# DATA MINING & DATA ANALYTICS LAB

**B.Tech. III Year I Sem. Course Code: 20-CS-PC-315**

**LIST OF EXPERIMENTS**

**Part-A: Data Mining**

1. Demonstration of preprocessing on dataset student.arff

2. Demonstration of Association rule process on dataset contactlenses.arff using apriori

algorithm.

3. Demonstration of classification rule process on dataset employee.arff using j48 algorithm.

4. Demonstration of classification rule process on dataset employee.arff using id3 algorithm.

5. Demonstration of classification rule process on dataset employee.arff using naïve bayes

algorithm.

6. Demonstration of clustering rule process on dataset iris.arff using simple k-means.

7. Demonstration of clustering rule process on dataset student.arff using hierarchical

clustering.

**1. Weka Introduction**

Weka is created by researchers at the university WIKATO in NewZealand. University of Waikato, Hamilton, New Zealand Alex Seewald (original Command-line primer) David Scuse (original Experimenter tutorial)

* It is java based application.
* It is collection often source, Machine Learning Algorithm.
* The routines (functions) are implemented as classes and logically arranged in packages.
* It comes with an extensive GUI Interface.
* Weka routines can be used standalone via the command line interface.

The Graphical User Interface

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 conducting

statistical tests between learning schemes.

• **Knowledge Flow** 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.

**I. Explorer**

The Graphical 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 & 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.

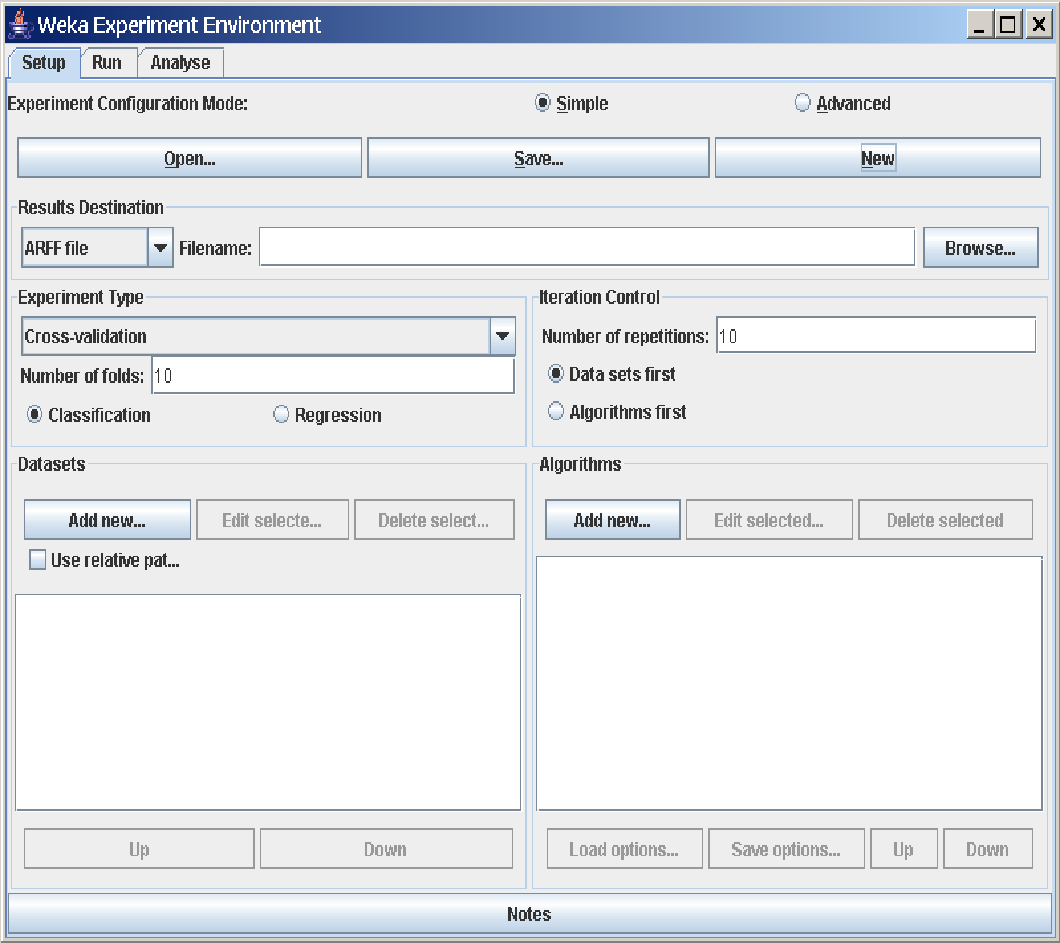
6. **Visualize.** View an interactive 2D plot of the data.

Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in. The Explorer can be easily extended with custom tabs. The Wiki article “Adding tabs in the Explorer” [7] explains this in detail.

**II. Experimenter**

**2.1 Introduction**

The Weka Experiment Environment enables the user to create, run, modify, and analyse experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyse the results to determine if one of the schemes is (statistically) better than the other schemes.



The Experiment Environment can be run from the command line using the Simple CLI. For example, the following commands could be typed into the CLI to run the OneR scheme on the Iris dataset using a basic train and test process. (Note that the commands would be typed on one line into the CLI.) While commands can be typed directly into the CLI, this technique is not particularly convenient and the experiments are not easy to modify. The Experimenter comes in two flavours, either with a simple interface that provides most of the functionality one needs for experiments, or with an interface with full access to the Experimenter’s capabilities. You can choose between those two with the Experiment Configuration Mode radio buttons:

• Simple

• Advanced

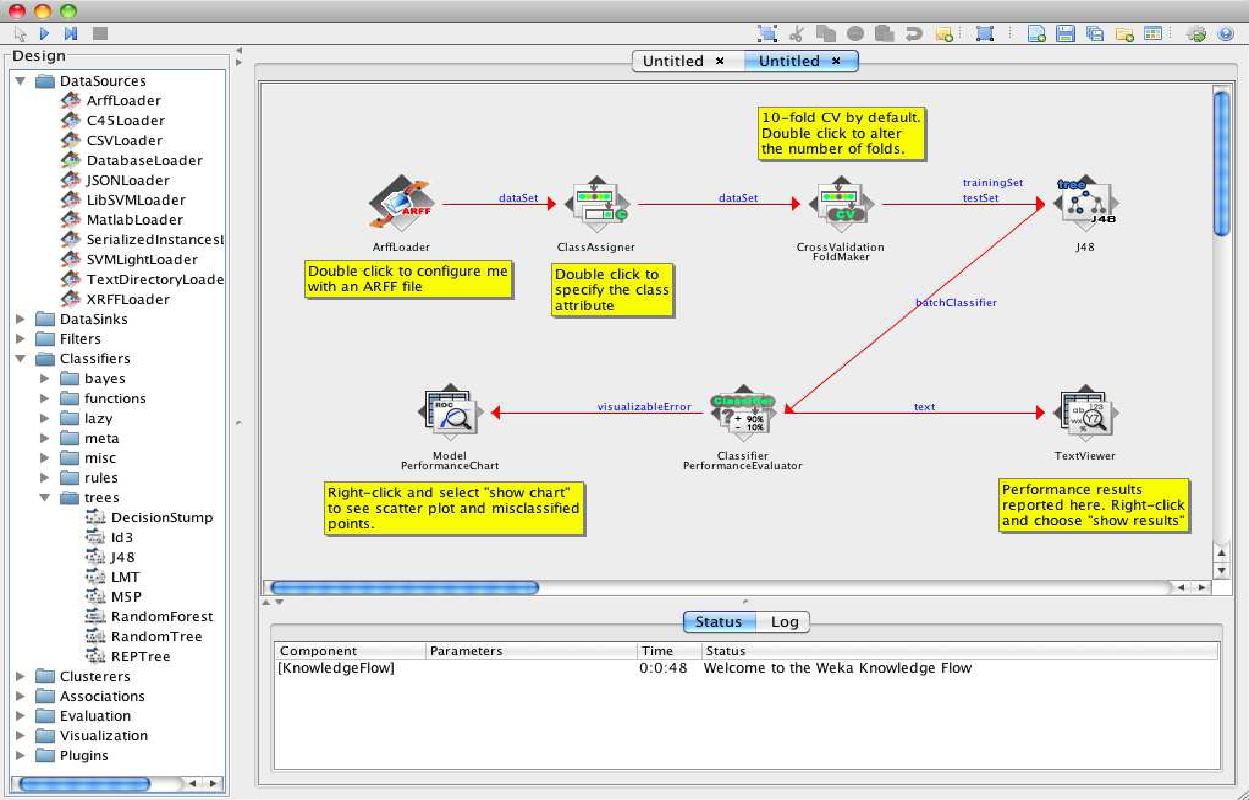
Both setups allow you to setup standard experiments, that are run locally on a single machine, or remote experiments, which are distributed between several hosts. The distribution of experiments cuts down the time the experiments will take until completion, but on the other hand the setup takes more time. The next section covers the standard experiments (both, simple and advanced), followed by the remote experiments and finally the analysing of the results.

**III. Knowledge Flow**

**3.1 Introduction**

The Knowledge Flow provides an alternative to the Explorer as a graphical front end to WEKA’s core algorithms.

The KnowledgeFlow presents a data-flow inspired interface to WEKA. The user can selectWEKA components from a palette, 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, clusterers, associators, loaders and savers are available in the KnowledgeFlow along withsome extra tools.



The Knowledge Flow can handle data either incrementally or in batches (the Explorer handles batch data only). Of course learning from data incremen- tally requires a classifier that can be updated on an instance by instance basis. Currently in WEKA there are ten classifiers that can handle data incrementally.

**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)

•**process multiple streams sequentially** via a user-specified order of execution

**•chain filters** together

•**view models** produced by classifiers for each fold in a cross validation

•**visualize performance** of incremental classifiers during processing

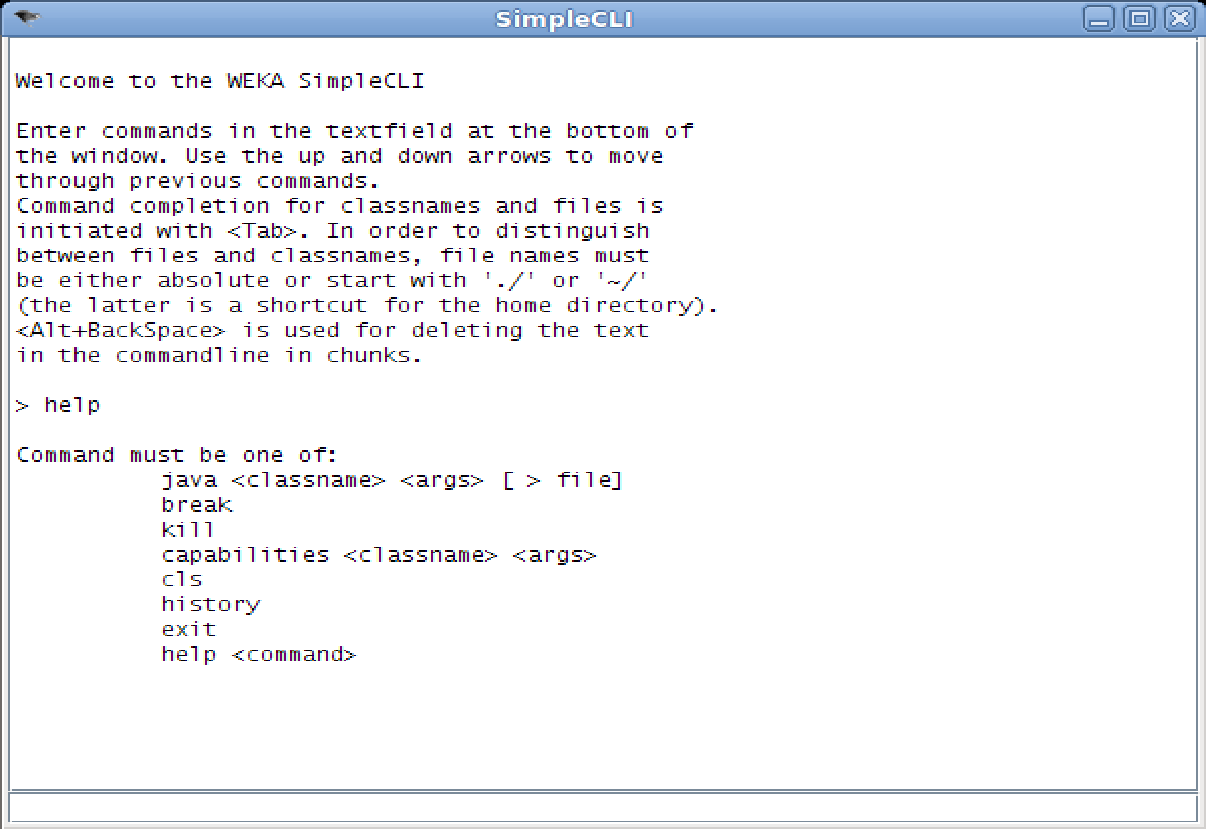
(scrolling plots of classification accuracy, RMS error, predictions etc.)

• **plugin** “perspectives” that add major new functionality (e.g. 3D data

visualization, time series forecasting environment etc.)

**IV. 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.



**4.1 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

killstops the current thread in an unfriendly fashion

• cls

clears the output area

• capabilities <classname> [<args>]

lists the capabilities of the specified class, e.g., for a classifier with its

option:

capabilities weka.classifiers.meta.Bagging -W weka.classifiers.trees.Id3

• exit

exits the Simple CLI

• help [<command>]

provides an overview of the available commands if without a command

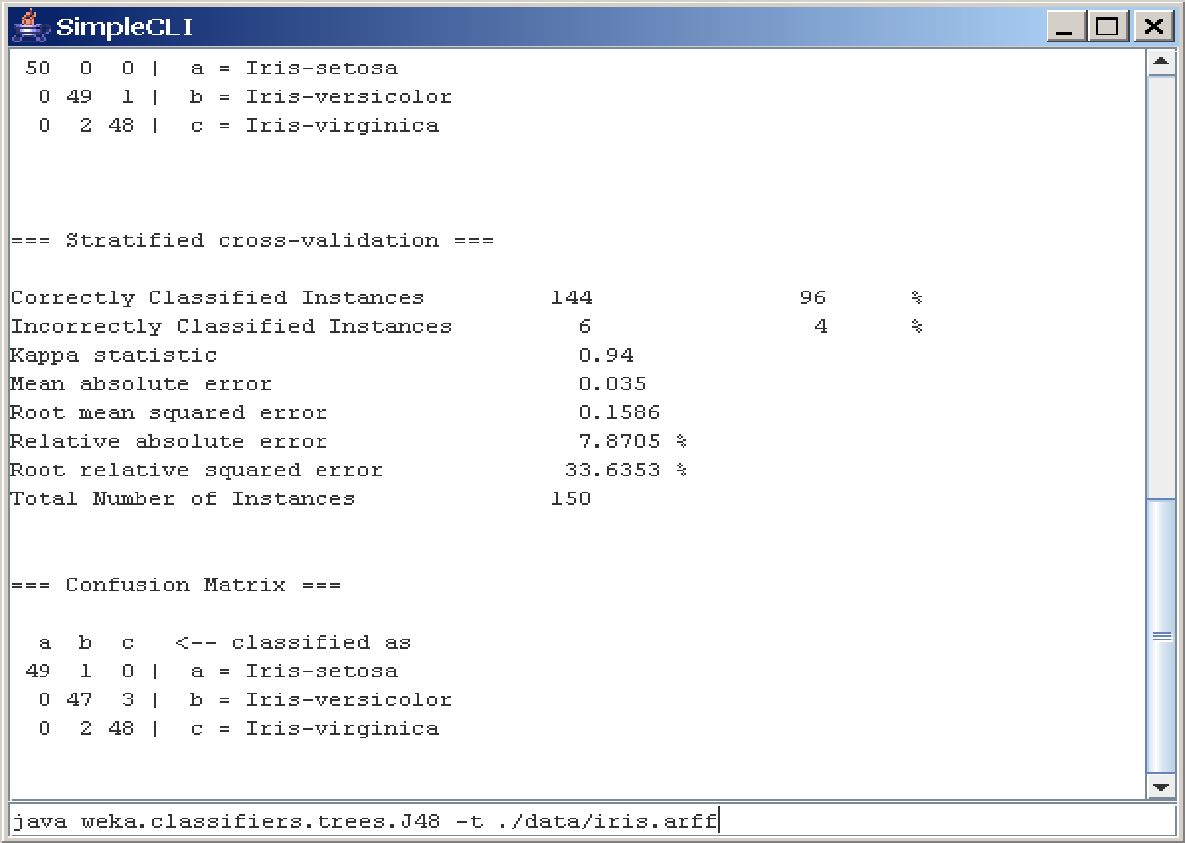
name as argument, otherwise more help on the specified command

**4.2 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

This results in the following output:



**4.3 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.

**4.4 Command completion**

Commands starting with java support completion for classnames and filenames via Tab (Alt+BackSpace deletes parts of the command again). In case that there are several matches, Weka lists all possible matches.

• package name completion

java weka.cl<Tab>

results in the following output of possible matches of package names:

Possible matches:

weka.classifiers

weka.clusterers

• classname completion

java weka.classifiers.meta.A<Tab>

lists the following classes

Possible matches:

weka.classifiers.meta.AdaBoostM1

weka.classifiers.meta.AdditiveRegression

weka.classifiers.meta.AttributeSelectedClassifier

• filename completion

In order for Weka to determine whether a the string under the cursor is a classname or a filename, filenames need to be absolute (Unix/Linx: /some/path/file;Windows: C:\Some\Path\file) or relative and starting with a dot (Unix/Linux: ./some/other/path/file;Windows: .\Some\Other\Path\file).

**ARFF File Format**

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 are not the only format one can load, but all files that can be converted with Weka’s “core converters”. The following formats are currently

supported:

• ARFF (+ compressed)

• C4.5

• CSV

• libsvm

• binary serialized instances

• XRFF (+ compressed)

**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.

**The ARFF Header Section**

The ARFF Header section of the file contains the relation declaration and at-

tribute 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 ordered 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 supported by Weka:**

• numeric

• integer is treated as numeric

• real is treated as numeric

• <nominal-specification>

• string

• date [<date-format>]

• relational for multi-instance data (for future use)

where <nominal-specification> and <date-format> are defined below. The keywords numeric, real, integer, 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 writeWeka Filters to manipulate strings (like String- ToWordVectorFilter). 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).

**Relational attributes**

Relational attribute declarations take the form:

@attribute <name> relational

<further attribute definitions>

@end <name>

For the multi-instance dataset MUSK1 the definition would look like this (”...” denotes an omission):

@attribute molecule\_name {MUSK-jf78,...,NON-MUSK-199}

@attribute bag relational

@attribute f1 numeric

...

@attribute f166 numeric

@end bag

@attribute class {0,1}

...

**The 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. A percent sign (%) introduces a comment, which continues to the end of the line.

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 or the comment-delimiter character % must be quoted. (The code suggests that double-quotes are acceptable and that a backslash will escape individual characters.)

An example 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"

Relational data must be enclosed within double quotes ”. For example an instance of the MUSK1 dataset (”...” denotes an omission):

MUSK-188,"42,...,30",1

**Preprocess Tab**

**1. Loading Data**

The first four buttons at the top of the preprocess section enable you to load

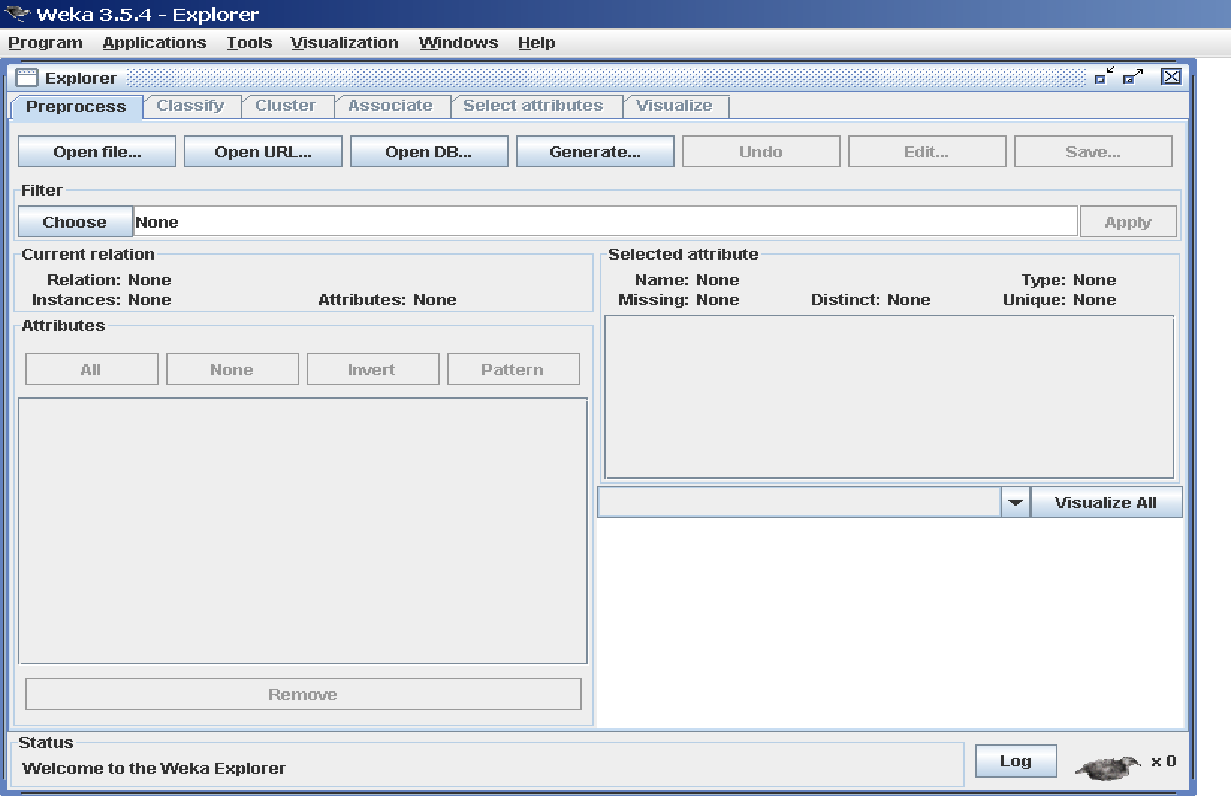
data into WEKA:

**1. Open file....** Brings up a dialog box allowing you to browse for the data file on the local file system.

**2. Open URL....** Asks for a Uniform Resource Locator address for where the data is stored.

**3. Open DB....** Reads data from a database. (Note that to make this work you might have to edit the file in weka/experiment/DatabaseUtils.props.)

**4. Generate....** Enables you to generate artificial data from a variety of Data Generators. Using the Open file... button you can read files in a variety of formats: WEKA’s ARFF format, CSV format, C4.5 format, or serialized Instances format. ARFF files typically have a .arff extension, CSV files a .csv extension, C4.5 files a .data and .names extension, and serialized Instances objects a .bsi extension.

****

**The Current Relation:** Once some data has been loaded, the Preprocess panel shows a variety of information. The Current relation box (the “current relation” is the currently loaded data, which can be interpreted as a single relational table in database terminology) has three entries:

**1. Relation.** The name of the relation, as given in the file it was loaded from. Filters (described below) modify the name of a relation.

**2. Instances.** The number of instances (data points/records) in the data.

**3. Attributes.** The number of attributes (features) in the data.

**Working With Attributes**

Below the Current relation box is a box titled Attributes. There are four buttons, and beneath them is a list of the attributes in the current relation.

The list has three columns:

**1. No..** A number that identifies the attribute in the order they are specified in the data file.

**2. Selection tick boxes**. These allow you select which attributes are present in the relation.

**3. Name.** The name of the attribute, as it was declared in the data file. When you click on different rows in the list of attributes, the fields change in the box to the right titled Selected attribute. This box displays the characteristics of the currently highlighted attribute in the list:

**1. Name.** The name of the attribute, the same as that given in the attribute list.

**2. Type.** The type of attribute, most commonly Nominal or Numeric.

**3. Missing.** The number (and percentage) of instances in the data for which this attribute is missing (unspecified).

**4. Distinct.** The number of different values that the data contains for this attribute.

**5. Unique.** The number (and percentage) of instances in the data having a value for this attribute that no other instances have.

Below these statistics is a list showing more information about the values stored in this attribute, which differ depending on its type. If the attribute is nominal, the list consists of each possible value for the attribute along with the number of instances that have that value. If the attribute is numeric, the list gives four statistics describing the distribution of values in the data—the minimum, maximum, mean and standard deviation. And below these statistics there is a coloured histogram, colour-coded according to the attribute chosen as the Class using the box above the histogram. (This box will bring up a drop-down list of available selections when clicked.) Note that only nominal Class attributes will result in a colour-coding. Finally, after pressing the Visualize All button, histograms for all the attributes in the data are shown in a separate window.

Returning to the attribute list, to begin with all the tick boxes are unticked.

They can be toggled on/off by clicking on them individually. The four buttons above can also be used to change the selection:

**PREPROCESSING**

**1. All.** All boxes are ticked.

**2. None.** All boxes are cleared (unticked).

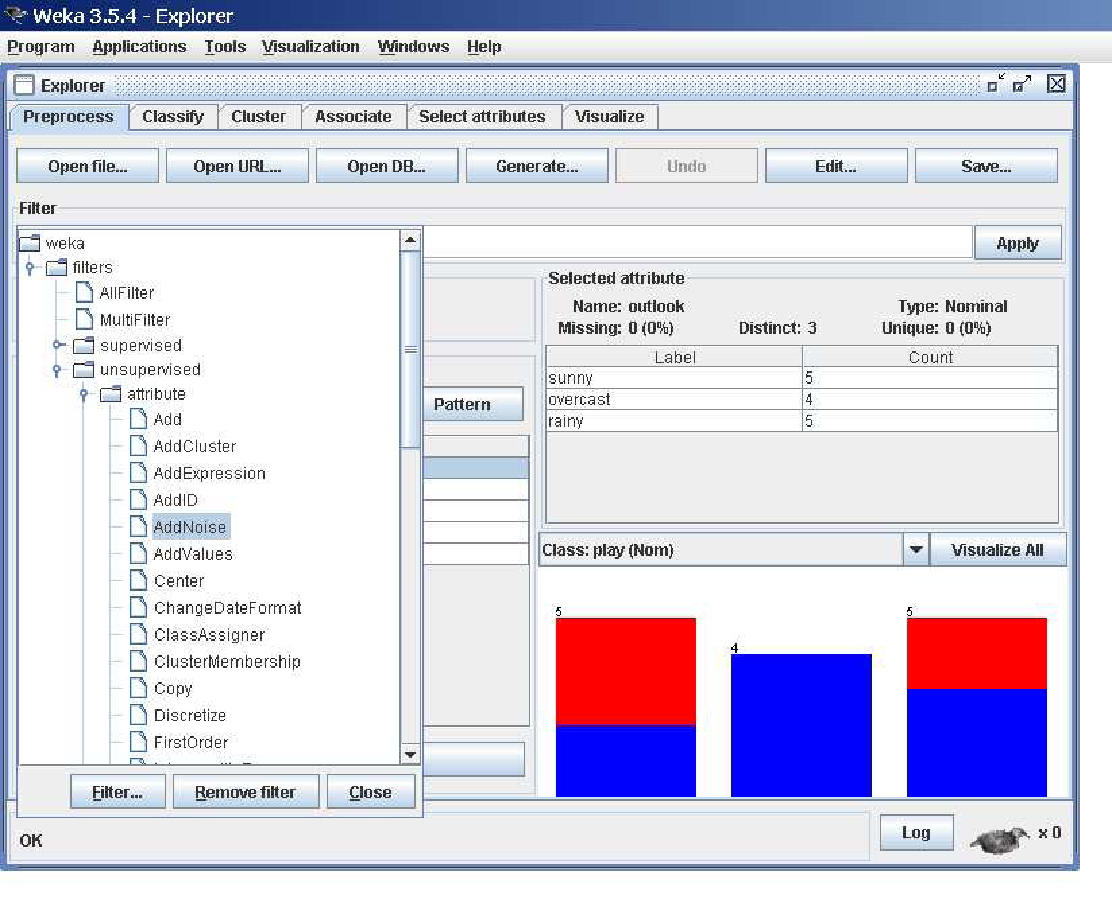
**3. Invert.** Boxes that are ticked become unticked and vice versa.

**4. Pattern.** Enables the user to select attributes based on a Perl 5 Regular Expression. E.g., .\* id selects all attributes which name ends with id.

Once the desired attributes have been selected, they can be removed by clicking the Remove button below the list of attributes. Note that this can be undone by clicking the Undo button, which is located next to the Edit button in the top-right corner of the Preprocess panel.

**Working With Filters**

The preprocess section allows filters to be defined that transform the data in various ways. The Filter box is used to set up the filters that are required. At the left of the Filter box is a Choose button. By clicking this button it is possible to select one of the filters in WEKA. Once a filter has been selected, its name and options are shown in the field next to the Choose button. Clicking on this box with the left mouse button brings up a GenericObjectEditor dialog box. A click with the right mouse button (or Alt+Shift+left click) brings up a menu where you can choose, either to display the properties in a GenericObjectEditor dialog box, or to copy the current setup string to the clipboard.

****

**The GenericObjectEditor Dialog Box**

The GenericObjectEditor dialog box lets you configure a filter. The same kind of dialog box is used to configure other objects, such as classifiers and clusterers (see below). The fields in the window reflect the available options. Right-clicking (or Alt+Shift+Left-Click) on such a field will bring up a popup menu, listing the following options:

**1. Show properties...** has the same effect as left-clicking on the field, i.e., a dialog appears allowing you to alter the settings.

**2. Copy configuration** to clipboard copies the currently displayed configuration string to the system’s clipboard and therefore can be used anywhere else in WEKA or in the console. This is rather handy if you have to setup complicated, nested schemes.

**3. Enter configuration...** is the “receiving” end for configurations that got copied to the clipboard earlier on. In this dialog you can enter a class name followed by options (if the class supports these). This also allows you to transfer a filter setting from the Preprocess panel to a Filtered Classifier used in the Classify panel.

Left-Clicking on any of these gives an opportunity to alter the filters settings. For example, the setting may take a text string, in which case you type the string into the text field provided. Or it may give a drop-down box listing several states to choose from. Or it may do something else, depending on the information required. Information on the options is provided in a tool tip if you let the mouse pointer hover of the corresponding field. More information on the filter and its options can be obtained by clicking on the More button in the About panel at the top of the GenericObjectEditor window.

Some objects display a brief description of what they do in an About box, along with a More button. Clicking on the More button brings up a window describing what the different options do. Others have an additional button, Capabilities, which lists the types of attributes and classes the object can handle.

At the bottom of the GenericObjectEditor dialog are four buttons. The first two, Open... and Save... allow object configurations to be stored for future use. The Cancel button backs out without remembering any changes that have been made. Once you are happy with the object and settings you have chosen, click OK to return to the main Explorer window.

**Applying Filters**

Once you have selected and configured a filter, you can apply it to the data by pressing the Apply button at the right end of the Filter panel in the Preprocess panel. The Preprocess panel will then show the transformed data. The change can be undone by pressing the Undo button. You can also use the Edit...button to modify your data manually in a dataset editor. Finally, the Save... button at the top right of the Preprocess panel saves the current version of the relation in file formats that can represent the relation, allowing it to be kept for future use.

**Note:** Some of the filters behave differently depending on whether a class attribute has been set or not (using the box above the histogram, which will bring up a drop-down list of possible selections when clicked). In particular, the “supervised filters” require a class attribute to be set, and some of the “unsupervised attribute filters” will skip the class attribute if one is set. Note that it is also possible to set Class to None, in which case no class is set.

**5. Classification Tab**

**5.1 Selecting a Classifier**

At the top of the classify section is the Classifier box. This box has a text field that gives the name of the currently selected classifier, and its options. Clicking on the text box with the left mouse button brings up a GenericObjectEditor dialog box, just the same as for filters, that you can use to configure the options of the current classifier. With a right click (or Alt+Shift+left click) you can once again copy the setup string to the clipboard or display the properties in a GenericObjectEditor dialog box. The Choose button allows you to choose one of the classifiers that are available in WEKA.

**5.2 Test Options**

The result of applying the chosen classifier will be tested according to the options that are set by clicking in the Test options box. There are four test modes:

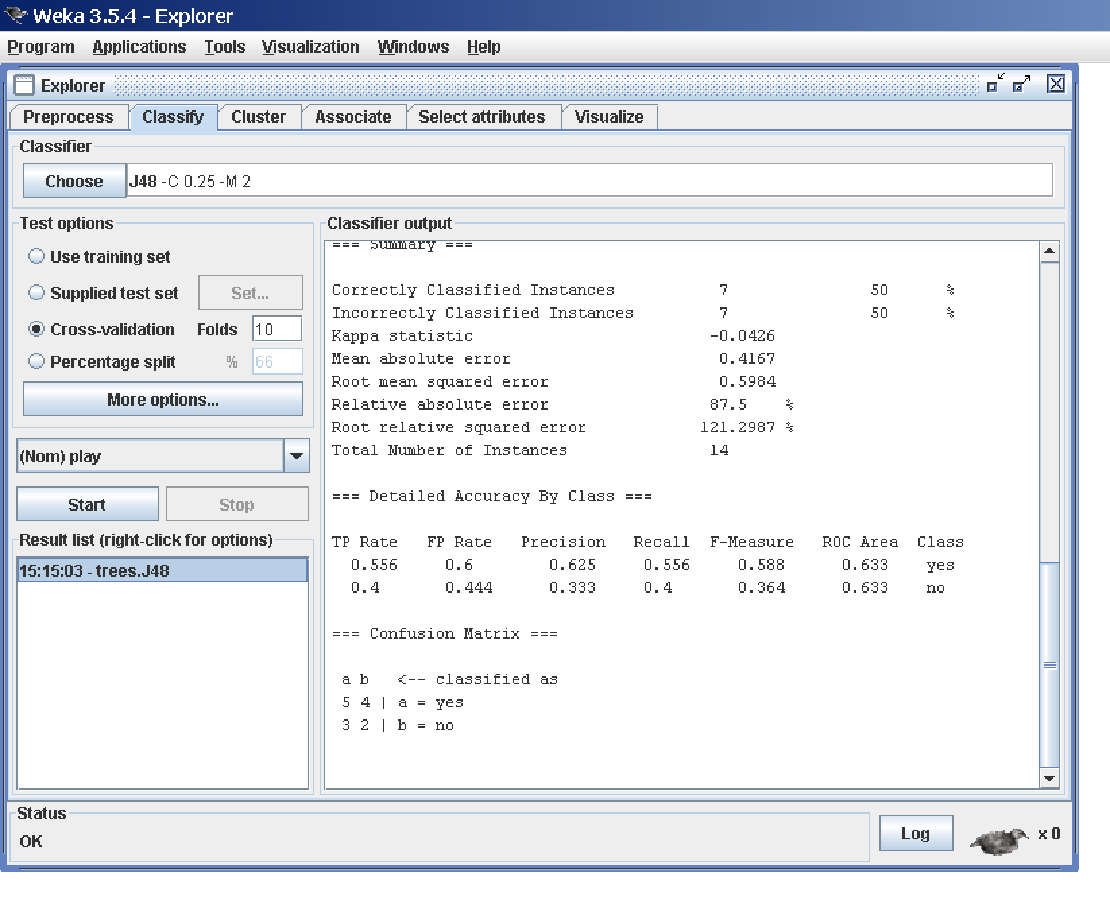
**1. Use training set.** The classifier is evaluated on how well it predicts the class of the instances it was trained on.

**2. Supplied test set.** The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file. Clicking the Set... button brings up a dialog allowing you to choose the file to test on.

**3. Cross-validation.** The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.

**4. Percentage split.** The classifier is evaluated on how well it predicts a certain percentage of the data which is held out for testing. The amount of data held out depends on the value entered in the % field.

Note: No matter which evaluation method is used, the model that is output is always the one build from all the training data. Further testing options can be set by clicking on the More options... button:

****

**1. Output model.** The classification model on the full training set is output so that it can be viewed, visualized, etc. This option is selected by default.

**2. Output per-class stats.** The precision/recall and true/false statistics for each class are output. This option is also selected by default.

**3. Output entropy evaluation measures.** Entropy evaluation measures are included in the output. This option is not selected by default.

**4. Output confusion matrix.** The confusion matrix of the classifier’s predictions is included in the output. This option is selected by default.

**5. Store predictions for visualization.** The classifier’s predictions are remembered so that they can be visualized. This option is selected by default.

**6. Output predictions.** The predictions on the evaluation data are output.

**Note** that in the case of a cross-validation the instance numbers do not correspond to the location in the data!

**7. Output additional attributes.** If additional attributes need to be output alongside the predictions, e.g., an ID attribute for tracking misclassifications, then the index of this attribute can be specified here. The usual Weka ranges are supported,“first” and “last” are therefore valid indices as well (example: “first-3,6,8,12-last”).

**8. Cost-sensitive evaluation.** The errors is evaluated with respect to a cost matrix. The Set... button allows you to specify the cost matrix used.

**9. Random seed for xval / % Split.** This specifies the random seed used when randomizing the data before it is divided up for evaluation purposes.

**10. Preserve order for % Split.** This suppresses the randomization of the data before splitting into train and test set.

**11. Output source code.** If the classifier can output the built model as Java source code, you can specify the class name here. The code will be printed in the “Classifier output” area.

**5.3 The Class Attribute**

The classifiers in WEKA are designed to be trained to predict a single ‘class’ attribute, which is the target for prediction. Some classifiers can only learn nominal classes; others can only learn numeric classes (regression problems) still others can learn both.

By default, the class is taken to be the last attribute in the data. If you want to train a classifier to predict a different attribute, click on the box below the Test options box to bring up a drop-down list of attributes to choose from.

**5.4 Training a Classifier**

Once the classifier, test options and class have all been set, the learning process is started by clicking on the Start button. While the classifier is busy being trained, the little bird moves around. You can stop the training process at any time by clicking on the Stop button. When training is complete, several things happen. The Classifier output area to the right of the display is filled with text describing the results of training and testing. A new entry appears in the Result list box. We look at the result list below; but first we investigate the text that has been output.

**5.5 The Classifier Output Text**

The text in the Classifier output area has scroll bars allowing you to browse the results. Clicking with the left mouse button into the text area, while holding Alt and Shift, brings up a dialog that enables you to save the displayed output in a variety of formats (currently, BMP, EPS, JPEG and PNG). Of course, you can also resize the Explorer window to get a larger display area.

The output is

**Split into several sections:**

1. Run information. A list of information giving the learning scheme options, relation name, instances, attributes and test mode that were involved in the process.

2. Classifier model (full training set). A textual representation of the classification model that was produced on the full training data.

3. The results of the chosen test mode are broken down thus.

4. Summary. A list of statistics summarizing how accurately the classifier was able to predict the true class of the instances under the chosen test mode.

5. Detailed Accuracy By Class. A more detailed per-class break down of the classifier’s prediction accuracy.

6. Confusion Matrix. Shows how many instances have been assigned to each class. Elements show the number of test examples whose actual class is the row and whose predicted class is the column.

7. Source code (optional). This section lists the Java source code if one

chose “Output source code” in the “More options” dialog.

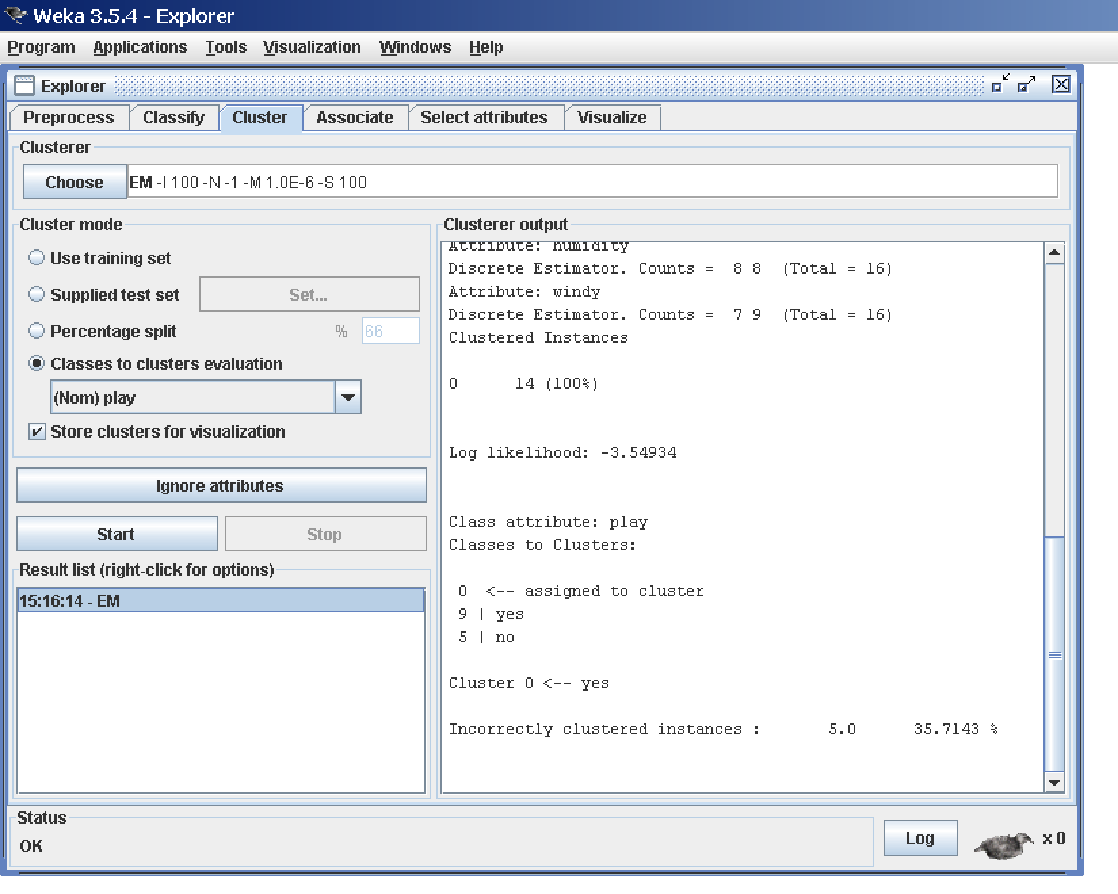
**6. Clustering Tab**

**6.1 Selecting a Clusterer**

By now you will be familiar with the process of selecting and configuring objects. Clicking on the clustering scheme listed in the Clusterer box at the top of the

window brings up a GenericObjectEditor dialog with which to choose a new

clustering scheme.



**Cluster Modes**

The Cluster mode box is used to choose what to cluster and how to evaluate the results. The first three options are the same as for classification: Use training set, Supplied test set and Percentage split (Section 5.3.1)—except that now the data is assigned to clusters instead of trying to predict a specific class. The fourth mode, Classes to clusters evaluation, compares how well the chosen clusters match up with a pre-assigned class in the data. The drop-down box below this option selects the class, just as in the Classify panel.

An additional option in the Cluster mode box, the Store clusters for visualization tick box, determines whether or not it will be possible to visualize the clusters once training is complete. When dealing with datasets that are so

large that memory becomes a problem it may be helpful to disable this option.

**6.2 Ignoring Attributes**

Often, some attributes in the data should be ignored when clustering. The Ignore attributes button brings up a small window that allows you to select which attributes are ignored. Clicking on an attribute in the window highlights it, holding down the SHIFT key selects a range of consecutive attributes, and holding down CTRL toggles individual attributes on and off. To cancel the selection, back out with the Cancel button. To activate it, click the Select button. The next time

is invoked, the selected attributes are ignored.

**6.3 Working with Filters**

The Filtered Clusterer meta-clusterer offers the user the possibility to apply filters directly before the clusterer is learned. This approach eliminates the manual application of a filter in the Preprocess panel, since the data gets processed on the fly. Useful if one needs to try out different filter setups.

**6.4 Learning Clusters**

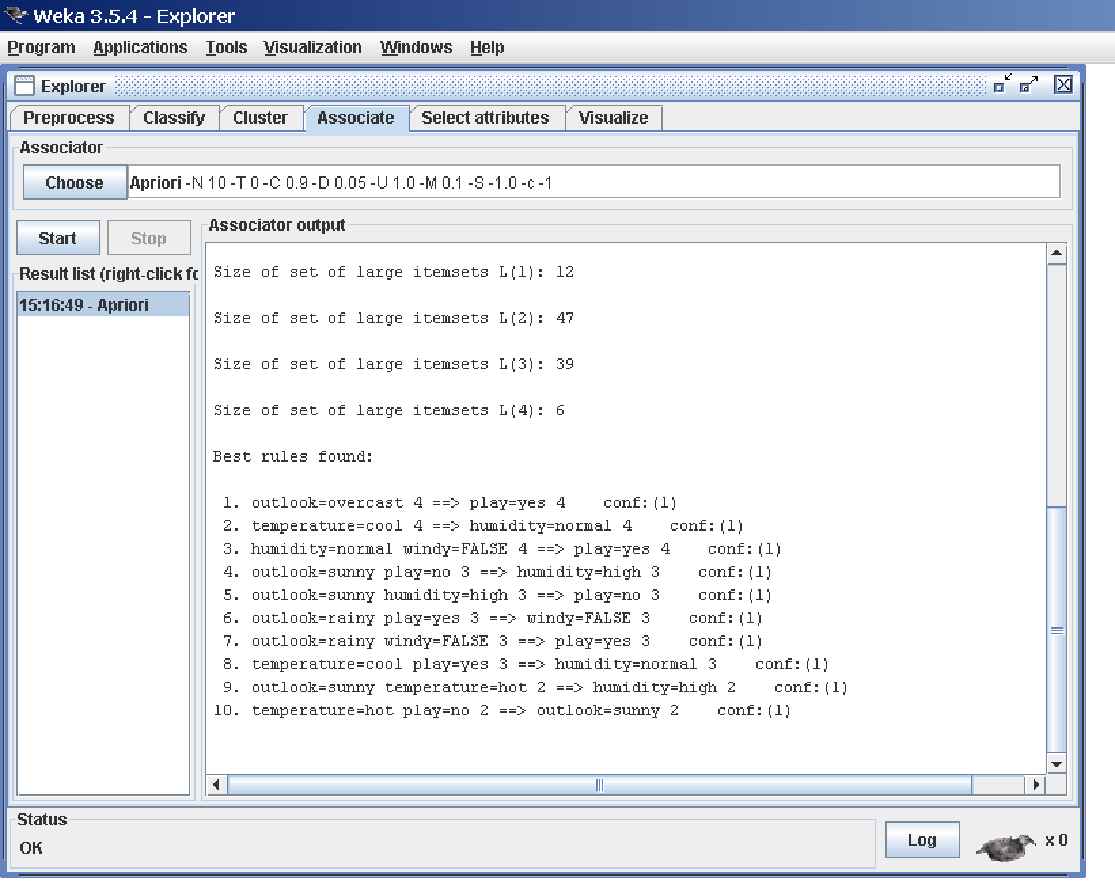
The Cluster section, like the Classify section, has Start/Stop buttons, a result text area and a result list. These all behave just like their classification counterparts. Right-clicking an entry in the result list brings up a similar menu, except that it shows only two visualization options: Visualize cluster assignments and Visualize tree. The latter is grayed out when it is not applicable.

**7. Associate Tab**

**7.1 Setting Up**

This panel contains schemes for learning association rules, and the learners are chosen and configured in the same way as the clusterers, filters, and classifiers

in the other panels.



**7.2 Learning Associations**

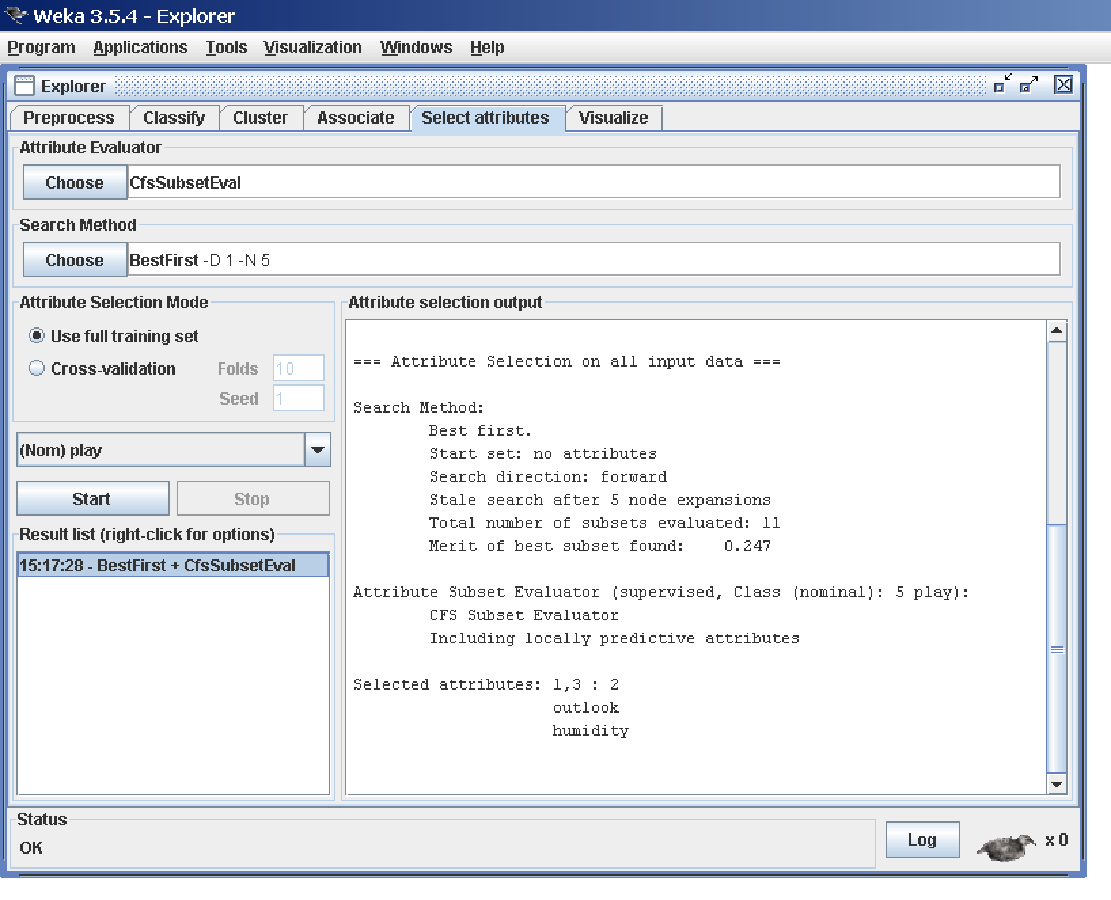
Once appropriate parameters for the association rule learner have been set, click the Start button. When complete, right-clicking on an entry in the result list allows the results to be viewed or saved.

**8. Selecting Attributes Tab**

**8.1 Searching and Evaluating**

Attribute selection involves searching through all possible combinations of attributes in the data to find which subset of attributes works best for prediction.

To do this, two objects must be set up: an attribute evaluator and a search method. The evaluator determines what method is used to assign a worth to each subset of attributes. The search method determines what style of search is performed.



**8.2 Options**

**The Attribute Selection Mode box has two options:**

**1. Use full training set.** The worth of the attribute subset is determined using the full set of training data.

**2. Cross-validation.** The worth of the attribute subset is determined by a process of cross-validation. The Fold and Seed fields set the number of folds to use and the random seed used when shuffling the data. As with Classify (Section 5.3.1), there is a drop-down box that can be used to specify which attribute to treat as the class.

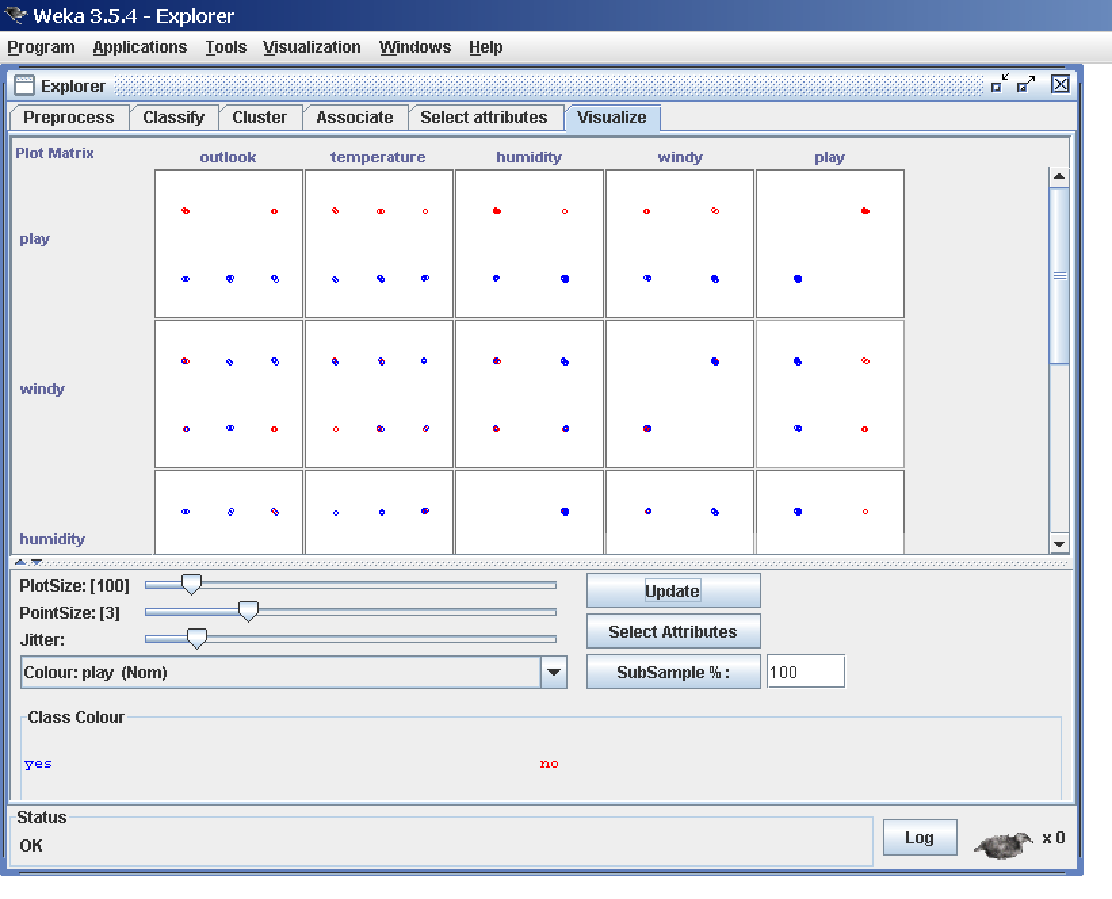
**8.3 Performing Selection**

Clicking Start starts running the attribute selection process. When it is finished, the results are output into the result area, and an entry is added to the result list. Right-clicking on the result list gives several options. The first three, (View in main window, View in separate window and Save result buffer), are the same as for the classify panel. It is also possible to Visualize reduced data, or if you have used an attribute transformer such as Principal Components, Visualize transformed data. The reduced/transformed data can be saved to a file with the Save reduced data... or Save transformed data... option.

In case one wants to reduce/transform a training and a test at the same time and not use the Attribute Selected Classifier from the classifier panel, it is best to use the Attribute Selection filter (a supervised attribute filter) in batch mode (’-b’) from the command line or in the Simple CLI. The batch mode allows one to specify an additional input and output file pair (options -r and -s), that is processed with the filter setup that was determined based on the training data.

**9. Visualizing Tab**

WEKA’s visualization section allows you to visualize 2D plots of the current relation.



**9.1 The scatter plot matrix**

When you select the Visualize panel, it shows a scatter plot matrix for all the attributes, colour coded according to the currently selected class. It is possible to change the size of each individual 2D plot and the point size, and to randomly jitter the data (to uncover obscured points). It also possible to change the attribute used to colour the plots, to select only a subset of attributes for inclusion in the scatter plot matrix, and to sub sample the data. Note that changes will only come into effect once the Update button has been pressed.

**9.2 Selecting an individual 2D scatter plot**

When you click on a cell in the scatter plot matrix, this will bring up a separate window with a visualization of the scatter plot you selected. (We described above how to visualize particular results in a separate window—for example, classifier errors—the same visualization controls are used here.)

Data points are plotted in the main area of the window. At the top are two drop-down list buttons for selecting the axes to plot. The one on the left shows which attribute is used for the x-axis; the one on the right shows which is used for the y-axis.

Beneath the x-axis selector is a drop-down list for choosing the colour scheme. This allows you to colour the points based on the attribute selected. Below the plot area, a legend describes what values the colours correspond to. If the values are discrete, you can modify the colour used for each one by clicking on them and making an appropriate selection in the window that pops up.

To the right of the plot area is a series of horizontal strips. Each strip represents an attribute, and the dots within it show the distribution of values of the attribute. These values are randomly scattered vertically to help you see concentrations of points. You can choose what axes are used in the main graph by clicking on these strips. Left-clicking an attribute strip changes the x-axis to that attribute, whereas right-clicking changes the y-axis. The ‘X’ and ‘Y’ written beside the strips shows what the current axes are (‘B’ is used for ‘both X and Y’).

Above the attribute strips is a slider labelled Jitter, which is a random displacement given to all points in the plot. Dragging it to the right increases the amount of jitter, which is useful for spotting concentrations of points. Without jitter, a million instances at the same point would look no different to just a

single lonely instance.

**9.3 Selecting Instances**

There may be situations where it is helpful to select a subset of the data using the visualization tool. (A special case of this is the User Classifier in the Classify panel, which lets you build your own classifier by interactively selecting instances.)

Below the y-axis selector button is a drop-down list button for choosing a selection method. A group of data points can be selected in four ways:

**1. Select Instance.** Clicking on an individual data point brings up a window listing its attributes. If more than one point appears at the same location, more than one set of attributes is shown.

**2. Rectangle.** You can create a rectangle, by dragging, that selects the points inside it.

**3. Polygon.** You can build a free-form polygon that selects the points inside it. Left-click to add vertices to the polygon, right-click to complete it. The polygon will always be closed off by connecting the first point to the last.

**4. Polyline.** You can build a polyline that distinguishes the points on one side from those on the other. Left-click to add vertices to the polyline, right-click to finish. The resulting shape is open (as opposed to a polygon, which is always closed).

Once an area of the plot has been selected using Rectangle, Polygon or Polyline, it turns grey. At this point, clicking the Submit button removes all instances from the plot except those within the grey selection area. Clicking on the Clear button erases the selected area without affecting the graph.

Once any points have been removed from the graph, the Submit button changes to a Reset button. This button undoes all previous removals and returns you to the original graph with all points included. Finally, clicking the Save button allows you to save the currently visible instances to a new ARFF file.

**Part-A: Data Mining**

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

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

**Weka** - **Preprocessing** the Data. The data that is collected from the field contains many unwanted things that leads to wrong analysis. For example, the data may contain null fields, it may contain columns that are irrelevant to the current analysis, and so on.

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

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

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

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

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

Step5:Selecting or filtering attributes Removing an attribute-When we need to remove an attribute,we can do this by using the attribute filters in weka.In the filter model panel,click on choose button,This will show a popup window with a list of available filters. Scroll down the list and select the “weka.filters.unsupervised.attribute.remove” filters.

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

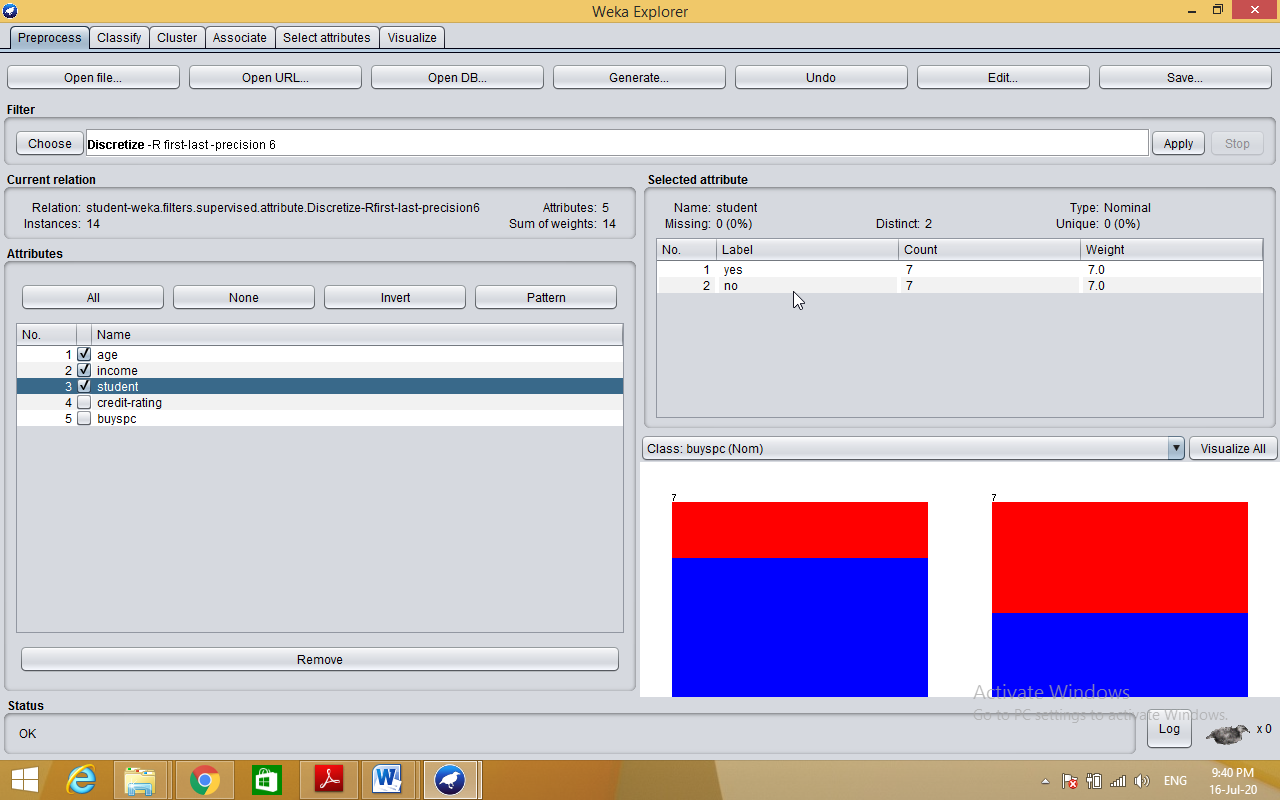
b)Make sure that invert selection option is set to false.The click OK now in the filter box.you will see “Remove-R-7”.

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

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

**Procedure:**

Choose Tab🡪Weka🡪filters🡪Supervised🡪attribute🡪Discretize



**Dataset student .arff**

@relation student

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

@attribute income {low, medium, high}

@attribute student {yes, no}

@attribute credit-rating {fair, excellent}

@attribute buyspc {yes, no}

@data

%

<30, high, no, fair, no

<30, high, no, excellent, no

30-40, high, no, fair, yes

>40, medium, no, fair, yes

>40, low, yes, fair, yes

>40, low, yes, excellent, no

30-40, low, yes, excellent, yes

<30, medium, no, fair, no

<30, low, yes, fair, no

>40, medium, yes, fair, yes

<30, medium, yes, excellent, yes

30-40, medium, no, excellent, yes

30-40, high, yes, fair, yes

>40, medium, no, excellent, no

%

**2. Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm.**

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

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

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

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

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

**Description:**

**The Apriori Algorithm:** Finding Frequent Itemsets Using Candidate Generation

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. Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are used to explore (*k*+1)-itemsets.

**Apriori property:** All nonempty subsets of a frequent itemset must also be frequent.

It has two steps.

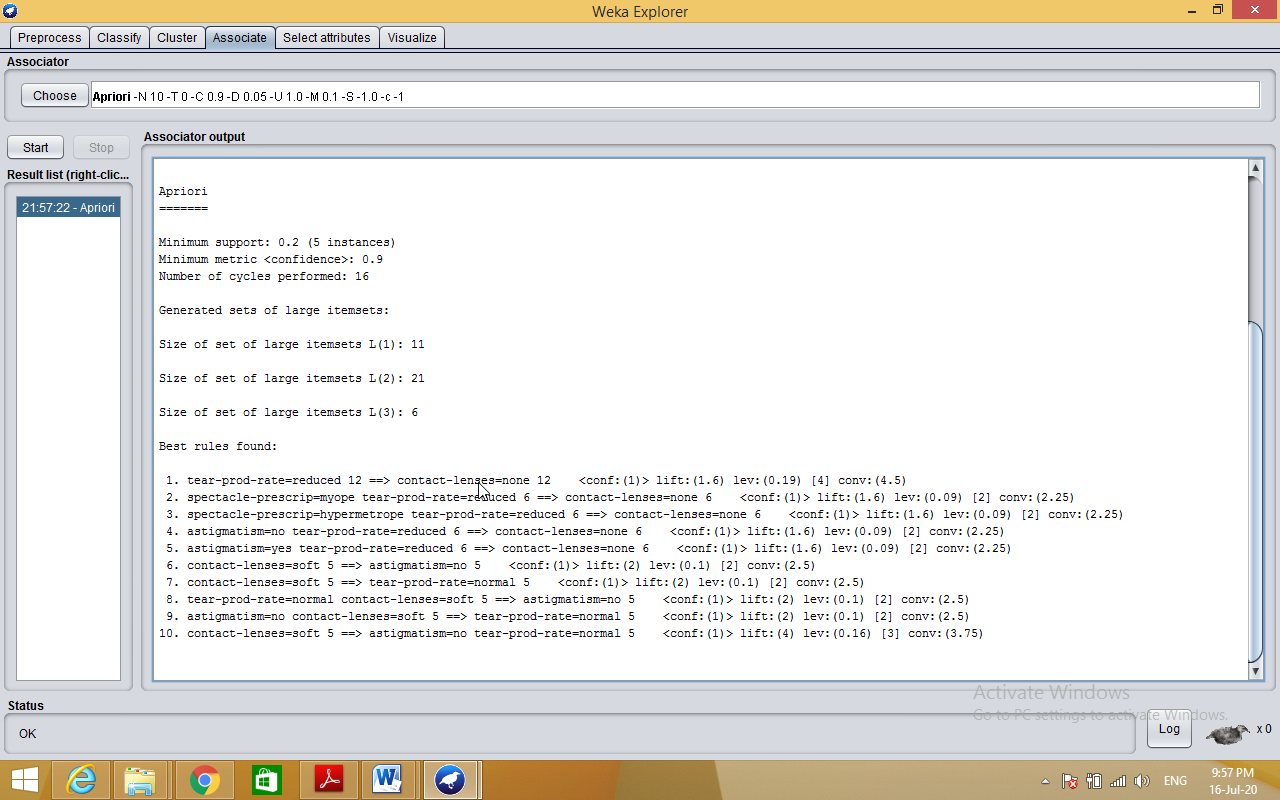
1) Join step

2) Prune step

**Procedure:**

Step 1:Load appropriate dataset into weka

Step 2: Select associate tab and select apriori algorithm



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

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

Steps involved in this experiment:

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

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

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

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

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

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

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

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

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

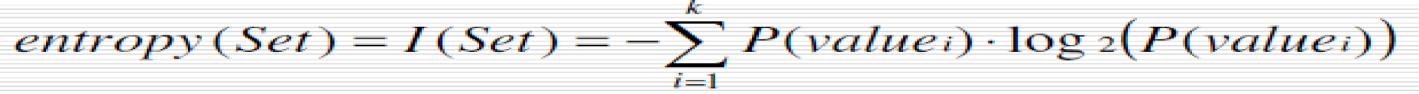
**Description:**

**Classification** is a **data mining** function that assigns items in a collection to target categories or classes. The goal of **classification** is to accurately predict the target class for each case in the **data**. For example, a **classification** model could be used to identify loan applicants as low, medium, or high credit risks.

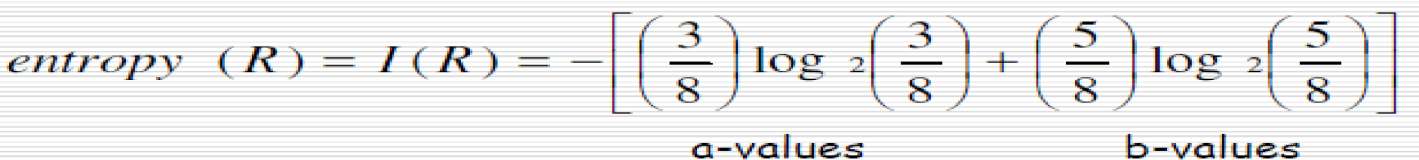
**Entropy** characterizes the (im) purity of an arbitrary collection of examples

Information Gain is the expected reduction in entropy caused by partitioning the examples according to a given attribute

If we have a set with k different values in it, we can calculate the entropy as follows:



Where P(valuei) is the probability of getting the ith value when randomly selecting one from the set. So, for the set R = {a,a,a,b,b,b,b,b}

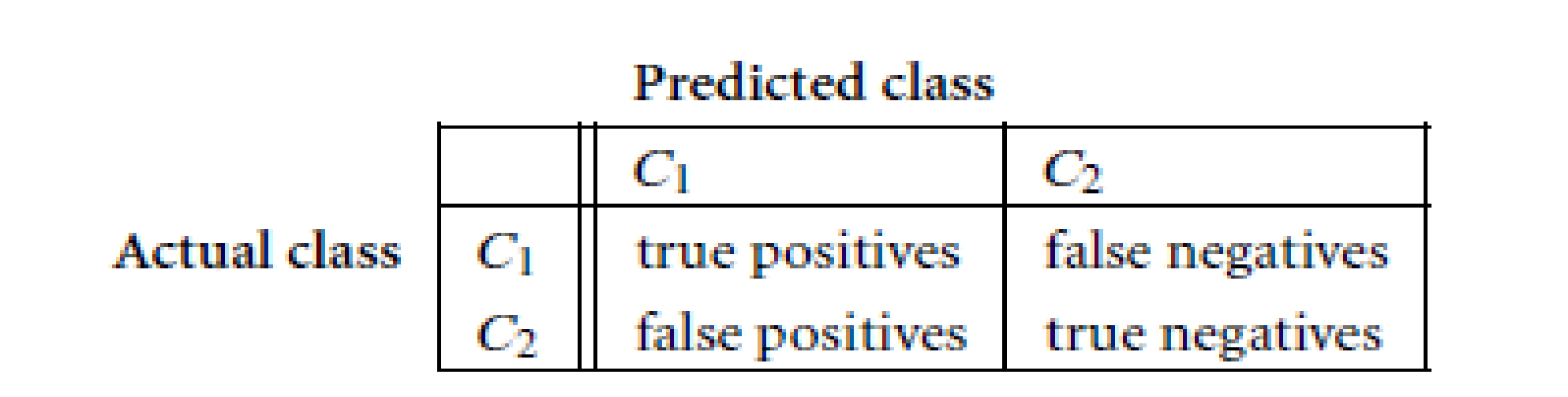
****

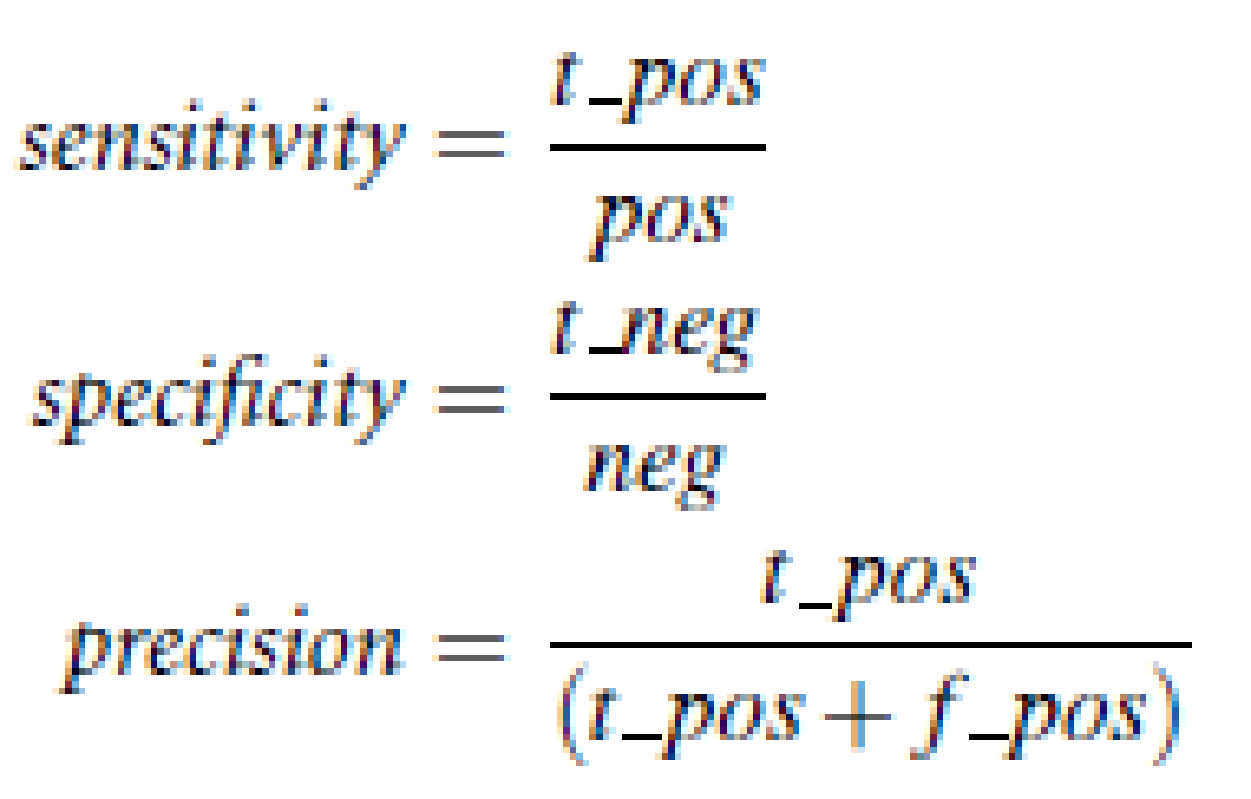
**Kappa statistic** is a measure of how closely the instances classified by the machine learning classifier matched the **data** labeled as ground truth, controlling for the accuracy of a random classifier as measured by the expected accuracy.

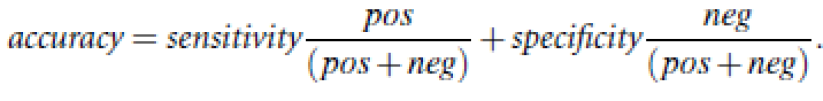
**Decision tree** builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**.

**Classifier Accuracy Measures**

The ***confusion matrix***is a useful tool for analyzing how well your classifier can recognize tuples of different classes.

****

****

****

**F measure** (F1 score or **F** score) is a **measure** of a test's accuracy and is defined as the weighted harmonic mean of the precision and recall of the test.

**TP Rate**: **rate** of true positives (instances correctly classified as a given class)

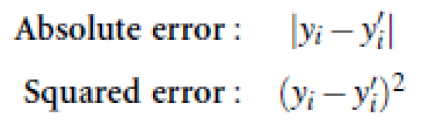
FP**Rate**: **rate** of false positives (instances falsely classified as a given class)

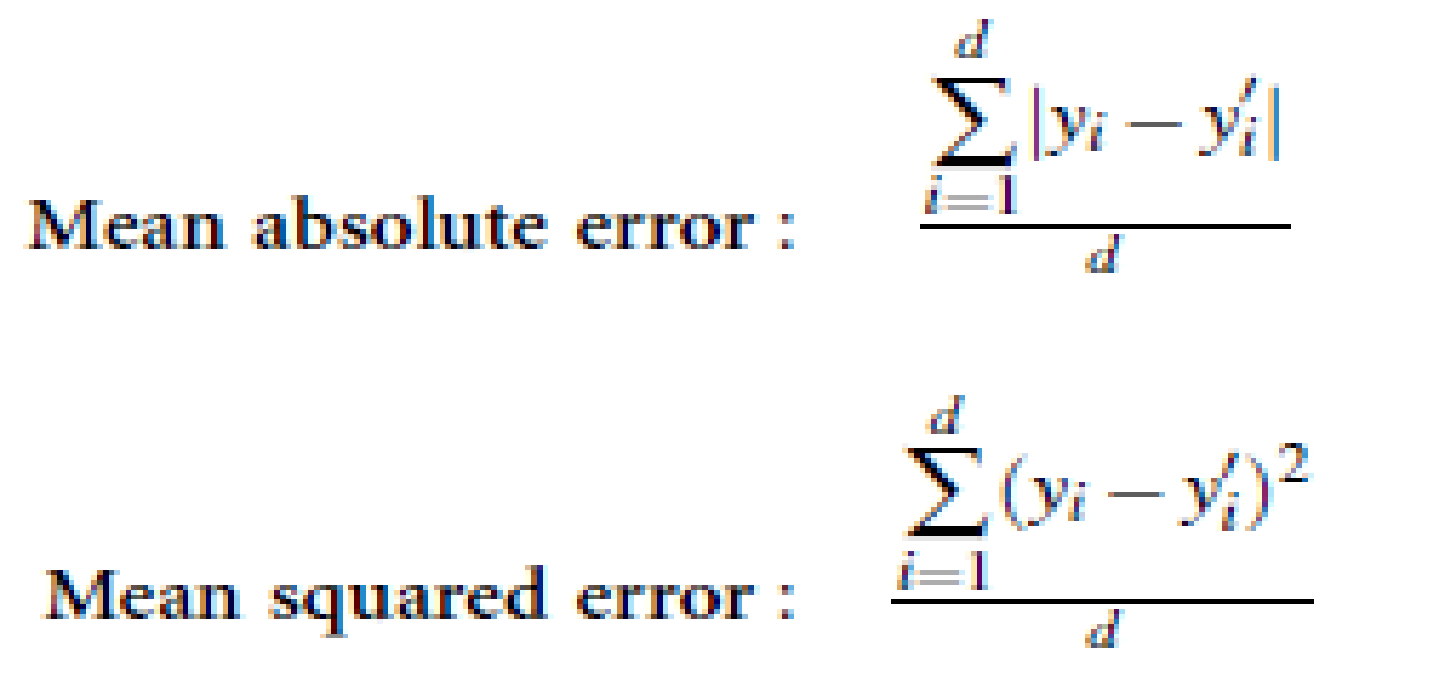
**Precision** is the fraction of retrieved data that are [relevant](https://en.wikipedia.org/wiki/Relevance_(information_retrieval)) to the query:

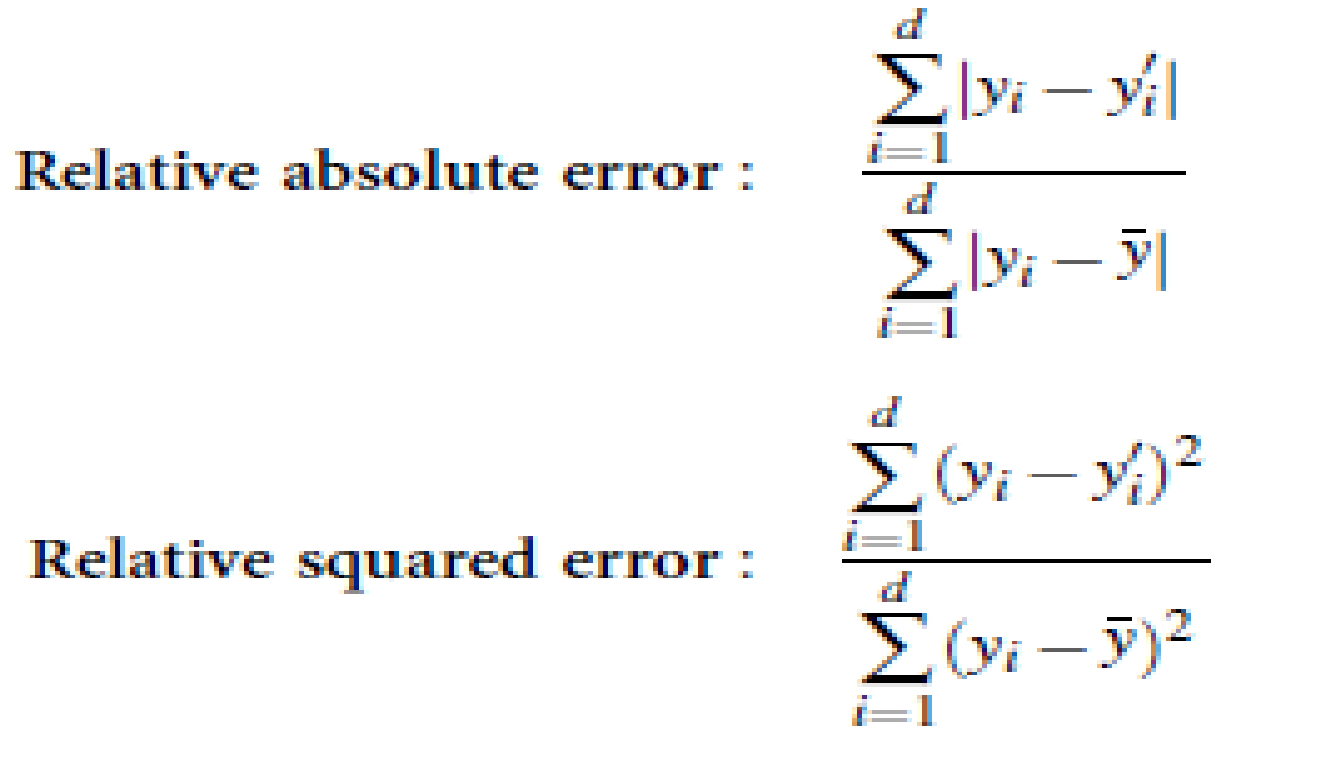
**Recall** is the fraction of the relevant data that are successfully retrieved.

**Cross-validation:** In ***k***-fold **cross-validation**, the initial data are randomly partitioned into *k* mutually exclusive subsets or “folds,” *D*1, *D*2: *Dk*, each of approximately equal size.

**Predictor Error Measures**

****

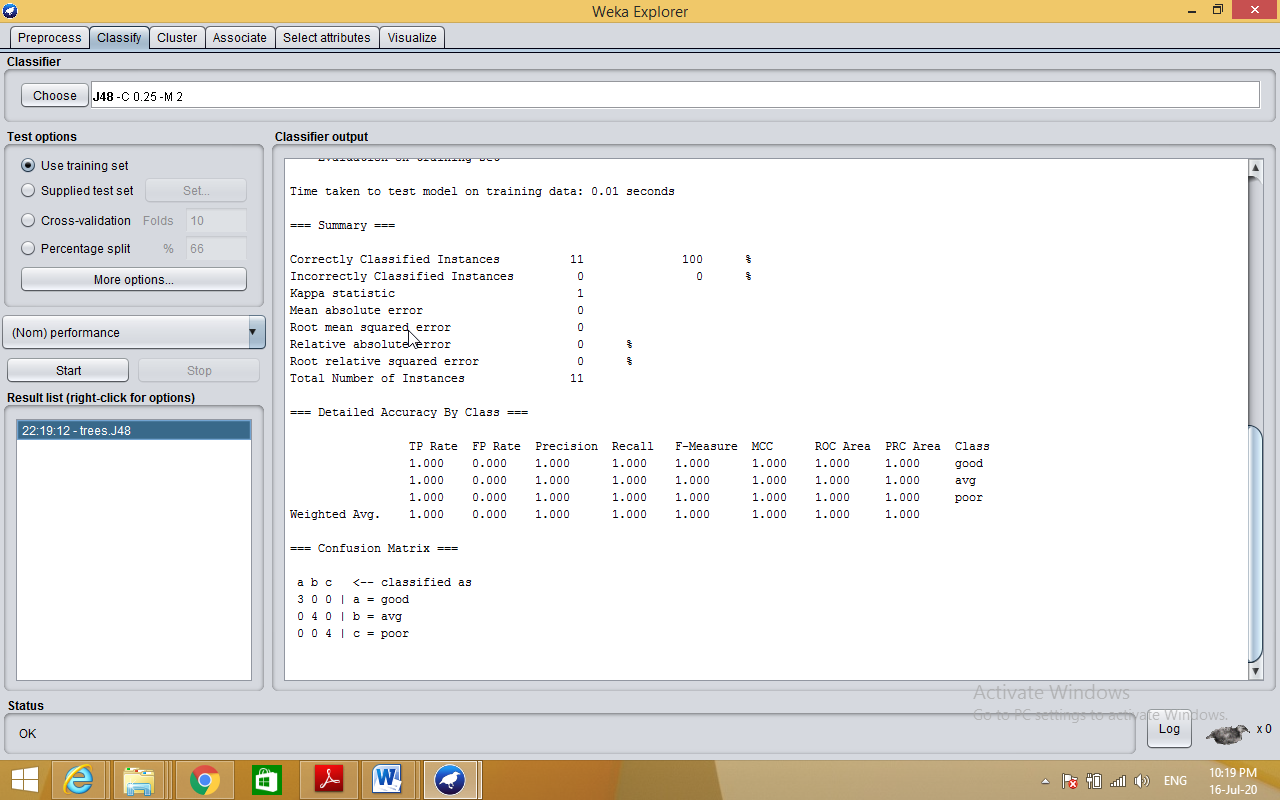
****

****

**Procedure:**

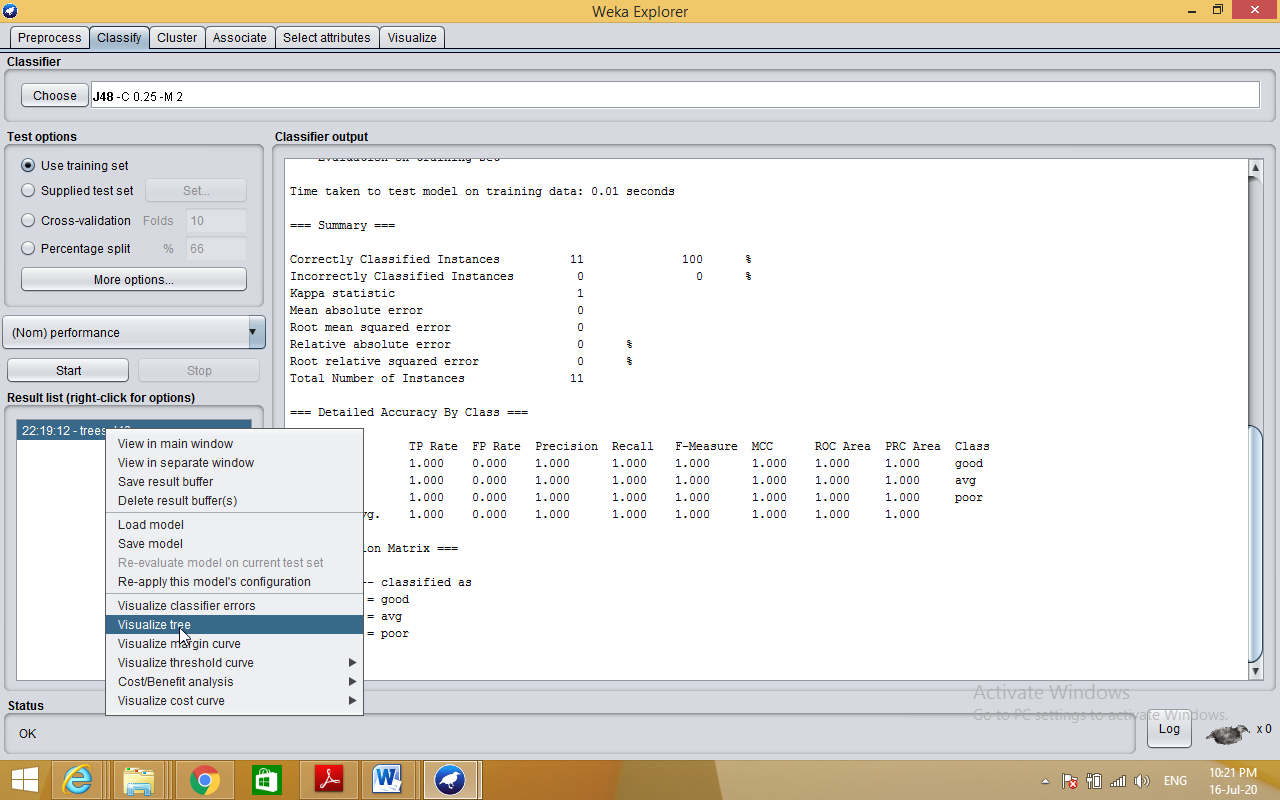
Step 1: Load appropriate dataset into WEKA.

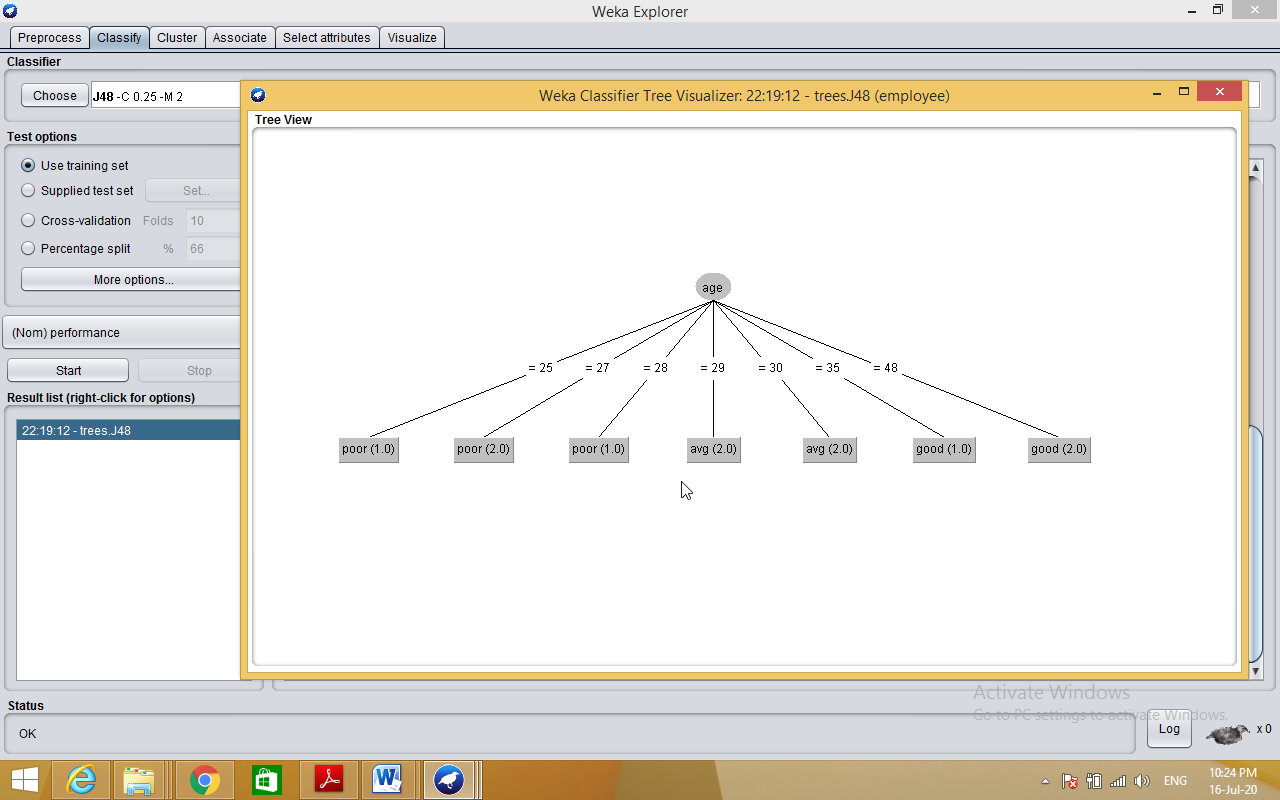
Step 2: Select classify TAB and select J48 algorithm.



**Generating Decision Tree:-**

Right click on result list🡪select visualize tree option





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

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

Steps involved in this experiment:

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

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

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

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

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

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

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

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

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

**Description: Classification** is a **data mining** function that assigns items in a collection to target categories or classes. The goal of **classification** is to accurately predict the target class for each case in the **data**. For example, a **classification** model could be used to identify loan applicants as low, medium, or high credit risks.

**Data set employee.arff:**

@relation employee

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

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

@attribute performance {good, avg, poor}

@data

%

25, 10k, poor

27, 15k, poor

27, 17k, poor

28, 17k, poor

29, 20k, avg

30, 25k, avg

29, 25k, avg

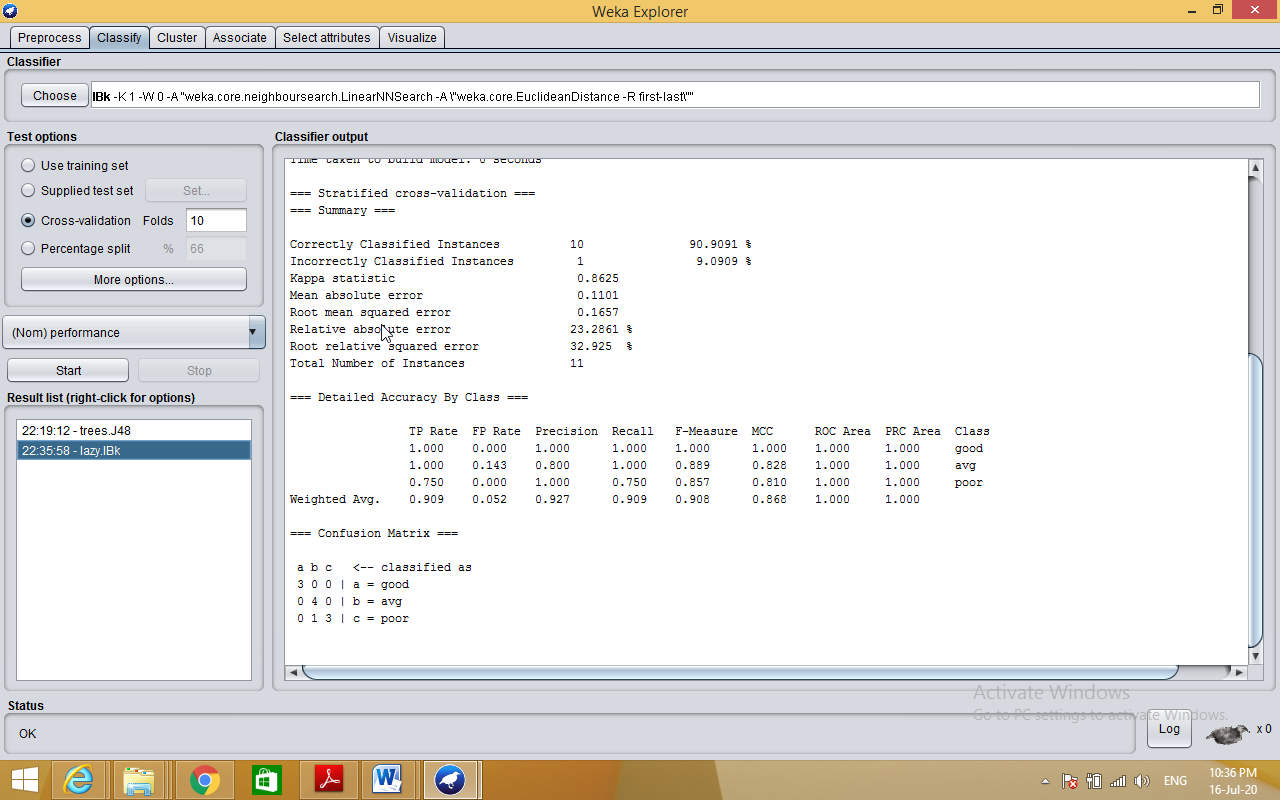
30, 20k, avg

35, 32k, good

48, 35k, good

48, 32k,good

%



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

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

Steps involved in this experiment:

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

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

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

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

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

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

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

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

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

**Description:**

**Bayesian classification** is based on Baye’s Theorem. Bayesian classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

## Baye's Theorem

Bayes' Theorem is named after Thomas Bayes. There are two types of probabilities

* Posterior Probability [P(H/X)]
* Prior Probability [P(H)]

where X is data tuple and H is some hypothesis.

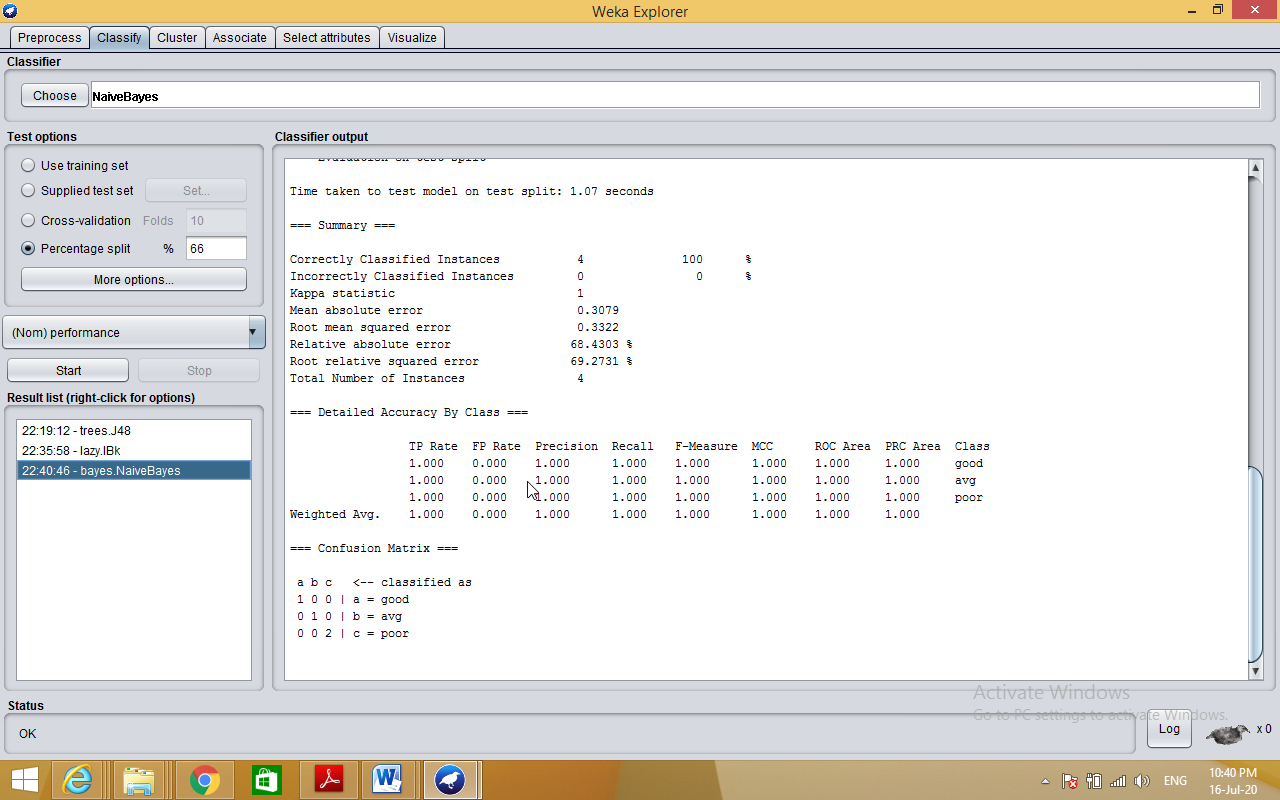
According to Bayes' Theorem,

P(H/X)= P(X/H)P(H) / P(X)

**Procedure:**

Step 1: Load appropriate dataset into WEKA

Step 2: Go to Classify tab🡪Bayes🡪NaiveBayes



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

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

Steps involved in this Experiment

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

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

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

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

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

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

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

**Description:**

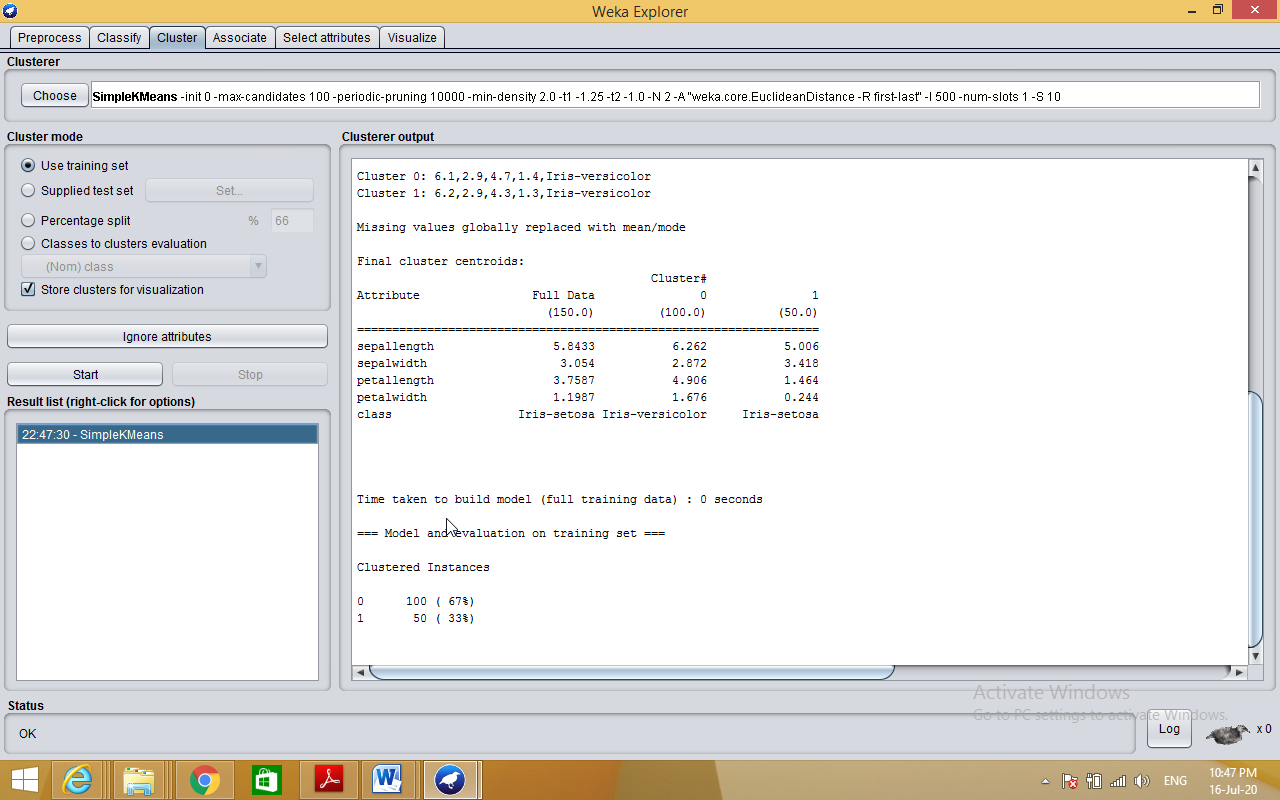
**Clustering** is a process of partitioning a set of **data** (or objects) into a set of meaningful sub-classes, called **clusters**.

**K-Means clustering** intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean.

**Procedure:**

Step 1: Load appropriate dataset into WEKA

Step 2: Go to Cluster tab🡪choose🡪SimpleKmeans



**7. Demonstration of clustering rule process on dataset student.arff using hierarchical clustering.**

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

Steps involved in this Experiment

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

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

Step 3 : In this case we select ‘hierarchical’.

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

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

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

Step 7: Another way of understanding characteristics of each cluster through visualization ,we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments. Interpretation of the above visualization From the above visualization, we can understand the distribution of age and instance number in each cluster. For instance, for each cluster is dominated by age. In this case by changing the color dimension to other attributes we can see their distribution with in each of the cluster.

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

**Description:**

**Hierarchical clustering** involves creating **clusters** that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a **hierarchy**. There are two types of **hierarchical clustering**, Divisive and Agglomerative.

**Procedure:**

Step 1: Load appropriate dataset into WEKA

Step 2: Go to Cluster tab🡪choose🡪Hierarchical clustering

