e., if the data is represented in a data matrix, the rows and columns are clustered simultaneously.

Many clustering algorithms require specification of the number of clusters to produce in the input data set, prior to execution of the algorithm. Barring knowledge of the proper value beforehand, the appropriate value must be determined, a problem for which a number of techniques have been developed.

k-means clustering:

In statistics and machine learning, *k*-means clustering is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean.

ALGORITHM:

Regarding computational complexity, the *k*-means clustering problem is:

- NP-hard in general Euclidean space *d* even for 2 clusters
- NP-hard for a general number of clusters *k* even in the plane

• If *k* and *d* are fixed, the problem can be exactly solved in time *O(ndk+1 log n)*, where *n* is the number of entities to be cluster

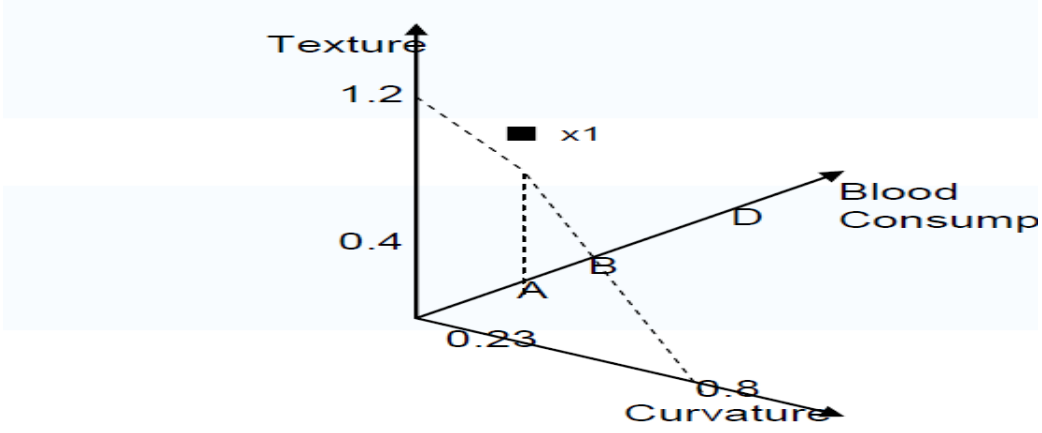
Example
for example, if our class (decision) attribute is tumor Type and its values are: malignant, benign, etc. - these will be the classes. They will be represented by cluster1, cluster2, etc. However, the class information is never provided to the algorithm. The class information can be used later on, to evaluate how accurately the algorithm classified the objects.

Curvature	Texture	Blood Consump	Tumor Type
0.8	1.2	A	Benign
0.75	1.4	B	Benign
0.23	0.4	D	Malignant
0.23	0.5	D	Malignant

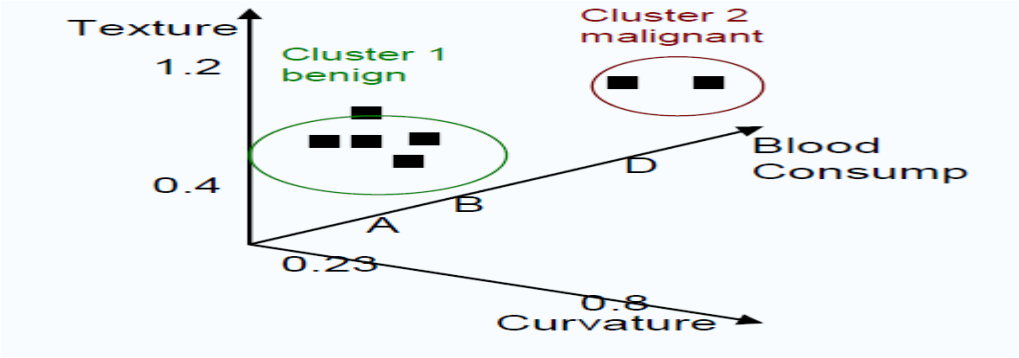
→

	Curvature	Texture	Blood Consump	Tumor Type
x1	0.8	1.2	A	Benign
x2	0.75	1.4	B	Benign
x3	0.23	0.4	D	Malignant
x4	0.23	0.5	D	Malignant
.				
.				

The way we do that, is by plotting the objects from the database into space. Each attribute is one dimension



After all the objects are plotted, we will calculate the distance between them, and the ones that are close to each other – we will group them together, i.e. place them in the same cluster.



Problem: Cluster the following eight points (with (x, y) representing locations) into three clusters A1(2, 10) A2(2, 5) A3(8, 4) A4(5, 8) A5(7, 5) A6(6, 4) A7(1, 2) A8(4, 9). Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).

The distance function between two points *a=(x1, y1)* and *b=(x2, y2)* is defined as:

$$\rho(a, b) = |x2 - x1| + |y2 - y1| .$$

Use k-means algorithm to find the three cluster centers after the second iteration.

Iteration 1

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)				
A2	(2, 5)				
A3	(8, 4)				
A4	(5, 8)				
A5	(7, 5)				
A6	(6, 4)				
A7	(1, 2)				
A8	(4, 9)				

First we list all points in the first column of the table above. The initial cluster centers – means, are (2, 10), (5, 8) and (1, 2) - chosen randomly. Next, we will calculate the distance from the first point (2, 10) to each of the three means, by using the distance function:

point	mean1
$x1, y1$	$x2, y2$
(2, 10)	(2, 10)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\begin{aligned}\rho(point, mean1) &= |x2 - x1| + |y2 - y1| \\ &= |2 - 2| + |10 - 10| \\ &= 0 + 0 \\ &= 0\end{aligned}$$

point	mean2
$x1, y1$	$x2, y2$
(2, 10)	(5, 8)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\begin{aligned}\rho(point, mean2) &= |x2 - x1| + |y2 - y1| \\ &= |5 - 2| + |8 - 10| \\ &= 3 + 2 \\ &= 5\end{aligned}$$

point	mean3
$x1, y1$	$x2, y2$
(2, 10)	(1, 2)

$$\rho(a, b) = |x2 - x1| + |y2 - y1|$$

$$\begin{aligned}\rho(point, mean2) &= |x2 - x1| + |y2 - y1| \\ &= |1 - 2| + |2 - 10| \\ &= 1 + 8 \\ &= 9\end{aligned}$$

So, we fill in these values in the table:

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)	0	5	9	1
A2	(2, 5)				
A3	(8, 4)				
A4	(5, 8)				
A5	(7, 5)				
A6	(6, 4)				
A7	(1, 2)				
A8	(4, 9)				

So, which cluster should the point (2, 10) be placed in? The one, where the point has the shortest distance to the mean – that is mean 1 (cluster 1), since the distance is 0.

Cluster 1 Cluster 2 Cluster 3
(2, 10)
So, we go to the second point (2, 5) and we will calculate the distance to each of the three means, by using the distance function:

point mean1
x1, y1 *x2, y2*
(2, 5) (2, 10)

$\rho(a, b) = |x2 - x1| + |y2 - y1|$
$\rho(point, mean1) = |x2 - x1| + |y2 - y1|$
$= |2 - 2| + |10 - 5|$
$= 0 + 5$
$= 5$

point mean2
x1, y1 *x2, y2*
(2, 5) (5, 8)

$\rho(a, b) = |x2 - x1| + |y2 - y1|$
$\rho(point, mean2) = |x2 - x1| + |y2 - y1|$
$= |5 - 2| + |8 - 5|$
$= 3 + 3$
$= 6$

point mean3
x1, y1 *x2, y2*
(2, 5) (1, 2)

$\rho(a, b) = |x2 - x1| + |y2 - y1|$
$\rho(point, mean2) = |x2 - x1| + |y2 - y1|$
$= |1 - 2| + |2 - 5|$
$= 1 + 3$
$= 4$
So, we fill in these values in the table:
Iteration 1

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)	0	5	9	1
A2	(2, 5)	5	6	4	3
A3	(8, 4)				
A4	(5, 8)				
A5	(7, 5)				
A6	(6, 4)				
A7	(1, 2)				
A8	(4, 9)				

So, which cluster should the point (2, 5) be placed in? The one, where the point has the shortest distance to the mean – that is mean 3 (cluster 3), since the distance is 0.

Cluster 1	Cluster 2	Cluster 3
(2, 10)		(2, 5)

Analogically, we fill in the rest of the table, and place each point in one of the clusters:

Iteration 1

		(2, 10)	(5, 8)	(1, 2)	
	Point	**Dist Mean 1**	**Dist Mean 2**	**Dist Mean 3**	**Cluster**
A1	(2, 10)	0	5	9	1
A2	(2, 5)	5	6	4	3
A3	(8, 4)	12	7	9	2
A4	(5, 8)	5	0	10	2
A5	(7, 5)	10	5	9	2
A6	(6, 4)	10	5	7	2
A7	(1, 2)	9	10	0	3
A8	(4, 9)	3	2	10	2

Cluster 1	Cluster 2	Cluster 3
(2, 10)	(8, 4)	(2, 5)
	(5, 8)	(1, 2)
	(7, 5)	
	(6, 4)	
	(4, 9)	

Next, we need to re-compute the new cluster centers (means). We do so, by taking the mean of all points in each cluster.

For Cluster 1, we only have one point A1(2, 10), which was the old mean, so the cluster center remains the same.

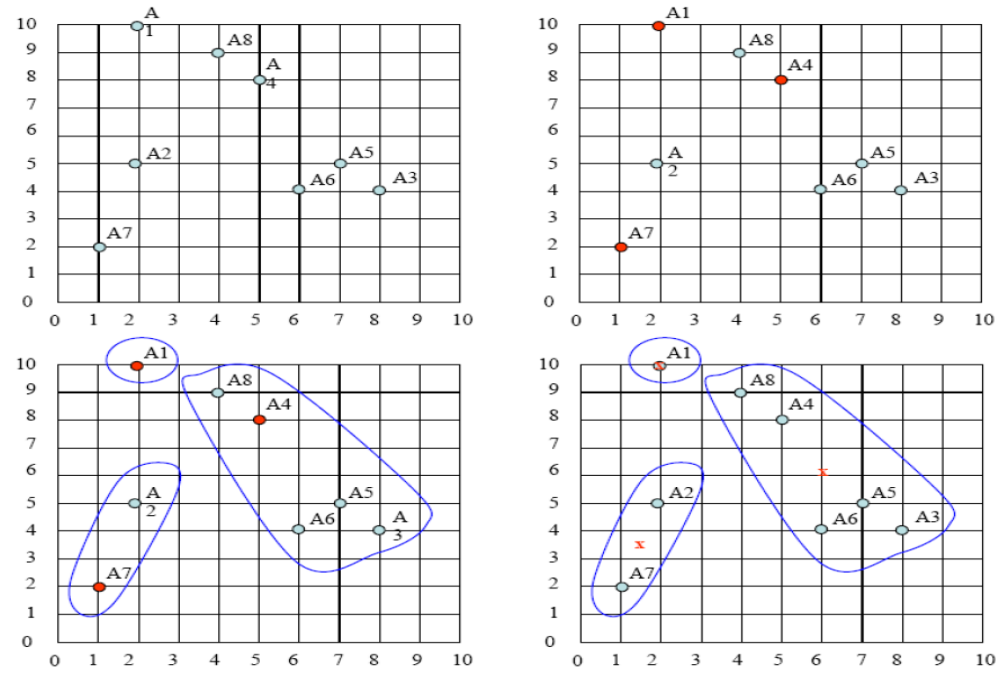
For Cluster 2, we have ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = (6, 6)

For Cluster 3, we have ((2+1)/2, (5+2)/2) = (1.5, 3.5)

new clusters: 1: {A1}, 2: {A3, A4, A5, A6, A8}, 3: {A2, A7}

b) centers of the new clusters:
C1= (2, 10), C2= ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = (6, 6), C3= ((2+1)/2, (5+2)/2) = (1.5, 3.5)

c)

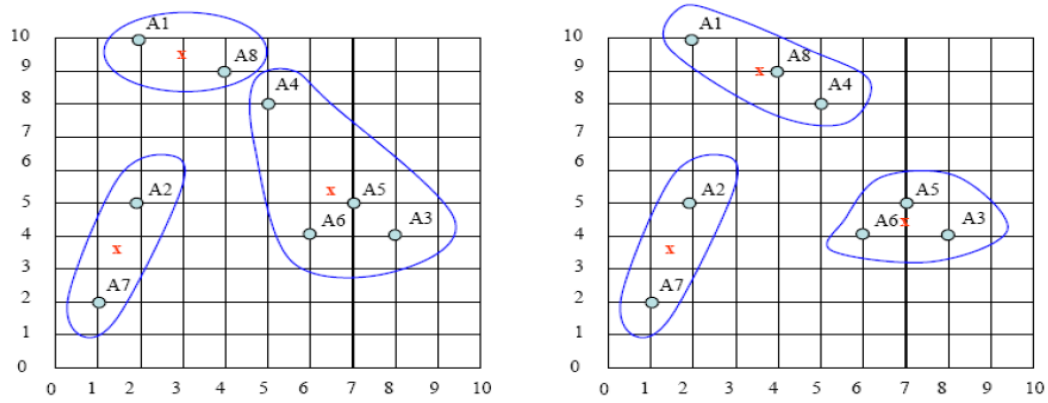


The initial cluster centers are shown in red dot. The new cluster centers are shown in red x.

That was Iteration1 (epoch1). Next, we go to Iteration2 (epoch2), Iteration3, and so on until the means do not change anymore.

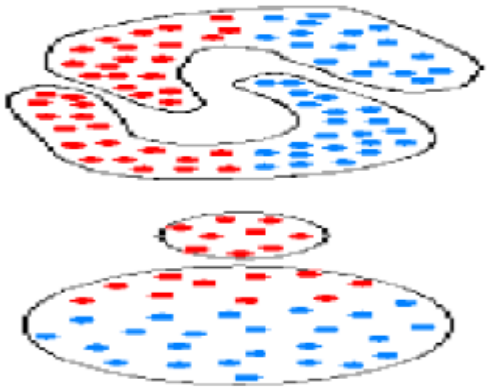
In Iteration2, we basically repeat the process from Iteration1 this time using the new means we computed.

d)
We would need two more epochs. After the 2nd epoch the results would be:
1: {A1, A8}, 2: {A3, A4, A5, A6}, 3: {A2, A7}
with centers C1=(3, 9.5), C2=(6.5, 5.25) and C3=(1.5, 3.5).
After the 3rd epoch, the results would be:
1: {A1, A4, A8}, 2: {A3, A5, A6}, 3: {A2, A7}
with centers C1=(3.66, 9), C2=(7, 4.33) and C3=(1.5, 3.5).



K-means: Problems and Limitations

Based on minimizing within cluster error -a criterion that is not appropriate for many situations.– Unsuitable when clusters have widely different sizes or have convex shapes.



K-mediod method:

The *k*-medoids algorithm is a clustering algorithm related to the *k*-means algorithm and the medoidshift algorithm. Both the *k*-means and *k*-medoids algorithms are partitional (breaking the dataset up into groups) and both attempt to minimize squared error, the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the *k*-means algorithm *k*medoids chooses datapoints as centers (medoids or exemplars).

ALGORITHM:

The most common realization of *k*-medoid clustering is the Partitioning Around Medoids (PAM) algorithm and is as follows:

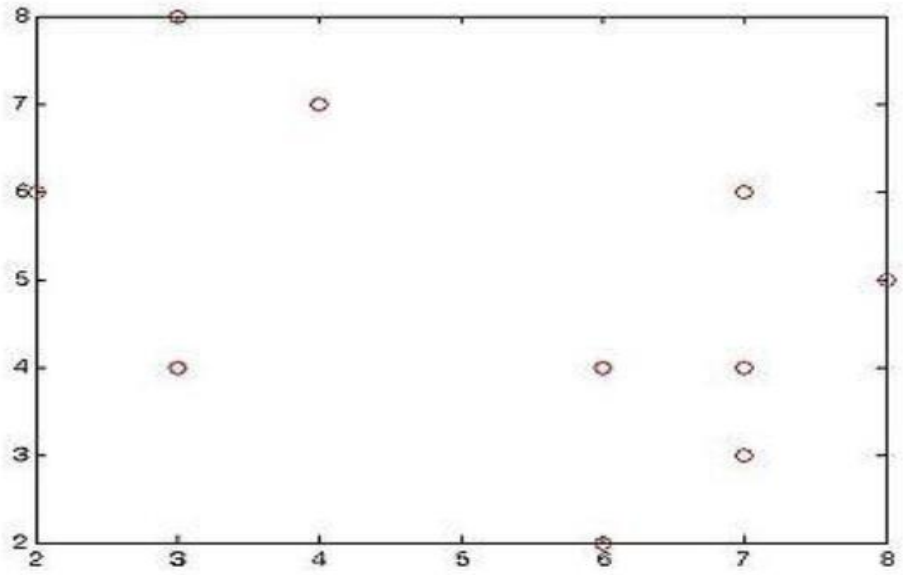
1. Initialize: randomly select *k* of the *n* data points as the mediods

2. Associate each data point to the closest medoid. ("closest" here is defined using any valid distance metric, most commonly Euclidean distance, Manhattan distance or Minkowski distance)
3. For each mediod *m*
    1. For each non-mediod data point *o*
        1. Swap *m* and *o* and compute the total cost of the configuration
4. Select the configuration with the lowest cost.
5. Repeat steps 2 to 5 until there is no change in the medoid.

EXAMPLE:

Cluster the following data set of ten objects into two clusters i.e *k* = 2.
Consider a data set of ten objects as follows:

X1	2	6
X2	3	4
X3	3	8
X4	4	7
X5	6	2
X6	6	4
X7	7	3
X8	7	4
X9	8	5
X10	7	6

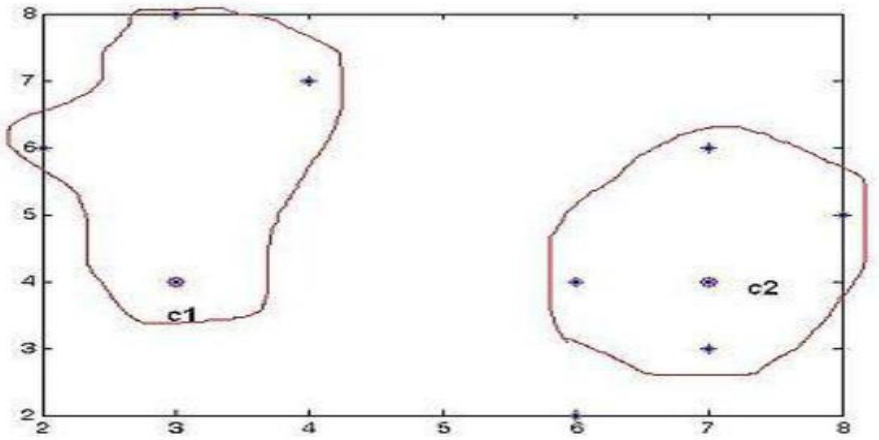


Step 1
Initialise *k* centre
Let us assume c1 = (3,4) and c2 = (7,4)
So here c1 and c2 are selected as medoid.
Calculating distance so as to associate each data object to its nearest medoid. Cost is calculated using Minkowski distance metric with *r* = 1.

C1		Data Objects (X_i)		Cost (distance)
3	4	2	6	3
3	4	3	8	4
3	4	4	7	4
3	4	6	2	5
3	4	6	4	3
3	4	7	3	5
3	4	8	5	6
3	4	7	6	6

C2		Data Objects (X_i)		Cost (distance)
7	4	2	6	7
7	4	3	8	8
7	4	4	7	6
7	4	6	2	3
7	4	6	4	1
7	4	7	3	1
7	4	8	5	2
7	4	7	6	2

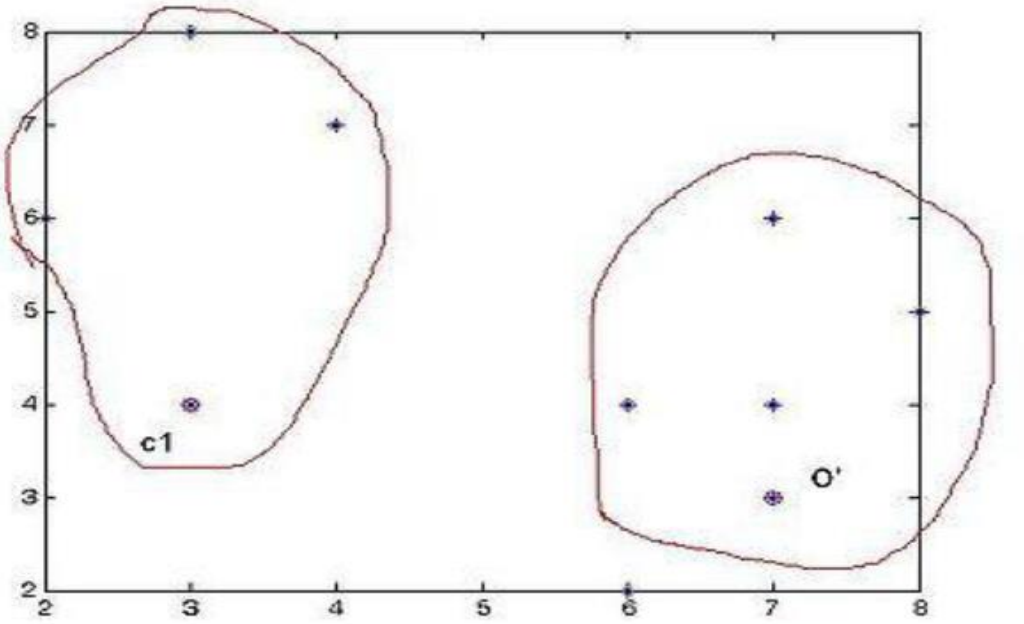
Then so the clusters become:
Cluster1 = {(3,4)(2,6)(3,8)(4,7)}
Cluster2 = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}
Since the points (2,6) (3,8) and (4,7) are close to c1 hence they form one cluster whilst remaining points form another cluster.
So the total cost involved is 20.
Where cost between any two points is found using formula
cost(x,c)=$\Sigma|x-c|$ from i=1 to d
where *x* is any data object, *c* is the medoid, and *d* is the dimension of the object which in this case is 2.
Total cost is the summation of the cost of data object from its medoid in its cluster so here:
Totaol cost={cost((3,4),(2,6)) +cost((3,4),(3,8))+cost((3,4),(4,7)) }
+cost((7,4),(6,2))+cost((7,4),(6,4))+cost((7,4)(7,3))
+cost((7,4),(8,5))+cost((7,4),(7,6))}
=(3+4+4) +(3+1+1+2+2)
=20



Step 2
Selection of nonmedoid O′ randomly
Let us assume O′ = (7,3)
So now the medoids are c1(3,4) and O′(7,3)
If c1 and O′ are new medoids, calculate the total cost involved By using the formula in the step 1

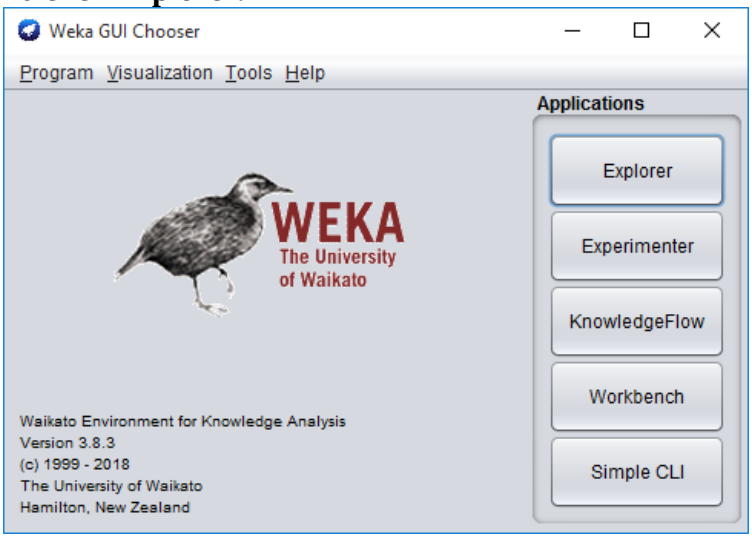
C1		Data Objects (X_i)		Cost (distance)
3	4	2	6	3
3	4	3	8	4
3	4	4	7	4
3	4	6	2	5
3	4	6	4	3
3	4	7	4	4
3	4	8	5	6
3	4	7	6	6

O1		Data Objects (X_i)		Cost (distance)
7	3	2	6	8
7	3	3	8	9
7	3	4	7	7
7	3	6	2	2
7	3	6	4	2
7	3	7	4	1
7	3	8	5	3
7	3	7	6	3

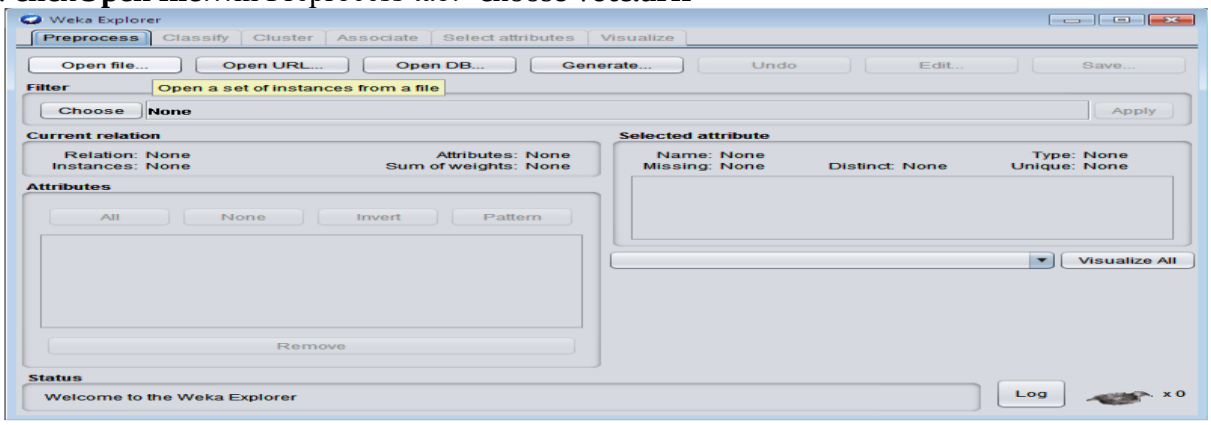


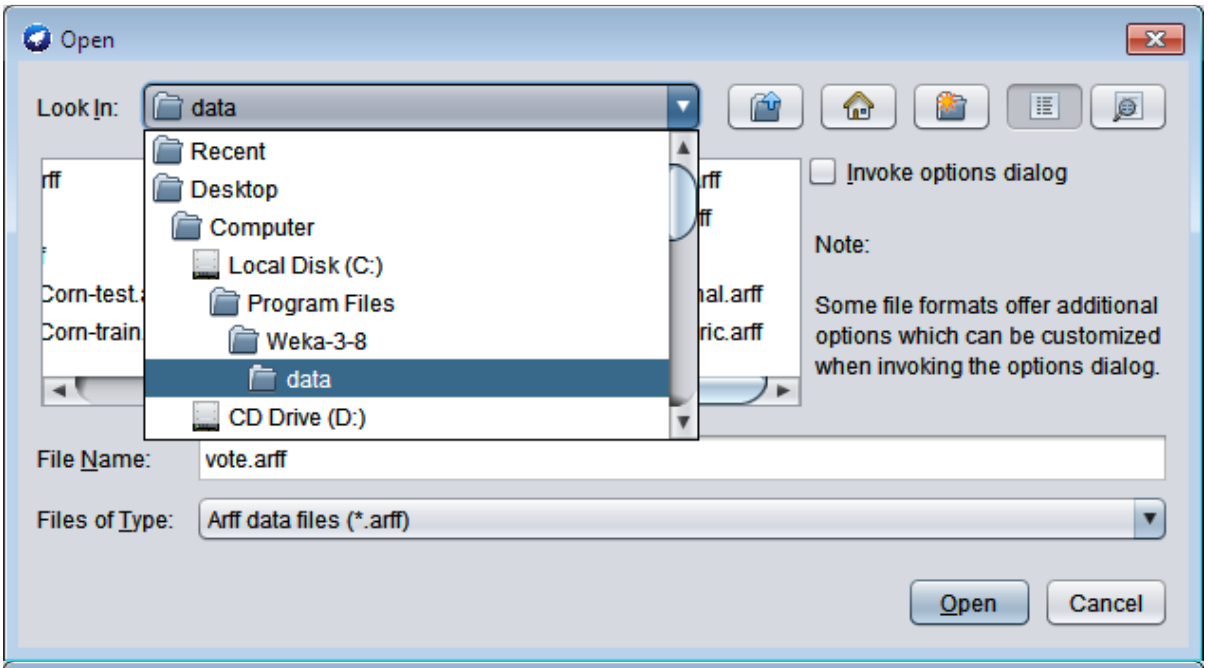
total cost=3+4+4+2+2+1+3+3
=22
So cost of swapping medoid from c2 to O′ is
S=current total cost-past total cost
=22-20
=2
So moving to O′ would be bad idea, so the previous choice was good and algorithm terminates here (i.e there is no change in the medoids).
It may happen some data points may shift from one cluster to another cluster depending upon their closeness to medoid.

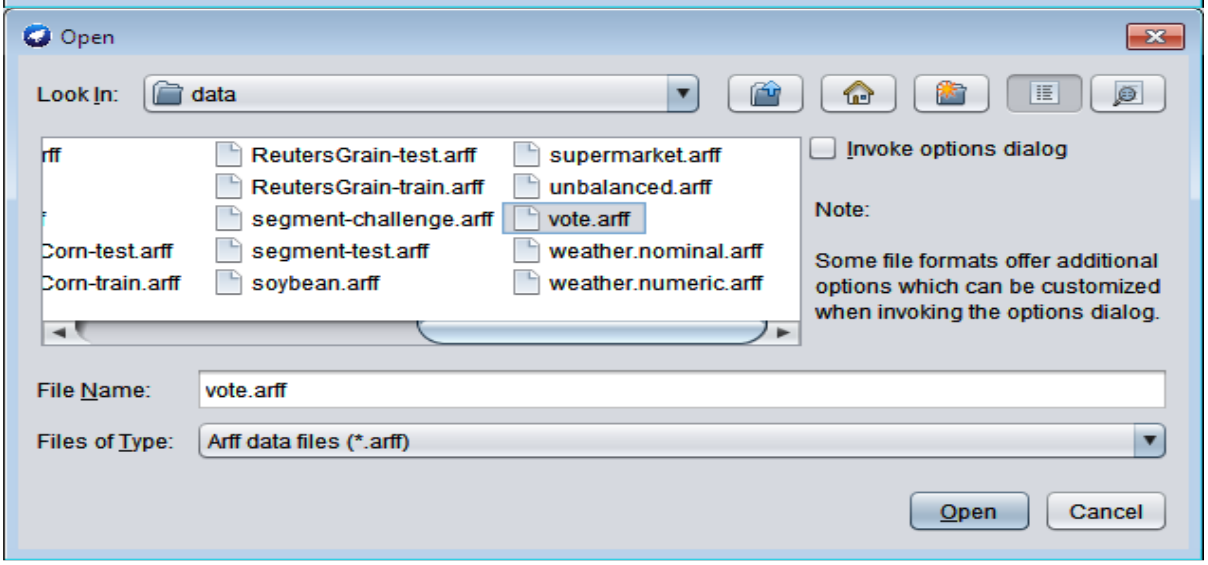
1.Open **weka** tool and click **Explorer.**

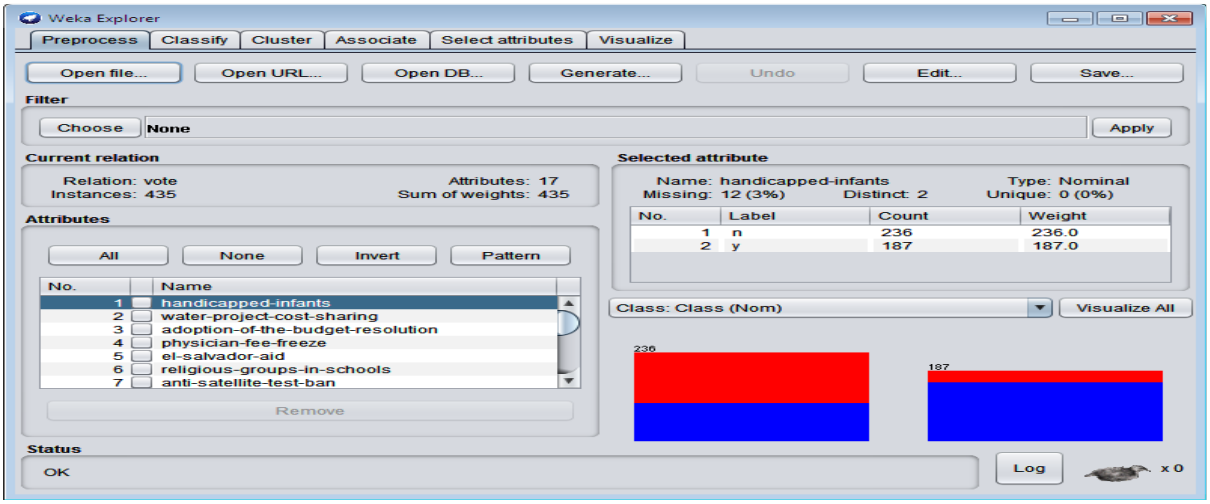


2. click**Open file**…in Preprocess tab.- choose **vote.arff**

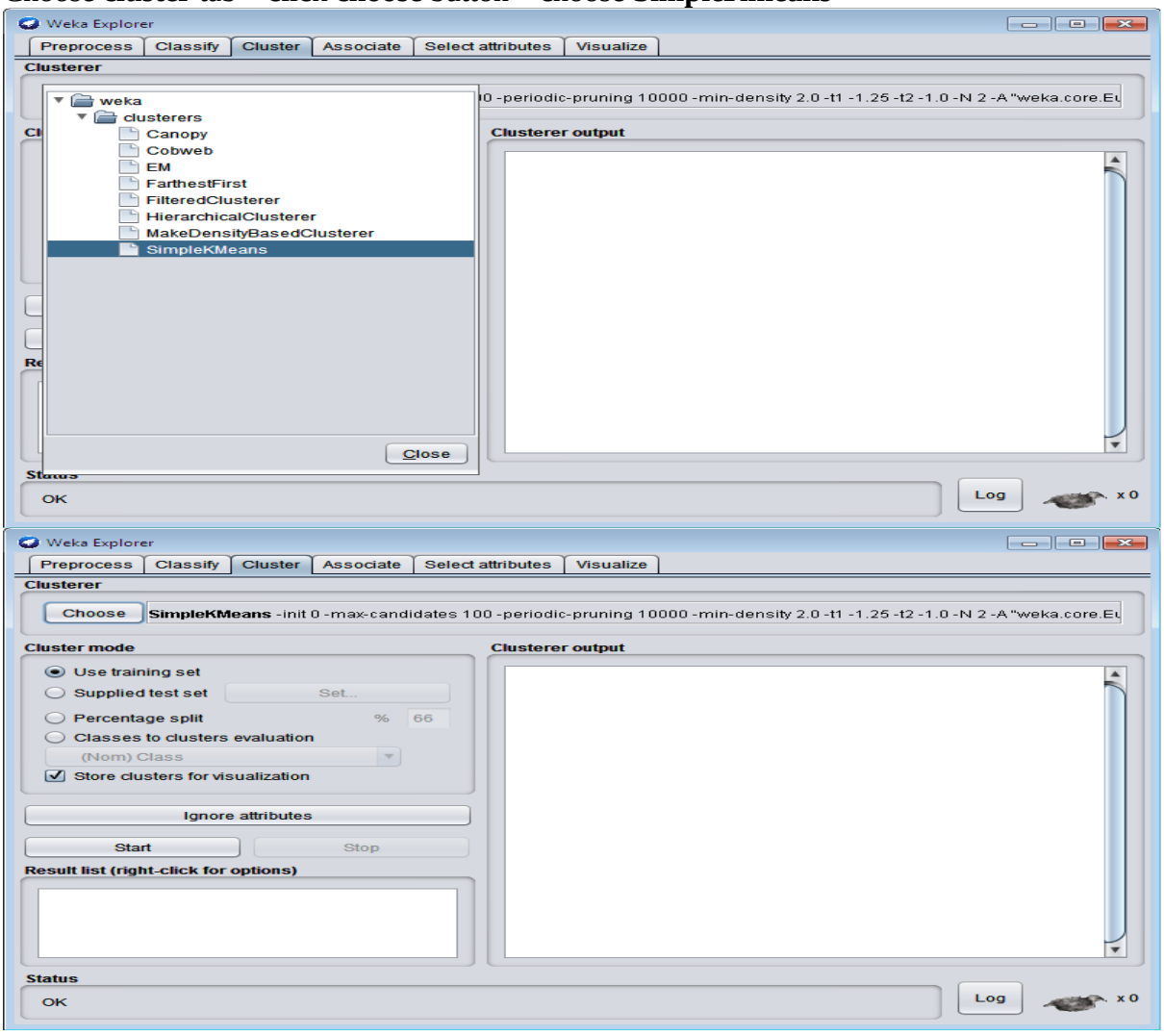




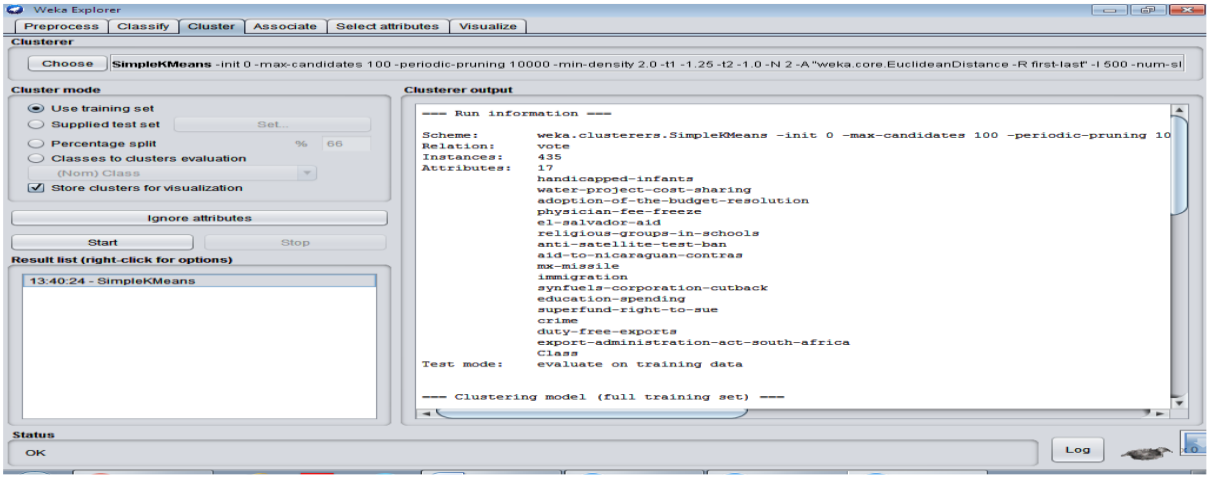


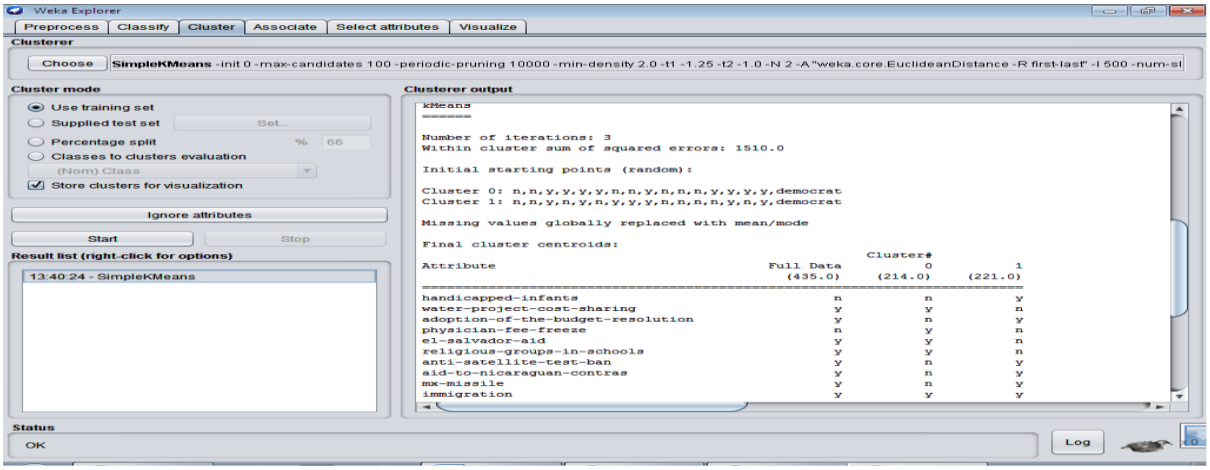


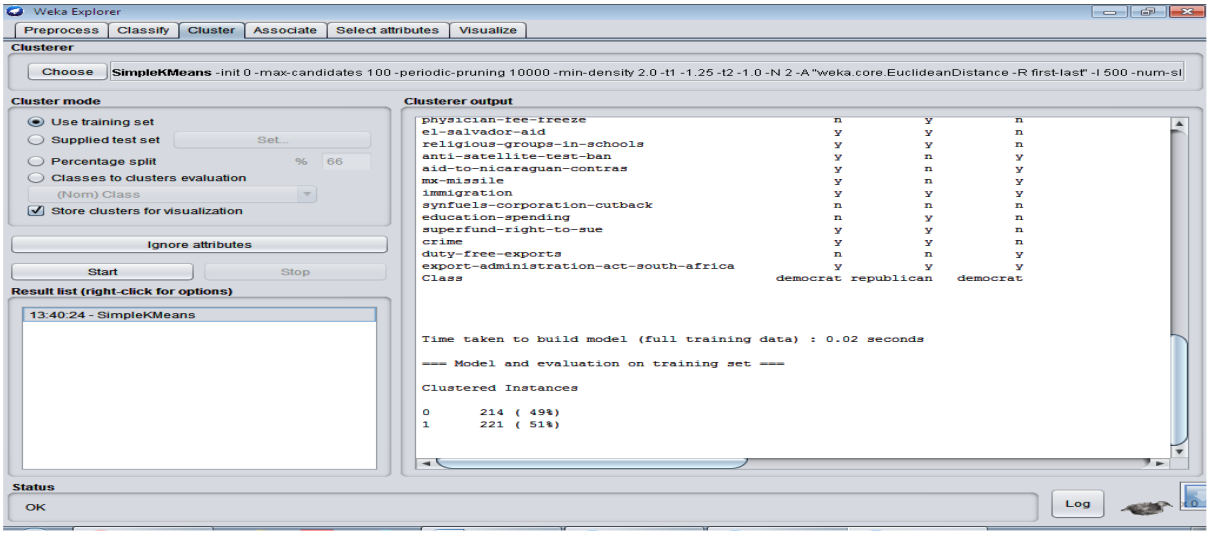
Choose **cluster** tab – click **choose** button – choose **SimpleKmeans**



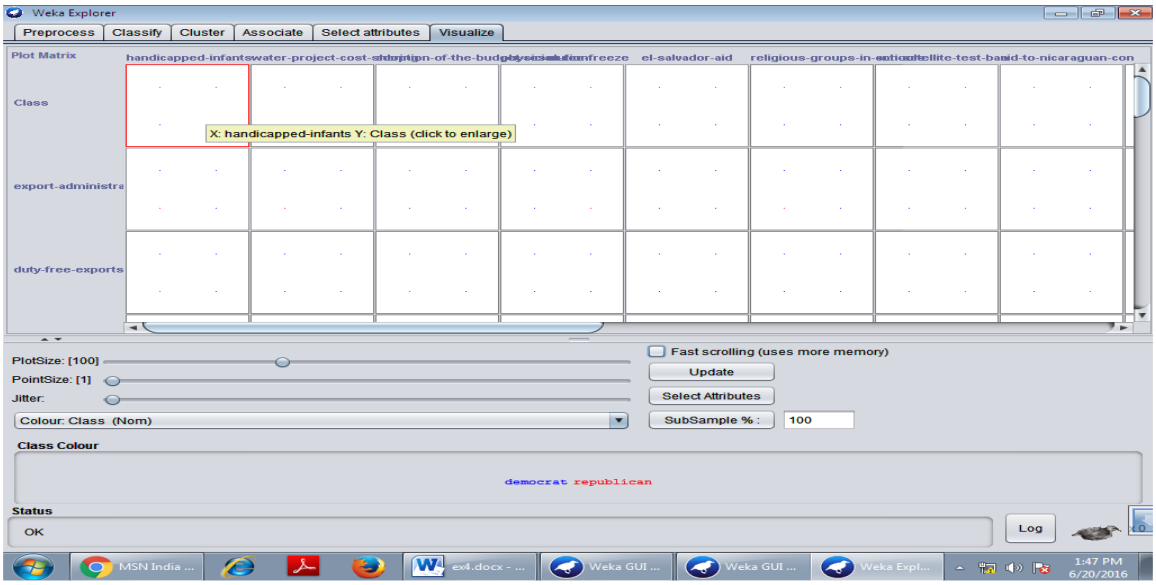
Click **Start** button

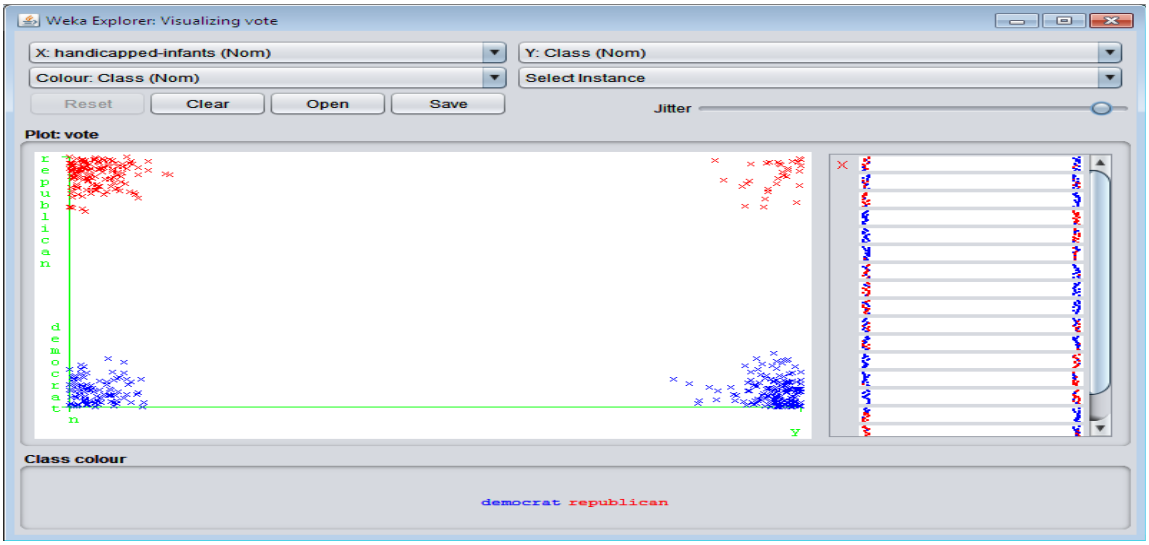






Goto - **Visualize** tab- click one box any visualize.





VIVA-QUESTIONS

1. A good clustering is one having
2. Which of the following is an exploratory data mining technique?
   A. Classification B. Clustering C. Regression
3. Which of the following tasks can be best solved using Clustering.
   A. Predicting the amount of rainfall based on various cues
   B. Detecting fraudulent credit card transactions
   C. Training a robot to solve a maze
4. Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8);
5. Compute the Manhattan distance between the two objects

6. Consider a set of five 2-dimensional points p1=(0, 0), p2=(0, 1), p3=(5, 8), p4=(5, 7), and p5=(0, 0.5).
   Euclidean distance is the distance function. The k-means algorithm is used to cluster the points into two clusters. The initial cluster centers are p1 and p4. The clusters after two iterations of k-means are:
7. Clustering is a
8. Which of the following algorithm is most sensitive to outliers?
   K-means clustering algorithm
   K-medians clustering algorithm
   K-modes clustering algorithm
   K-medoids clustering algorithm
9. K means and K-medioids are example of which type of clustering method?

P1 P2 P3 P4				
P1	0	1.000	1.414	1.000
P2	1.000	0	1.000	1.414
P3	1.414	1.000	0	1.000
P4	1.000	1.414	1.000	0

   The Euclidean distance matrix between four 2-dimensional points, p1, p2, p3, and p4, is shown below. A possible set of co-ordinate values of these points are:

10. Consider a set of five 2-dimensional points D1=(2,0), D2=(1,3), D3=(3,5), D4=(2,2) and D5=(4,6).
    Euclidean distance is the distance function. The k-means algorithm is used to cluster the points into two clusters. The initial cluster centers are D2 and D4. The clusters after two iterations of k-means are:

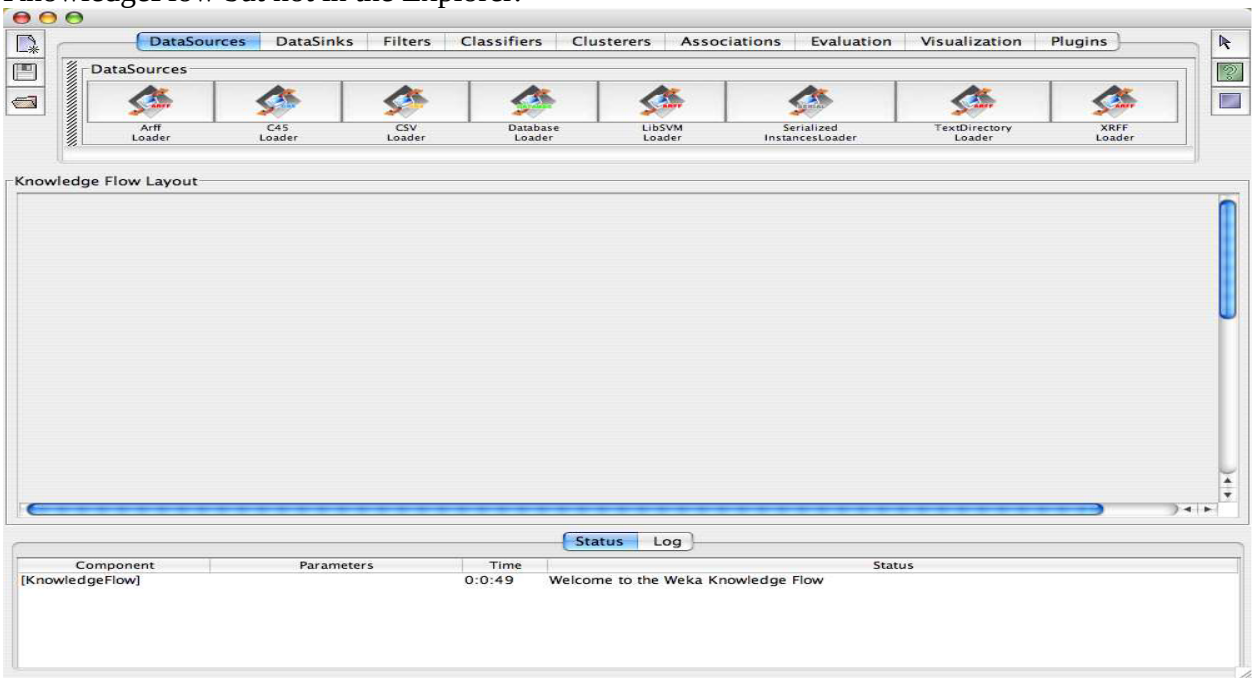
Experiment No. 8: Demonstrate performing knowledge flow on WEKA

i. Perform pre-processing tasks. ii. Perform decision-tree iii. Perform clustering

KnowledgeFlow

Introduction

The KnowledgeFlow provides an alternative to the Explorer as a graphical front end to WEKA's core algorithms. The KnowledgeFlow is a work in progress so some of the functionality from the Explorer is not yet available. On the other hand, there are things that can be done in the KnowledgeFlow but not in the Explorer.



The Knowledge Flow presents a data-flow inspired interface to WEKA. The user can select WEKA components from a tool bar, place them on a layout canvas and connect them together in order to form a knowledge flow for processing and analyzing data. At present, all of WEKA's classifiers, filters, clusters, loaders and savers are available in the Knowledge Flow along with some extra tools.

WEKA there are ten classifiers that can handle data incrementally:

- AODE
- IB1
- IBk
- KStar
- NaiveBayesMultinomialUpdateable
- NaiveBayesUpdateable
- NNge
- Winnow

Features

The Knowledge Flow offers the following features:

- intuitive data flow style layout

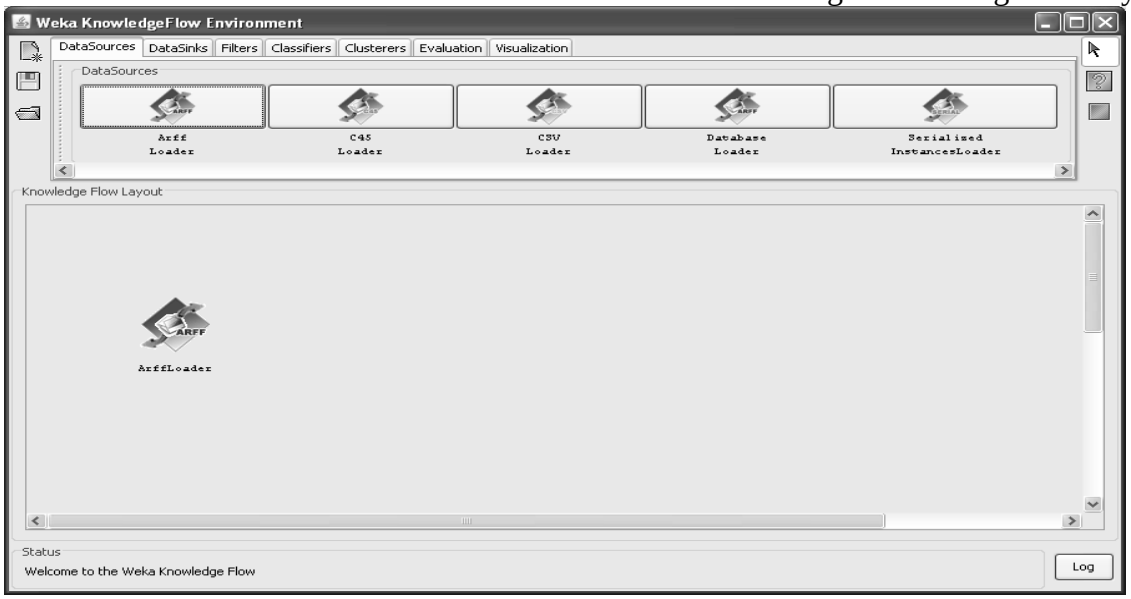
• process data in batches or incrementally
• process multiple batches or streams in parallel (each separate flow executes in its own thread)
• chain filters together
• view models produced by classifiers for each fold in a cross validation
• visualize performance of incremental classifiers during processing (scrolling plots of classification accuracy, RMS error, predictions etc.)
• plugin facility for allowing easy addition of new components to the Knowledge Flow

KNOWLEDGE FLOW FOR PRE-PROCESSING

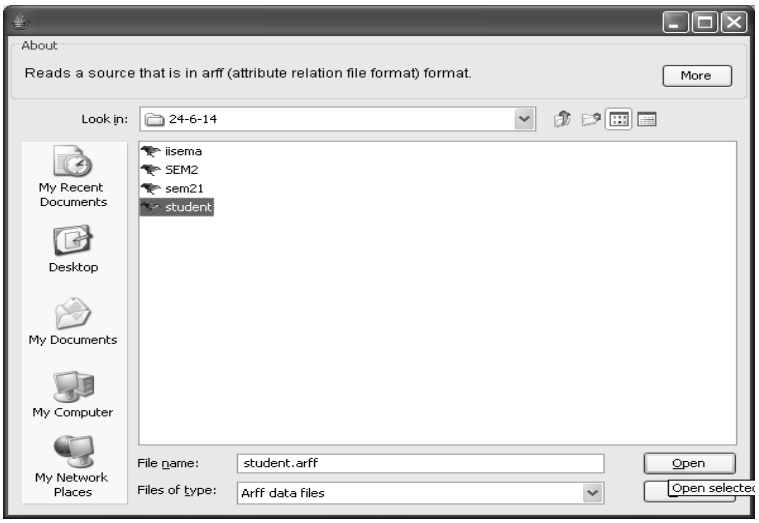
ADD ATTRIBUTE

Step-1: - Click on start button and then select All Programs and choose WEKA 3.8.3 in the WEKA 3.8(with console).

Step-2: - In the WEEKA GUI to select the knowledge flow.
Step-3: - Select the data sources and then select ARFF loader and dragon knowledge flow layout.

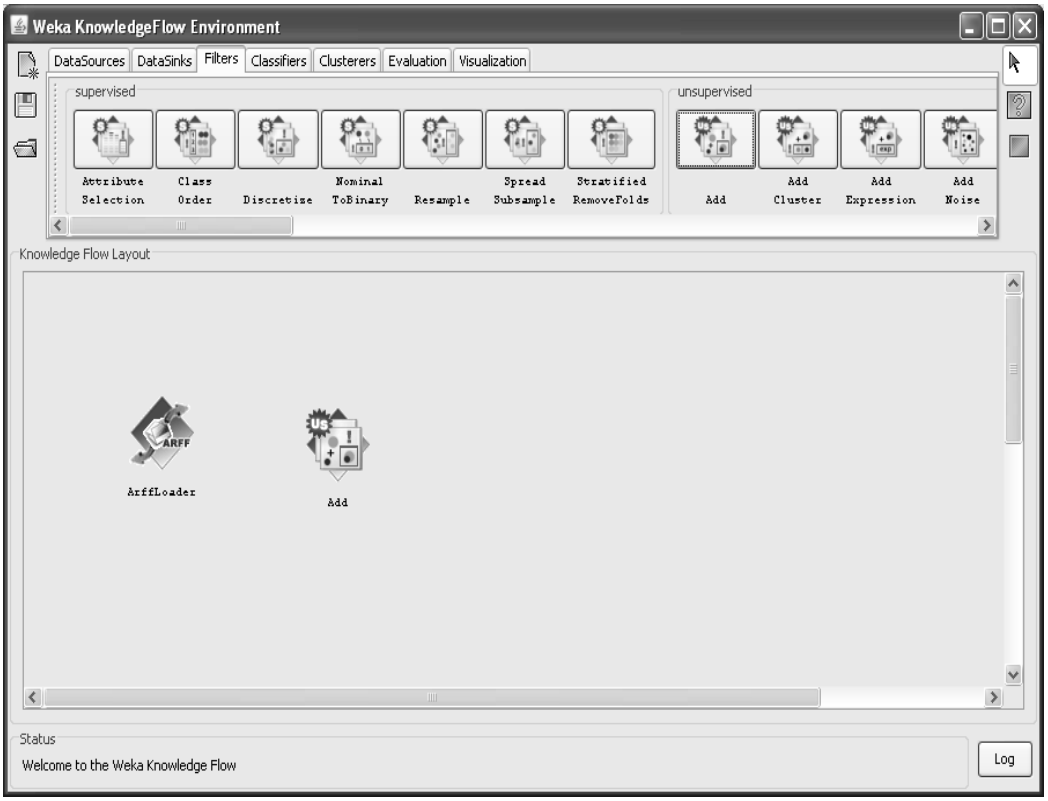


Step-4: - Select ARFF File right click in click configure to attach the file.

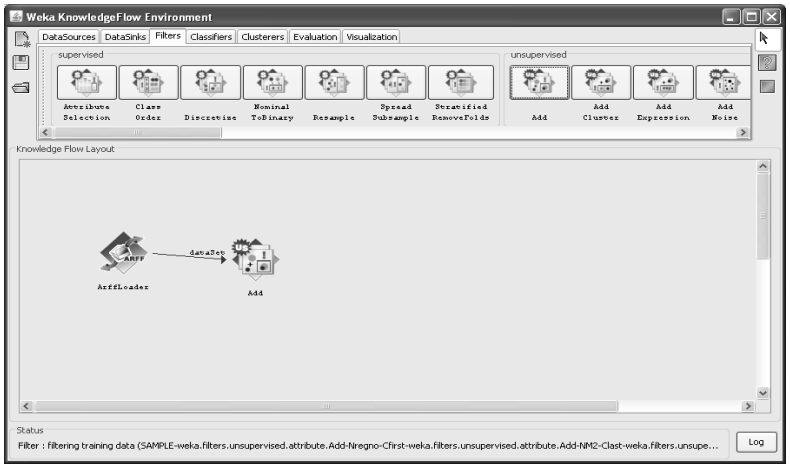


Click on open button.

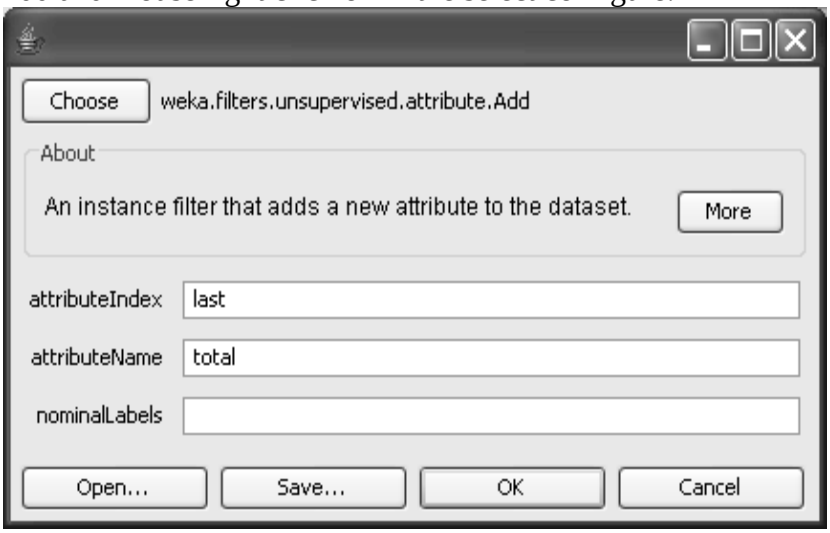
Step-5: - Go to Filters and select unsupervised in the Add Field and dragon knowledge flow layout.



Step-6: - Select ARFF loader mouse right click in the connection to select data set and connection between ARFF loader to Add.

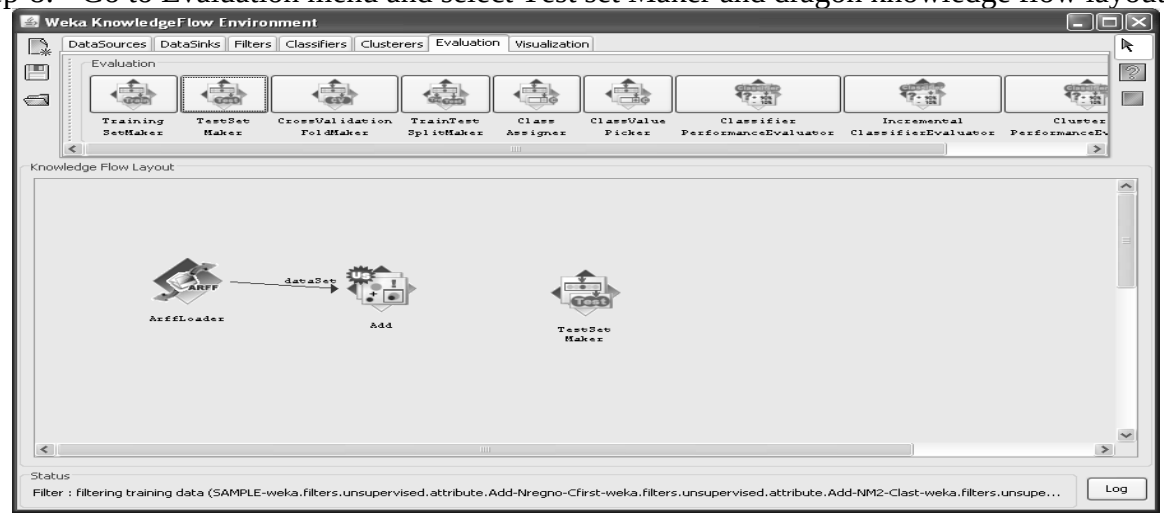


Step-7: - Select Add and mouse right click on in the select configure.

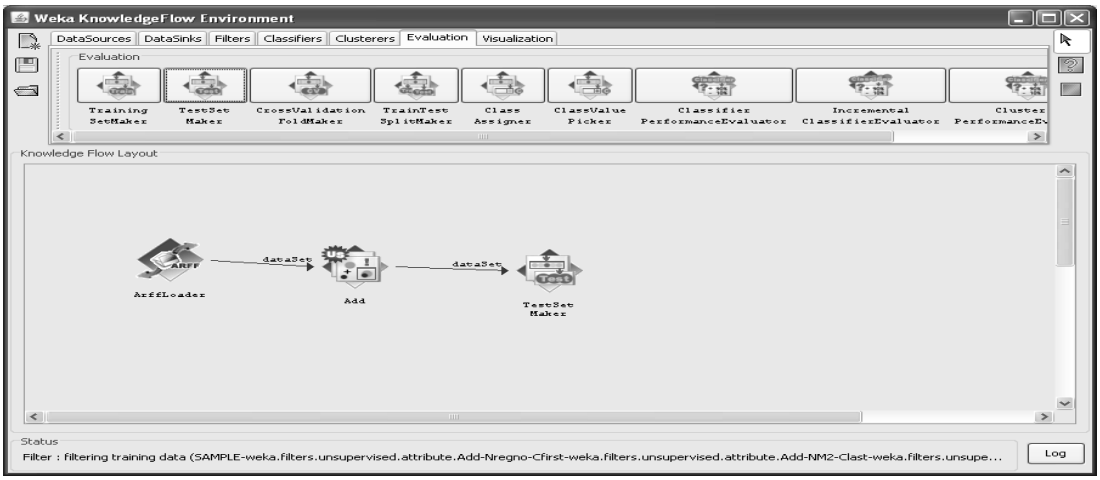


Click on ok button.

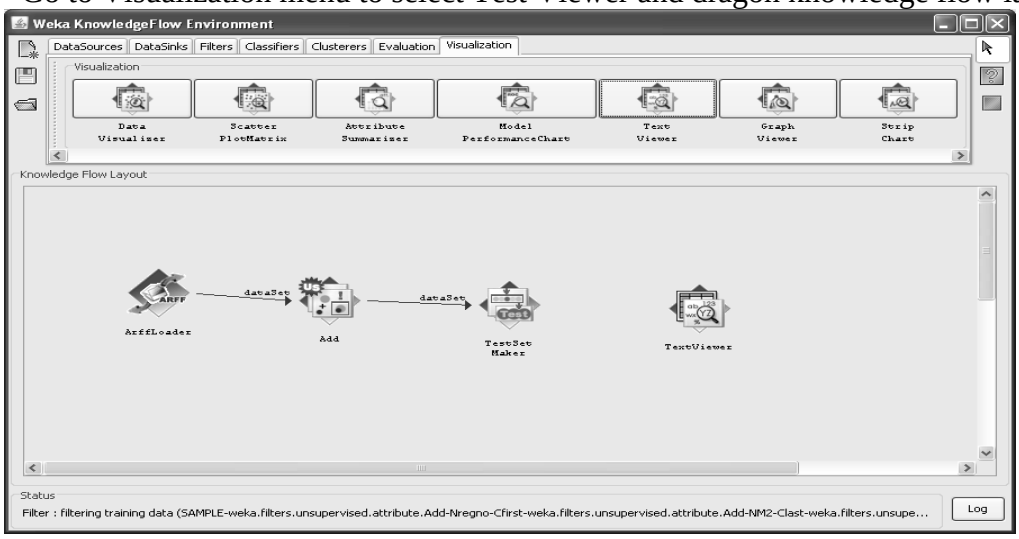
Step-8: - Go to Evaluation menu and select Test set Maker and dragon knowledge flow layout.



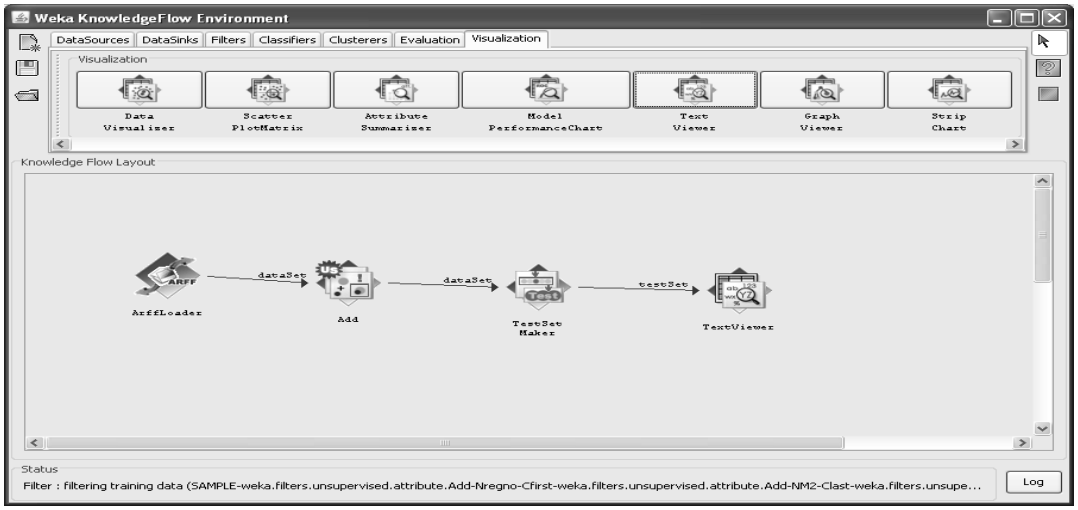
Step-9: - Select Add in the knowledge flow layout and mouse right click in the connection data set and connection between Add to Test Set Maker.



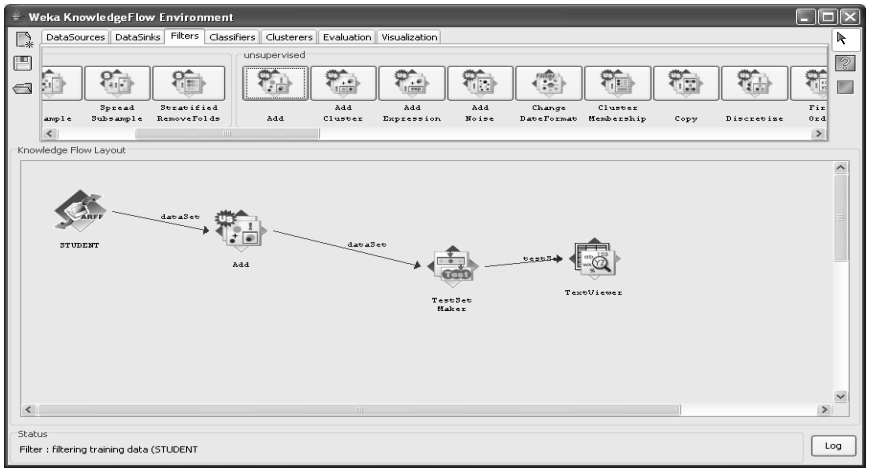
Step-10: - Go to Visualization menu to select Test Viewer and dragon knowledge flow layout.



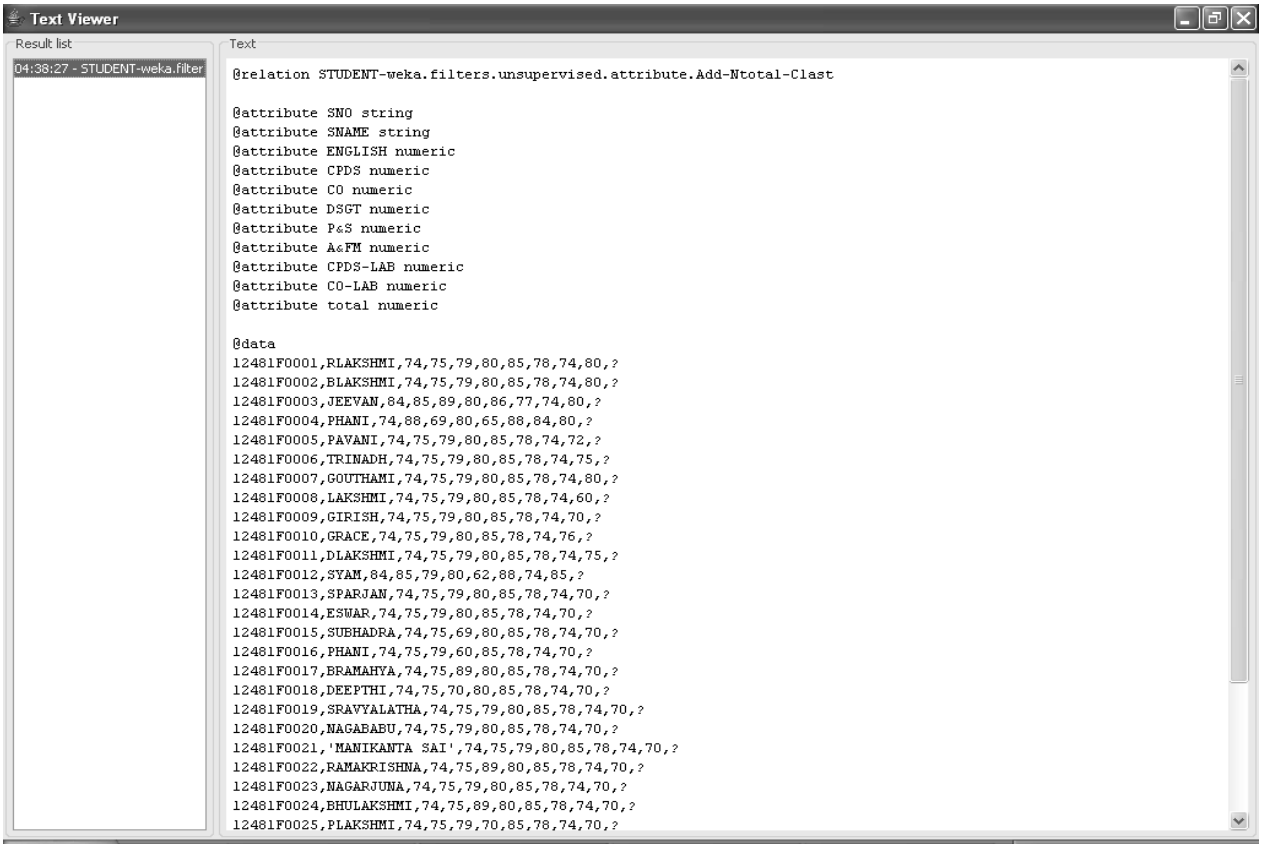
Step-11: - Select Test Set Maker in knowledge flow layout and mouse right click in connection to select test set and connection between test set to Text viewer.



Step-12: - Select ARFF Loader in the knowledge flow layout in the action select start loading.



Step-13: - Select Text Viewer in the knowledge flow layout mouse right click in the action to select show result.

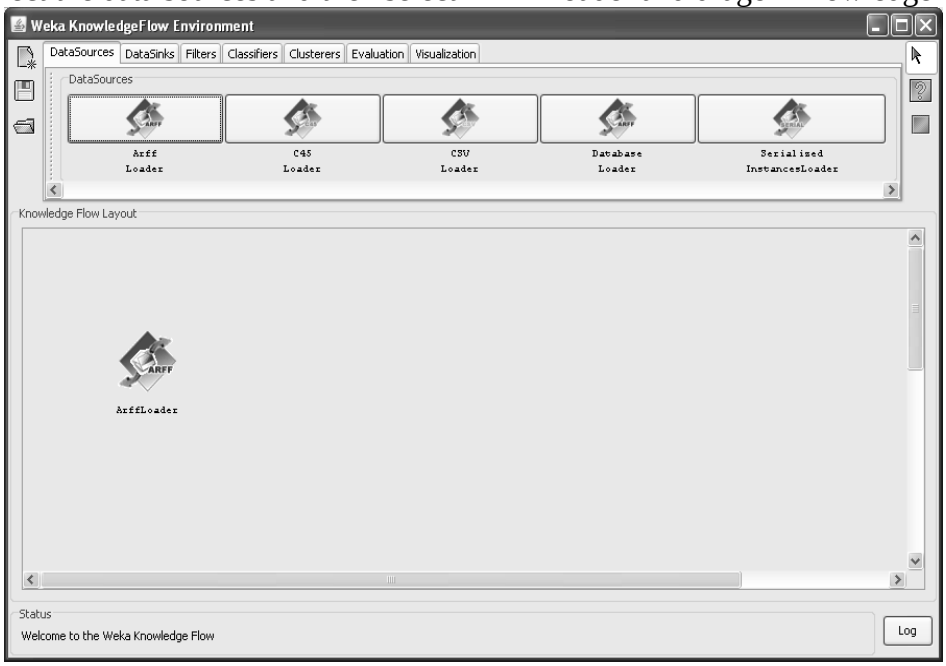


Add Expression.

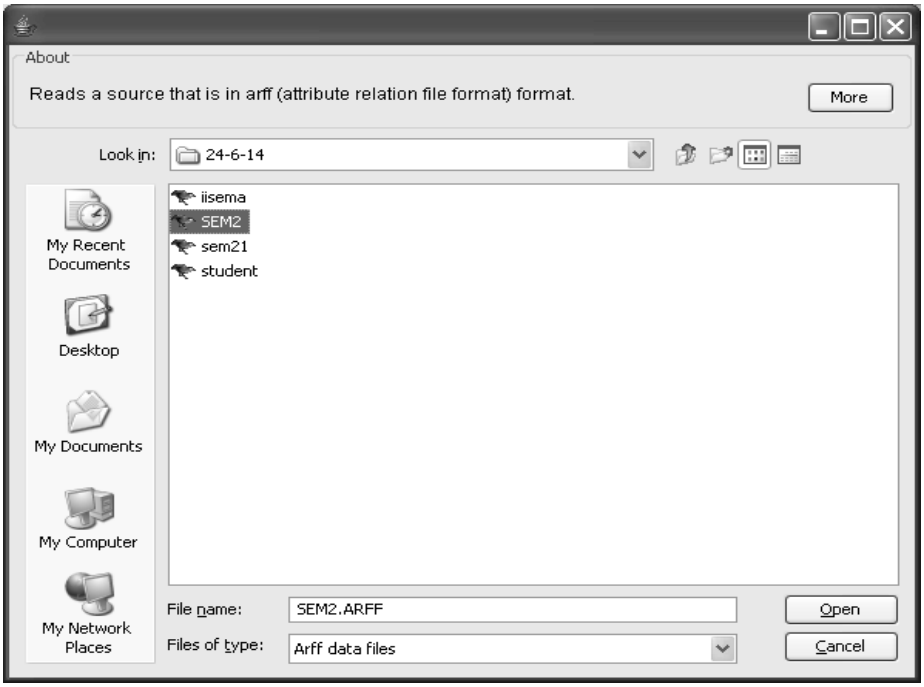
Step-1: - Click on start button and then select All Programs and choose WEKA 3.8.3 in the WEKA 3.8(with console).

Step-2: - In the WEEKA GUI to select the knowledge flow.

Step-3: - Select the data sources and then select ARFF loader and dragon knowledge flow layout.

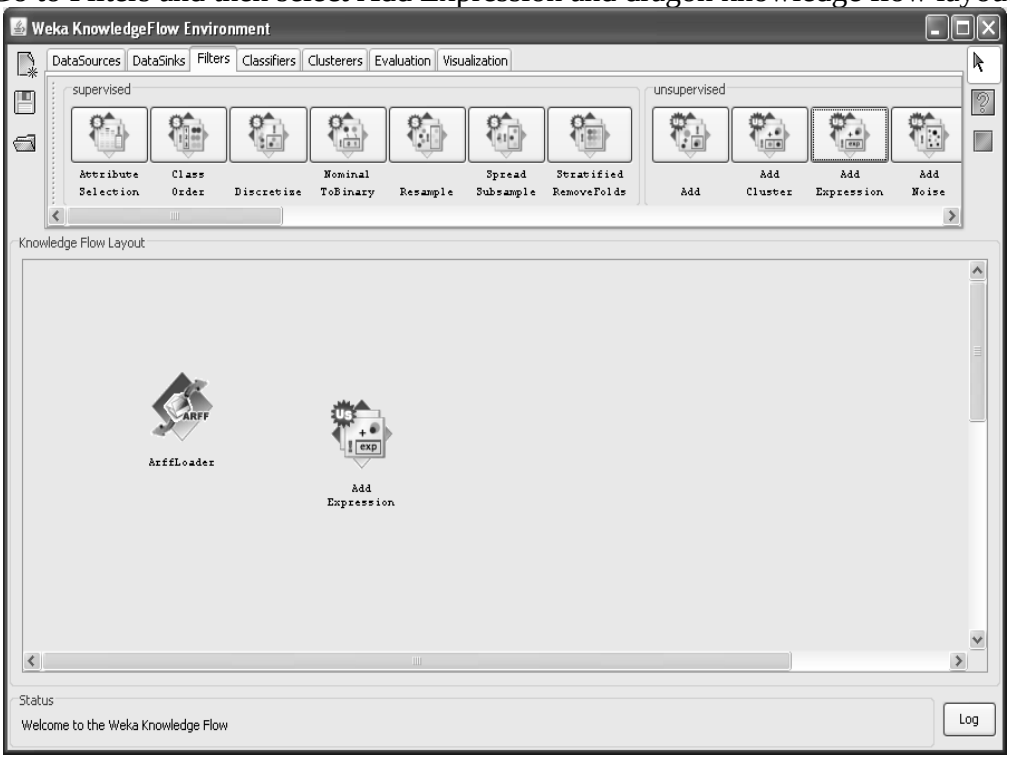


Step-4: - Select ARFF File right click in click configure to attach the file.

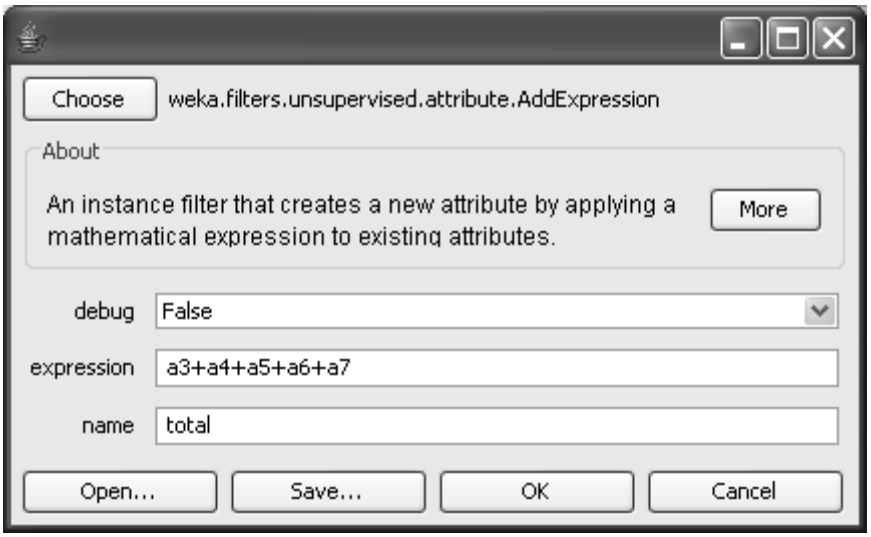


Click on open button.

Step-5: Go to Filters and then select Add Expression and dragon knowledge flow layout.

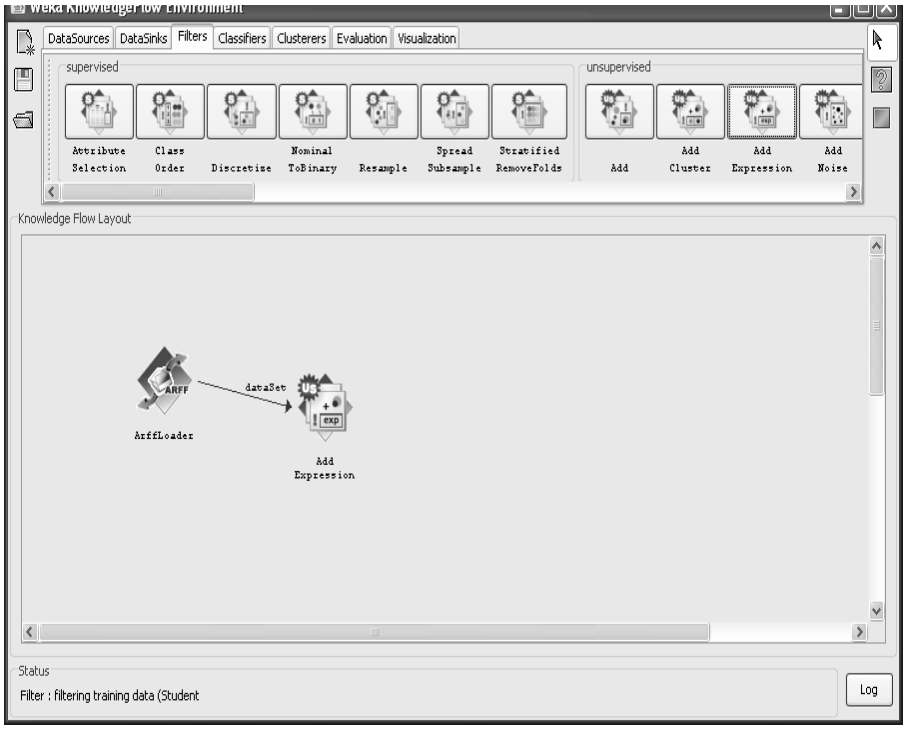


Step-6: - Select Add Expression and right click on select configure.

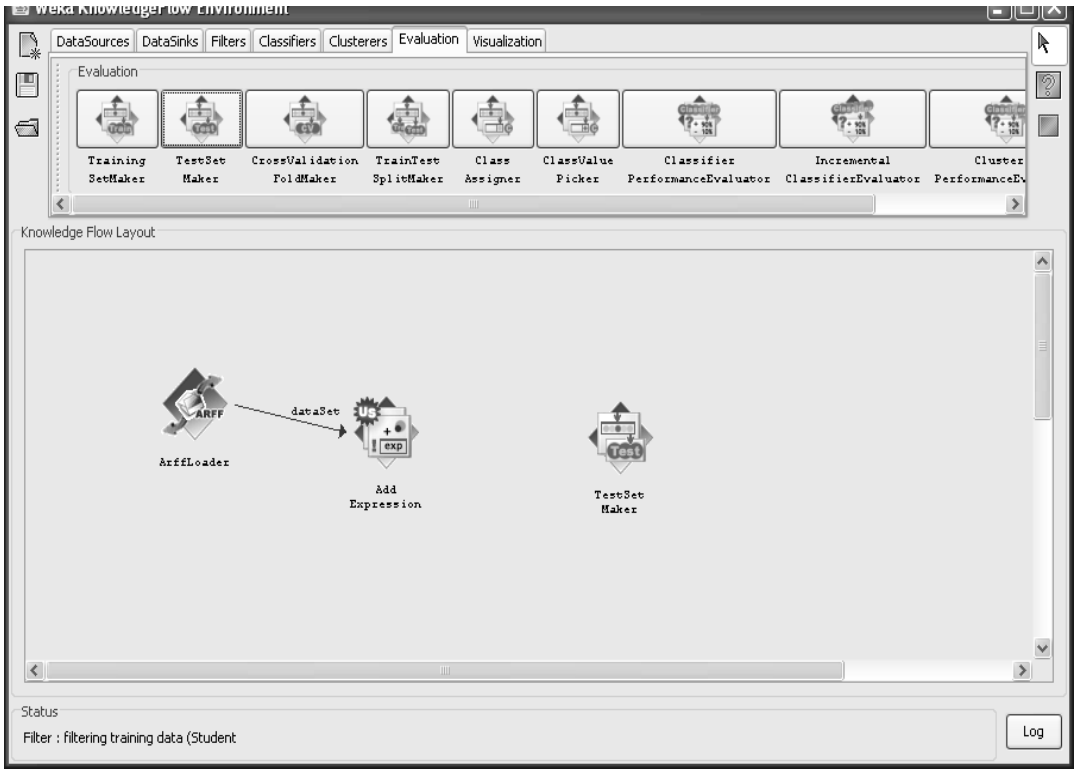


Click on ok button.

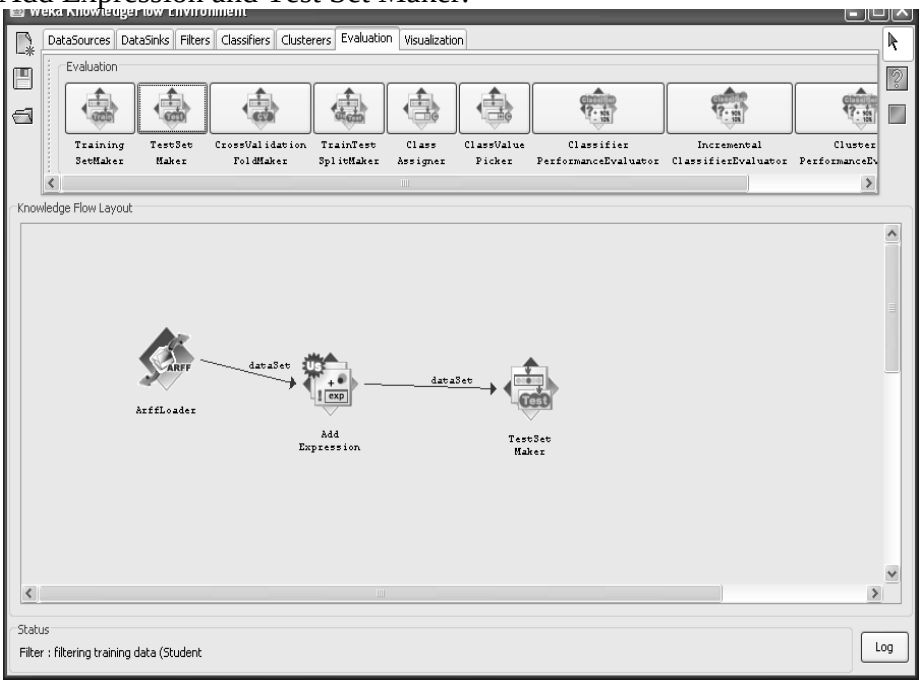
Step-7: - Select ARFF file to connection between the Add Expression to right click on ARFF file to select Data set.



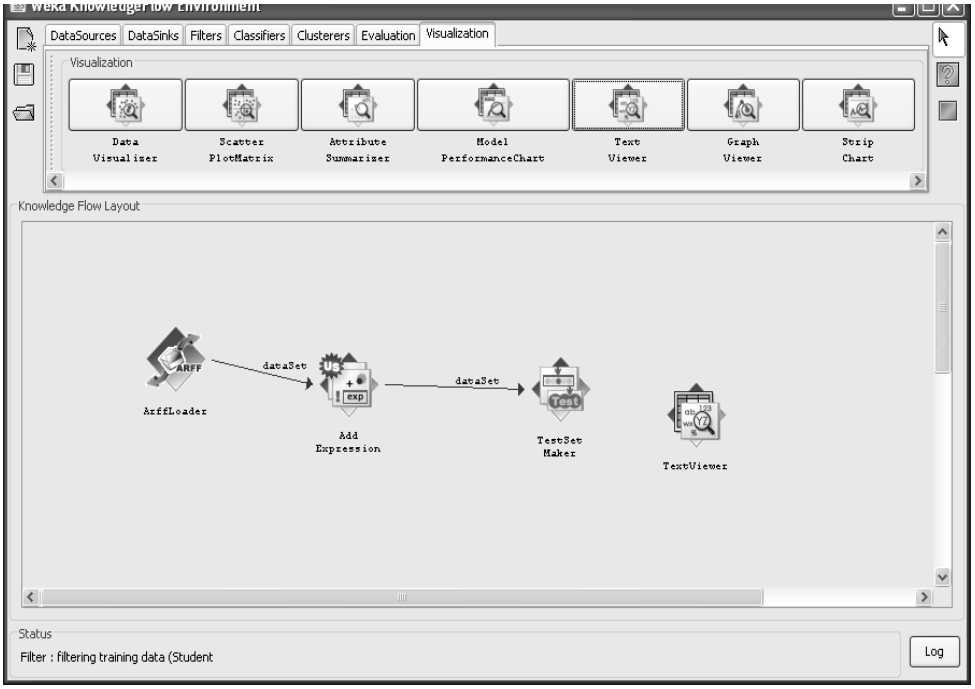
Step-8: - Go to Evaluation and select Test Set Maker drag on knowledge flow layout.



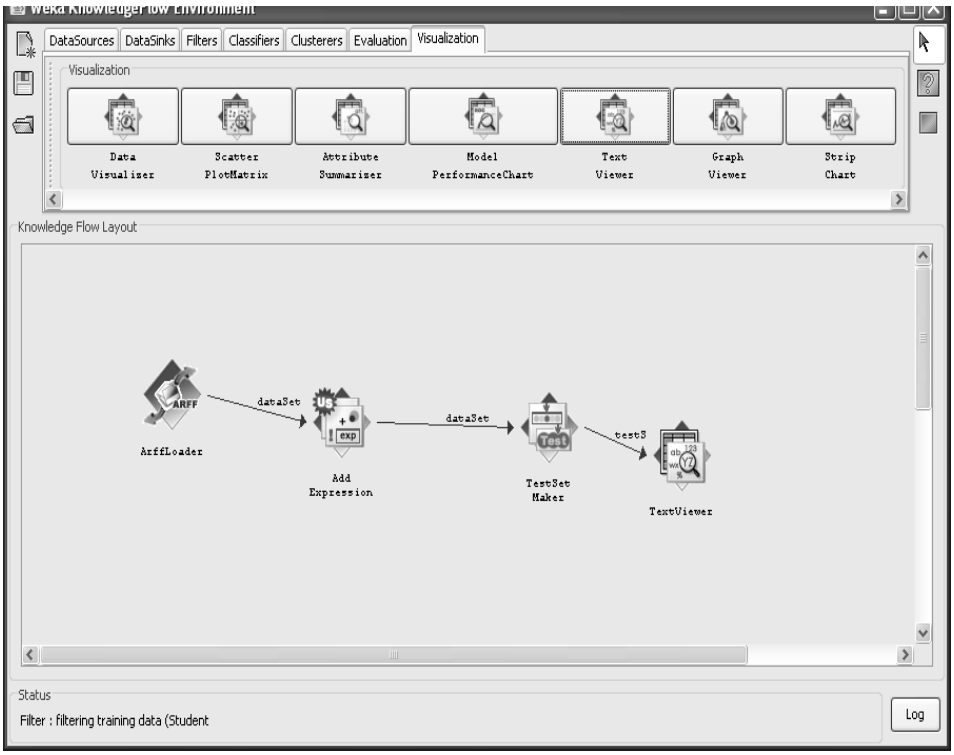
Step-9: - Select Add Expression and right click on to select Data set option to connection between the Add Expression and Test Set Maker.



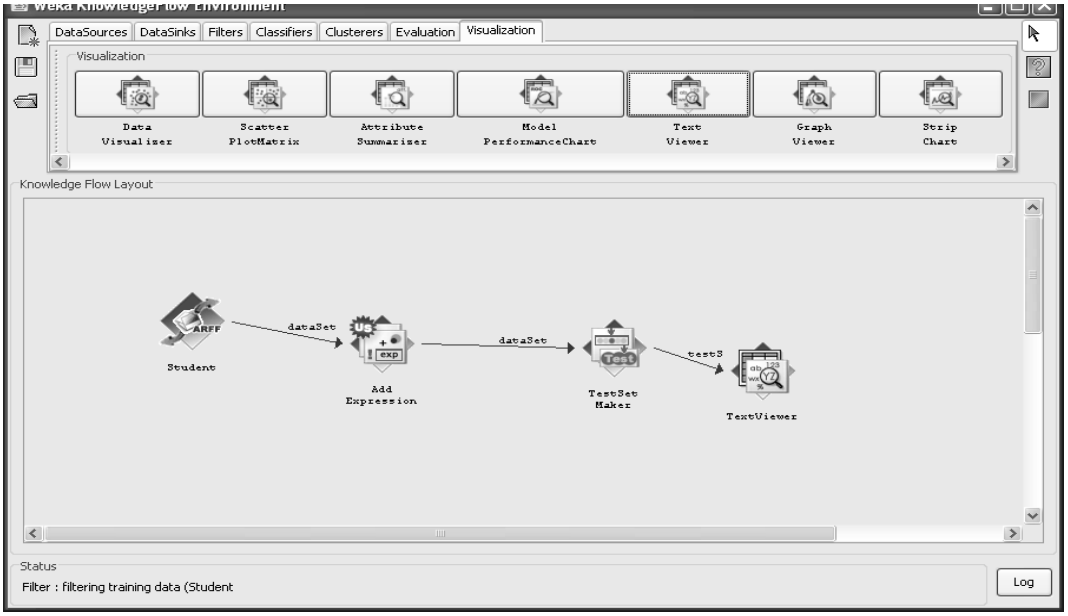
Step-10: - Go to visualization and select Text Viewer to drag on knowledge flow layout.



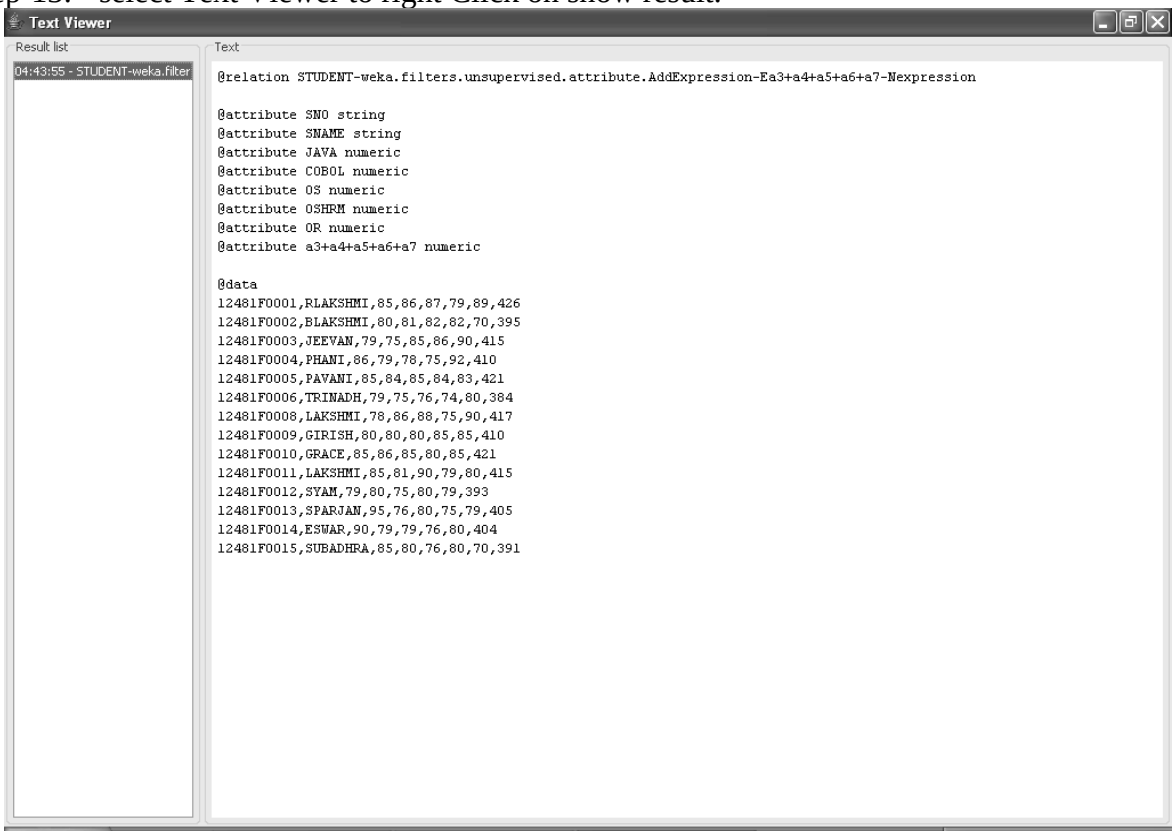
Step-11: - select Test Set maker to right click on to select test set to connetion between Test Set Maker and Text Viewer.



Step-12: - select ARFF Loader to right click on start loading.



Step-13: - select Text Viewer to right Click on show result.

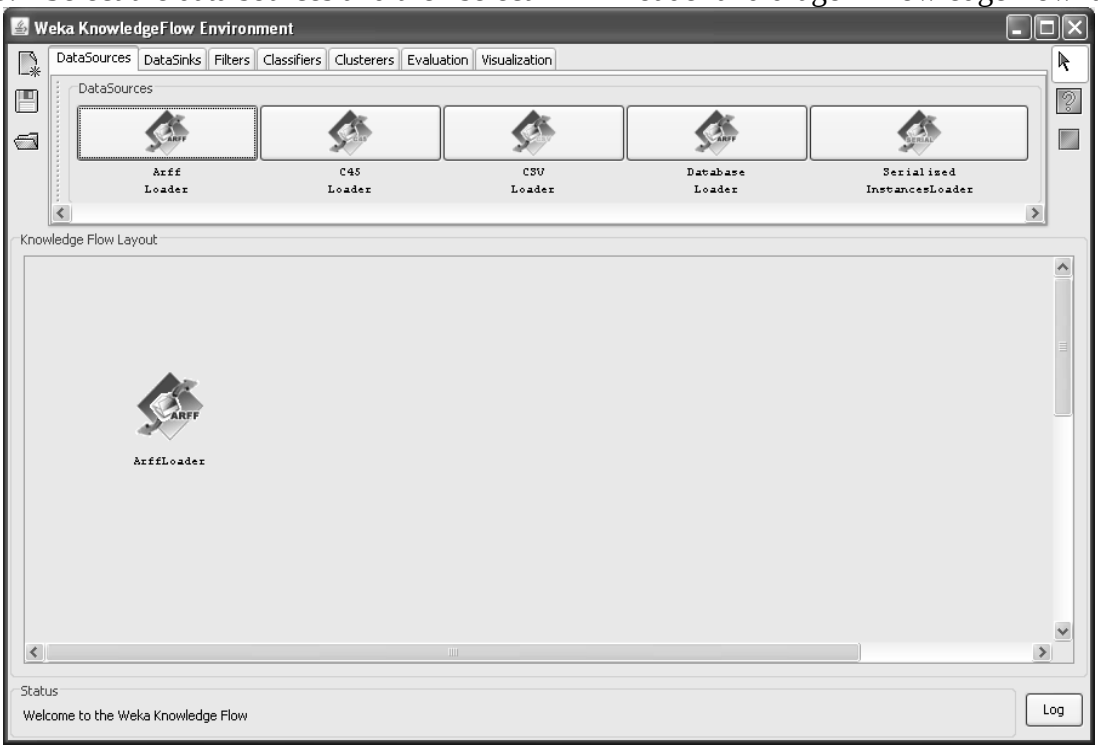


Copy Attribute.

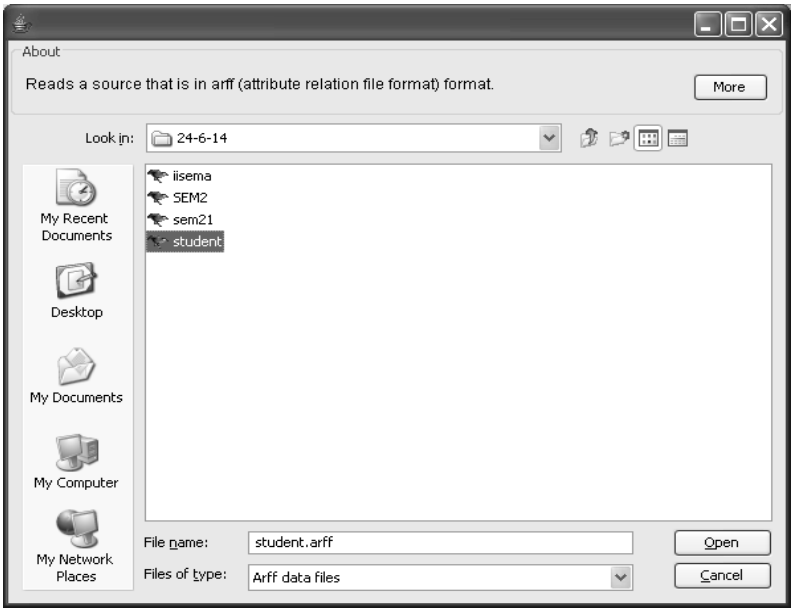
Step-1: - Click on start button and then select All Programs and choose WEKA 3.8.3 in the WEEKA 3.8(with console).

Step-2: - In the WEEKA GUI to select the knowledge flow.

Step-3: - Select the data sources and then select ARFF loader and dragon knowledge flow layout.

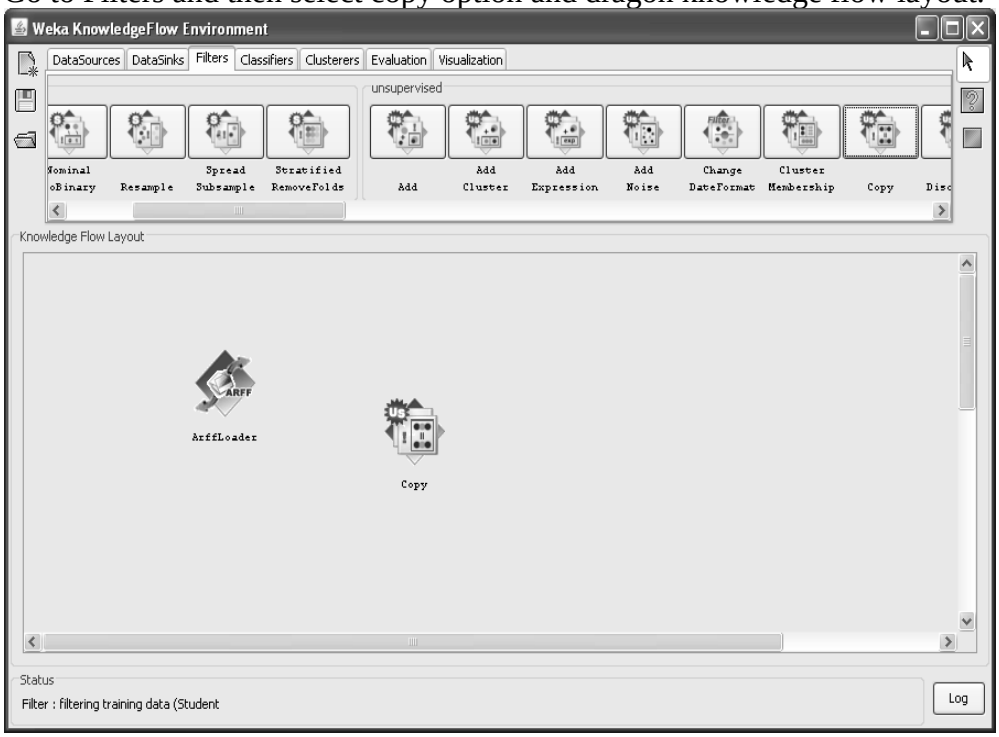


Step-4: - Select ARFF File right click in click configure to attach the file.

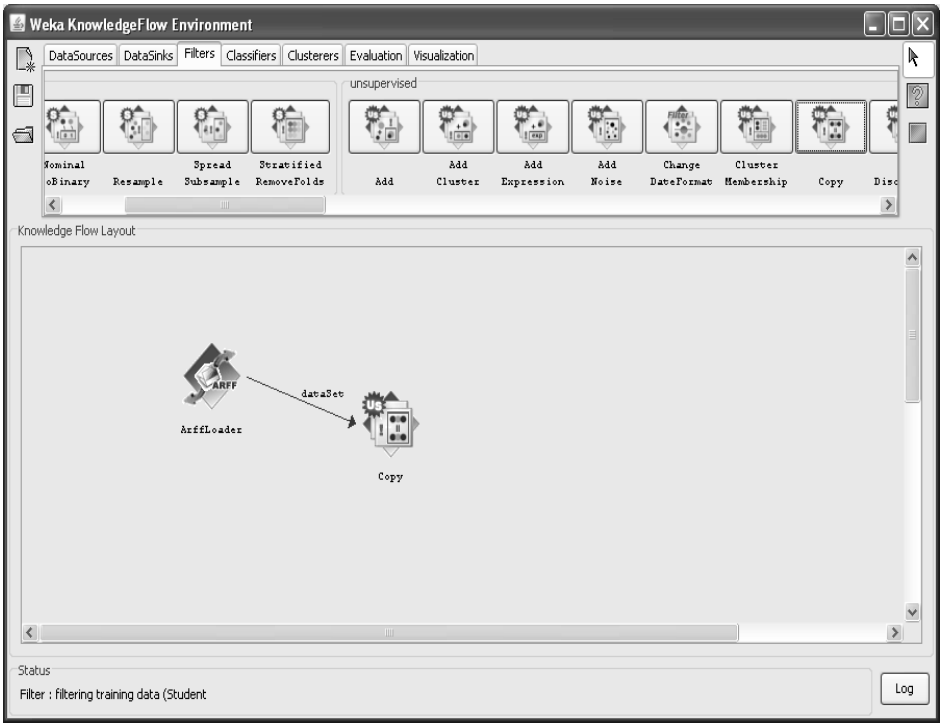


Click on open button.

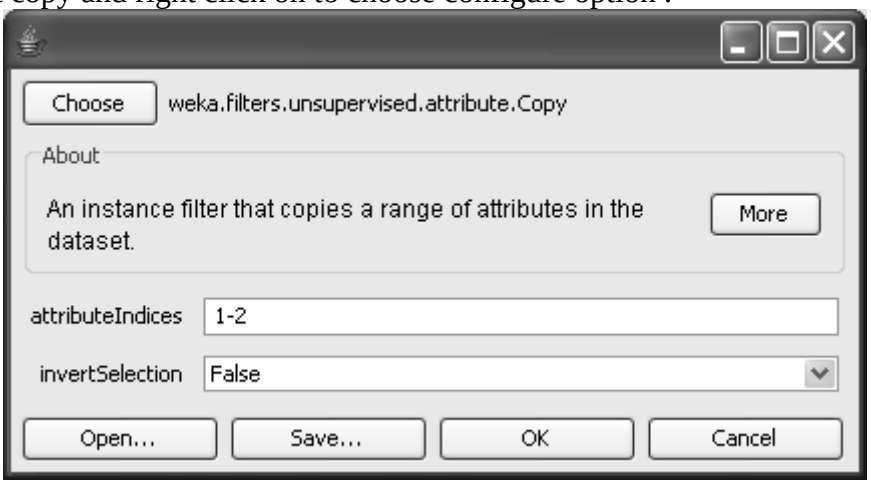
Step-5: - Go to Filters and then select copy option and dragon knowledge flow layout.



Step-6: - select the ARFF Loader and right click to choose Data Set to create connection between ARFF Loader and copy.

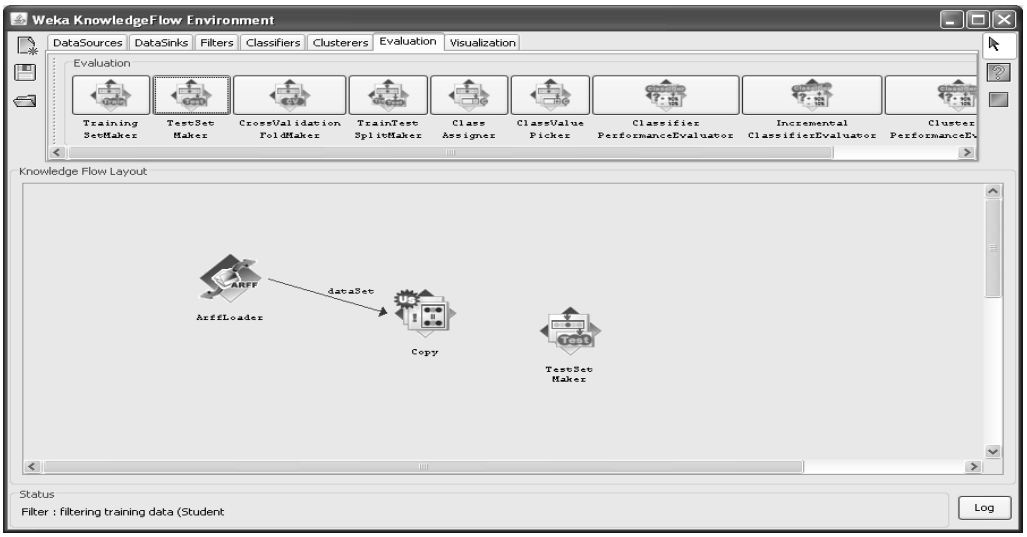


Step-7: - select copy and right click on to choose configure option .

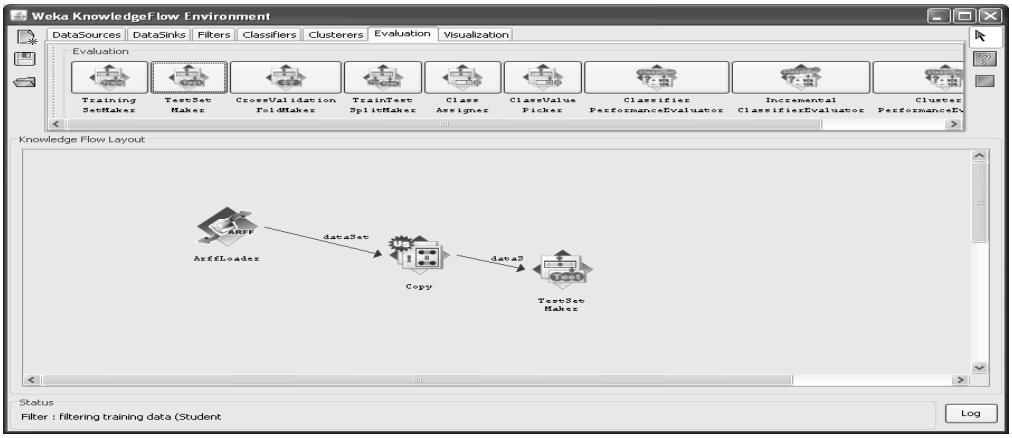


Click on ok button.

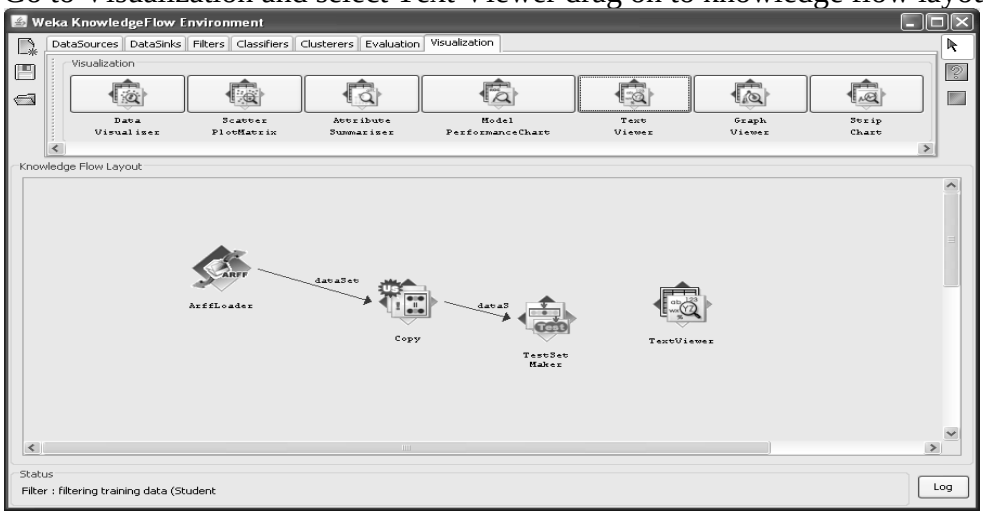
Step-8: - Go to Evaluation and select Test Set Maker drag on to knowledge flow layout.



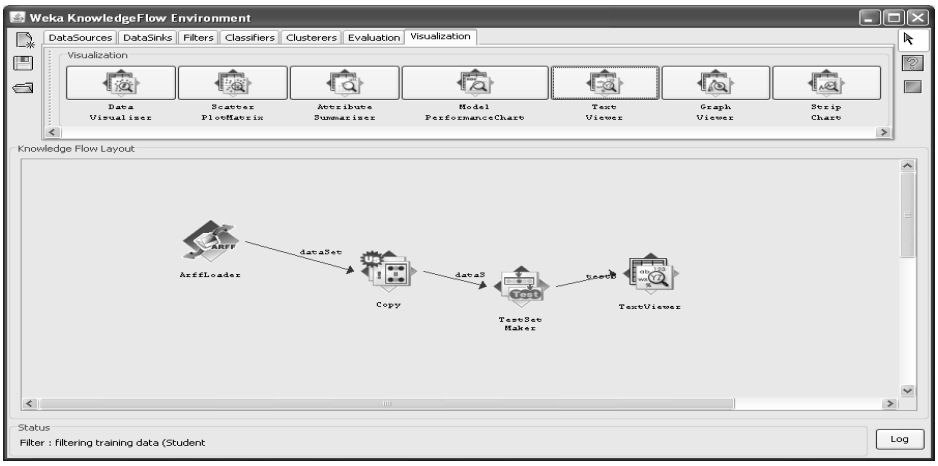
Step-9: - select copy and right click on to choose Data Set to connection between the copy and Test Set Maker.



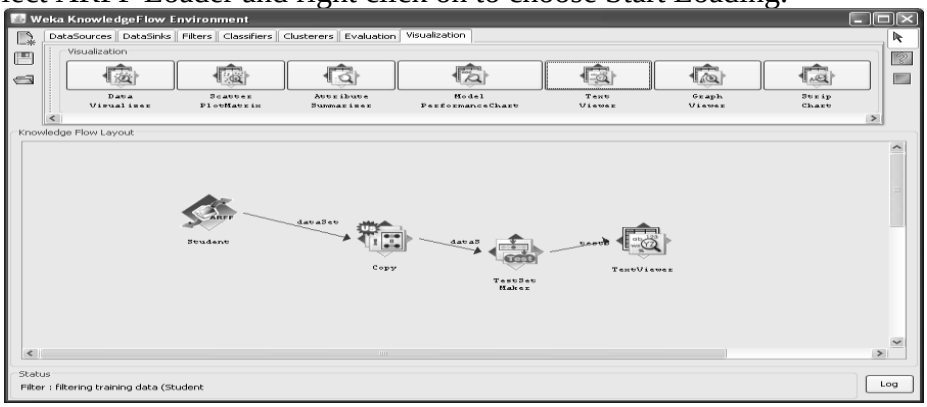
Step-10: - Go to Visualization and select Text Viewer drag on to knowledge flow layout.



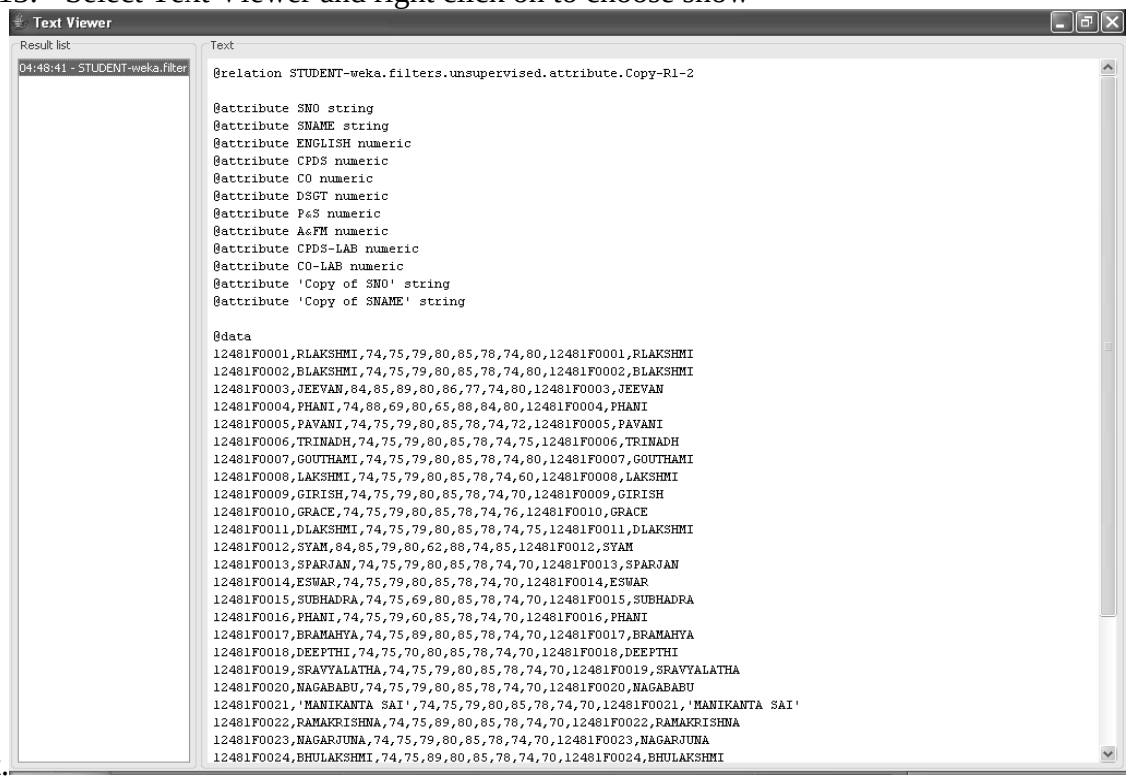
Step-11: - select Test Set Maker and right click on to choose Test Set to connection between Test Set Maker to Text Viewer.



Step-12: - select ARFF Loader and right click on to choose Start Loading.



Step-13: - Select Text Viewer and right click on to choose show result.

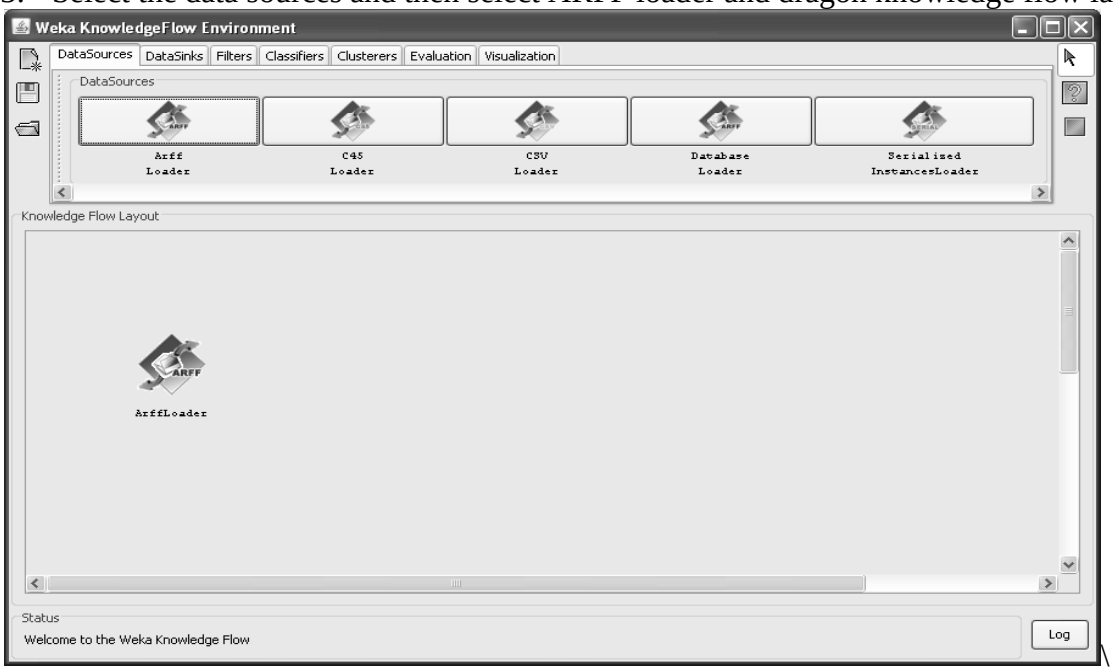


Remove Attribute

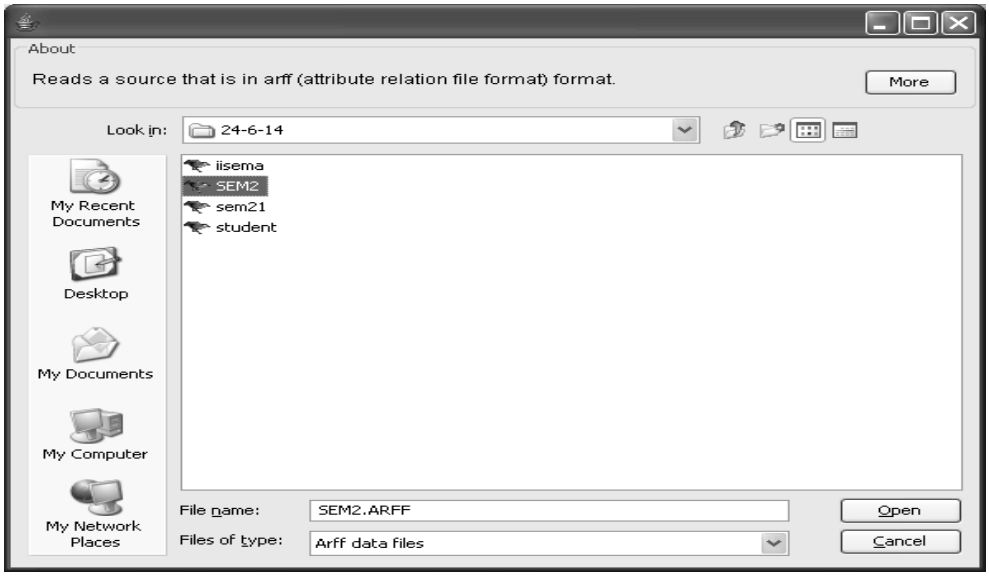
Step-1: - Click on start button and then select All Programs and choose WEKA 3.8.3 in the WEKA 3.8(with console).

Step-2: - In the WEEKA GUI to select the knowledge flow.

Step-3: - Select the data sources and then select ARFF loader and dragon knowledge flow layout.

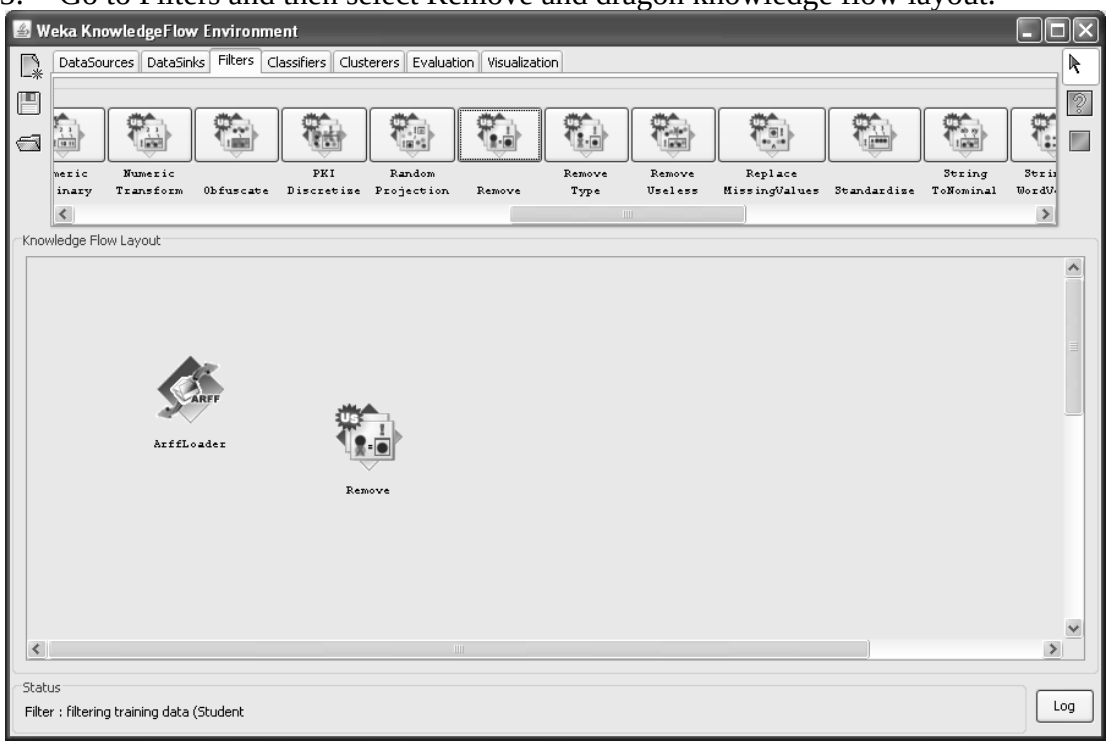


Step-4: - Select ARFF File right click in click configure to attach the file.

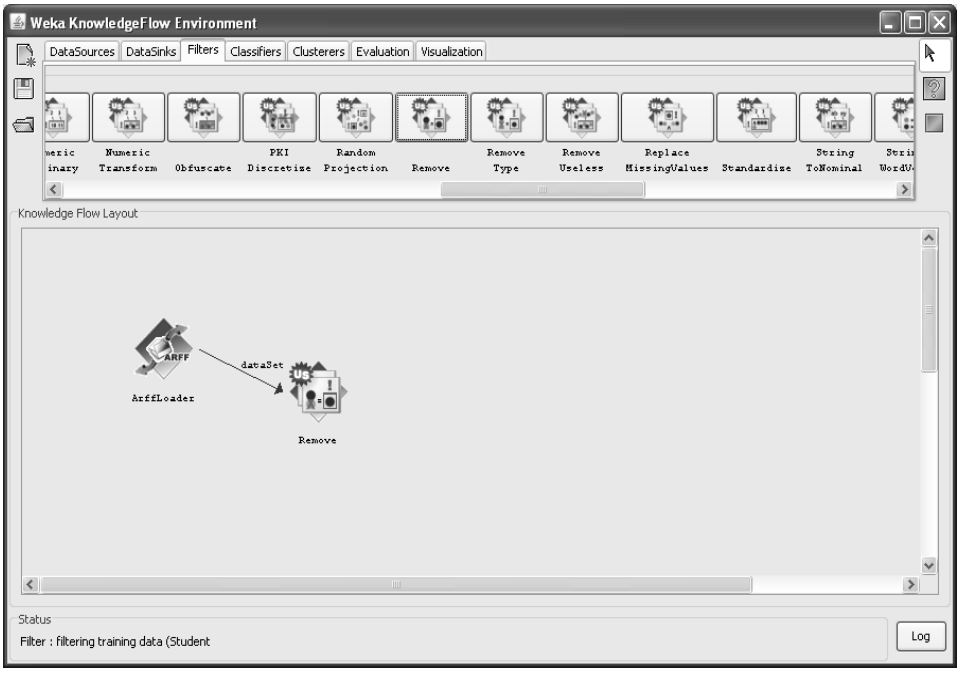


Click on open button.

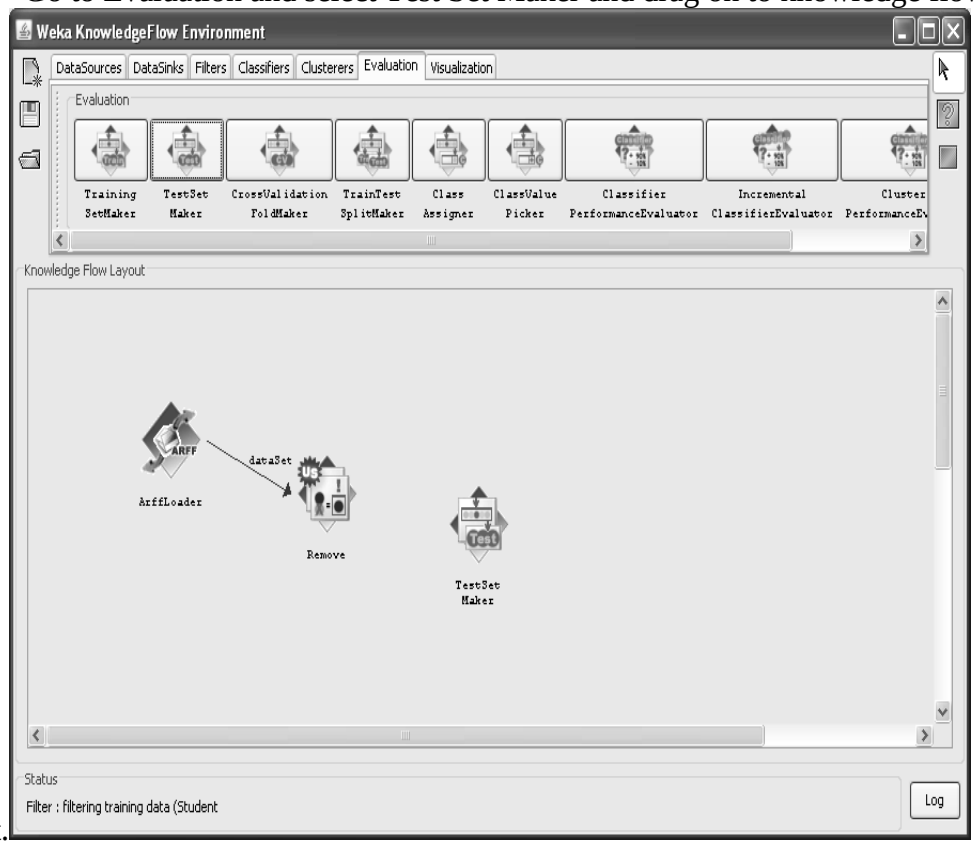
Step-5: - Go to Filters and then select Remove and dragon knowledge flow layout.



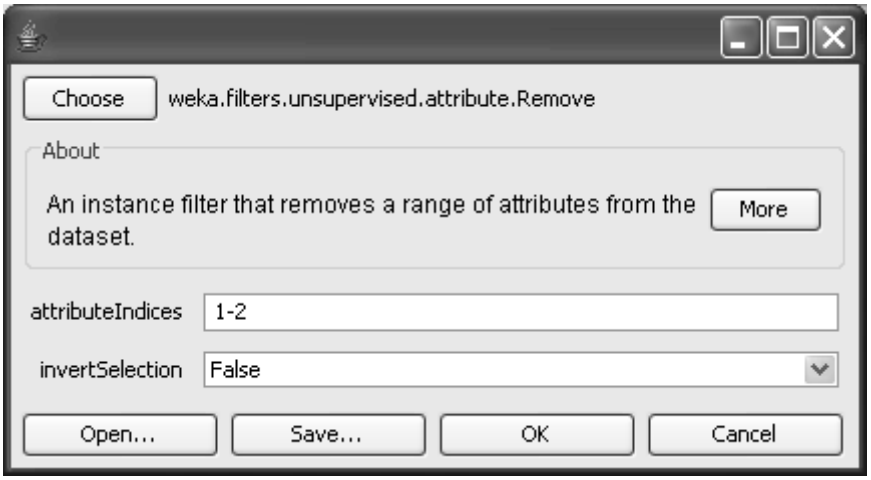
Step-6: - select ARFF Loader and right click on to choose Data Set to connection between the ARFF Loader and Remove.



Step-7: - Go to Evaluation and select Test Set Maker and drag on to knowledge flow layout.

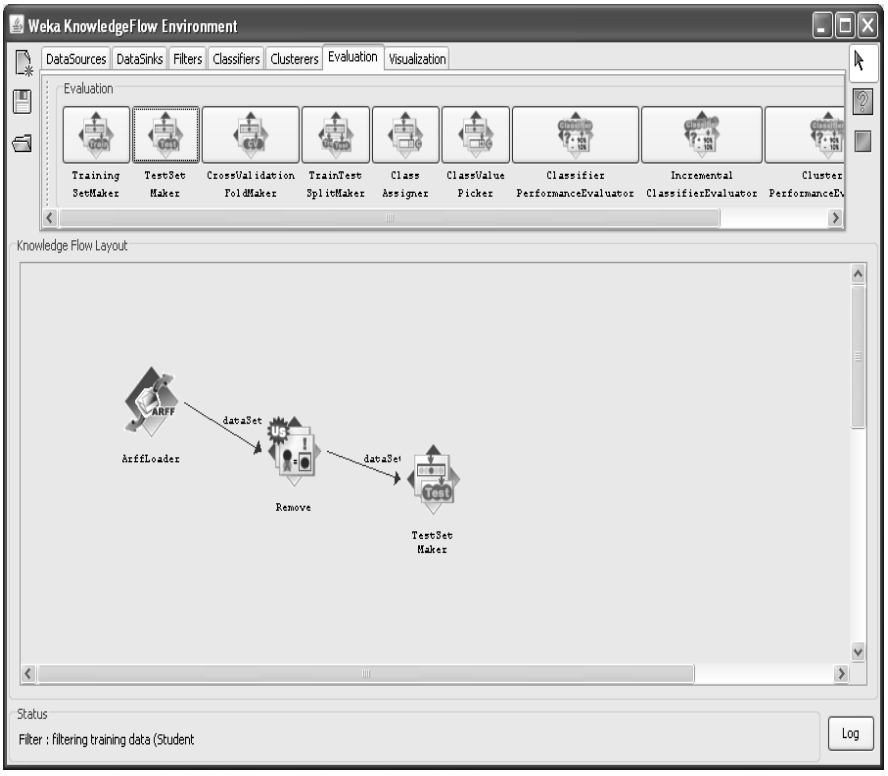


Step-8: -select Remove and right click on to choose configure .

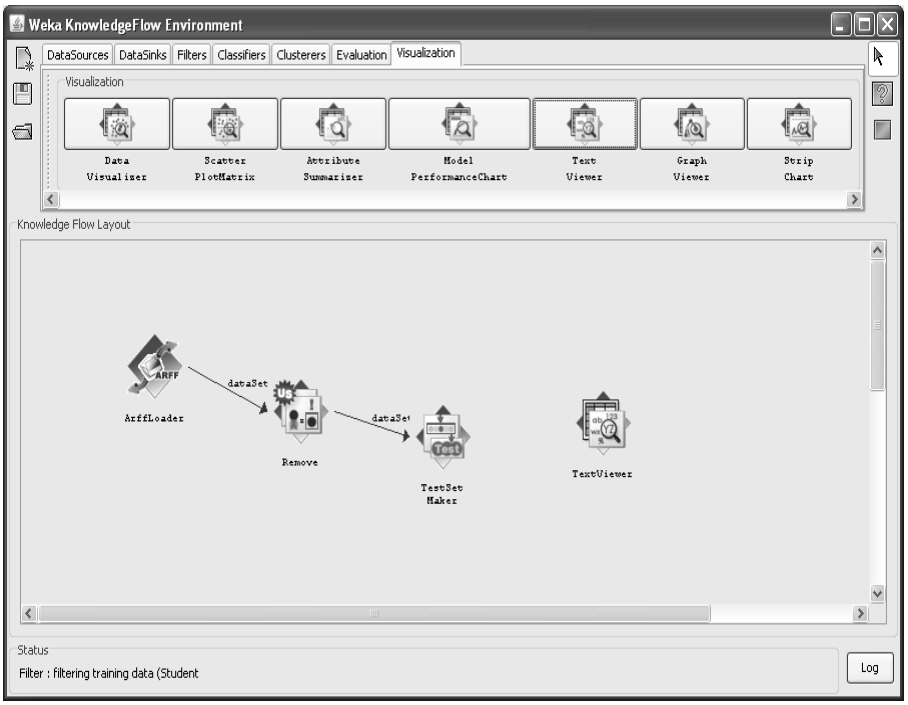


Click on ok button.

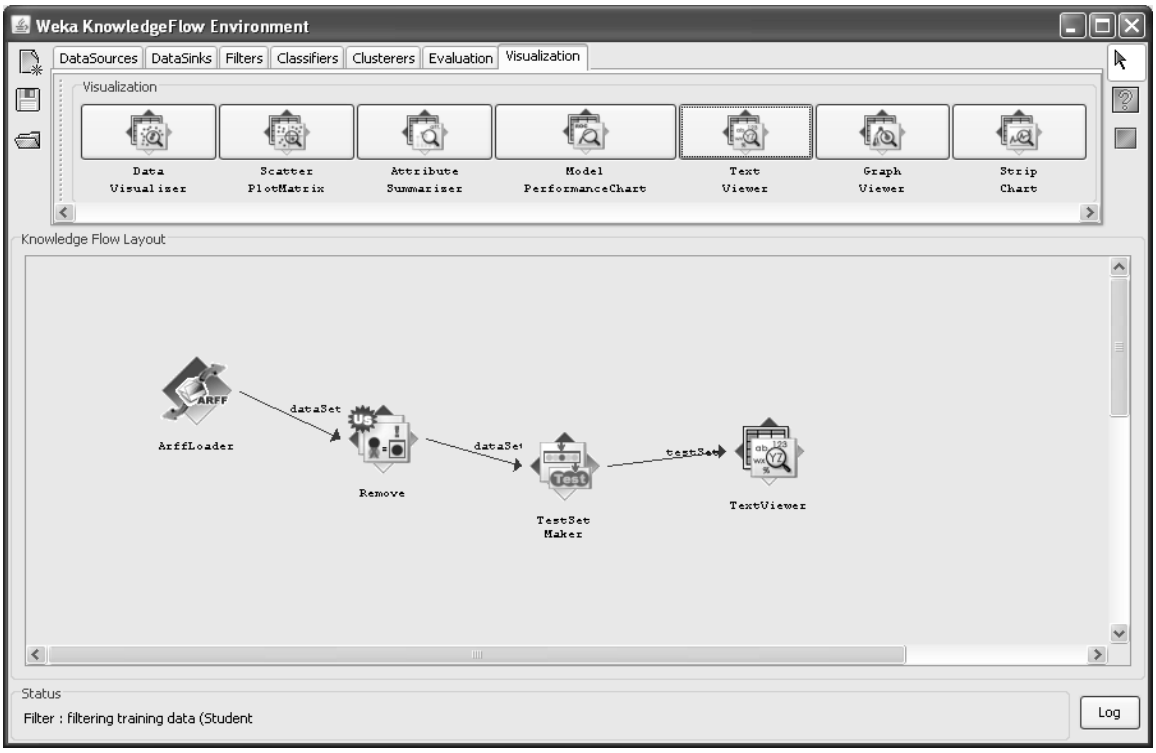
Step-9: - select Remove and right click on to choose Data Set to connection between to Remove and Test Set Maker.



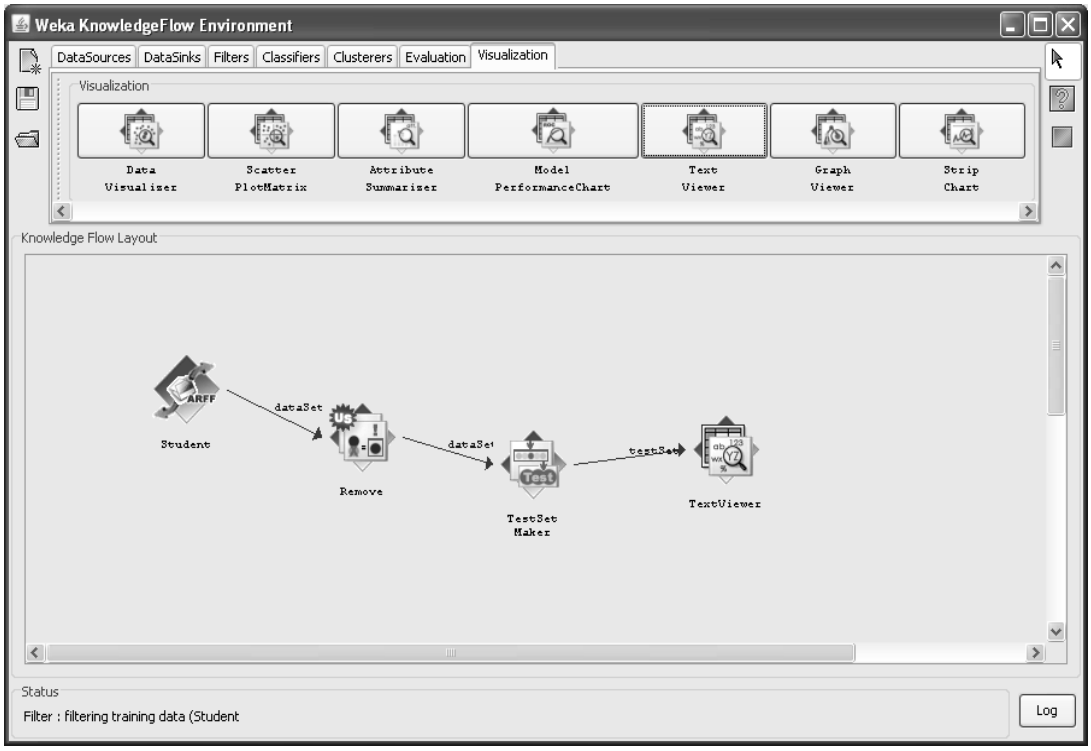
Step-10: -Go to Visualization and select Text Viewer drag on to knowledge flow layout.



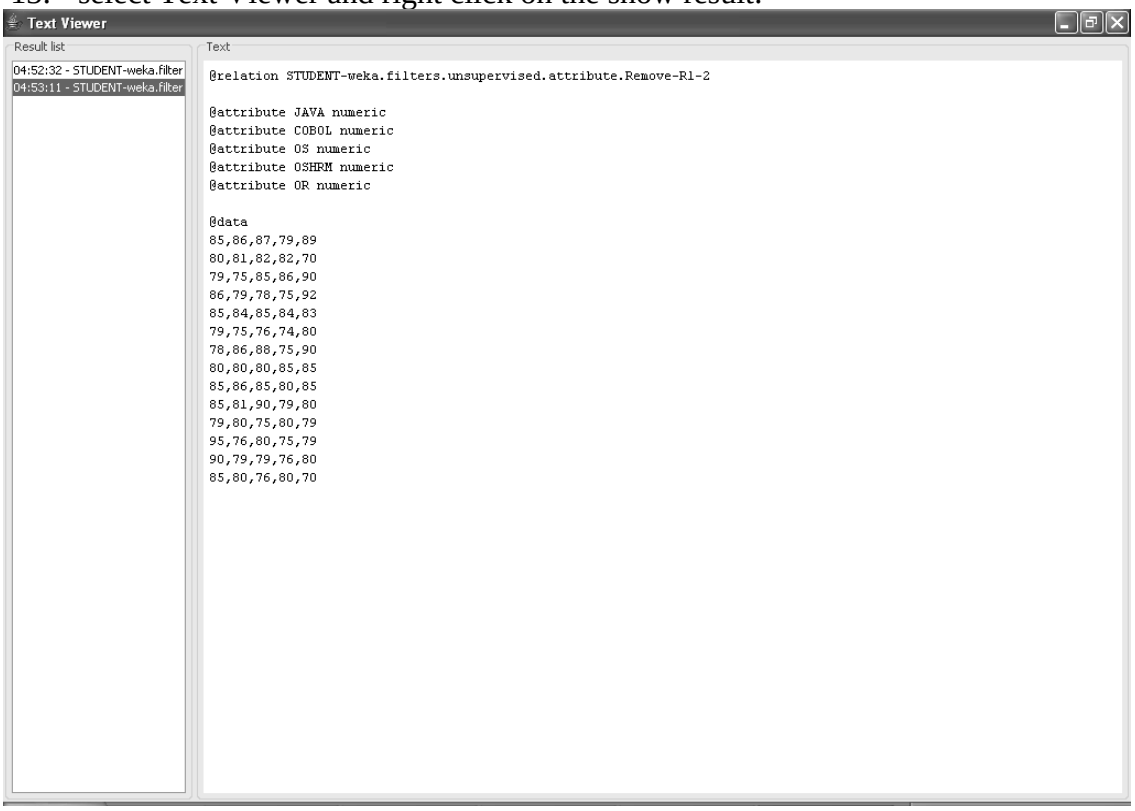
Step-11: - select Test Set Maker and right click on choose Test Set to connection between Test Set Maker to Text Viewer.



Step-12: - Select ARFF Loader and right click on start loading.



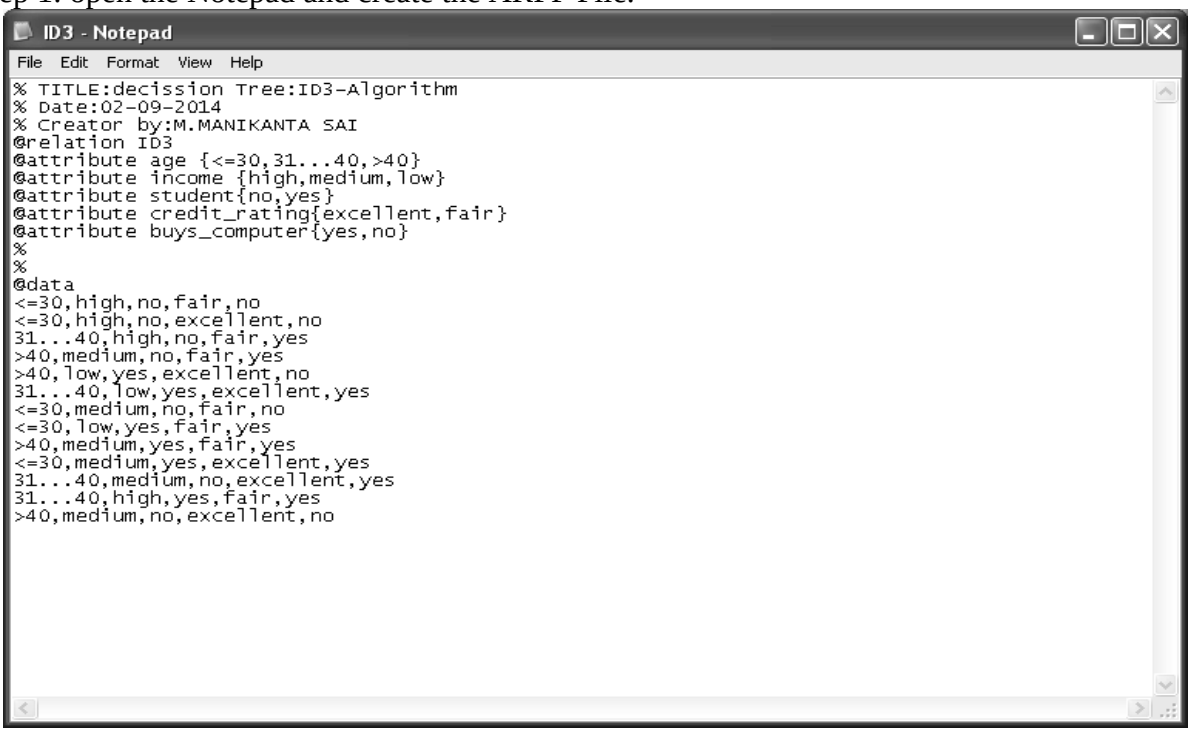
Step-13: - select Text Viewer and right click on the show result.



KNOWLEDGE FLOW FOR DECISSION TREE

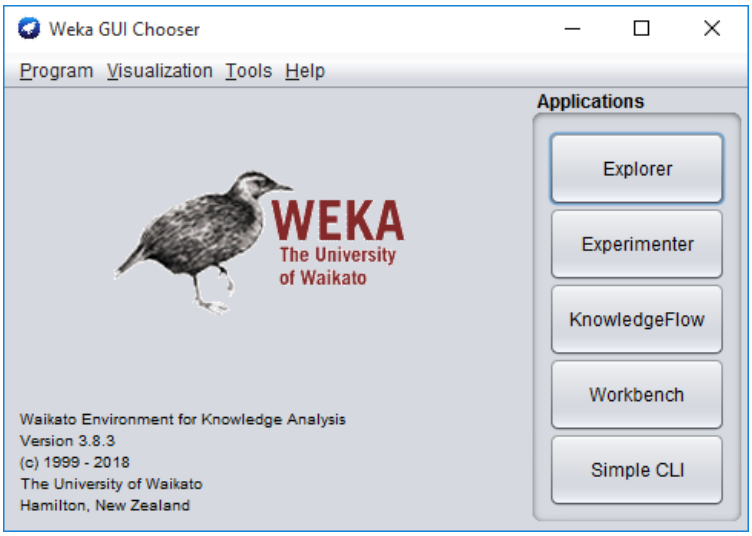
Knowledge flow for ID3 .

Step 1: open the Notepad and create the ARFF File.



Step2: To open All Programms ⟶weka3.8.3 ⟶ weka3.8(with console) .

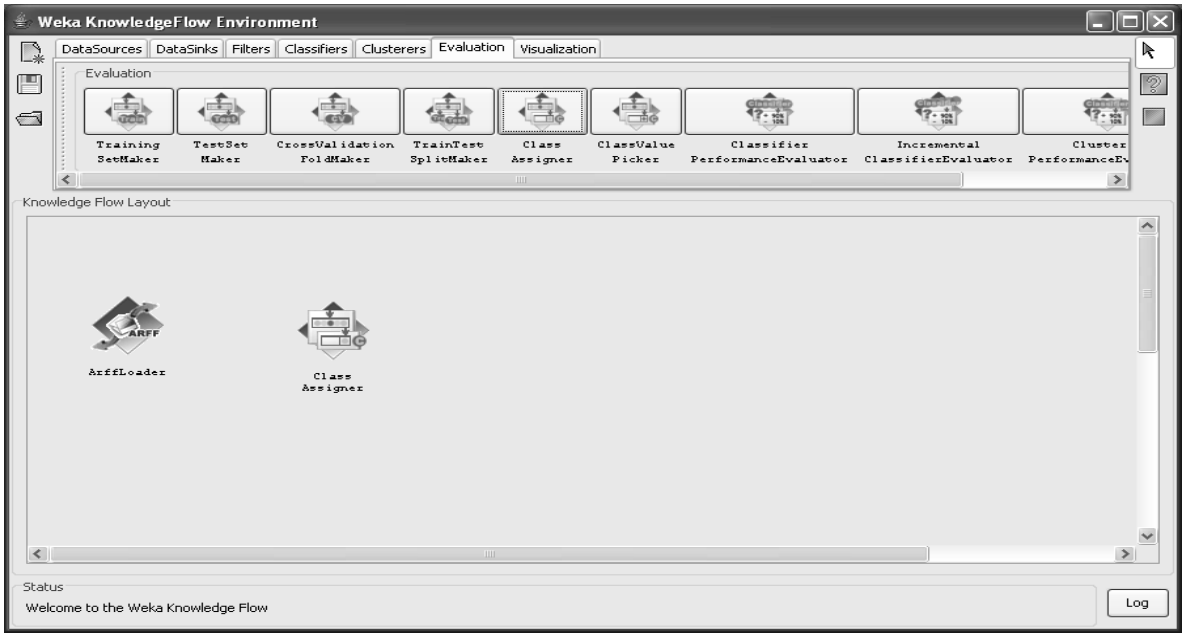
Step 3:click on the **Knowledge Flow.**



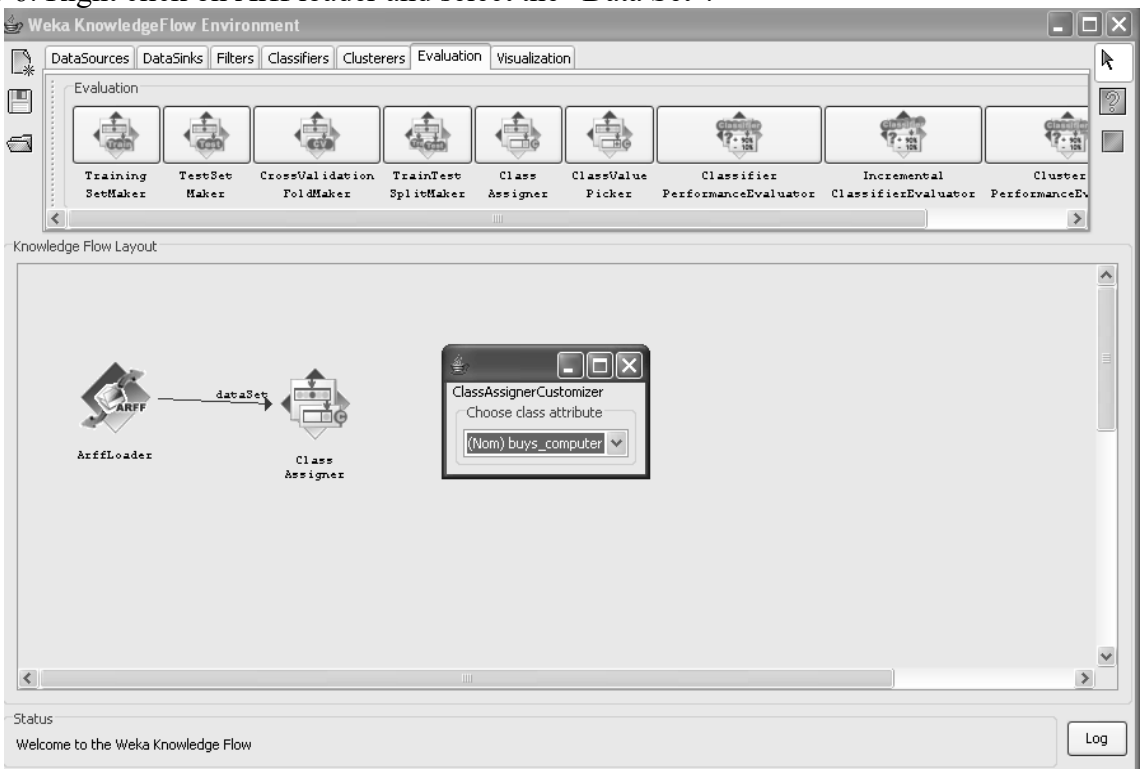
Step 4: click on the **Datasources** tab and drag the **Arff Loader ,**Right click over the Arff loader and select “configure” from the pop menu.

Click on **open.**

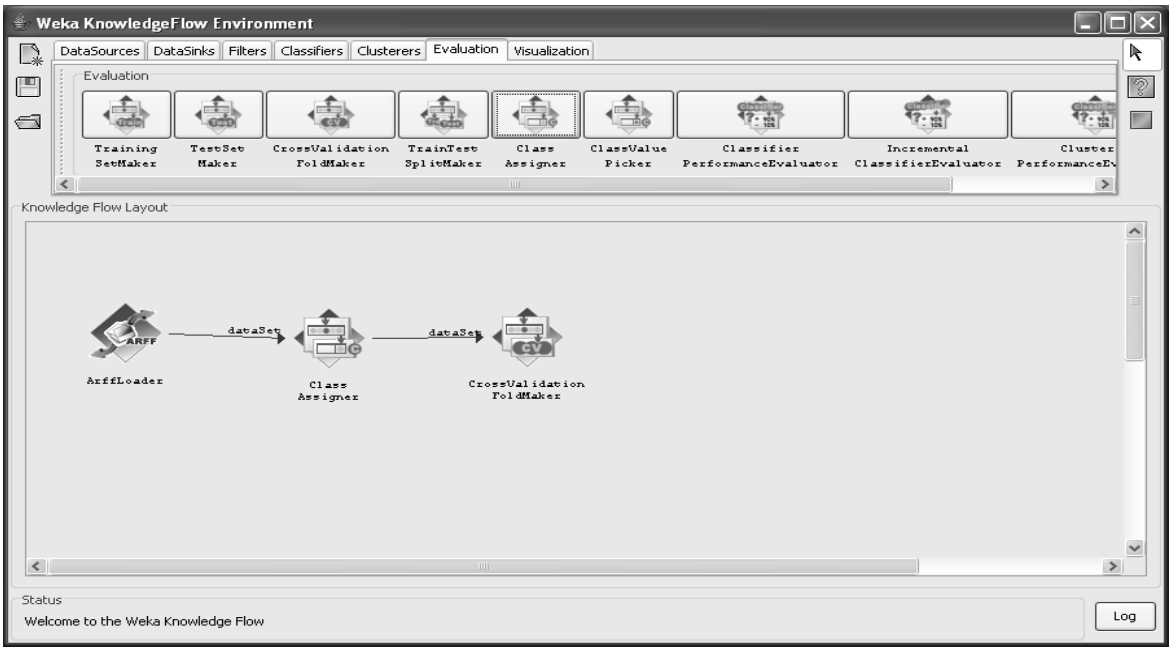
Step 5:click on the **Evaluation** tab at top of window,choose **class Assigner,**



Step 6: Right click on Arff loader and select the "Data Set".

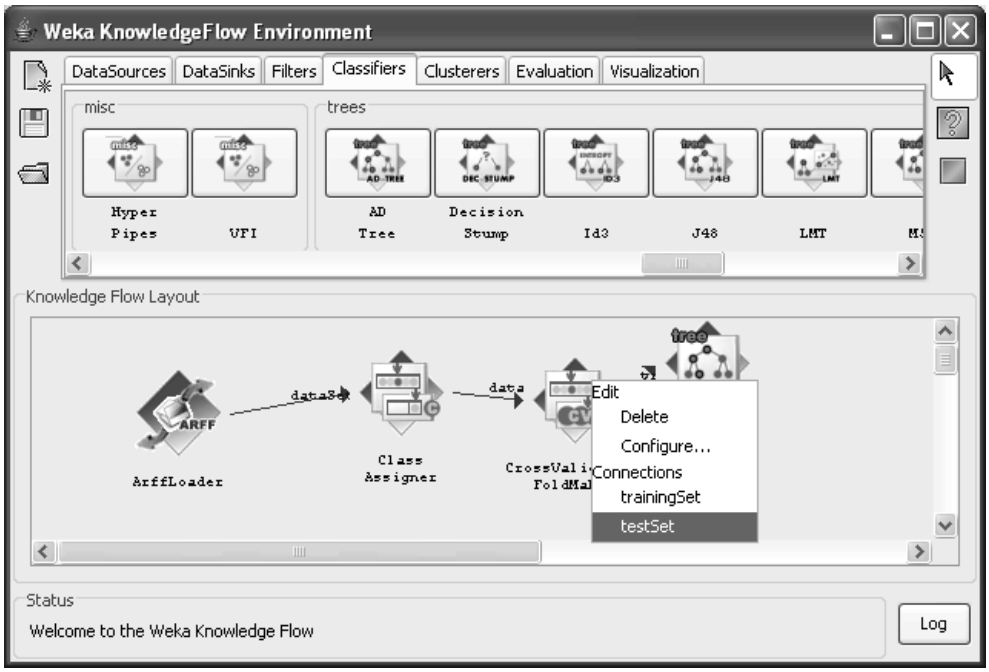


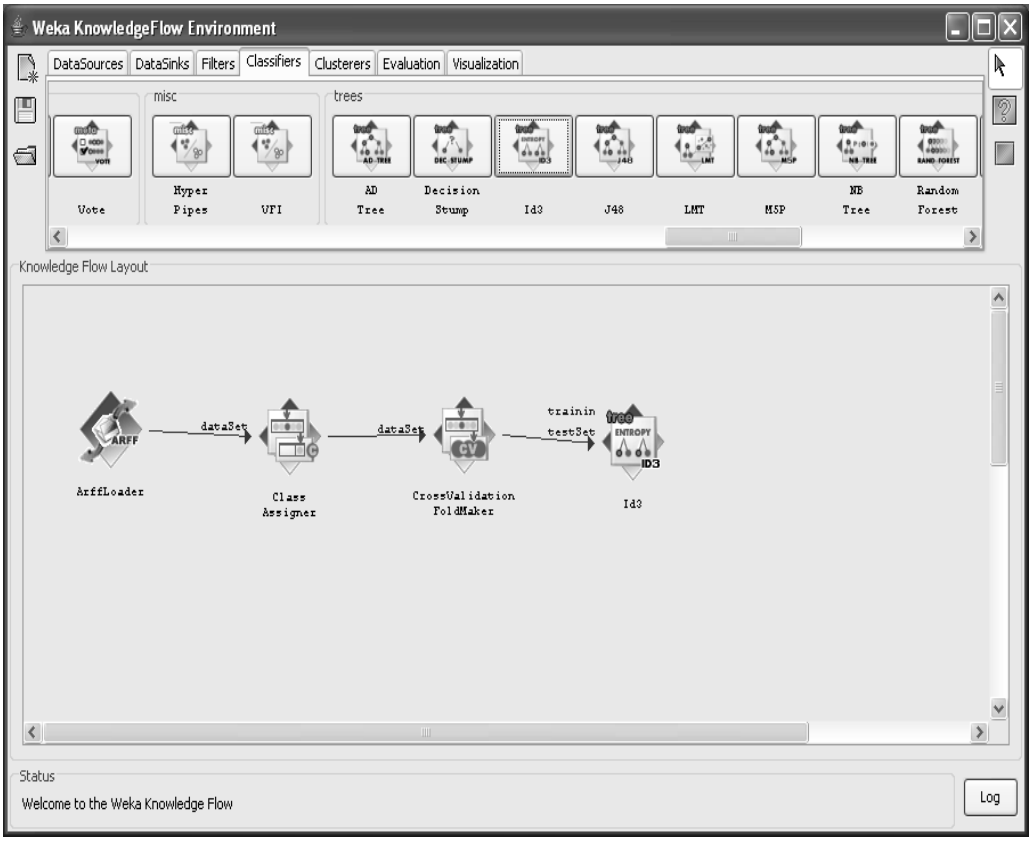
Step 7 : Choose cross validation fold maker, component from the evaluation toolbar
Step 8: connect class assigner to cross validation fold maker and right click over class assigner, select dataset under the connection menu.



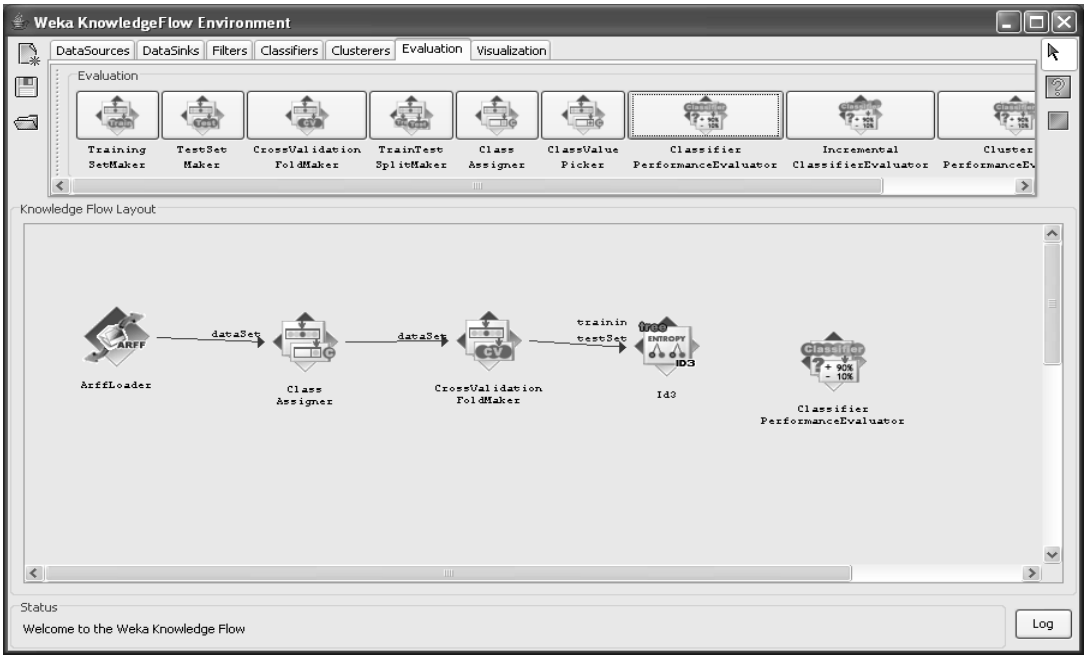
Step 9: click on classifiers tab on the window scroll the toolbar, specify ID3.

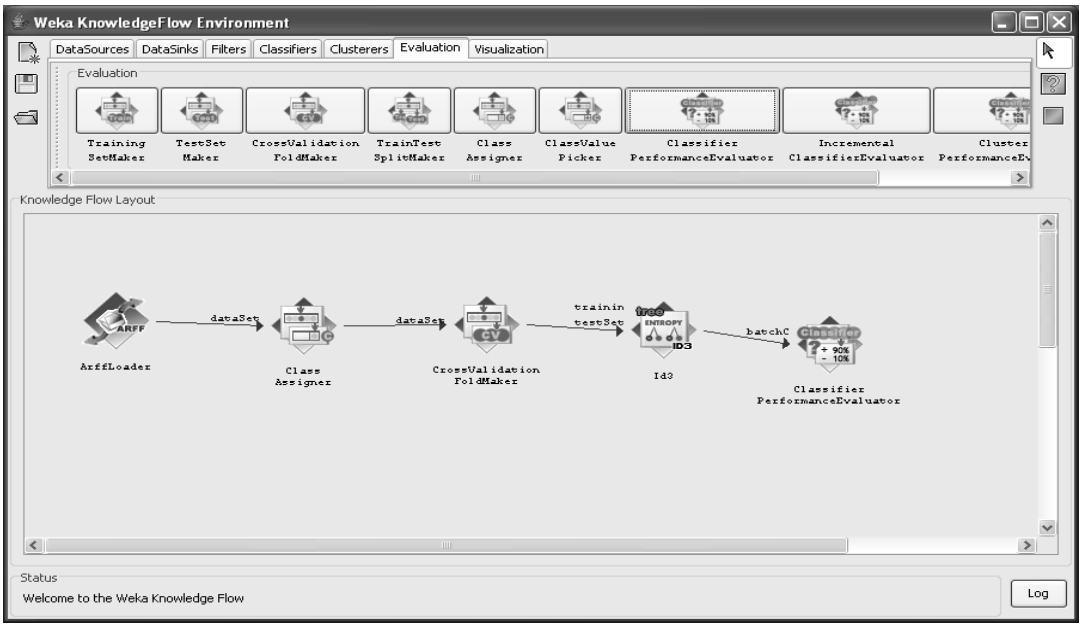
Step 10: connect the cross validation fold maker to ID3 by first choosing training set and then text set for the cross validation fold maker.





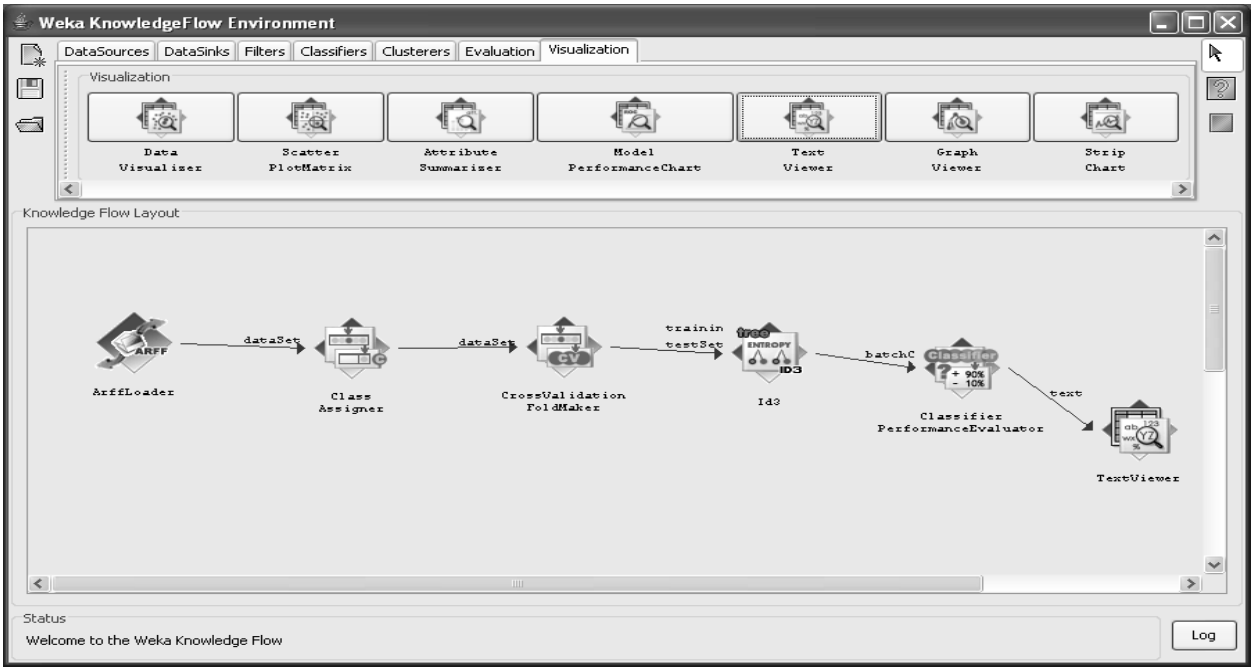
Step 11: Go to evaluation tab and place a classifier performance evaluator component on the layout Connect ID3 to this component by selecting the batch classifier entry from the popup menu for ID3.





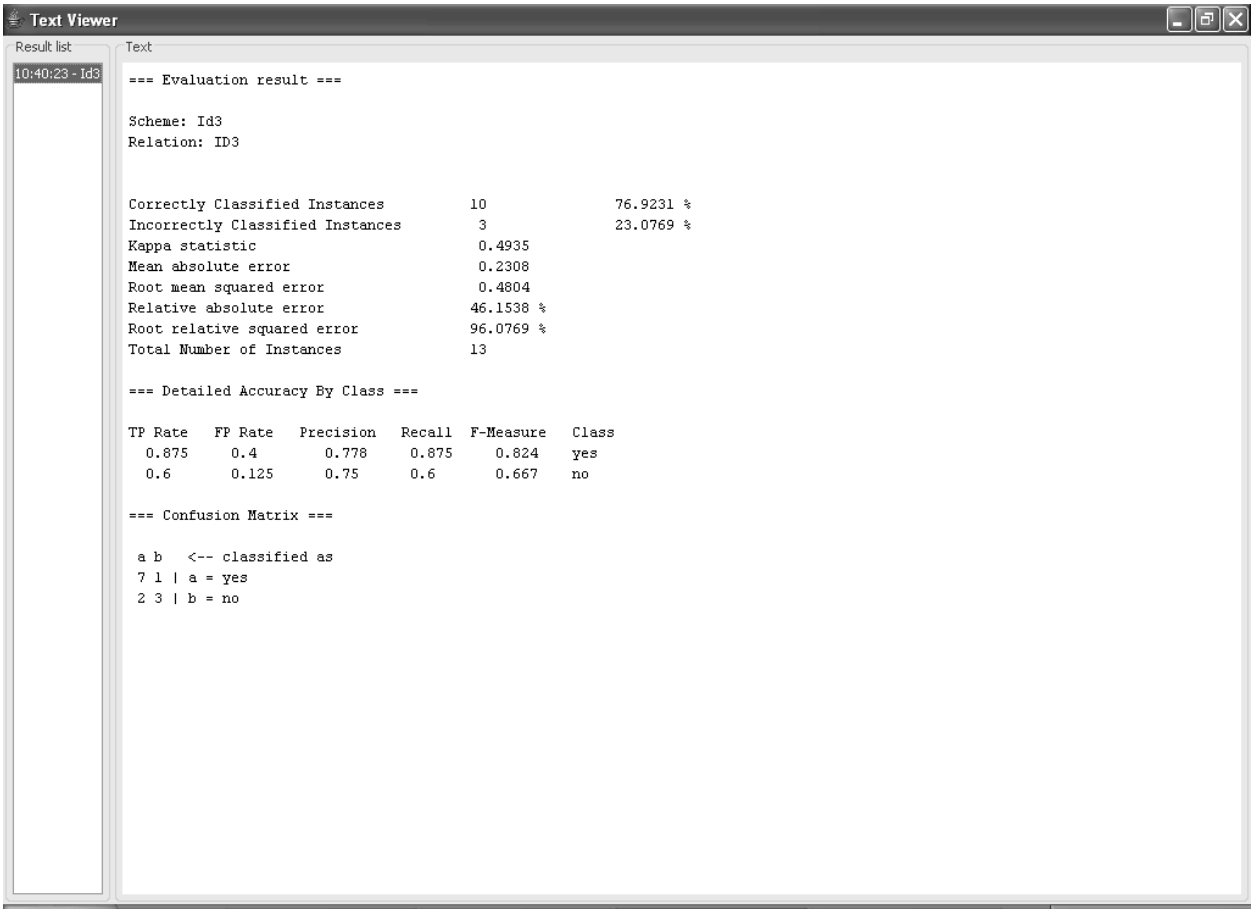
Step13: Go to visualization toolbar and place text viewer component on the layout. connect the classifier performance evaluator to the text viewer by selecting the text entry form the popup for classifier performance evaluator.

step14: Now start the flow executing by selecting start loading from the popup menu for arffloader we also see some progress information in the status bar and log at bottom of the window.



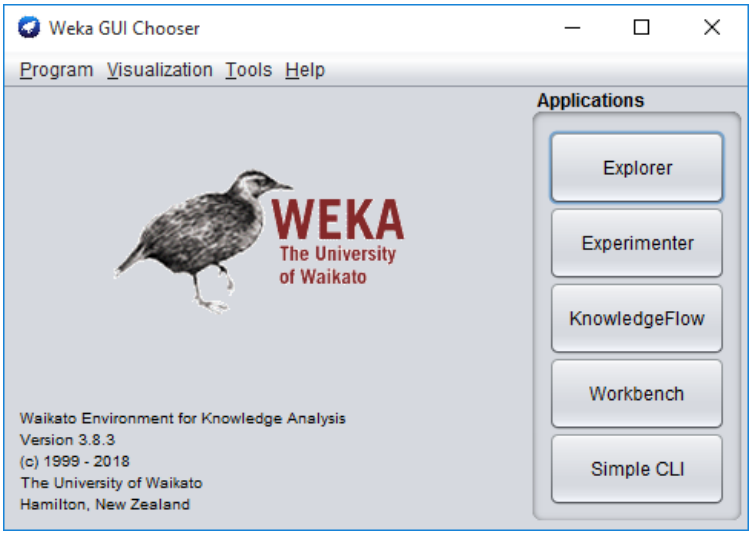
step15: when finished, you can view the results by choosing show results from the popup menu for the test viewer component.

Output:



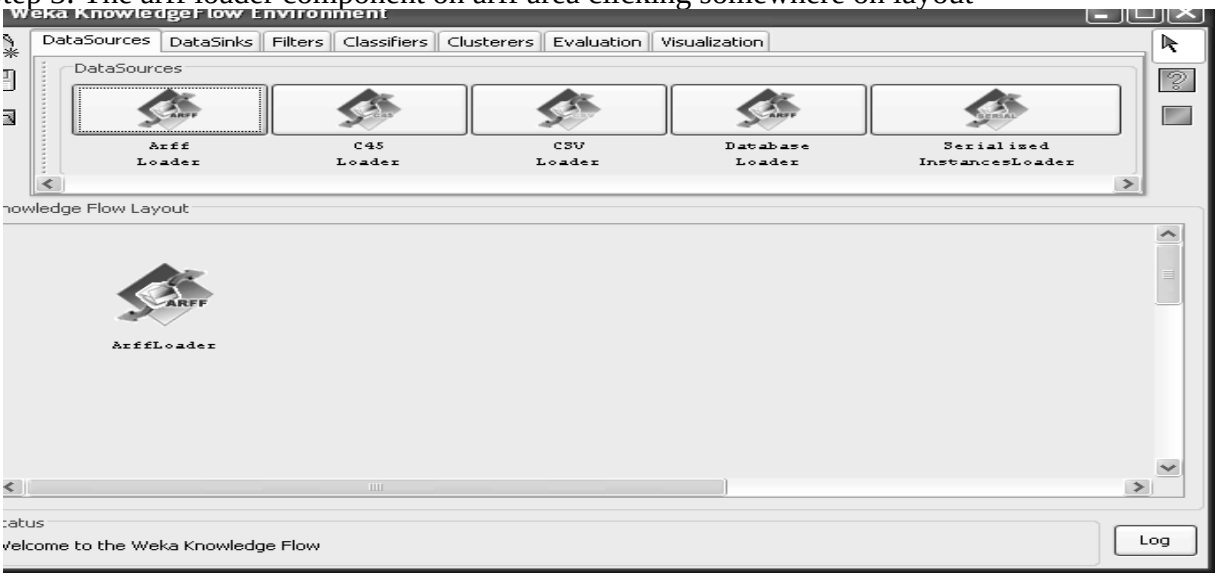
Knowledge flow for decision tree J48.

Step 1: Select knowledge flow in Weka GUI chooser and click on it.

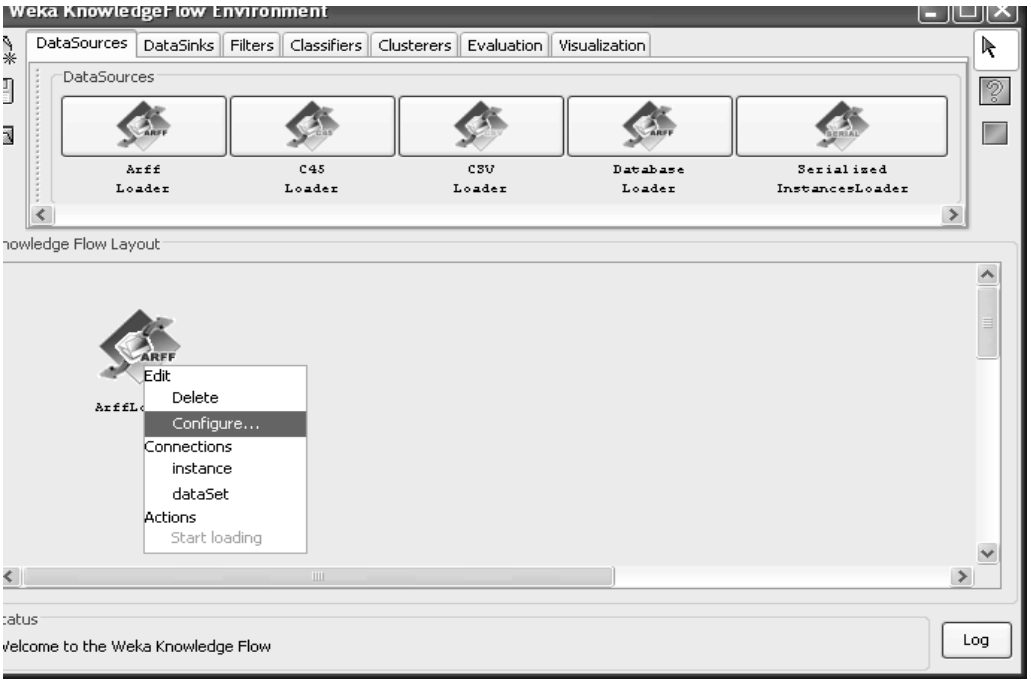


Step 2: click on data source tab and choose arff loader from the tool bar

Step 3: The arff loader component on arff area clicking somewhere on layout



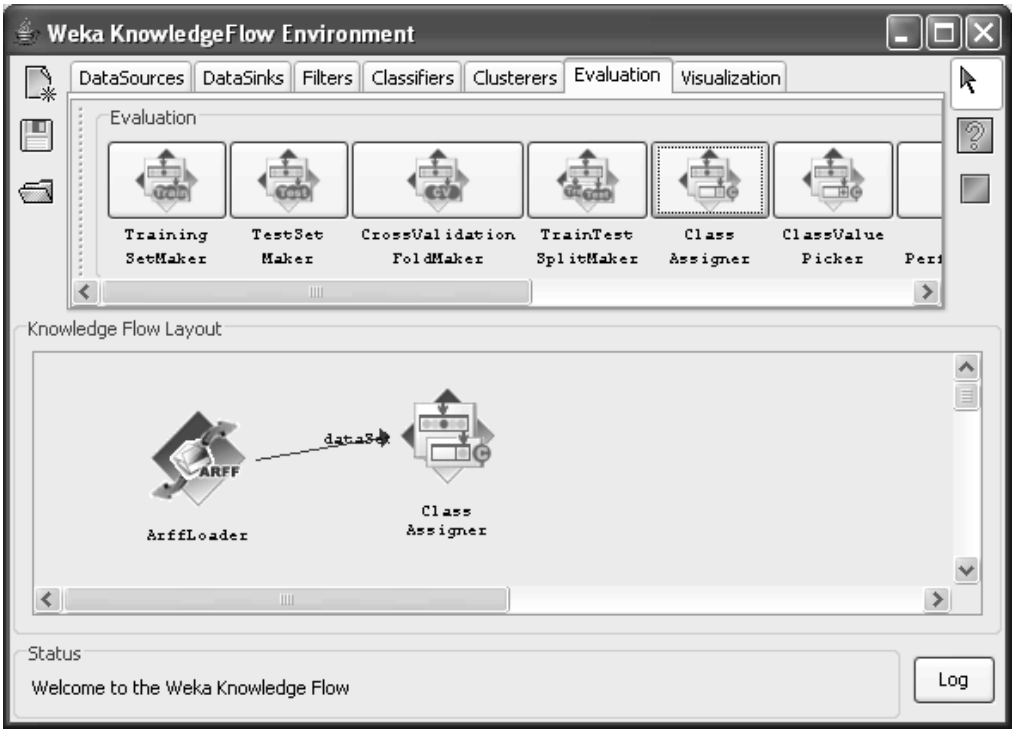
Step 4: specify j48. arff file to load by right clicking mouse over the arffloader icon on the layout.A popup menu will appear select **configure**.



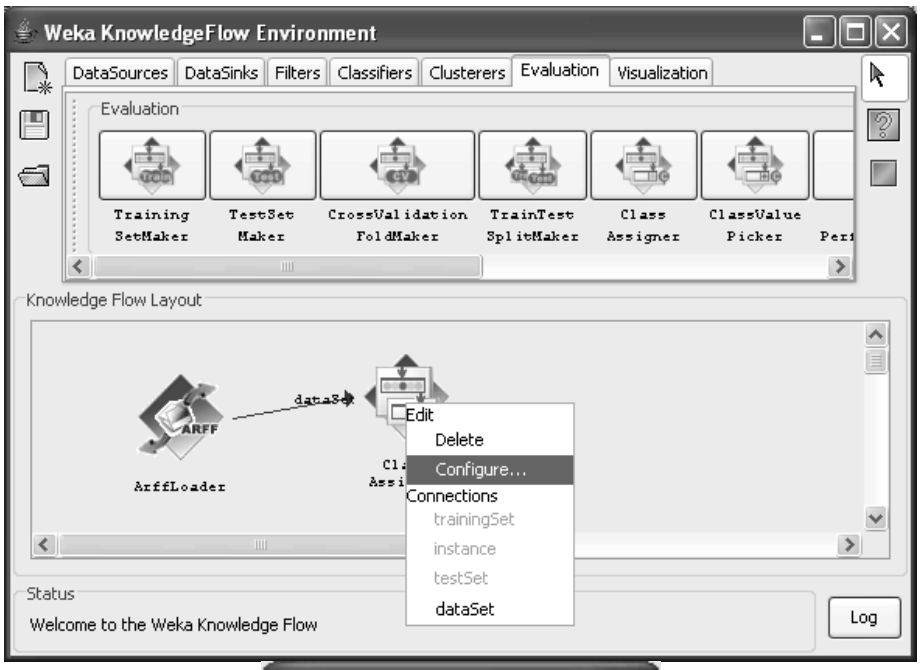
Click on open button.

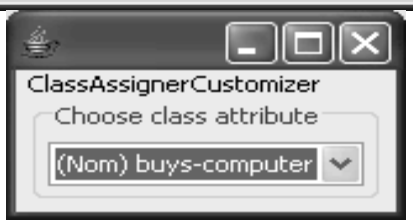
Step 5: Click on evaluation tab, at the top of window, choose **Class Assigner.**

Step6: Now connect arff loader to Class Assigner , first right click over Arffloader and select **dataset** under connection in the menu.

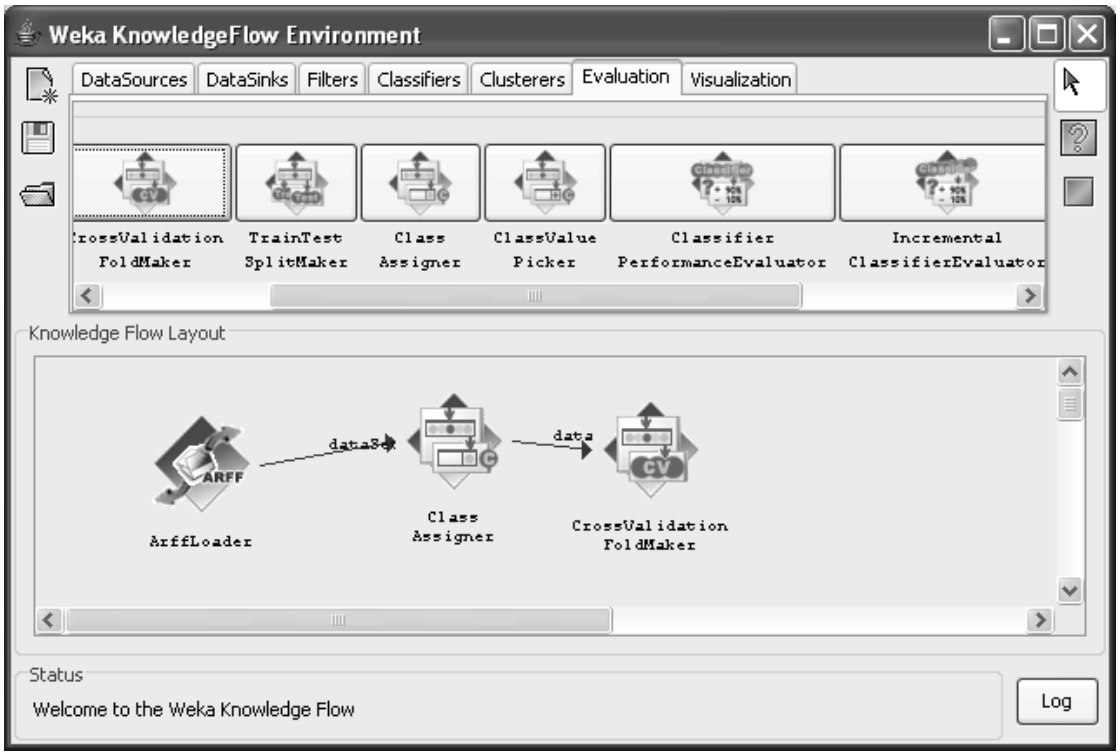


Step7:click on class assigner configure menu, choose specify columns is the class in our data given as(buys-computer).



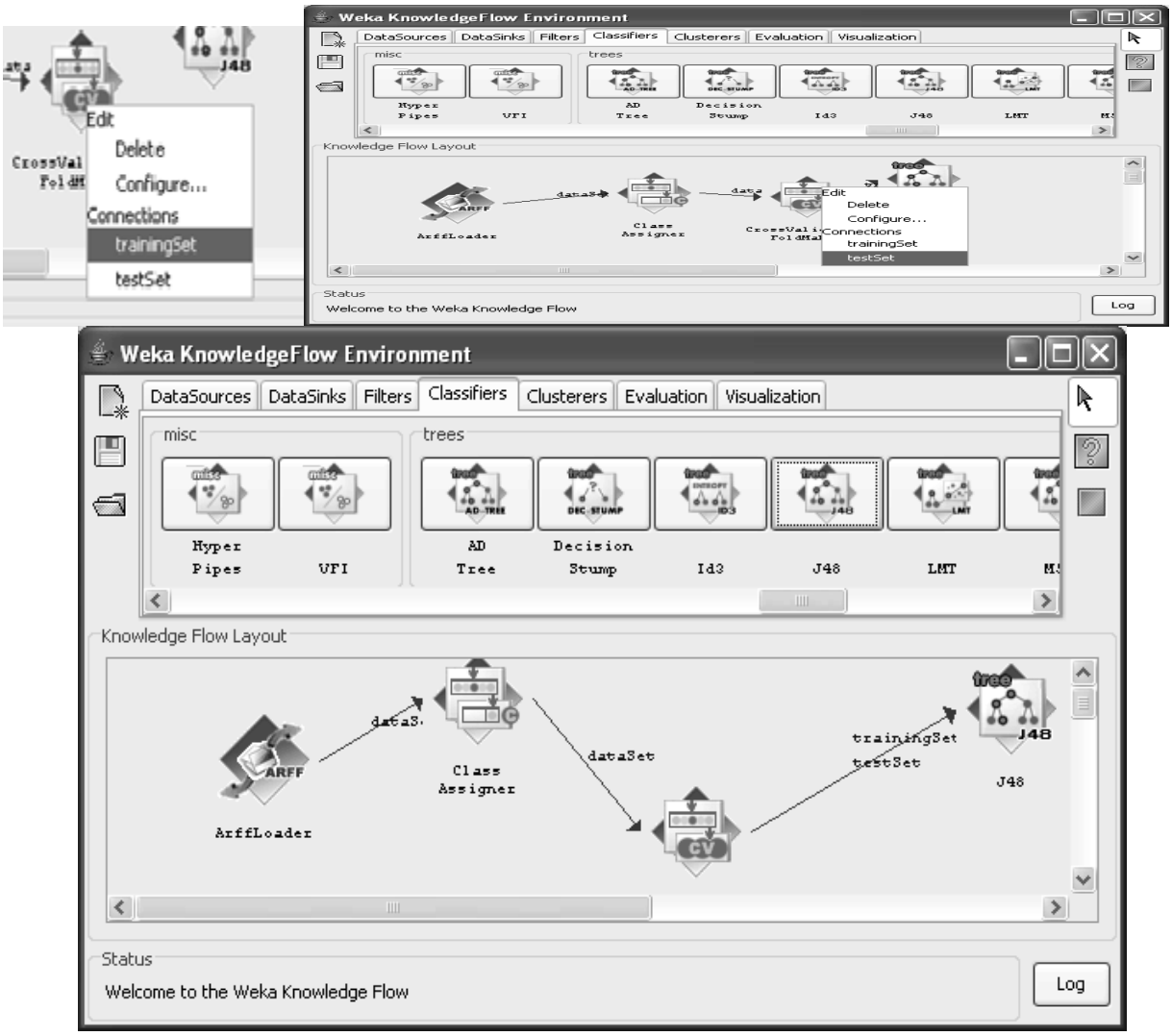


Step 8: Choose **cross validation fold maker**, component from the evaluation toolbar
Step 9: connect class assigner to cross validation fold maker and right click over class assigner, select dataset under the connection menu.

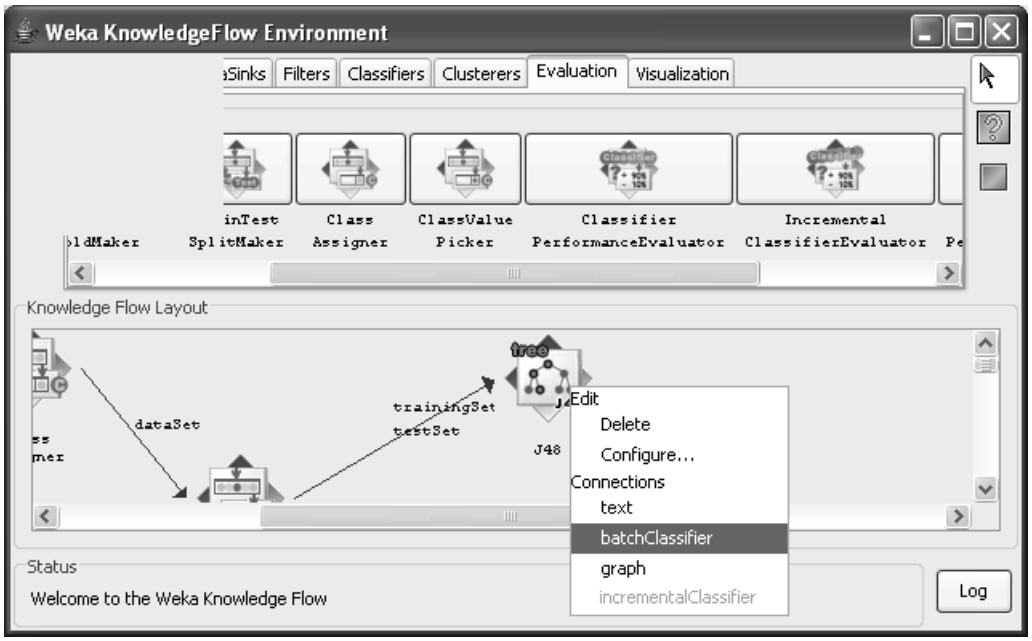


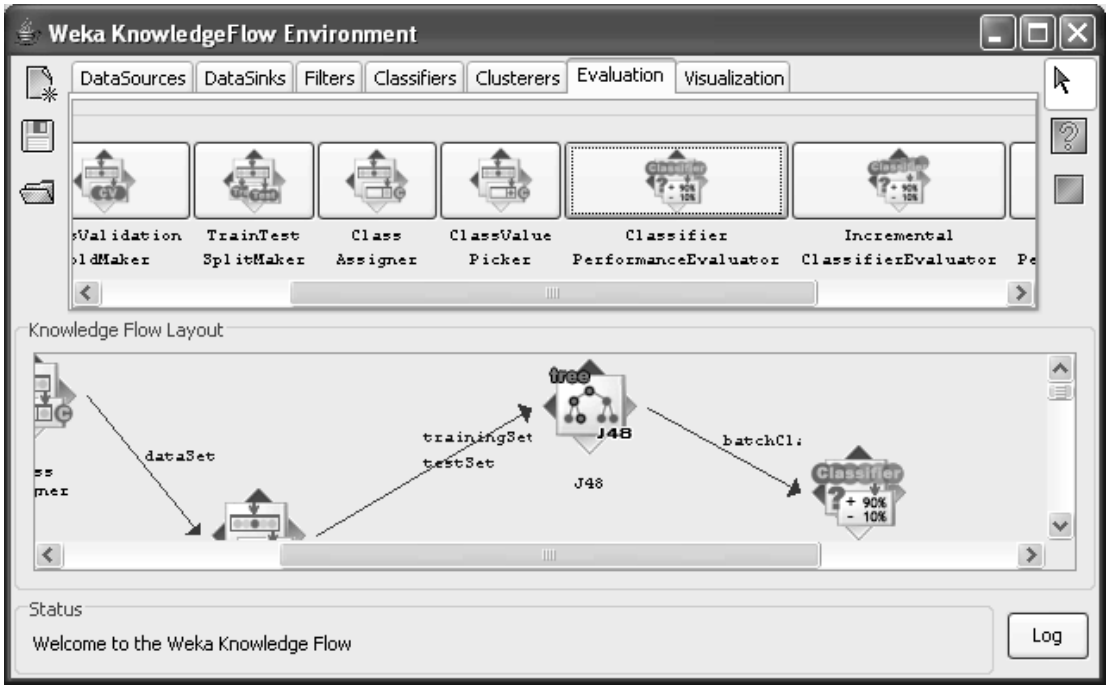
Step 10: click on classifiers tab on the window scroll to the toolbar, specify **j48**

Step 11: connect the cross validation fold maker to j48 by first choosing **training set** and then **text set** for the cross validation fold maker

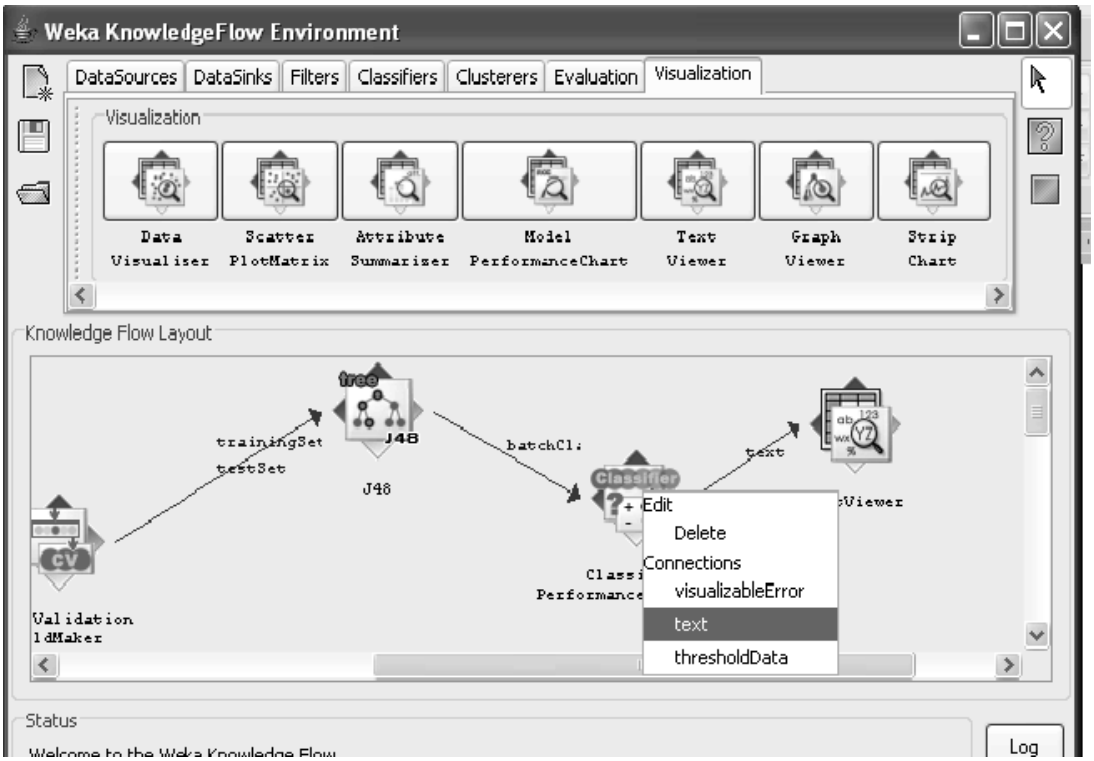


Step 12: Go to evaluation tab and place a classifier performance evaluator component on the layout Connect j48 to this component by selecting the **batch classifier** entry from the popup menu for j48

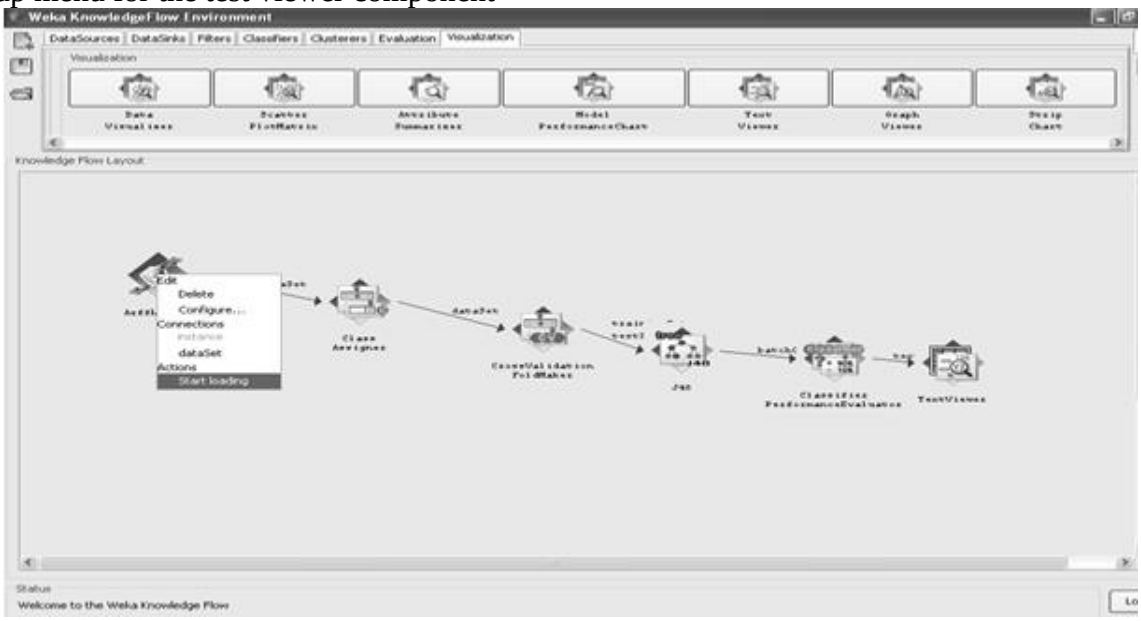




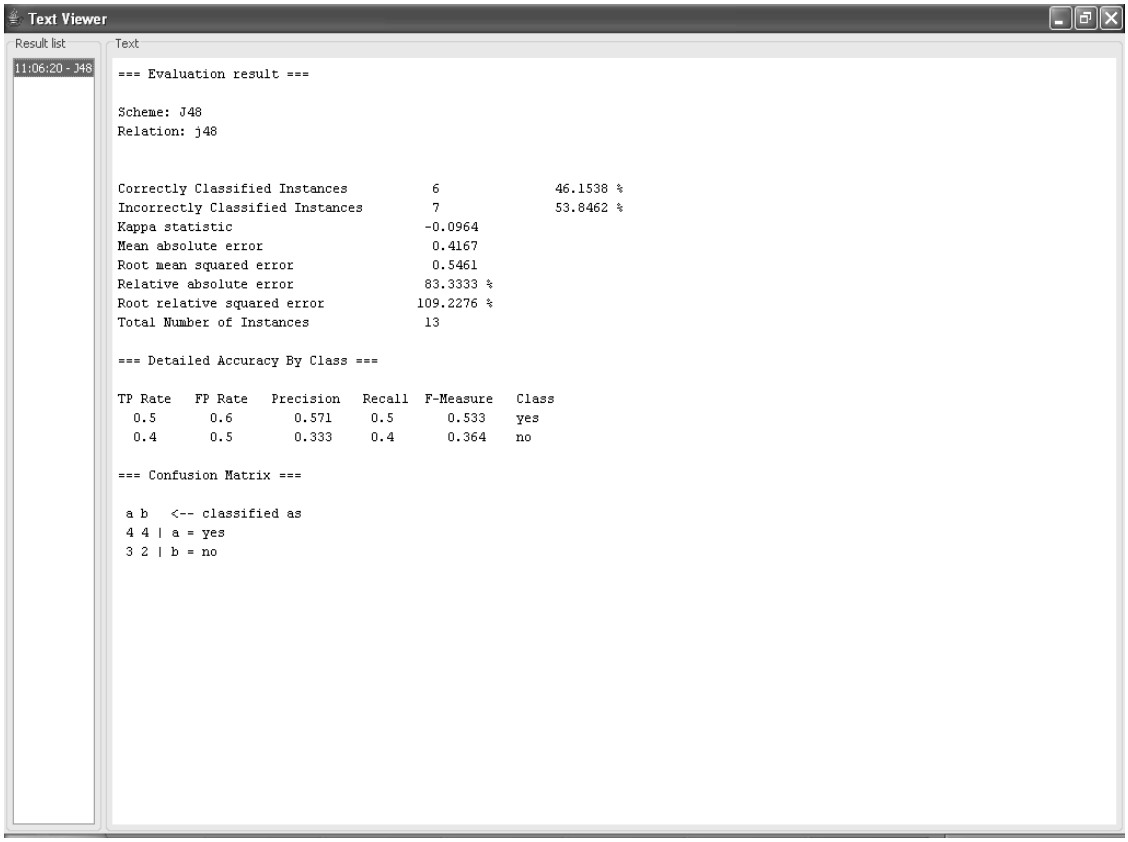
Step 13: Go to visualization toolbar and place **text viewer** component on the layout.connect the classifier performance evaluator to the text viewer by selecting he text entry form the popup for classifier performance evaluator



step14: Now start the flow executing by selecting **start loading** from the popup menu of arffloader. step15: when finished, you can view the results by choosing show results from the popup menu for the test viewer component



OUTPUT:



KNOWLEDGE FLOW FOR CLUSTERING

- First start the Knowledge Flow.
- Next click on the Data Sources tab and choose "ArffLoader" from the toolbar (the mouse pointer will change to a "cross hairs").
- Next place the ArffLoader component on the layout area by clicking somewhere on the layout (A copy of the ArffLoader icon will appear on the layout area).
- Next specify an arff file to load by first right clicking the mouse over the ArffLoader icon on the layout. A pop-up menu will appear. Select "Configure" under "Edit" in the list from this menu and browse to the location of your arff file.
- Next click the "Evaluation" tab at the top of the window and choose the "Class Assigner" (allows you to choose which column to be the class) component from the toolbar. Place this on the layout.
- Now connect the ArffLoader to the Class Assigner: first right click over the ArffLoader and select the "data Set" under "Connections" in the menu. A "rubber band" line will appear. Move the mouse over the Class Assigner component and left click - a red line labeled "data Set" will connect the two components.

- Next right click over the Class Assigner and choose "Configure" from the menu. This will pop up a window from which you can specify which column is the class in your data (last is the default).

- Next grab a "Cross Validation Fold Maker" component from the Evaluation toolbar and place it on the layout. Connect the Class Assigner to the Cross Validation Fold Maker by right clicking over "Class Assigner" and selecting "dataset" from under "Connections" in the menu.

- Next click the “Clustrer” tab the top of the window and choose the "SimpleKMeans" component from the toolbar. Place this on the layout.

- Connect the CrossValidationFoldMaker to SimpleKMeans by first choosing "training Set" and then "test Set" from the pop-up menu for the CrossValidationFoldMaker

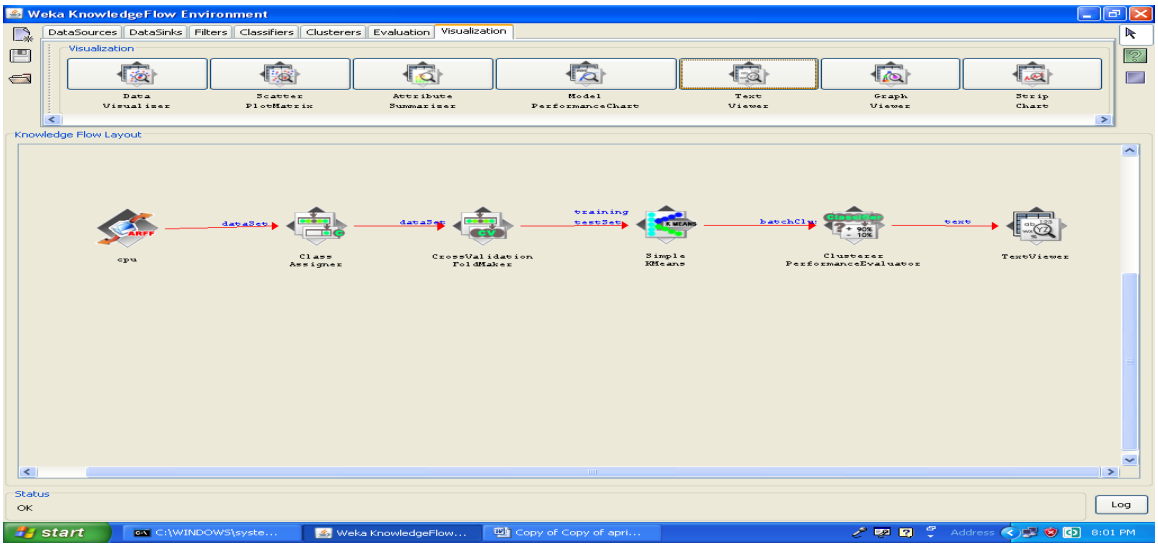
- Next go back to the "Evaluation" tab and place a "ClustererPerformanceEvaluator" component on the layout. Connect SimpleKMeans to this component by selecting the "batch Clusterer" entry from the pop-up menu for simpleKMeans.

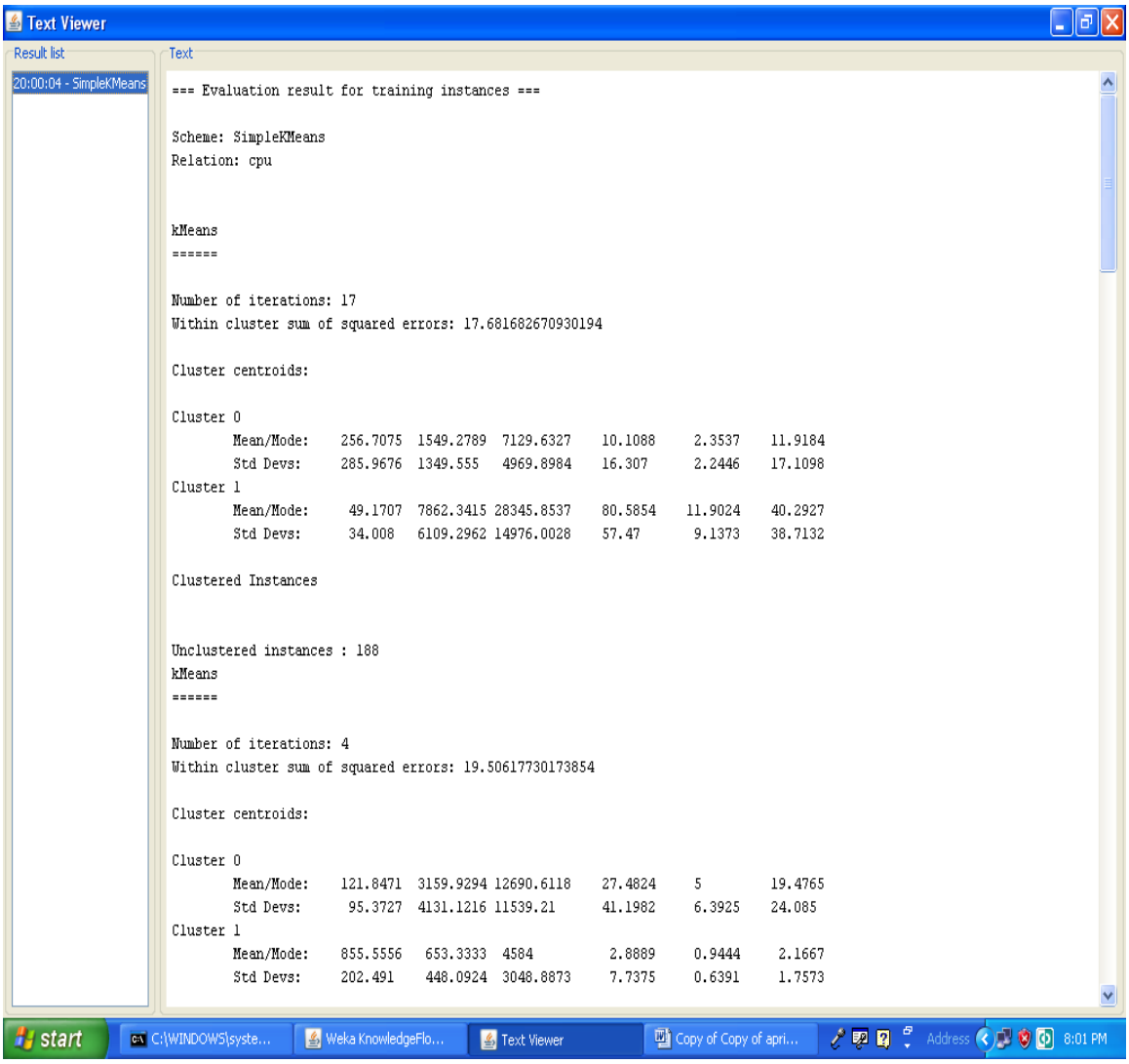
- Next go to the "Visualization" toolbar and place a "Text Viewer" component on the layout. Connect the ClustererPerformanceEvaluator to the Text Viewer by selecting the "text" entry from the pop-up menu for ClustererPerformanceEvaluator.

- Now start the flow executing by selecting "Start loading" from the pop-up menu for ArffLoader.

- When finished you can view the results by choosing show results from the pop-up menu for the TextViewer component.

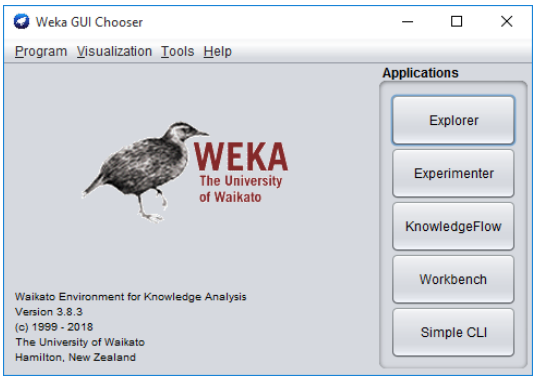
Output:



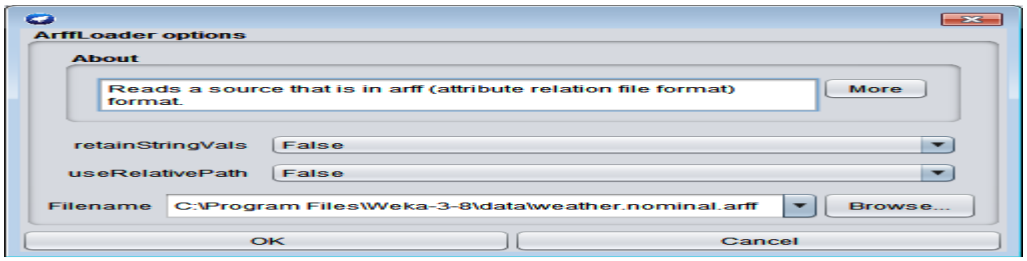


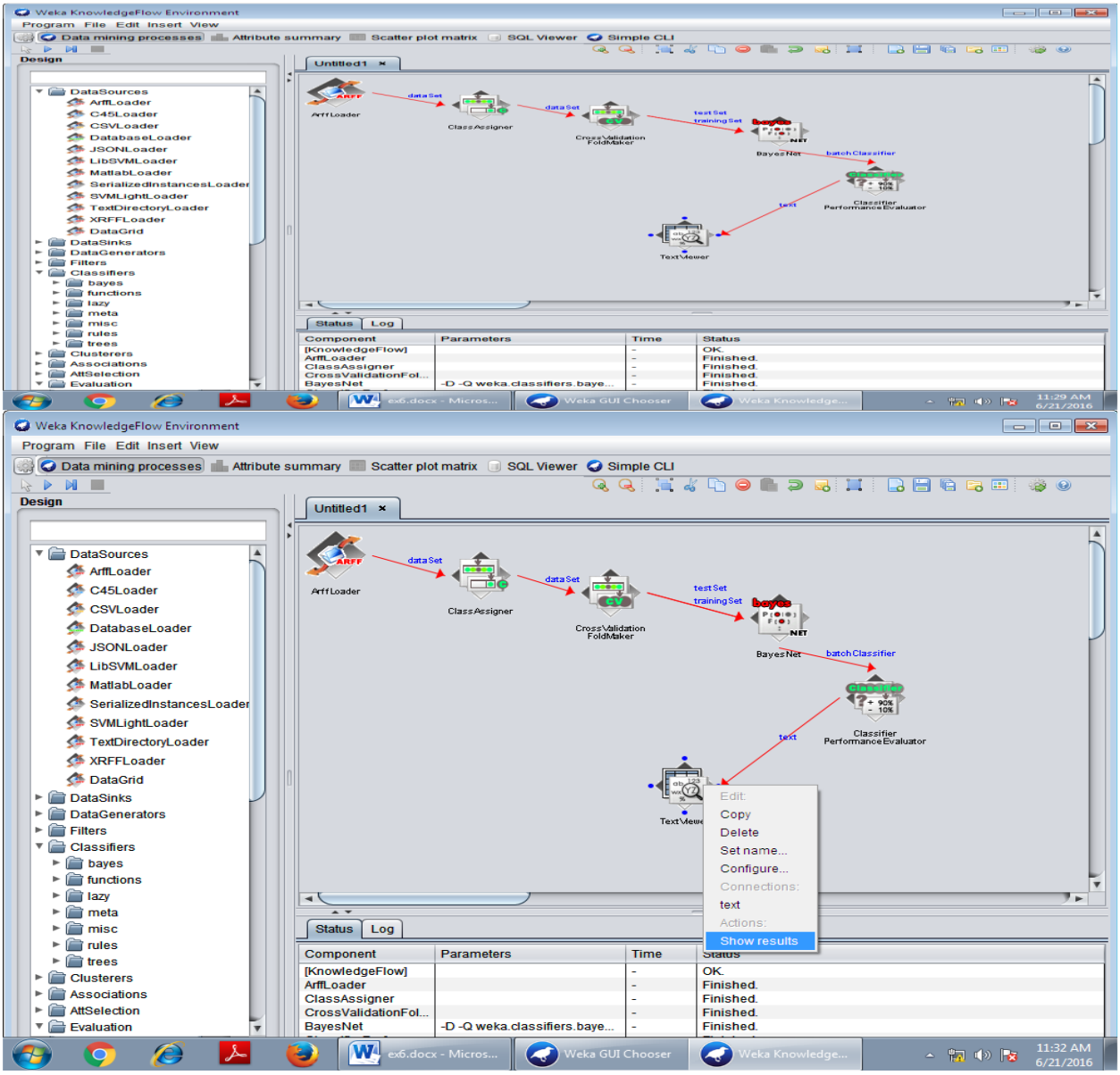
Naïve Bayes

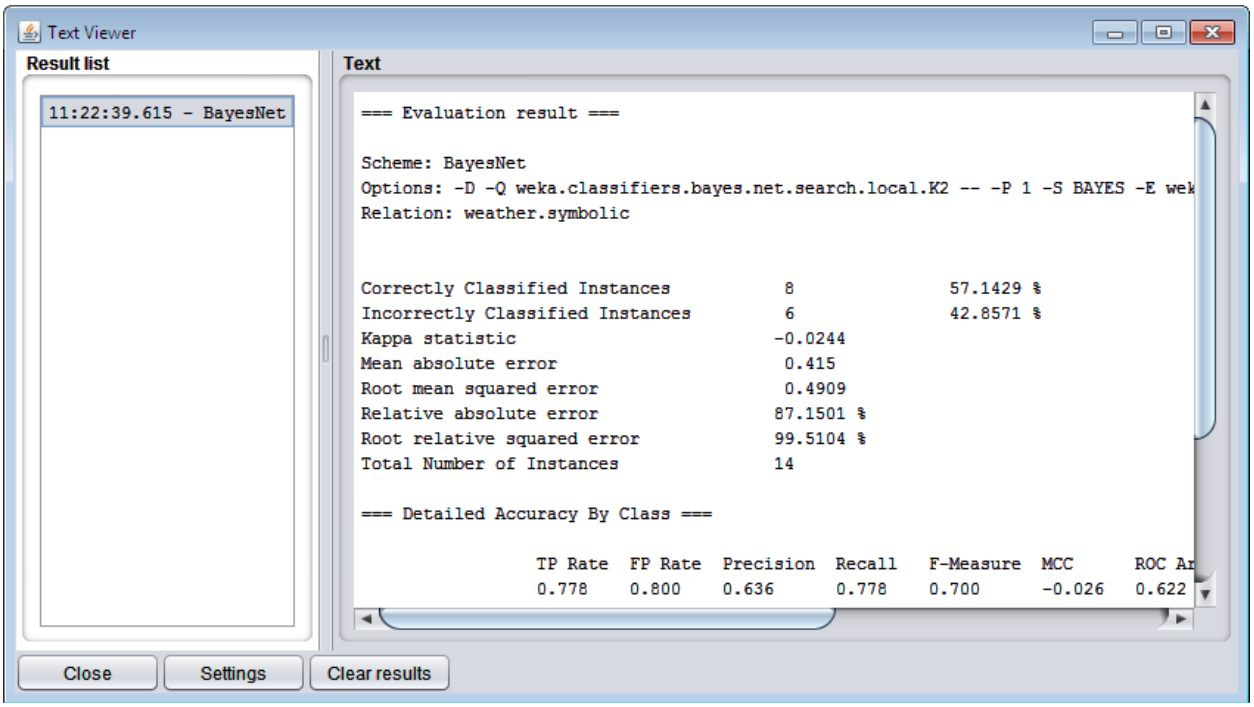
Click **Knowledge Flow** in **Weka GUI Chooser**



To create the following and click **Run** button and **Right click** the **TextViwer** and select **Show Result.**
Draw **ArrfLoader** and select the filename.







VIVA-QUESTIONS

1____ is the graphical representation of information and data.

2Which types of chart shows the relationship between a numerical variable and categorical variable?

A. Line B. Bar C. Pie D. x-y plot

3Which of the following are Data sources of Knowledge Flow?

A. CSV B. Notepad C. Excel D. Acess

4Data Visualization in mining cannot be done using

5A bubble chart is a variation of

6Which method shows hierarchical data in a nested format?

7Data visualization is also an element of the broader _____.

8Which of the following is a Trees in data visualization

A. B-Tree B. B+-Tree C. Heap-Tree D. AD-Tree

9which of the following not support data source

A. ARFF Loader B. CSV Loader C. Access Loader D. XML Loader

10what is use of class assigner in weka

ADDITIONAL EXPERIMENTS

FILE FORMATES FOR WEKA

1. Create CSV(Comma Separated Values) file.

   Step1: Create an excel file and save with specified format as CSV(Comma Delimited).
   Step2: Now open with notepad and check the values.Here,the fields of data in each row are delimited with a comma and individual rows are separated by new line.

2. Create arff(Attribute Relation File Format) file.

   Step1: Open a notepad and type the data as instructed below:

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
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa

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

Step 2: Save the file as .arff.
Step 3: Open with Weka Explorer and check the file values.

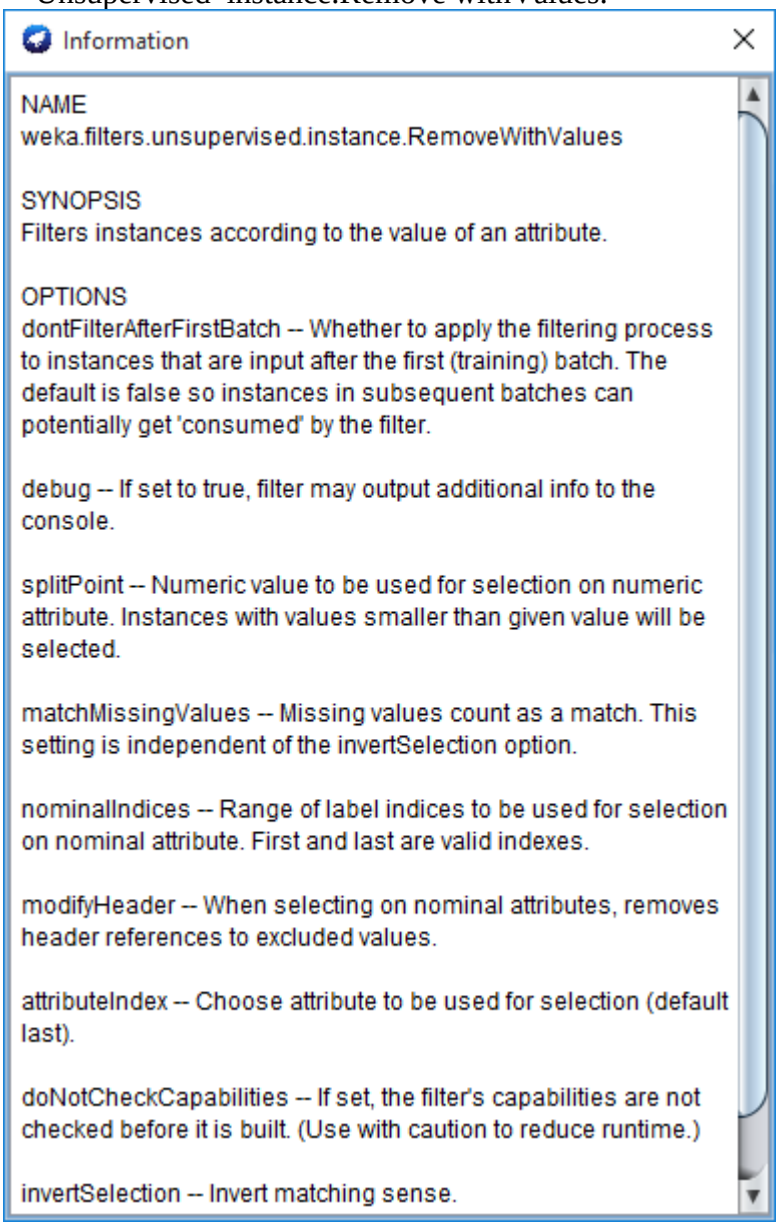
3. Convert CSV to ARFF file format.

Step 1: Open CSV file with notepad
Step 2: To fill header and data section in CSV file.
Step 3: save the file type as arff.
Step 4: Open with Weka Explorer.

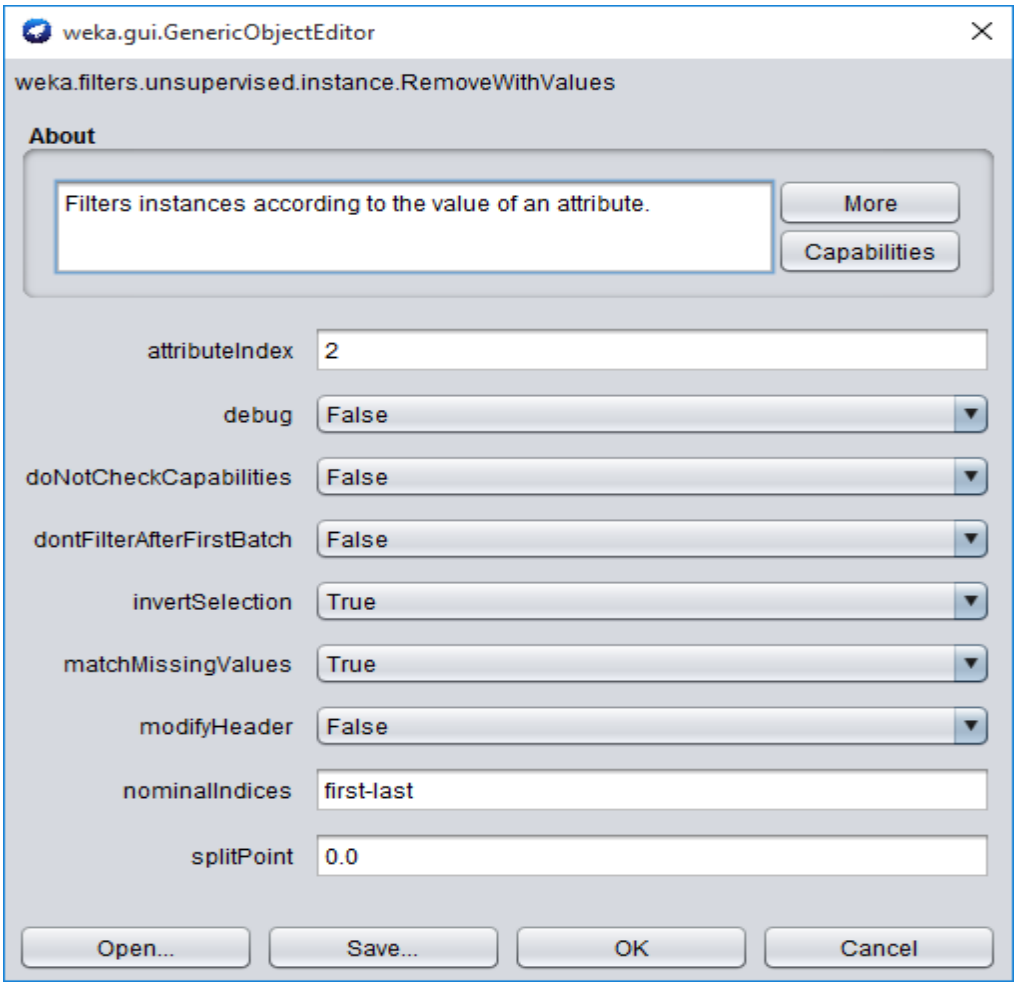
Missing Values

Ignore the tuple (Remove Missing Data)

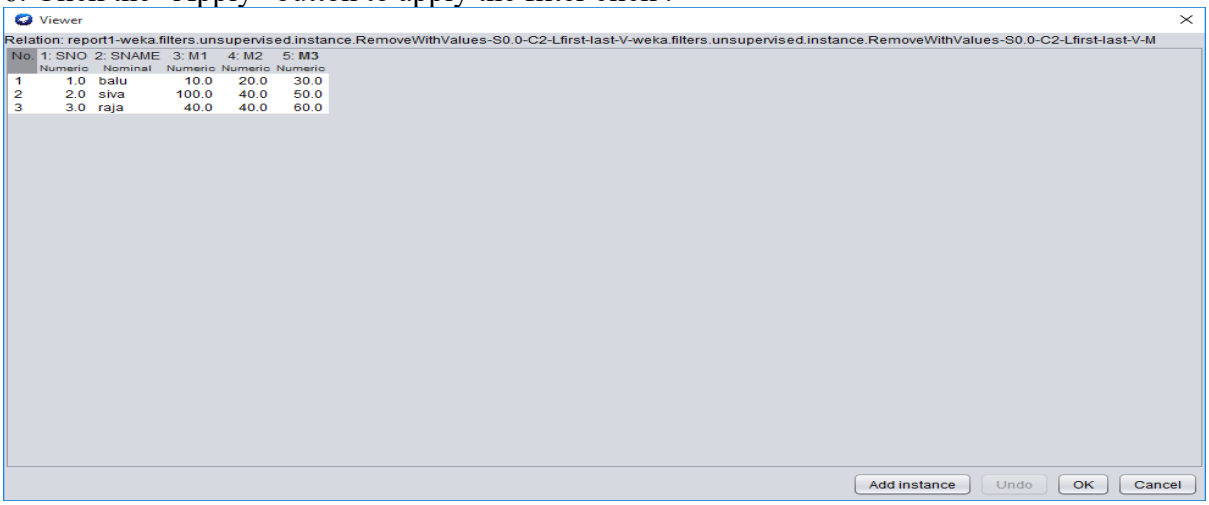
1.Open the Weka Explorer.
2.Load the Student Data set
3.Click the “Choose” button for the filter and select “Remove with values” as under Unsupervised instance.Remove withValues.



4. Set Match Missing value to “True”



5. Click the "OK" button to use the configuration for the filter.
6. Click the "Apply" button to apply the filter click .



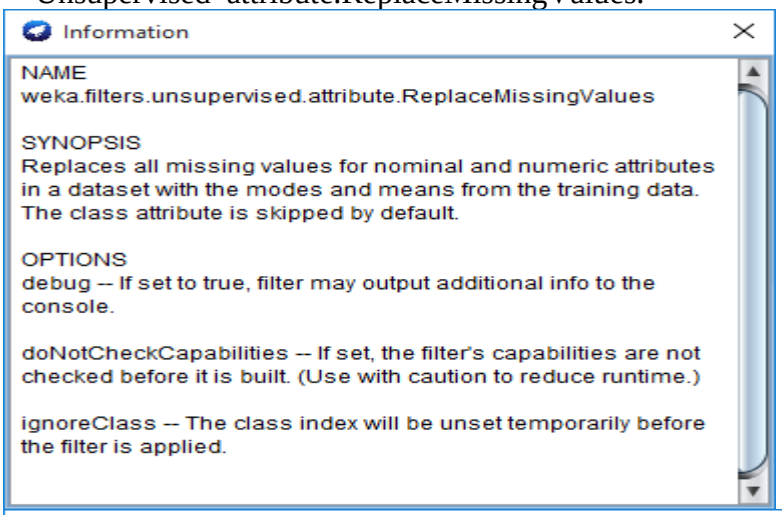
Impute Missing Values

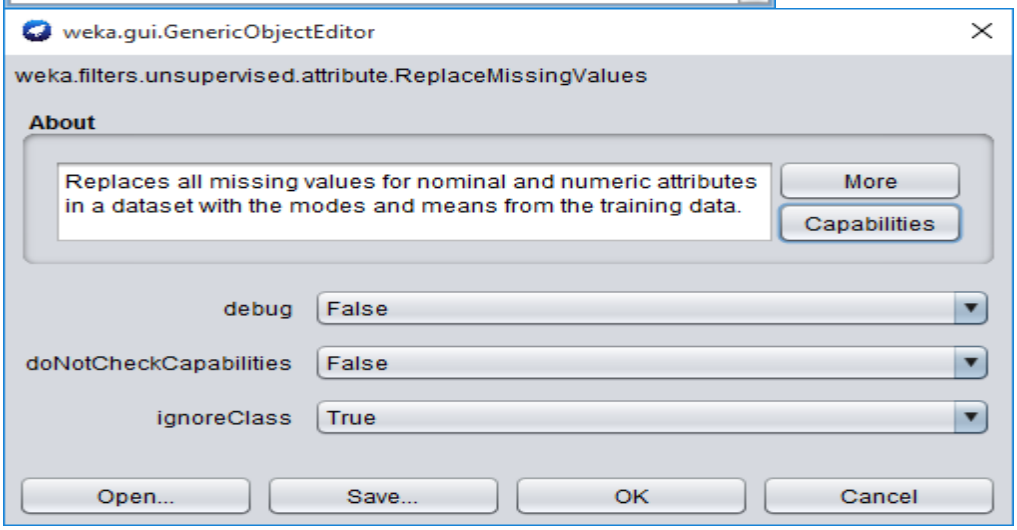
How to Impute mean values for missing values

1.Open the Weka Explorer.

2.Load the Student Data set

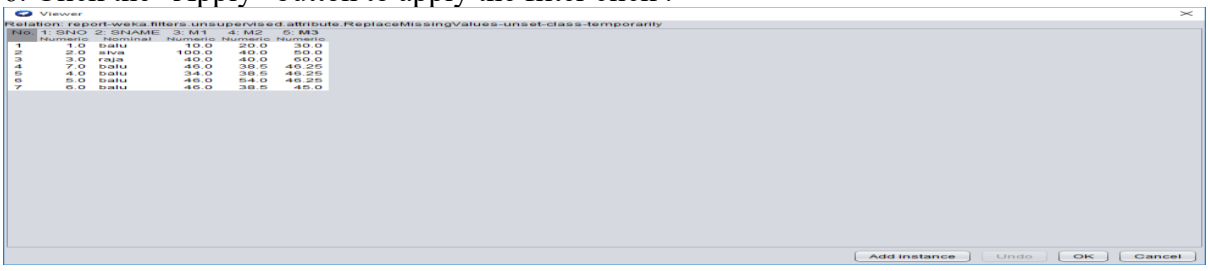
3.Click the “Choose” button for the filter and select “ReplaceMissingValues” as under Unsupervised attribute.ReplaceMissingValues.





5. Click the “OK” button to use the configuration for the filter.

6. Click the “Apply” button to apply the filter click .



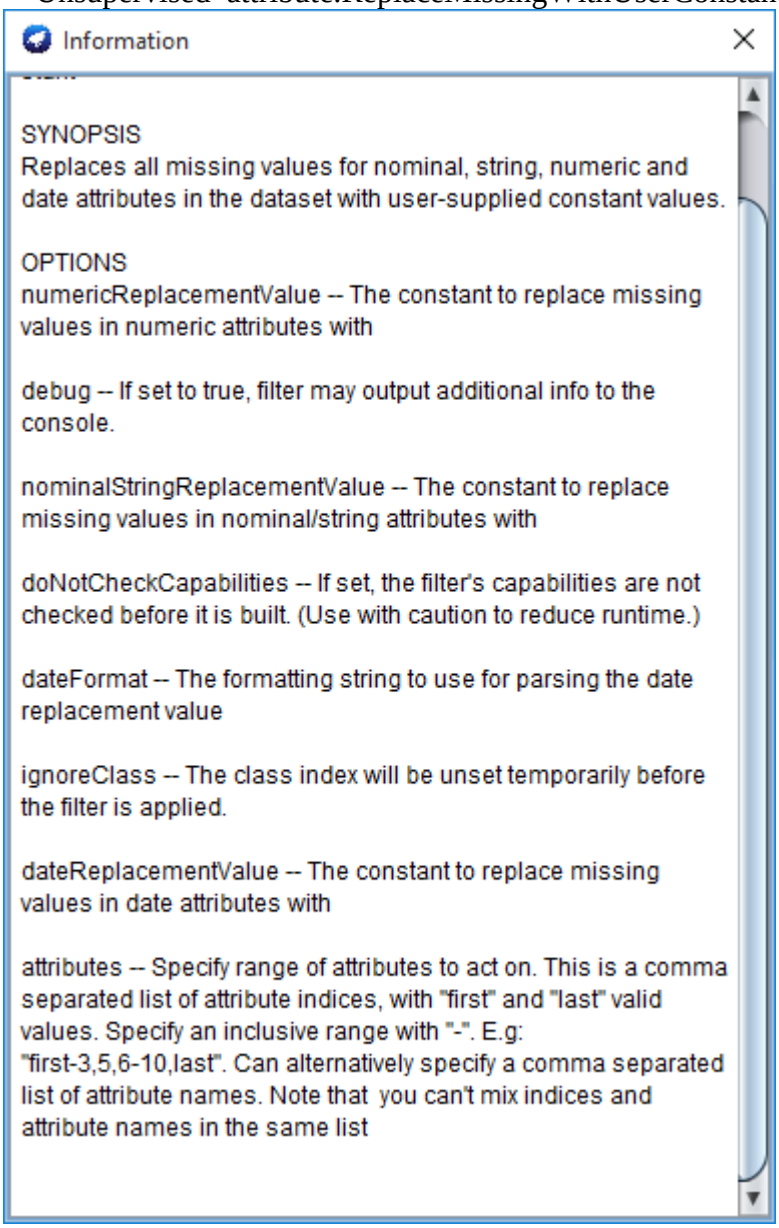
How to Impute constant values for missing values

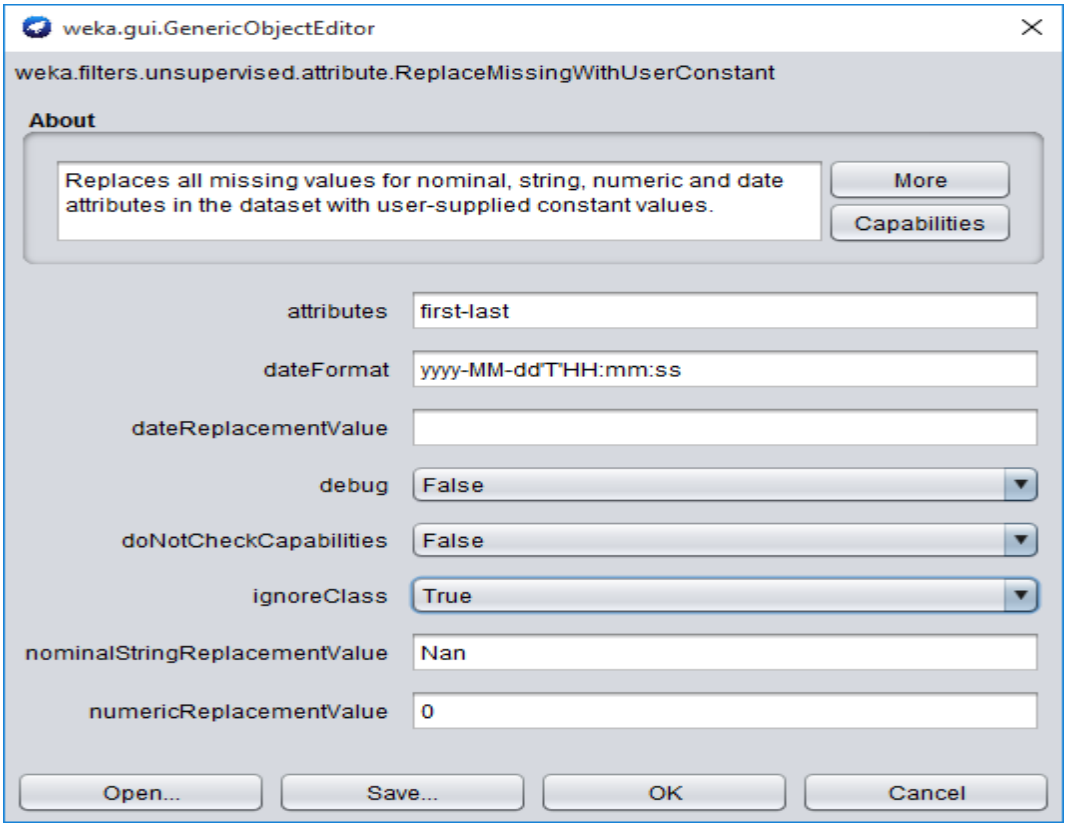
1.Open the Weka Explorer.

2.Load the Student Data set

3.Click the “Choose” button for the filter and select “ReplaceMissingWithUserConstant” as under

Unsupervised attribute.ReplaceMissingWithUserConstant.





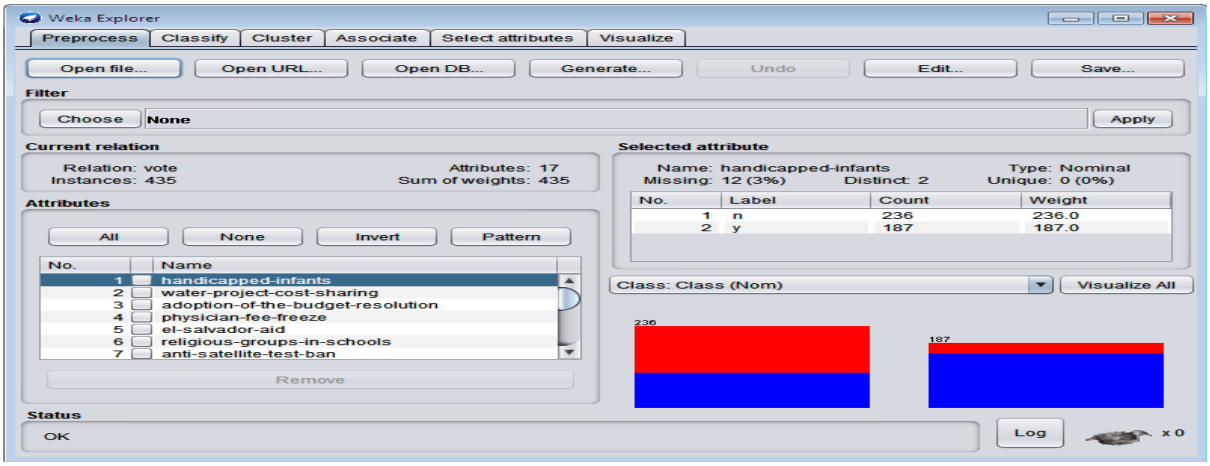
5. Click the “OK” button to use the configuration for the filter.
6. Click the “Apply” button to apply the filter click .

HIERARCHICAL CLUSTERING

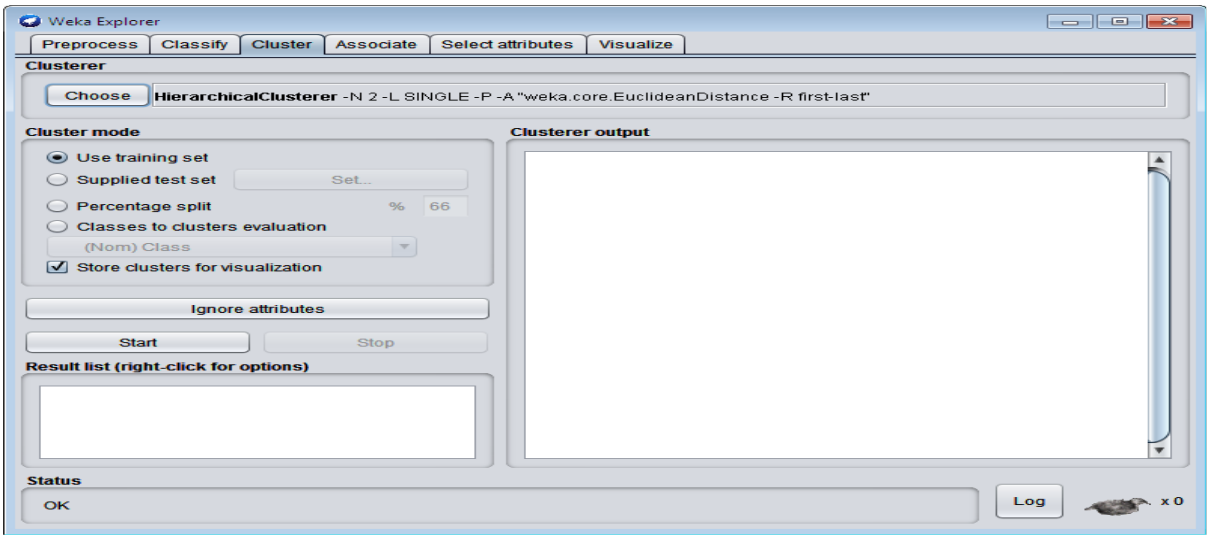
Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types. Agglomerative is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Divisive is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

1.Open**Weka** tool and choose **Explorer.**

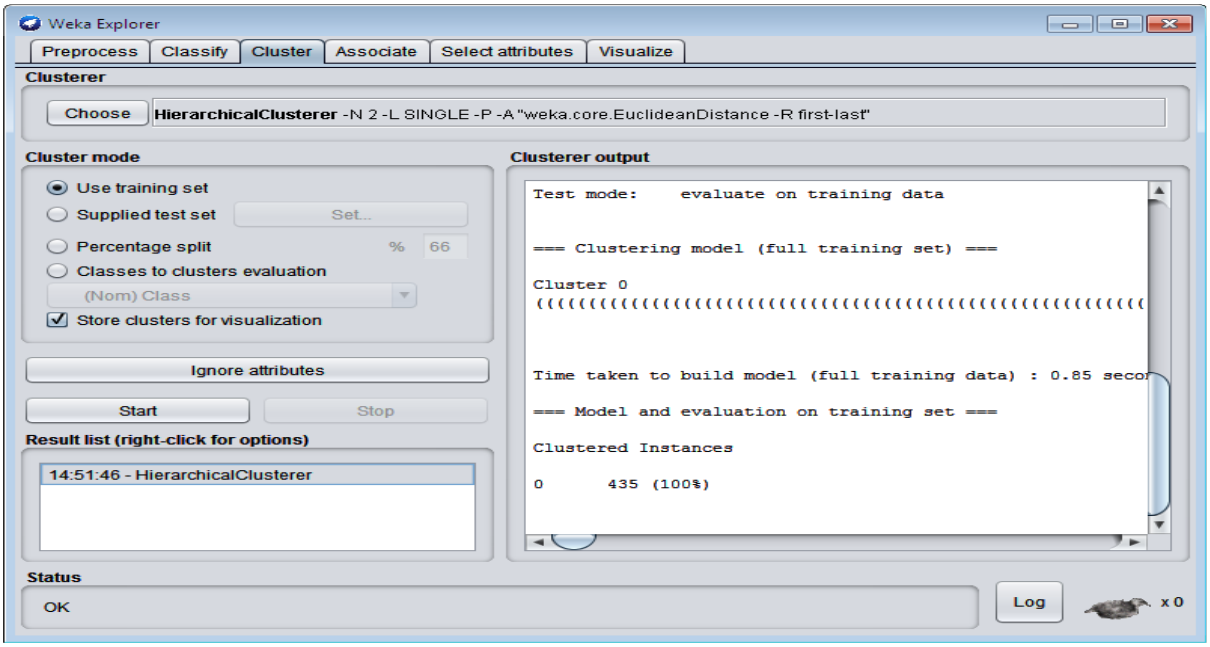
2.Click **- Open file**… in **preprocess** tab –choose **vote.arff.**



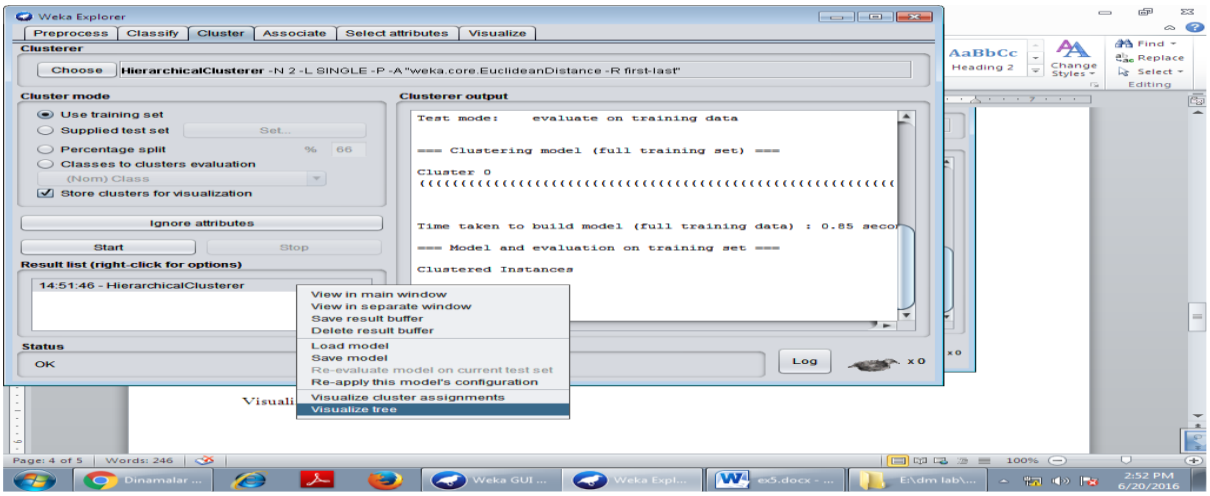
3.Goto **Cluster** tab – click **choose** button - select **HierarchicalClusterer.**

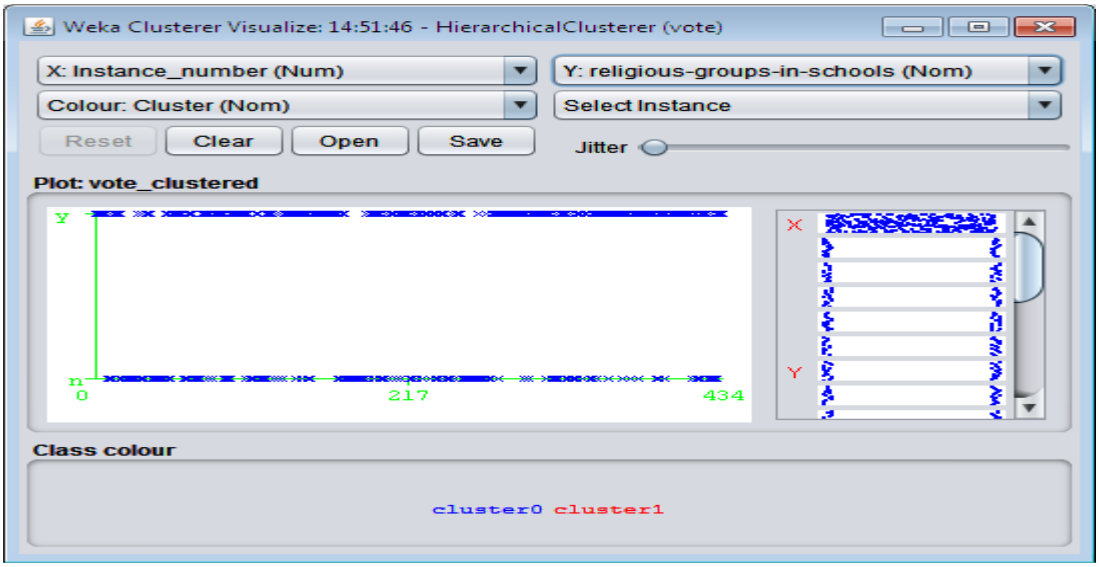


5.Click **Start** button



6. Visualize the tree by **right click**ing and choose **Visualize Tree** option.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
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)
14:51:46 - HierarchicalClusterer
Clusterer output
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
Time taken to build model (full training data) : 0.85 seco
=== Model and evaluation on training set ===
Clustered Instances
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize cluster assignments
Visualize tree
Status
OK
Log
x 0


Weka Clusterer Visualize: 14:51:46 - HierarchicalClusterer (vote)
X: Instance_number (Num)
Y: religious-groups-in-schools (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: vote_clustered
Y
n
0
217
434
X
Y
Class colour
cluster0 cluster1