

EX : 1 Installation of WEKA Tool

WEKA Machine Learning Tool

WEKA is used by machine learning and applied sciences researchers for learning purposes. It is an efficient tool for carrying out many data mining tasks.

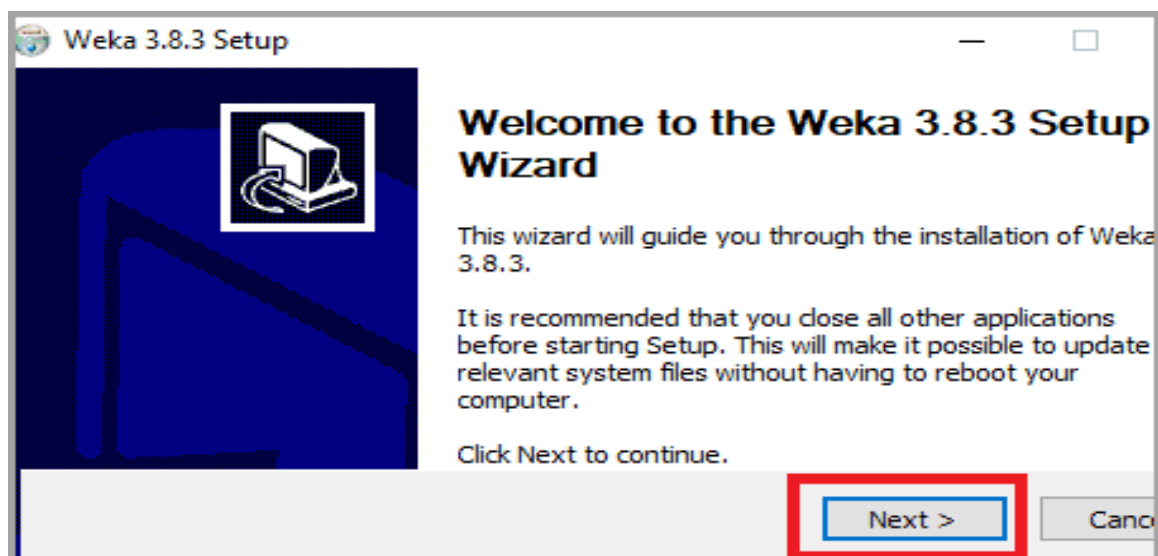
WEKA Download And Installation

STEP 1 : Download the software from [here](https://www.cs.waikato.ac.nz/ml/weka/downloading.html).

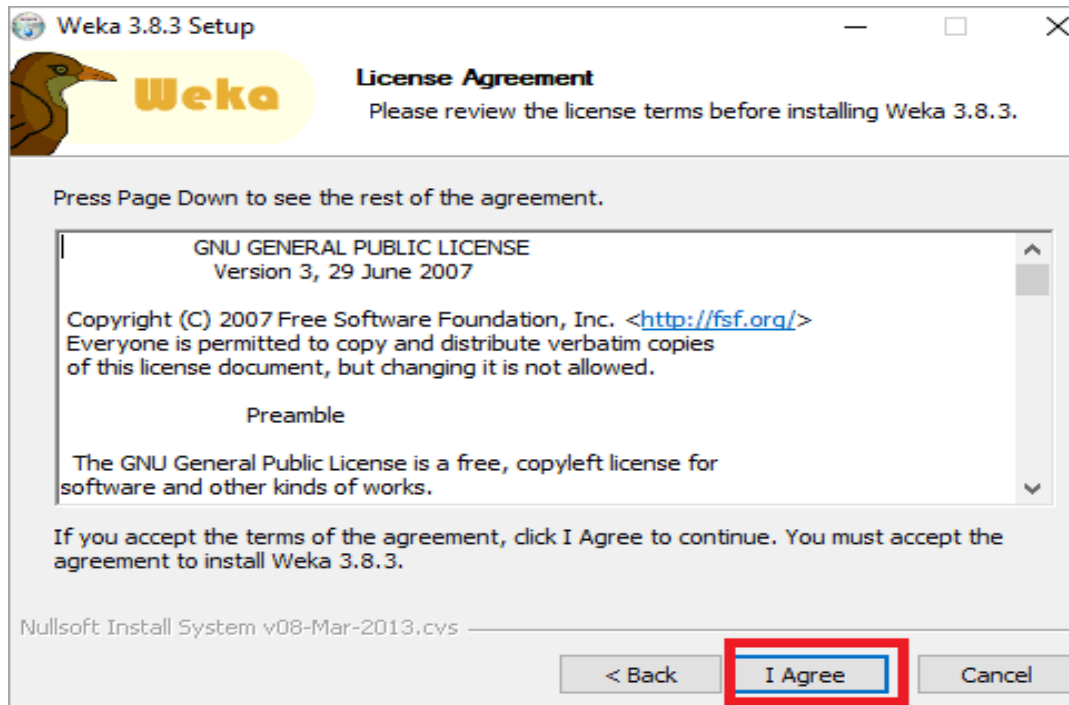
Check the configuration of the computer system and download the stable version of WEKA (currently 3.8) from this page.



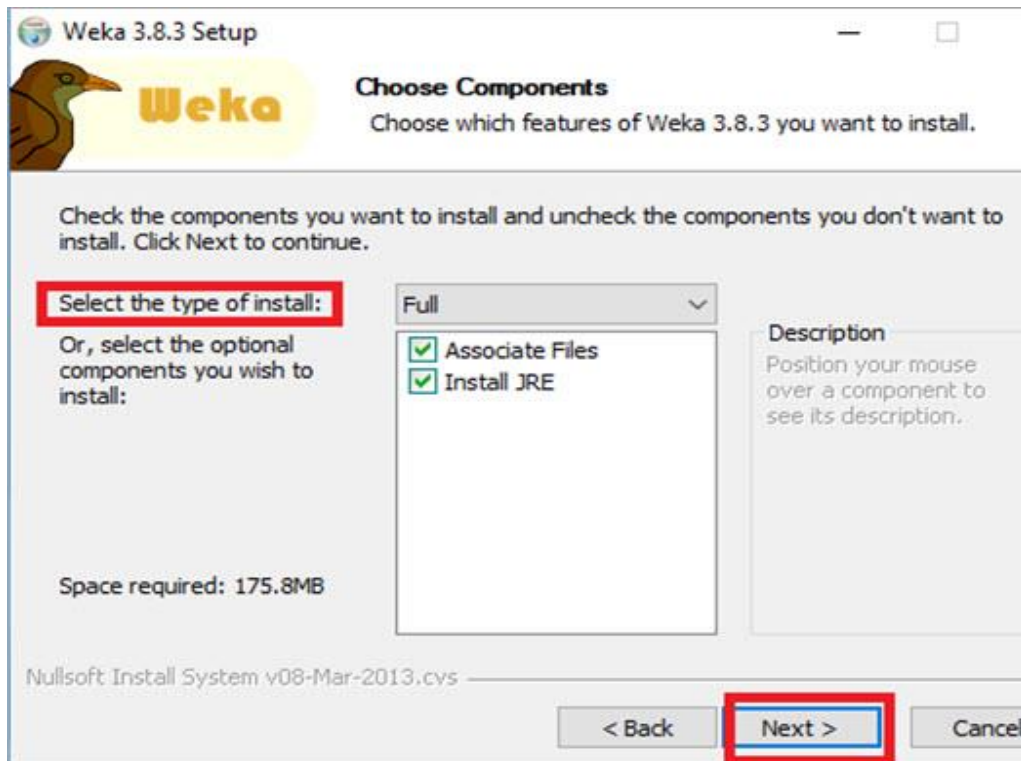
STEP 2 : After successful download, open the file location and double-click on the downloaded file. The Step Up wizard will appear. Click on Next.



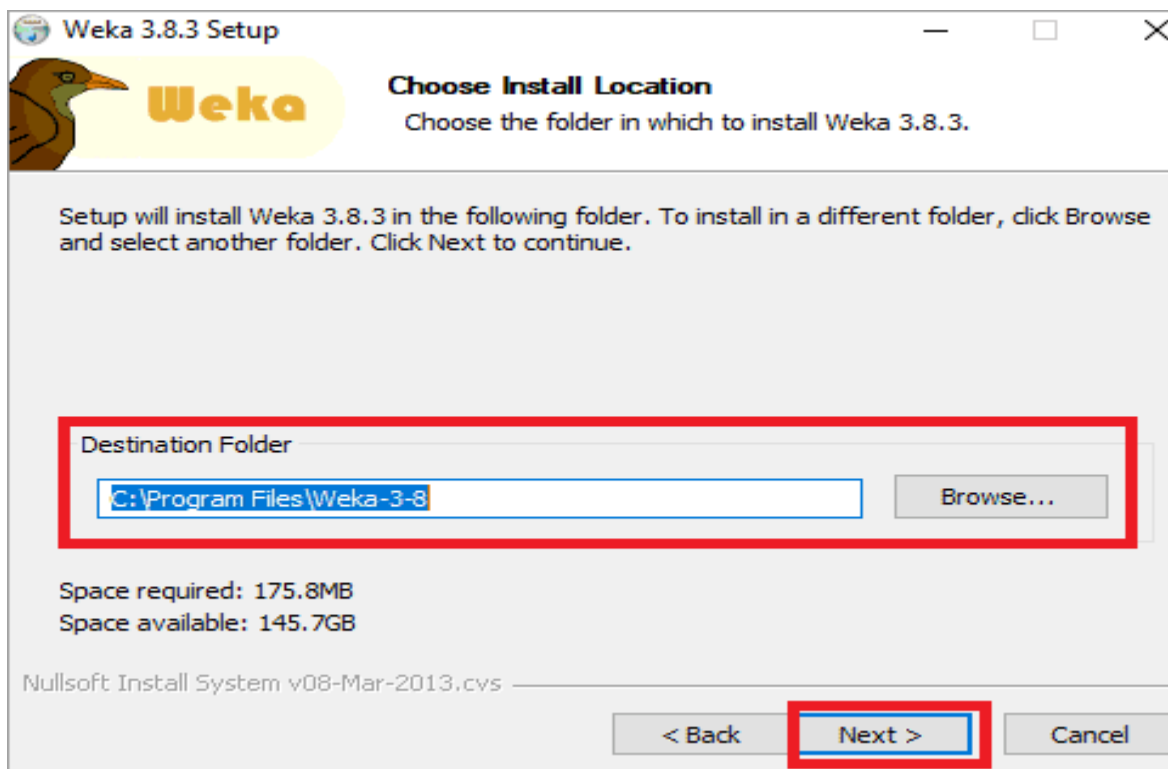
STEP 3 : The License Agreement terms will open. Read it thoroughly and click on “I Agree”.



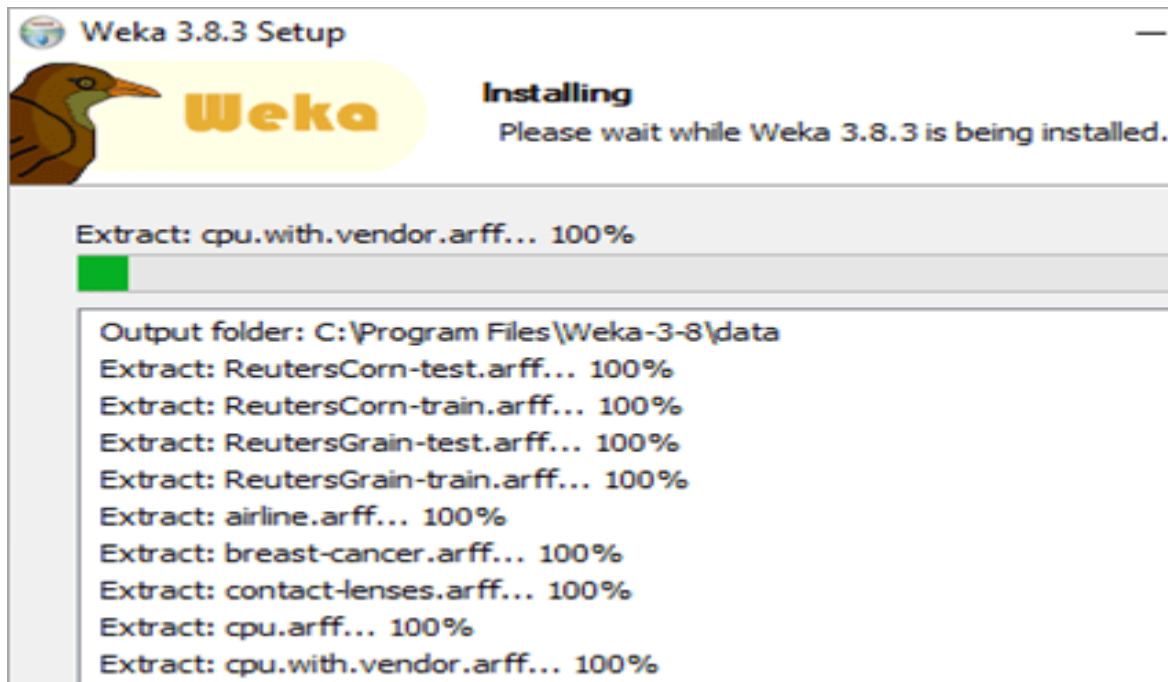
STEP 4 : According to your requirements, select the components to be installed. Full component installation is recommended. Click on Next.



STEP 5 : Select the destination folder and Click on Next.



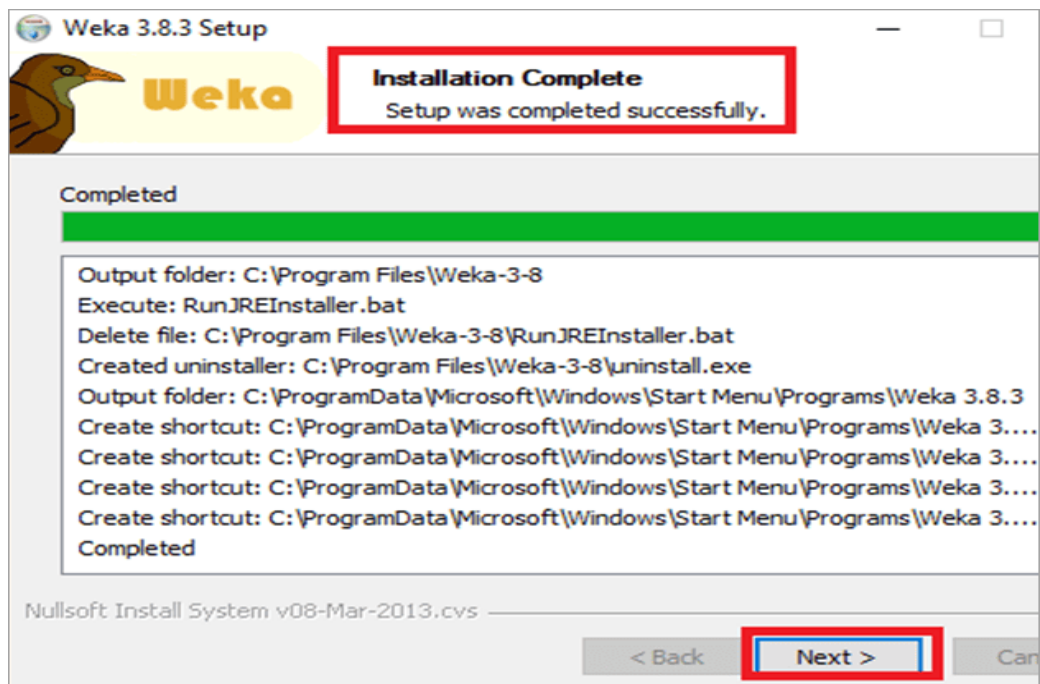
STEP 6 : Then, Installation will start.



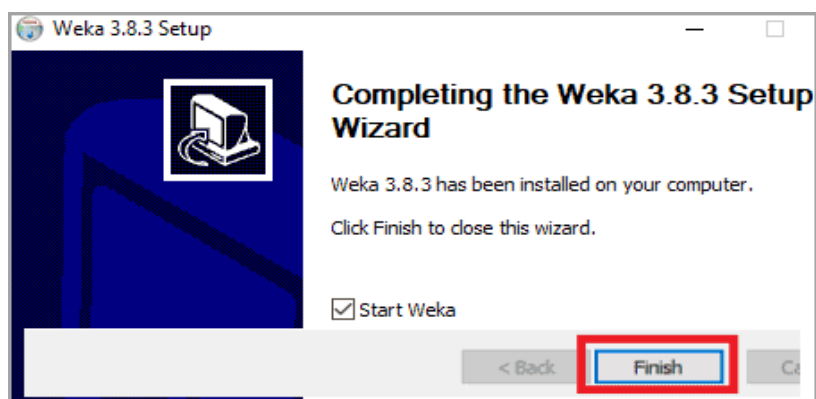
STEP 7 : If Java is not installed in the system, it will install Java first.



STEP 8 : After the installation is complete, the following window will appear. Click on Next.



STEP 9 : Select the Start Weka checkbox. Click on Finish.



STEP 10 : WEKA Tool and Explorer window opens.



STEP 11 : The WEKA manual can be downloaded from [here](#).

Graphical User Interface Of WEKA

The GUI of WEKA gives five options: Explorer, Experimenter, Knowledge flow, Workbench, and Simple CLI. Let us understand each of these individually.

1) Simple CLI

```
java <classname> <args>
    Lists the capabilities of the specified class.
    If the class is a weka.core.OptionHandler then
    trailing options after the classname will be
    set as well.

kill
    Kills the running job, if any.

script <script_file>
    Executes commands from a script file.

set [name=value]
    Sets a variable.
    If no key=value pair is given all current variables are 1

unset name
    Removes a variable.

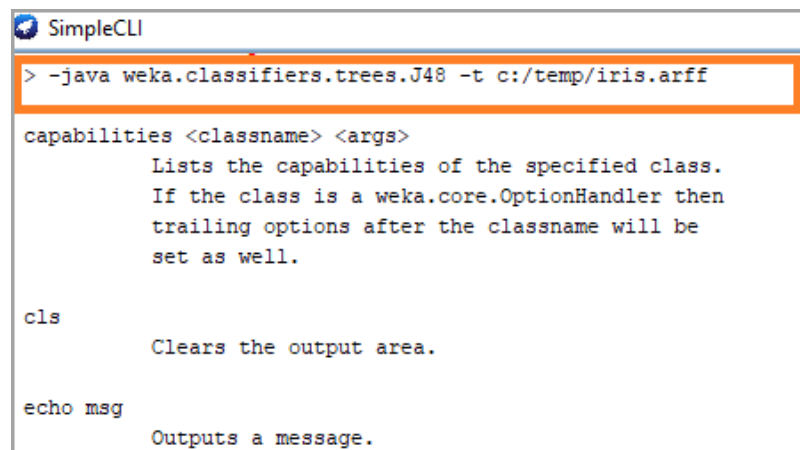
Notes:
- Variables can be used anywhere using '${<name>}' with '<name>'
  being the name of the variable.
- Environment variables can be used with '${env.<name>}',
  e.g., '${env.PATH}' to retrieve the PATH variable.
```

help

Simple CLI is Weka Shell with command line and output. With “help”, the overview of all the commands can be seen. Simple CLI offers access to all classes such as classifiers, clusters, filters, etc.

Some of the simple CLI commands are:

- **Break:** To stop the current thread
- **Exit:** Exit the CLI
- **Help[<command>]** : Outputs the help for the specified command
- **-java weka.classifiers.trees.J48 -t c:/temp/iris.arff** : To invoke a WEKA class, prefix it with Java. This command will direct WEKA to load the class and execute it with the given parameters. In this command, J48 classifier is invoked on the IRIS dataset.



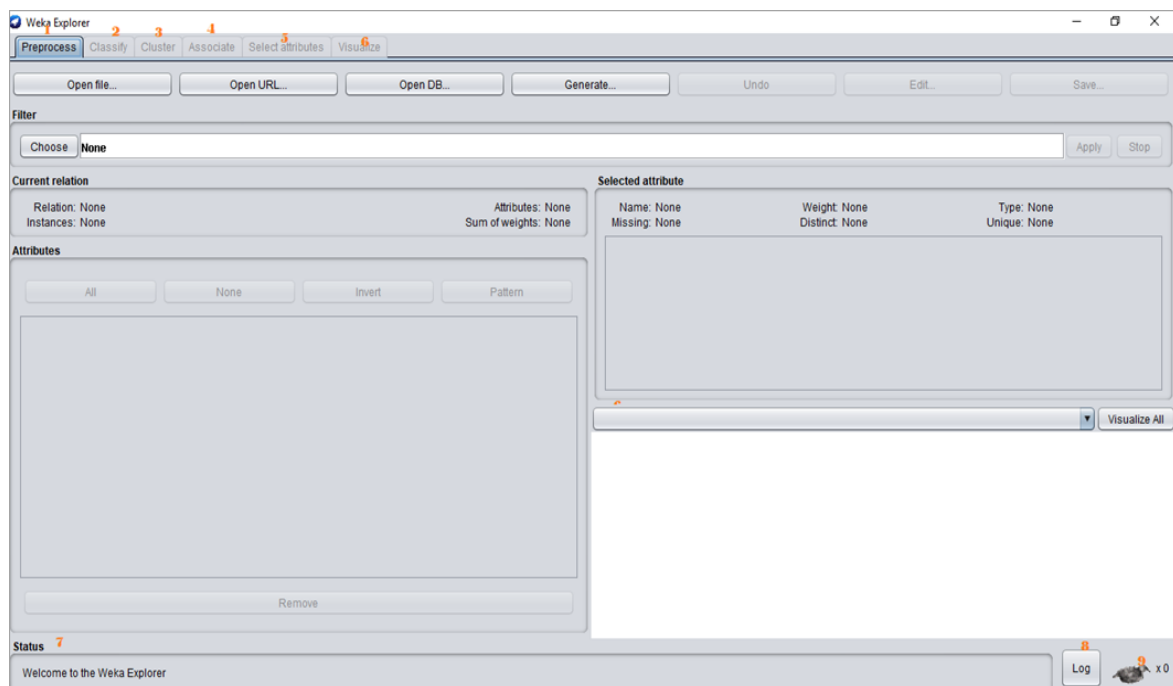
```
SimpleCLI
> -java weka.classifiers.trees.J48 -t c:/temp/iris.arff

capabilities <classname> <args>
    Lists the capabilities of the specified class.
    If the class is a weka.core.OptionHandler then
    trailing options after the classname will be
    set as well.

cls
    Clears the output area.

echo msg
    Outputs a message.
```

2) Explorer



The WEKA Explorer windows show different tabs starting with preprocessing. Initially, the preprocess tab is active, as first the data set is preprocessed before applying algorithms to it and exploring the dataset.

RESULT

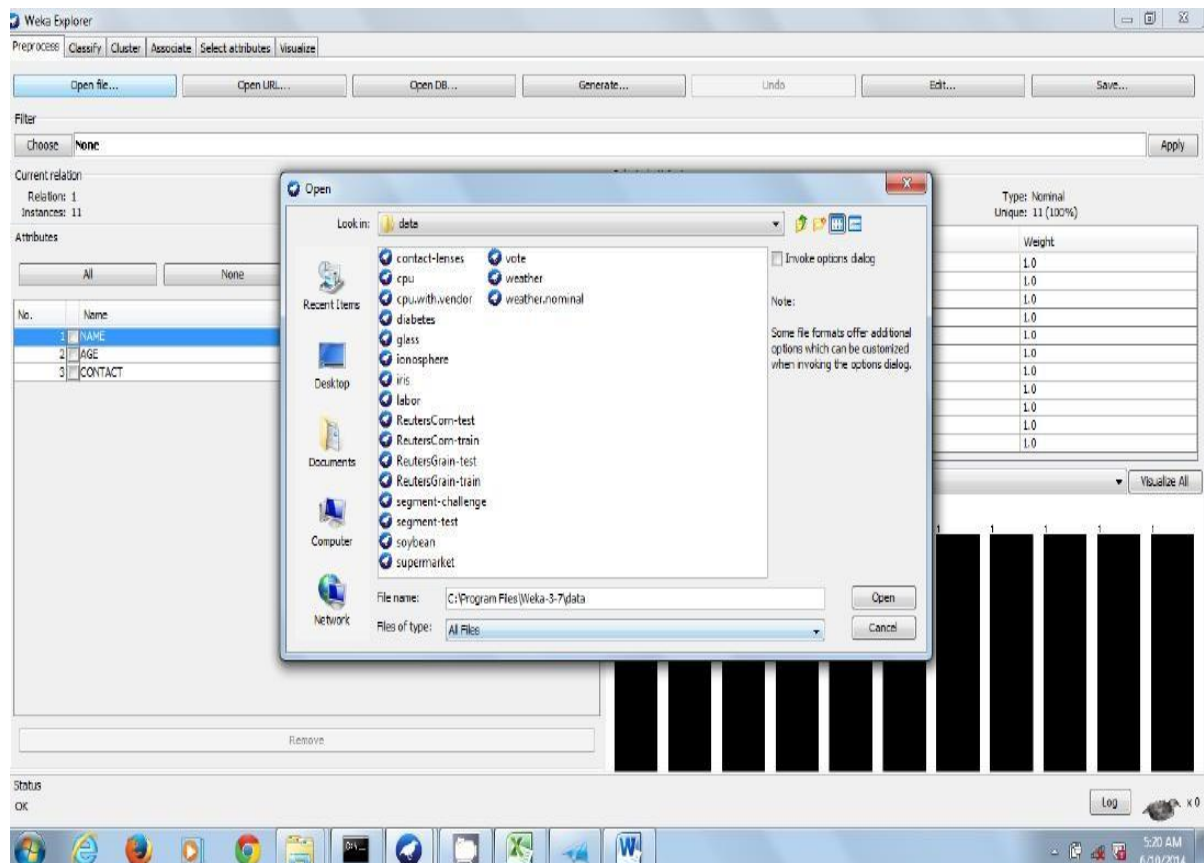
EX : 2 Creating new Arff File

To Create Data in .arff format

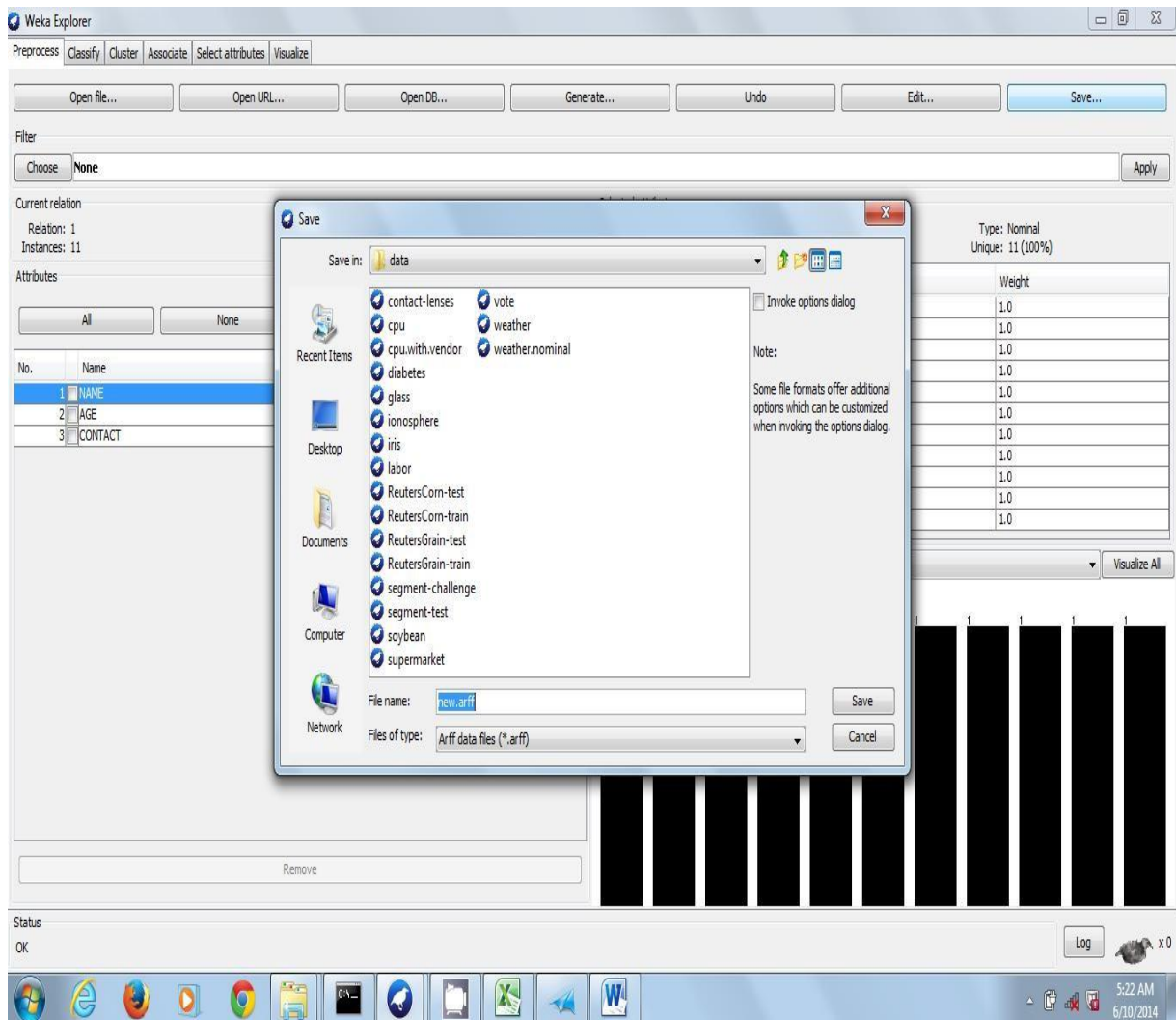
Input File: bank_data.csv

Procedure:

- Open MS Excel.
- Create a new worksheet with respective headings and data.
- Save the file with .csv extension
- Open Preprocessor tab in explorer
- Click open the file button and browse the file to open.
- Load the desired .csv file using open File tab.



Click **SAVE** shown... dialog box opens save with extension as **.arff**



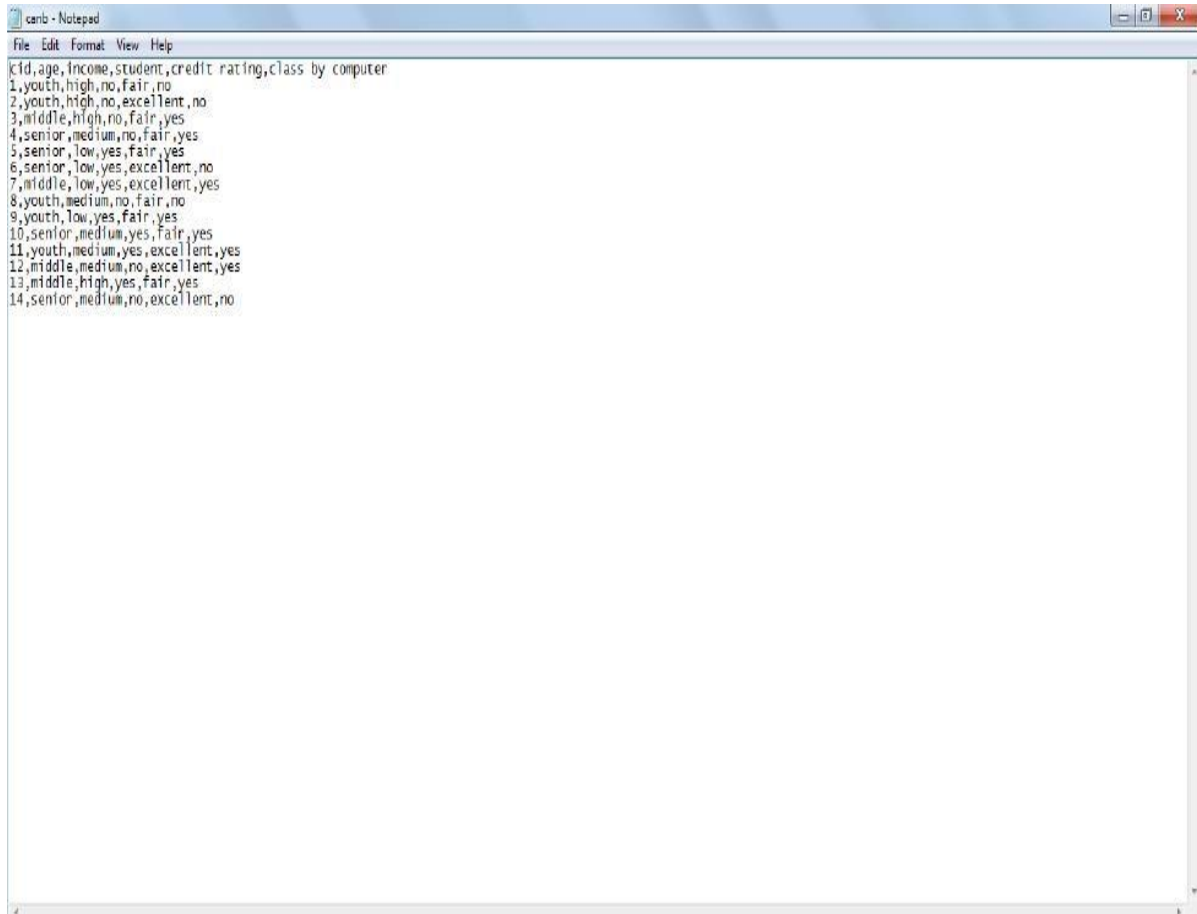
2. Create data in .csv format and store it in .arff format

Input: bank_data

Procedure:

- Open notepad and type the arff header information.
- Add data with respect to the given field separated by commas
- Save file as bank_data with .arff extension
- Open preprocessor tab of weka in the explorer
- Click open the button and browse the file bank_data.arff
- If the data has been entered without any errors then the file details will be available on the preprocessor screen.

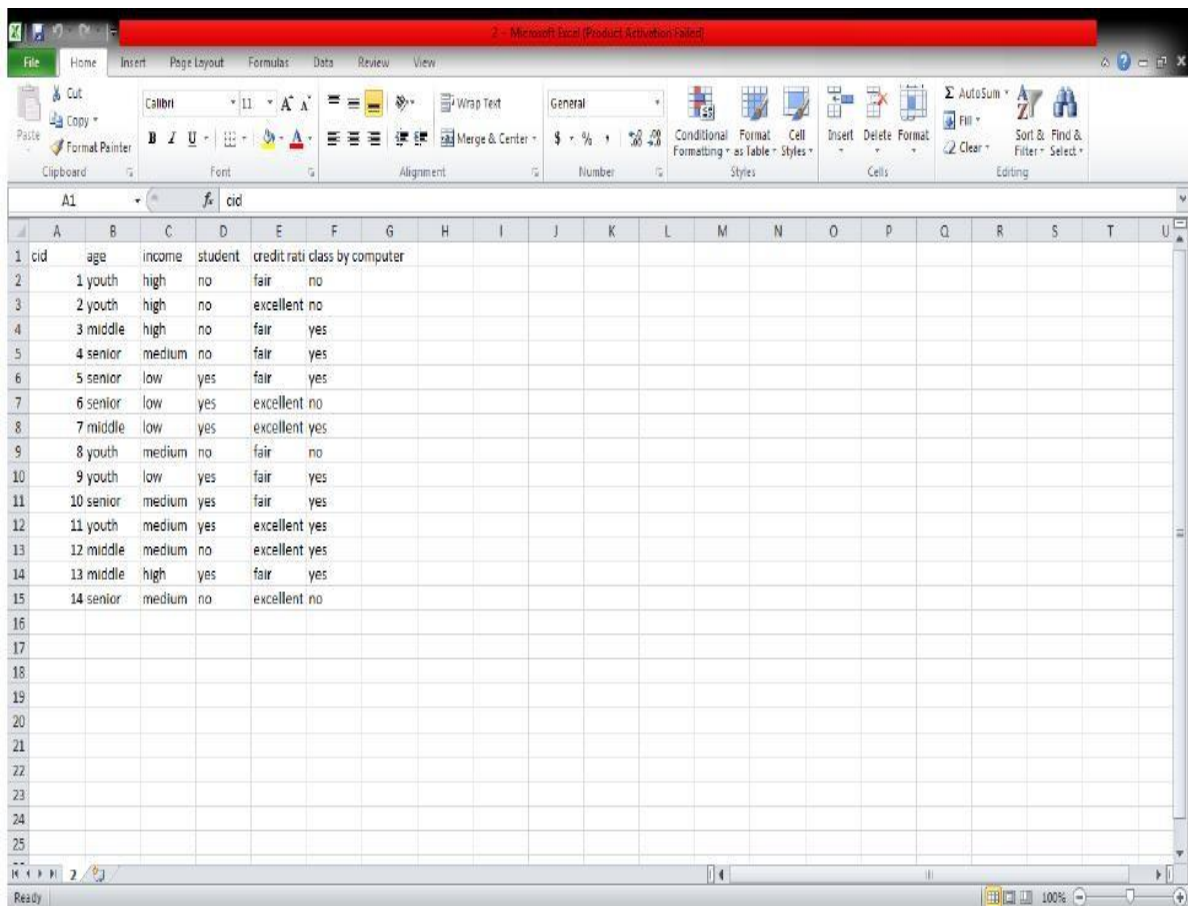
Insert data fields in note pad like shown and save with extension .csv choosing all files of type:



The screenshot shows a Notepad window titled "canb - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text content is a CSV file with the following data:

```
xid,age,income,student,credit rating,class by computer
1,youth,high,no,fair,no
2,youth,high,no,excellent,no
3,middle,high,no,fair,yes
4,senior,medium,no,fair,yes
5,senior,low,yes,fair,yes
6,senior,low,yes,excellent,no
7,middle,low,yes,excellent,yes
8,youth,medium,no,fair,no
9,youth,low,yes,fair,yes
10,senior,medium,yes,fair,yes
11,youth,medium,yes,excellent,yes
12,middle,medium,no,excellent,yes
13,middle,high,yes,fair,yes
14,senior,medium,no,excellent,no
```

A csv file looks like



The screenshot shows a Microsoft Excel spreadsheet with a table of data. The table has 5 columns and 14 rows of data. The columns are labeled 'cid', 'age', 'income', 'student', and 'credit rating'. The data is as follows:

	cid	age	income	student	credit rating
1	cid	age	income	student	credit rating
2	1	youth	high	no	fair
3	2	youth	high	no	excellent
4	3	middle	high	no	fair
5	4	senior	medium	no	fair
6	5	senior	low	yes	fair
7	6	senior	low	yes	excellent
8	7	middle	low	yes	excellent
9	8	youth	medium	no	fair
10	9	youth	low	yes	fair
11	10	senior	medium	yes	fair
12	11	youth	medium	yes	excellent
13	12	middle	medium	no	excellent
14	13	middle	high	yes	fair
15	14	senior	medium	no	excellent

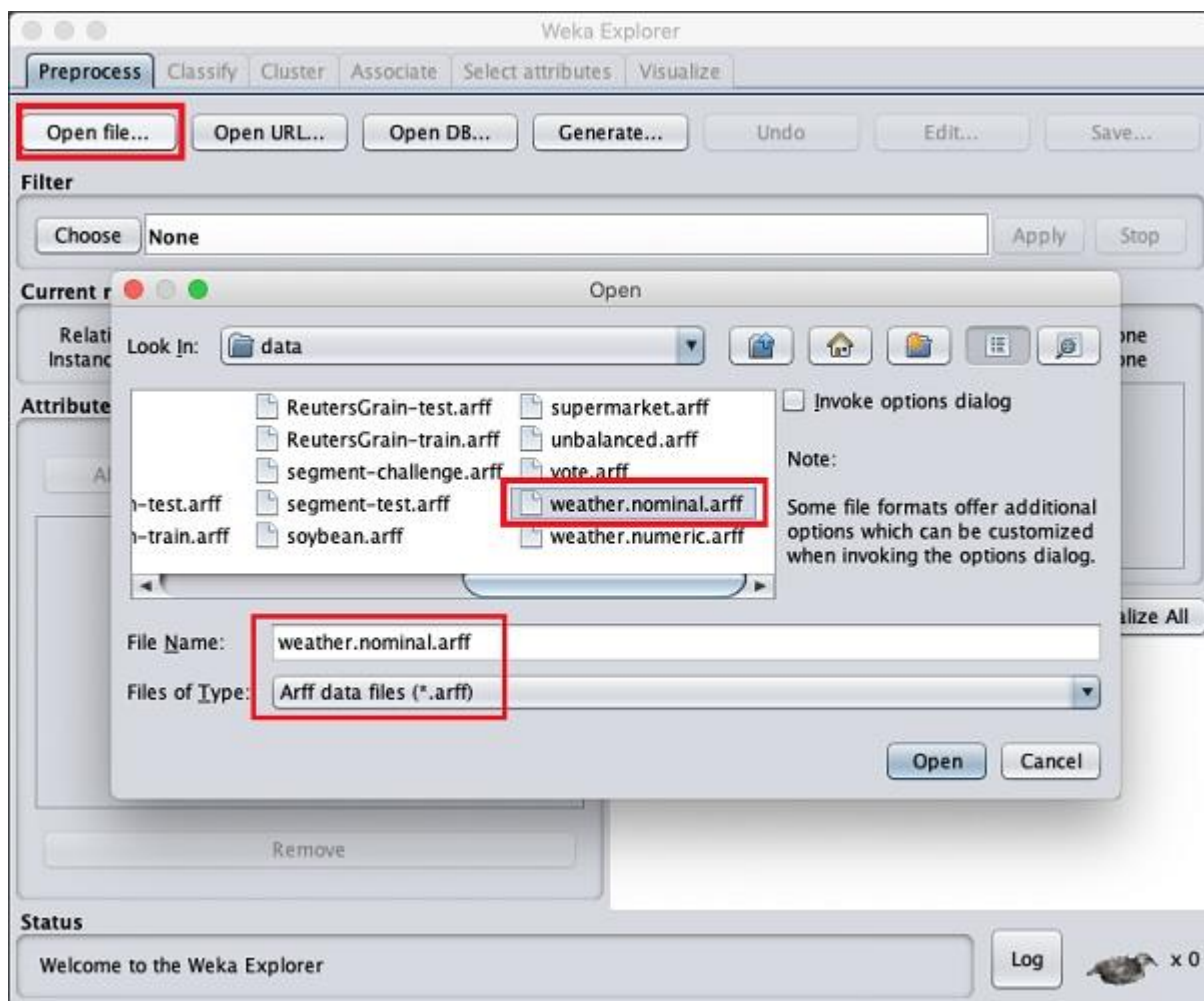
RESULT: Data are successfully created and uploaded.

EX : 3 Pre-Processes Techniques on Data Set and Pre-process a given dataset based on Handling Missing Values

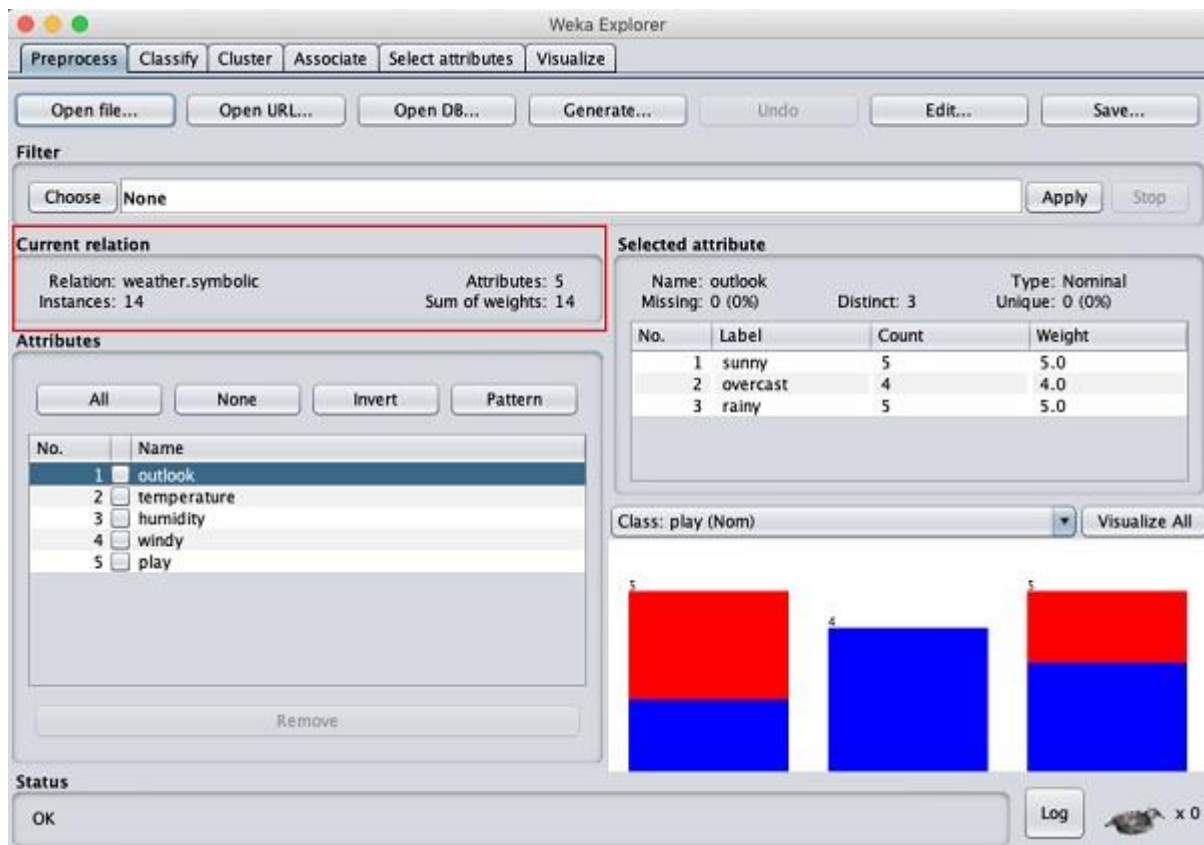
The data that is collected from the field contains many unwanted things that leads to wrong analysis. For example, the data may contain null fields, it may contain columns that are irrelevant to the current analysis, and so on. Thus, the data must be preprocessed to meet the requirements of the type of analysis you are seeking. This is the done in the preprocessing module.

To demonstrate the available features in preprocessing, we will use the **Weather** database that is provided in the installation.

Using the **Open file ...** option under the **Preprocess** tag select the **weather-nominal.arff** file.



When you open the file, your screen looks like as shown here –



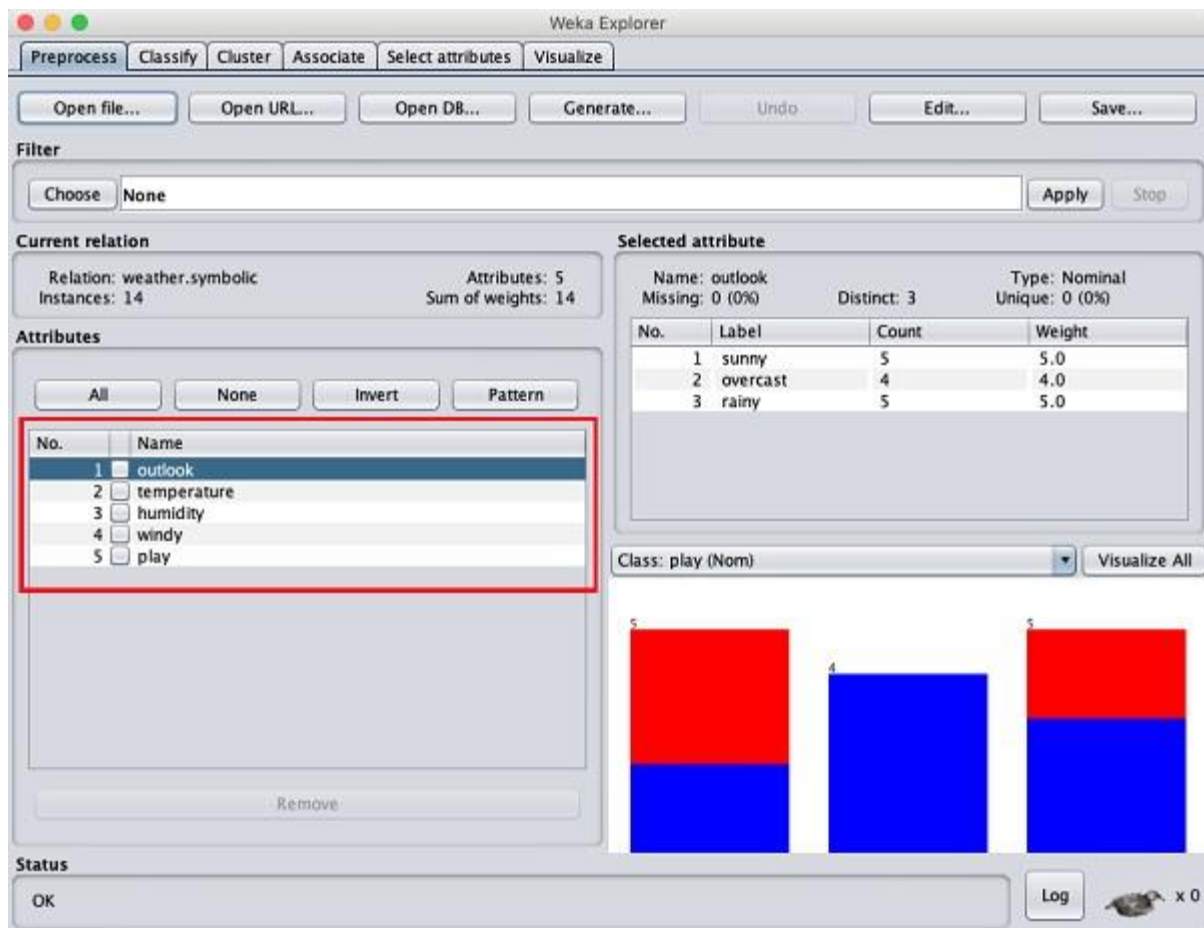
This screen tells us several things about the loaded data, which are discussed further in this chapter.

Understanding Data

Let us first look at the highlighted **Current relation** sub window. It shows the name of the database that is currently loaded. You can infer two points from this sub window –

- There are 14 instances - the number of rows in the table.
- The table contains 5 attributes - the fields, which are discussed in the upcoming sections.

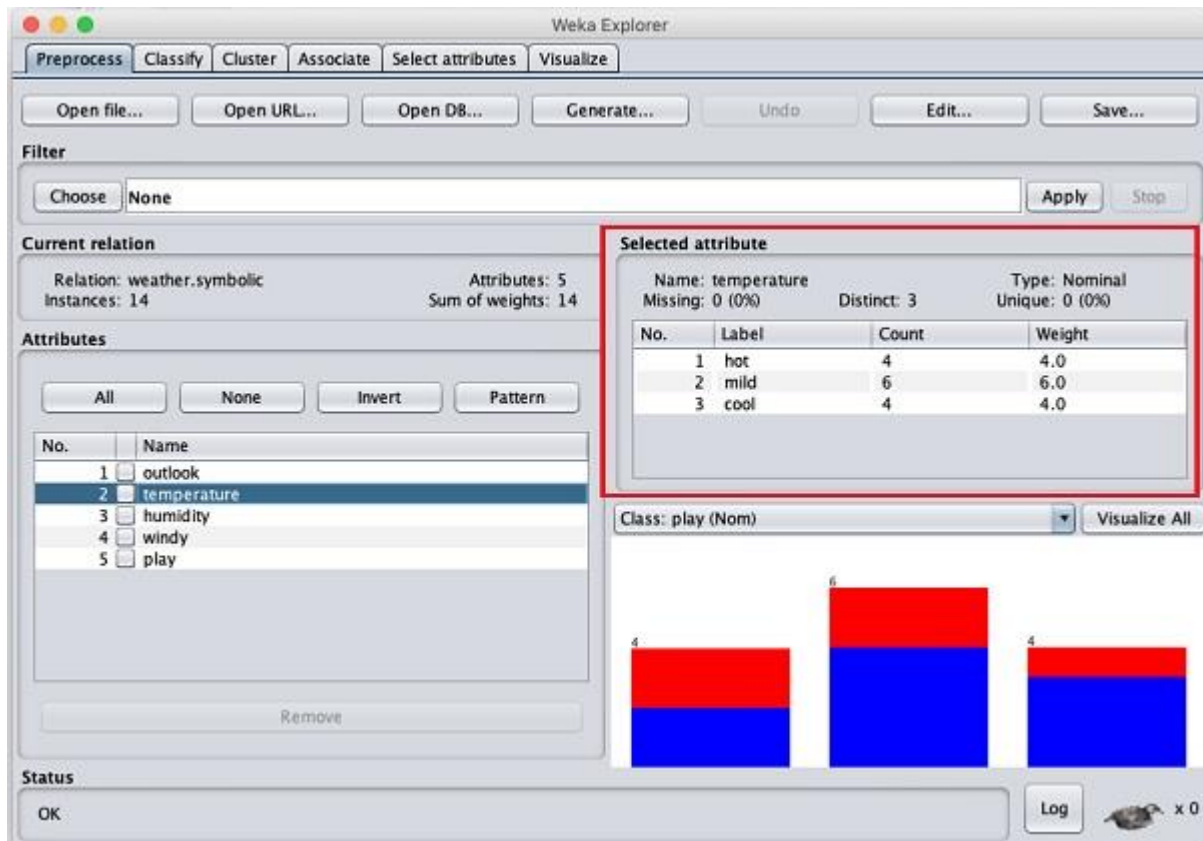
On the left side, notice the **Attributes** sub window that displays the various fields in the database.



The **weather** database contains five fields - outlook, temperature, humidity, windy and play.

When you select an attribute from this list by clicking on it, further details on the attribute itself are displayed on the right hand side.

Let us select the temperature attribute first. When you click on it, you would see the following screen –

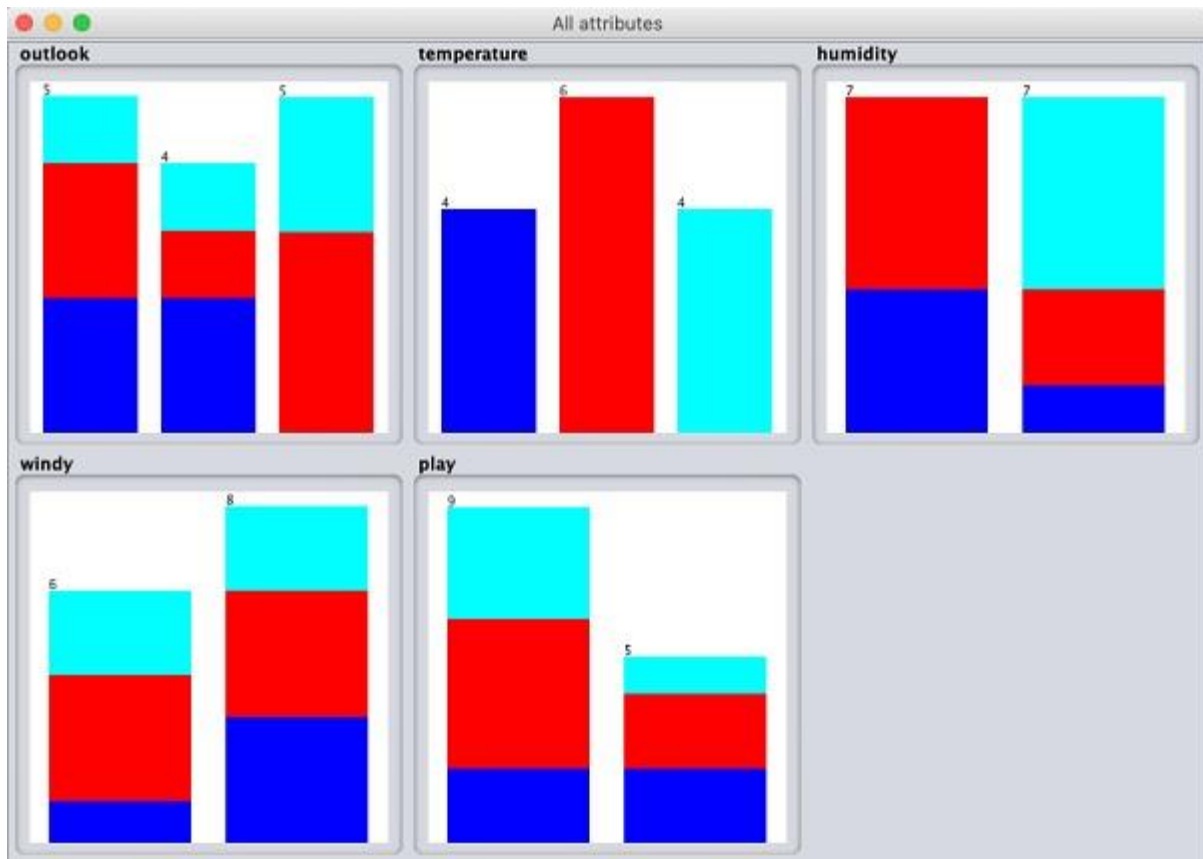


In the **Selected Attribute** subwindow, you can observe the following –

- The name and the type of the attribute are displayed.
- The type for the **temperature** attribute is **Nominal**.
- The number of **Missing** values is zero.
- There are three distinct values with no unique value.
- The table underneath this information shows the nominal values for this field as hot, mild and cold.
- It also shows the count and weight in terms of a percentage for each nominal value.

At the bottom of the window, you see the visual representation of the **class** values.

If you click on the **Visualize All** button, you will be able to see all features in one single window as shown here –



Removing Attributes

Many a time, the data that you want to use for model building comes with many irrelevant fields. For example, the customer database may contain his mobile number which is relevant in analysing his credit rating.

The figure shows a window titled "Attributes" with four buttons: "All", "None", "Invert", and "Pattern". Below these buttons is a table with 5 rows, each with a checkbox and a name. The "Remove" button is at the bottom.

No.		Name
1	<input checked="" type="checkbox"/>	outlook
2	<input type="checkbox"/>	temperature
3	<input checked="" type="checkbox"/>	humidity
4	<input type="checkbox"/>	windy
5	<input type="checkbox"/>	play

To remove Attribute/s select them and click on the **Remove** button at the bottom.

The selected attributes would be removed from the database. After you fully preprocess the data, you can save it for model building.

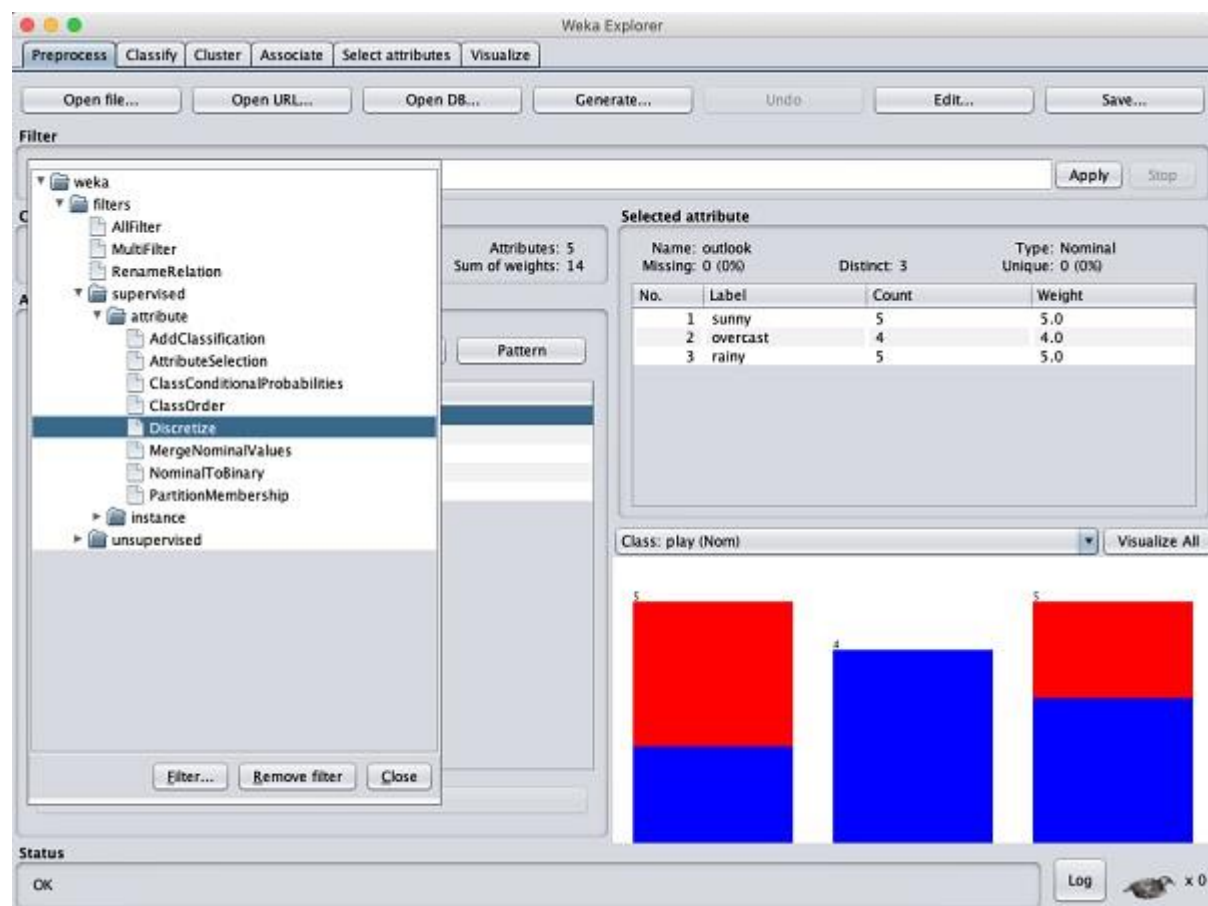
Next, you will learn to preprocess the data by applying filters on this data.

Applying Filters

Some of the machine learning techniques such as association rule mining requires categorical data. To illustrate the use of filters, we will use **weather-numeric.arff** database that contains two **numeric** attributes - **temperature** and **humidity**.

We will convert these to **nominal** by applying a filter on our raw data. Click on the **Choose** button in the **Filter** subwindow and select the following filter –

weka→filters→supervised→attribute→Discretize



Click on the **Apply** button and examine the **temperature** and/or **humidity** attribute. You will notice that these have changed from numeric to nominal types.

Name: temperature		Type: Nominal	
Missing: 0 (0%)		Distinct: 1	
		Unique: 0 (0%)	
No.	Label	Count	Weight
1	'All'	14	14.0

Let us look into another filter now. Suppose you want to select the best attributes for deciding the **play**. Select and apply the following filter –

weka→filters→supervised→attribute→AttributeSelection

You will notice that it removes the temperature and humidity attributes from the database.

The screenshot shows the Weka Explorer window with the 'AttributeSelection' filter applied. The 'Current relation' is 'weather.symbolic-weka.filters.superv...' with 14 instances and 3 attributes. The 'Attributes' list on the left shows 'outlook', 'humidity', and 'play', with 'humidity' and 'play' selected. The 'Selected attribute' table on the right shows the resulting data with 3 attributes: outlook, humidity, and play.

No.	Label	Count	Weight
1	sunny	5	5.0
2	overcast	4	4.0
3	rainy	5	5.0

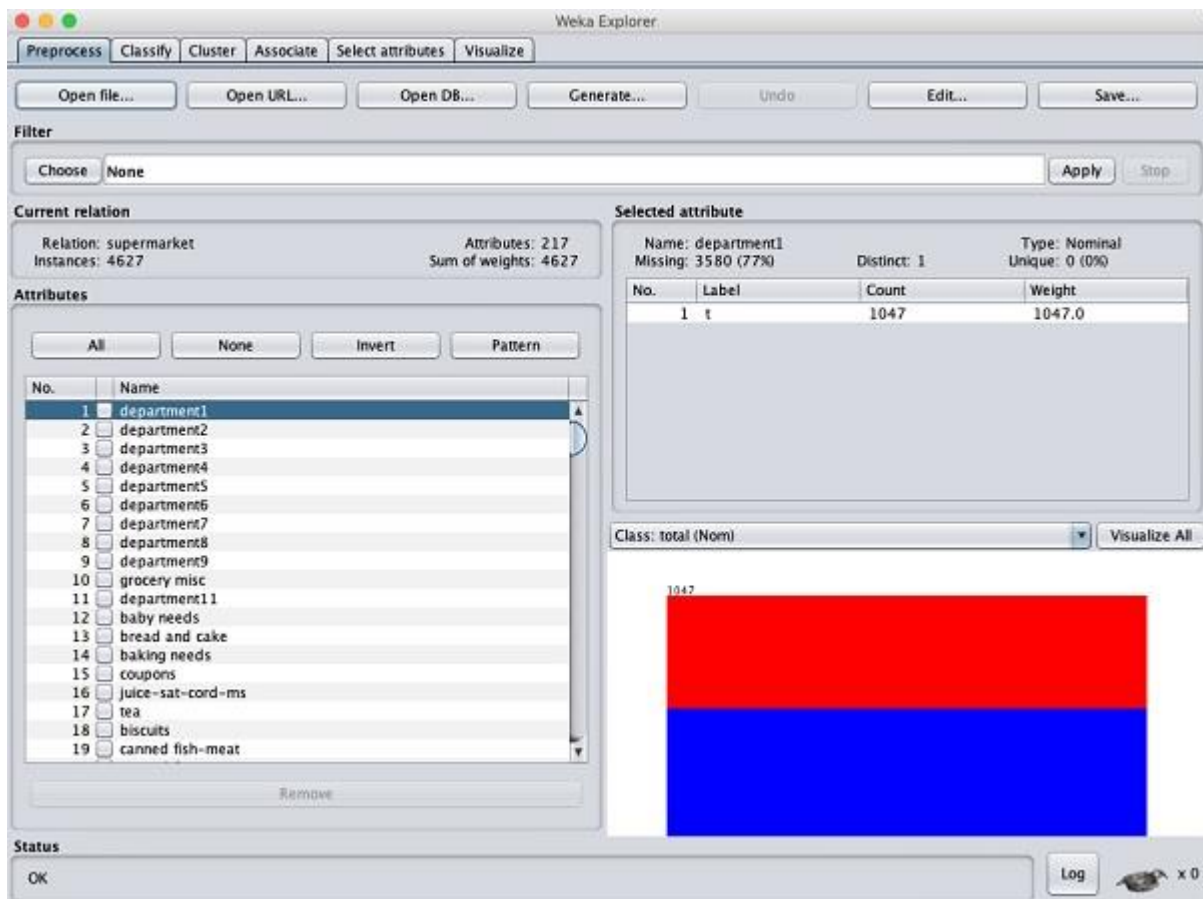
After you are satisfied with the preprocessing of your data, save the data by clicking the Save ... button. You will use this saved file for model building.

EX : 4 Generate Association Rules using the Apriori Algorithm

The **Apriori** algorithm is one such algorithm in ML that finds out the probable associations and creates association rules. WEKA provides the implementation of the Apriori algorithm. You can define the minimum support and an acceptable confidence level while computing these rules. You will apply the **Apriori** algorithm to the **supermarket** data provided in the WEKA installation.

Loading Data

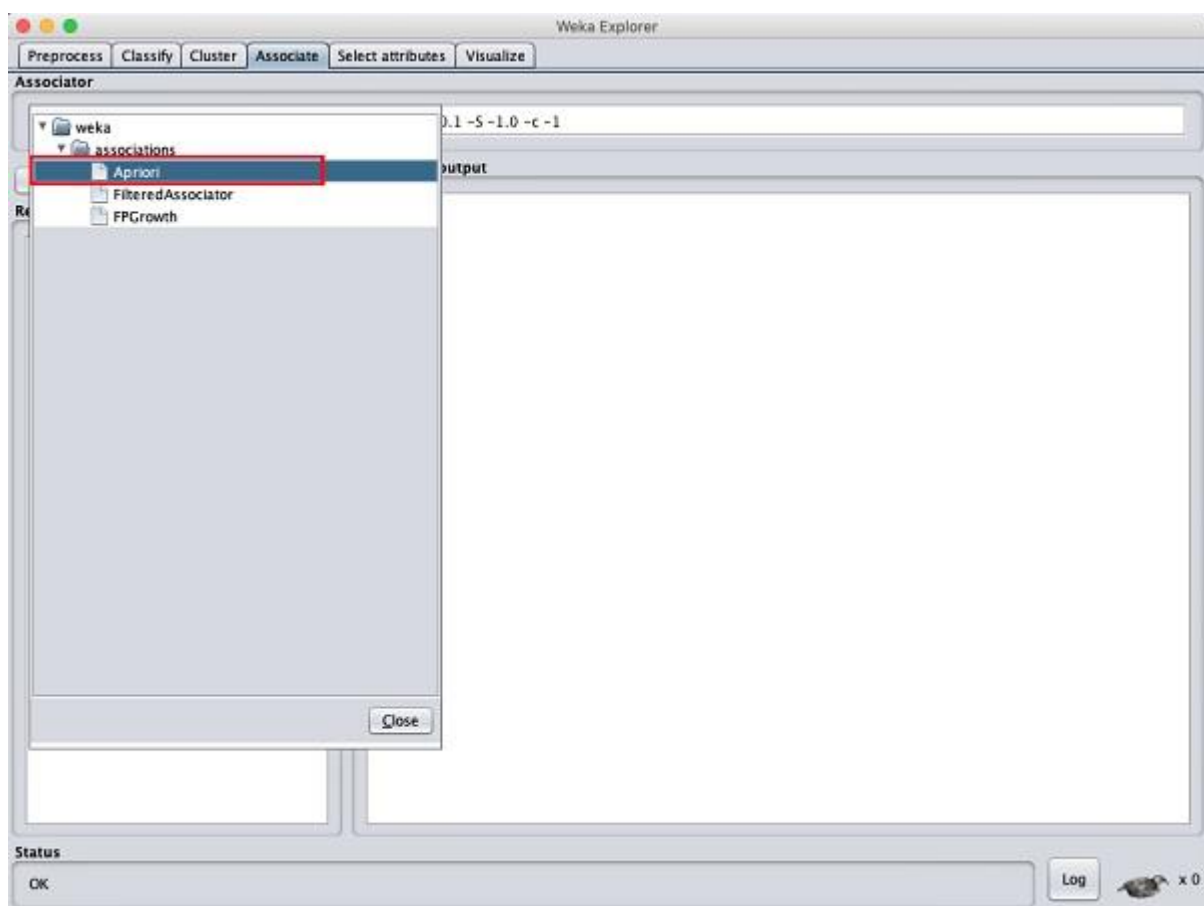
In the WEKA explorer, open the **Preprocess** tab, click on the **Open file ...** button and select **supermarket.arff** database from the installation folder. After the data is loaded you will see the following screen –



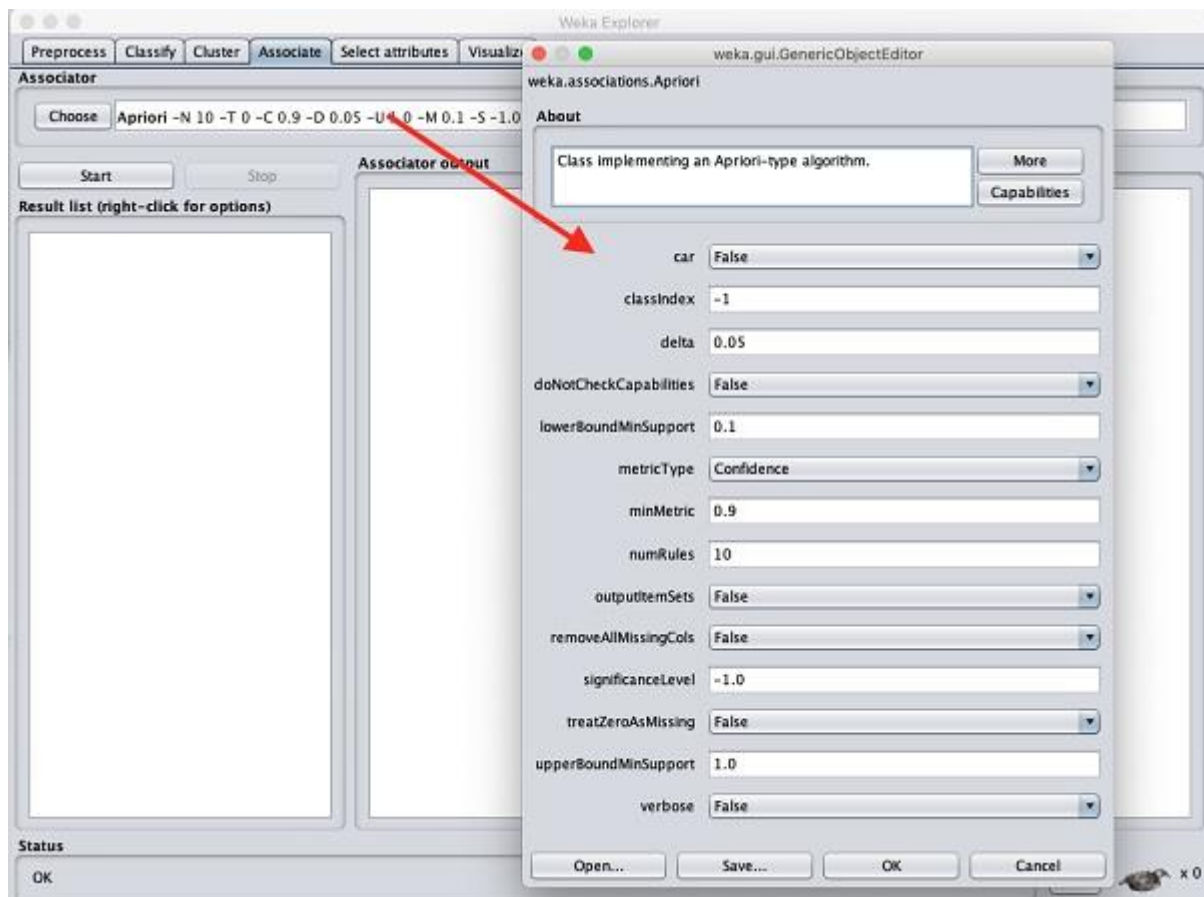
The database contains 4627 instances and 217 attributes. You can easily understand how difficult it would be to detect the association between such a large number of attributes. Fortunately, this task is automated with the help of Apriori algorithm.

Associator

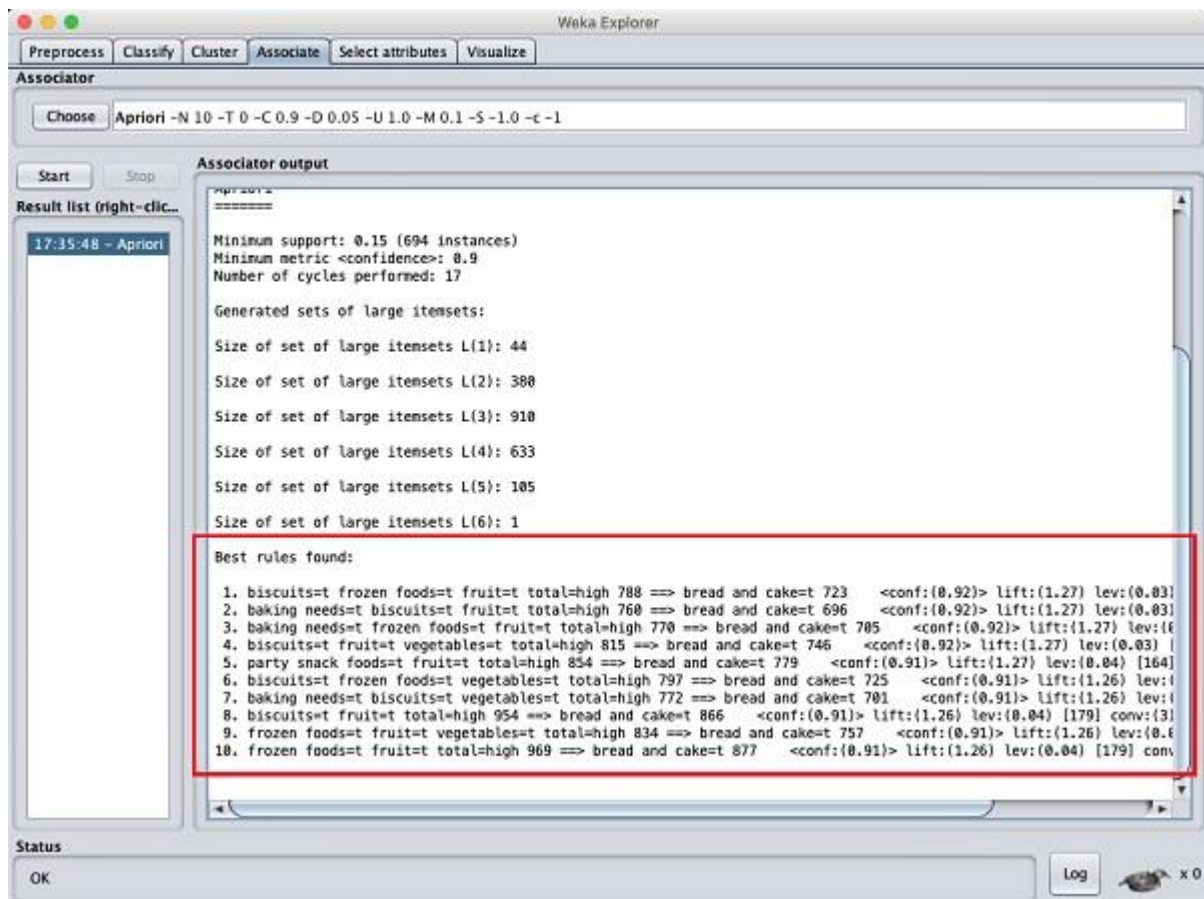
Click on the **Associate** TAB and click on the **Choose** button. Select the **Apriori** association as shown in the screenshot –



To set the parameters for the Apriori algorithm, click on its name, a window will pop up as shown below that allows you to set the parameters –



After you set the parameters, click the **Start** button. After a while you will see the results as shown in the screenshot below –



At the bottom, you will find the detected best rules of associations. This will help the supermarket in stocking their products in appropriate shelves.

RESULT :

EX : 5 Generating association rules using fp-growth algorithm

Procedure:

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 FP-Growth 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.

Data set:

Shopping.arff

@relation shopping

@attribute milk{yes,no}

@attribute bread{yes,no}

@attribute honey{yes,no}

@attribute ghee{yes,no}

@attribute jam{yes,no}

@data

yes,yes,no,no,yes

no,yes,no,yes,no

no,yes,yes,no,no

yes,yes,no,yes,no

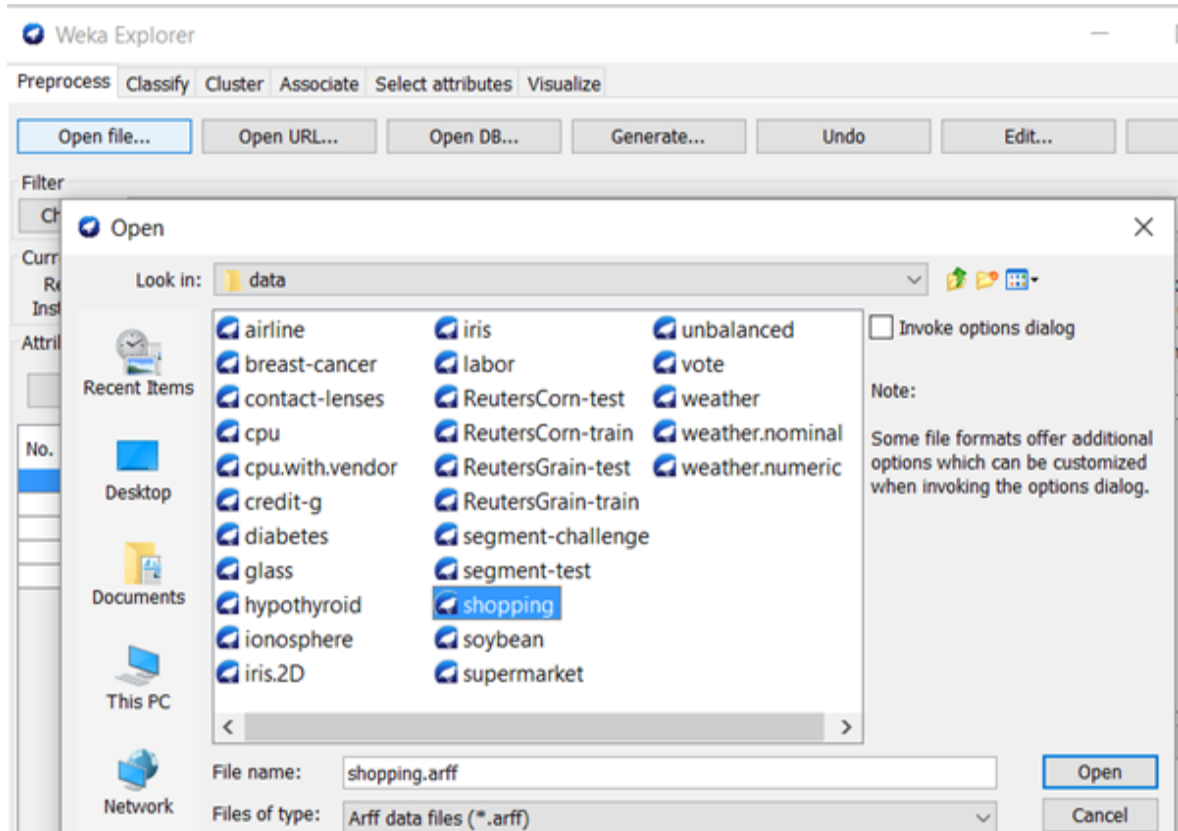
yes,no,yes,no,no

no,yes,yes,no,no

yes,no,yes,no,no

yes,yes,yes,no,yes

yes,yes,yes,no,no

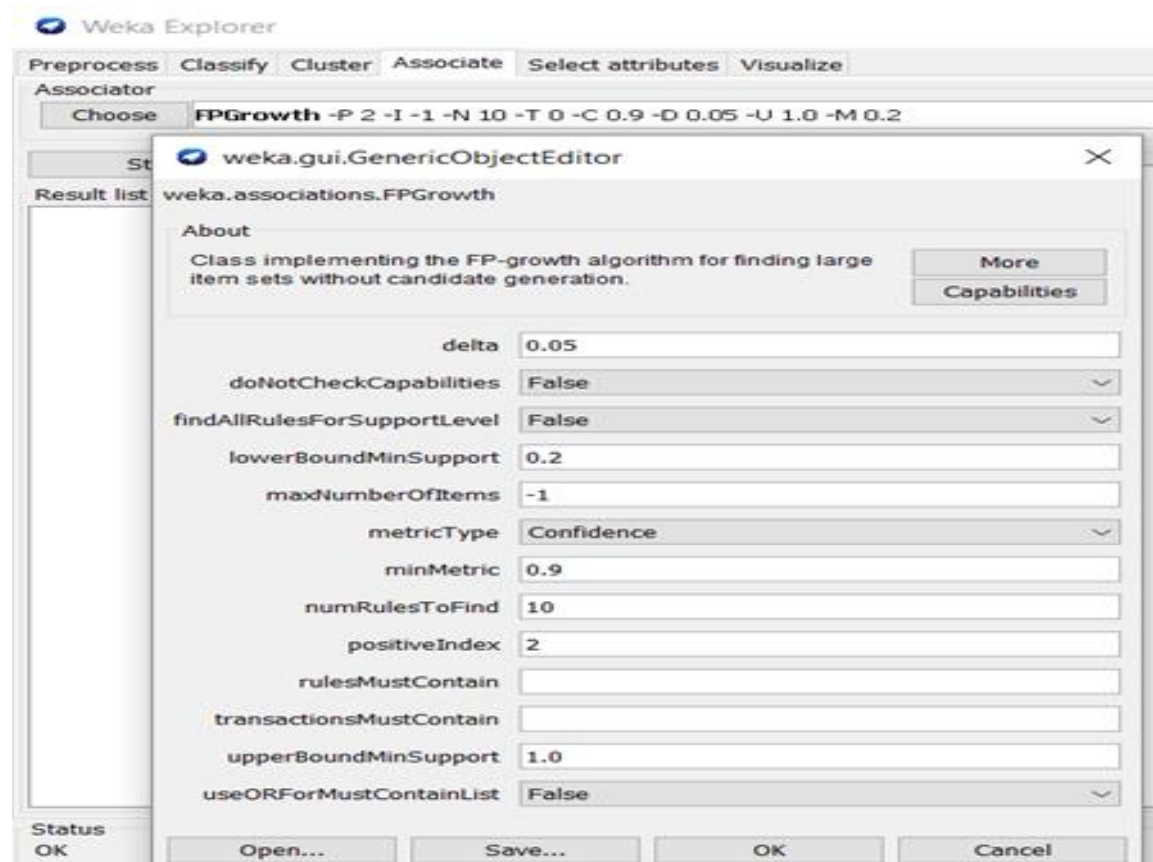
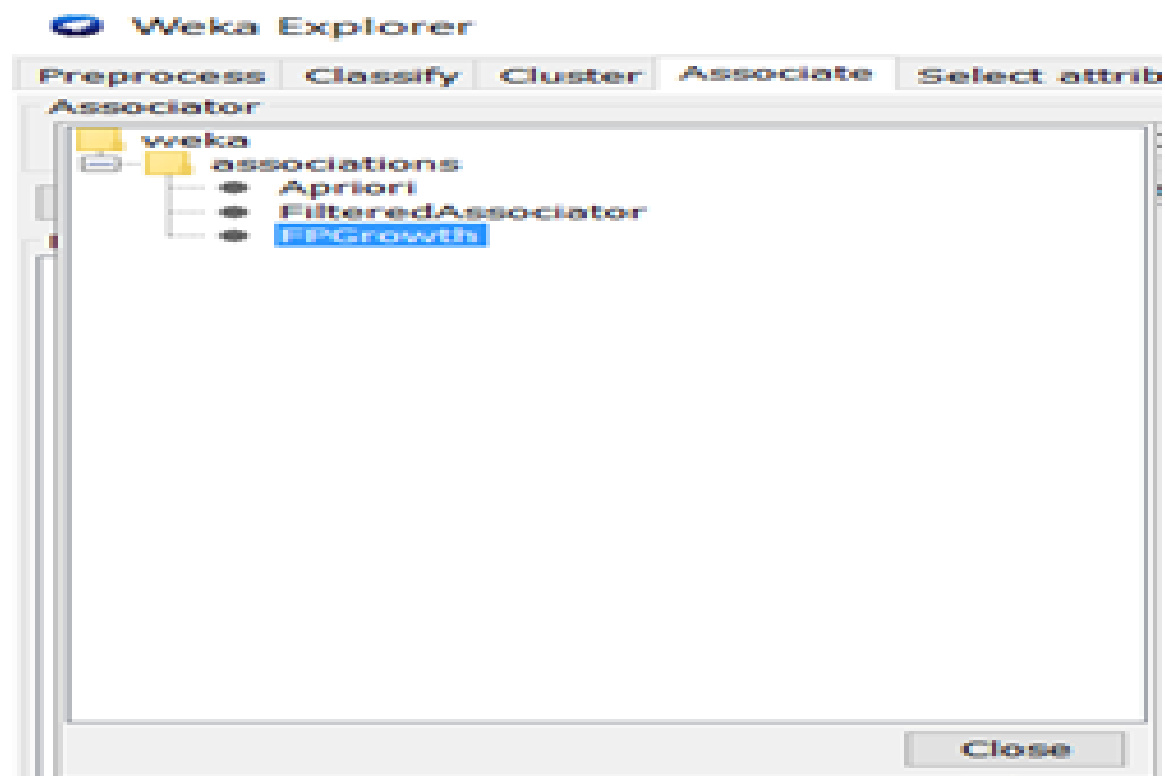


Viewer

Relation: shopping

No.	1: milk Nominal	2: bread Nominal	3: honey Nominal	4: ghee Nominal	5: jam Nominal
1	yes	yes	no	no	yes
2	no	yes	no	yes	no
3	no	yes	yes	no	no
4	yes	yes	no	yes	no
5	yes	no	yes	no	no
6	no	yes	yes	no	no
7	yes	no	yes	no	no
8	yes	yes	yes	no	yes
9	yes	yes	yes	no	no

Undo OK Cancel



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Associator

Choose **FPGrowth** -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.2

Start Stop

Associator output

Result list (right-click to open)

19:57:09 - FPGrowth

```
=== Run information ===
Scheme:      weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.2
Relation:     shopping
Instances:    9
Attributes:   5
              milk
              bread
              honey
              ghee
              jam
=== Associator model (full training set) ===

FPGrowth found 7 rules (displaying top 7)

1. [milk=no]: 3 ==> [jam=no]: 3 <conf:(1)> lift:(1.29) lev:(0.07) conv:(0.67)
2. [bread=no]: 2 ==> [jam=no]: 2 <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
3. [bread=no]: 2 ==> [ghee=no]: 2 <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
4. [ghee=no, milk=no]: 2 ==> [jam=no]: 2 <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
5. [bread=no]: 2 ==> [jam=no, ghee=no]: 2 <conf:(1)> lift:(1.8) lev:(0.1) conv:(0.89)
6. [jam=no, bread=no]: 2 ==> [ghee=no]: 2 <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
7. [ghee=no, bread=no]: 2 ==> [jam=no]: 2 <conf:(1)> lift:(1.29) lev:(0.05) conv:(0.44)
```

Status
OK

Log x 0

EX : 6 Build a Decision Tree by using J48algorithm

J48 is a machine learning decision tree classification algorithm based on Iterative Dichotomiser 3. It is very helpful in examine the data categorically and continuously.

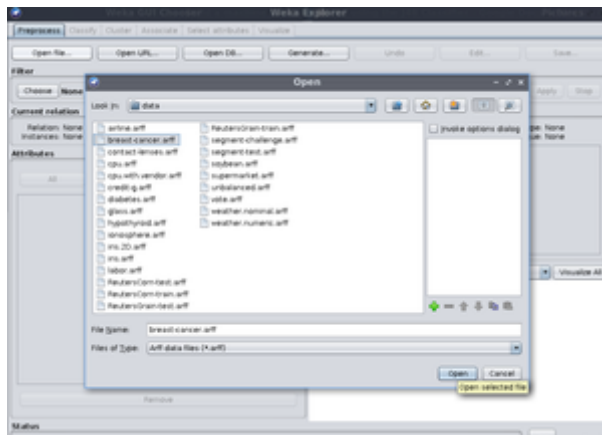
Steps to follow:

Step 1: Create a model using GUI

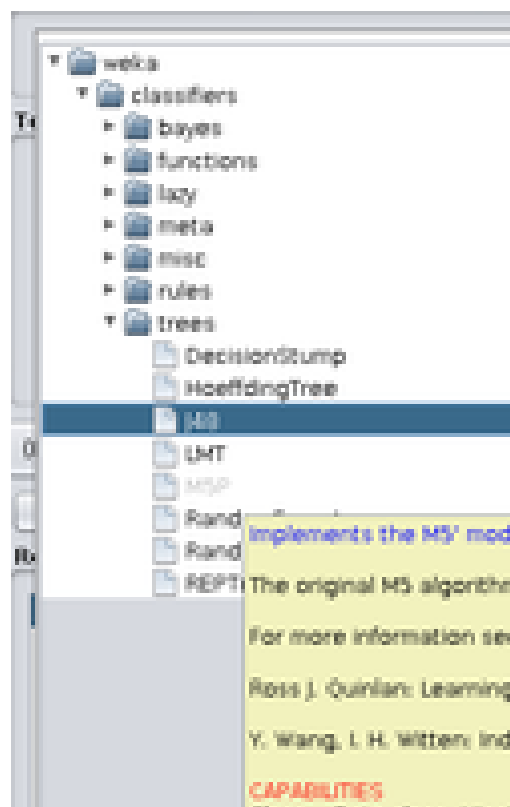
Step 2: After opening Weka click on the “Explorer” Tab



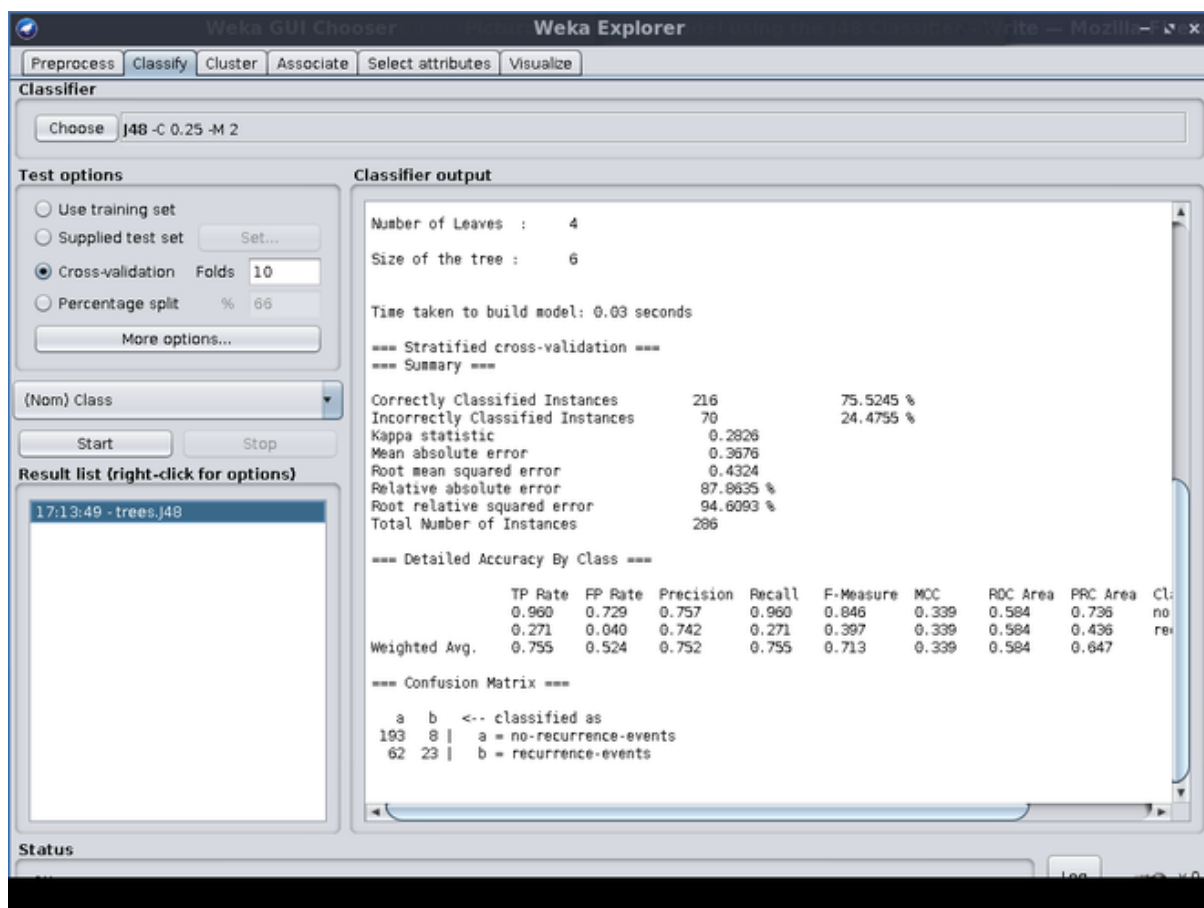
Step 3: In the “Preprocess” Tab Click on “Open File” and select the “breast-cancer.arff” file which will be located in the installation path, inside the data folder.



Step 4: In the “Classify” tab click on the choose button. Now under weka/classifiers/trees/ select J48



Step 5: Now one can click on the J48 Classifier selection and play around with it like changing batch size, confidence factor, etc. There under “Test Options” we’ll use the default cross-validation option as folds 10 and click on start.



RESULT :

EX : 7 Naïve bayes classification on a given data set



Preparing data for classification

We will use the same data set as in the previous example with weather features, **temperature** and **humidity**, and class **yes/no** for playing golf.

Data is stored in *arff* file format specific for WEKA software and looks like this:

```
@relation 'weather.symbolic-weka.filters.unsupervised.attribute.Remove-R1,4'
```

```
@attribute temperature {hot,mild,cool}
```

```
@attribute humidity {high,normal}
```

```
@attribute play {yes,no}
```

```
@data
```

```
hot,high,no
```

```
hot,high,no
```

```
hot,high,yes
```

```
mild,high,yes
```

```
cool,normal,yes
```

```
cool,normal,no
```

```
cool,normal,yes
```

```
mild,high,no
```

```
cool,normal,yes
```

```
mild,normal,yes
```

mild,normal,yes

mild,high,yes

hot,normal,yes

mild,high,no

Here we can see the attribute denominators: temperature, humidity, and play, followed by the data table. Using this data set, we will train the Naive Bayes model and then apply it to new data with temperature **cool** and humidity **high** to see to which class it will be assigned.

First of all, in WEKA explorer *Preprocess* tab, we need to open our ARFF data file:

The screenshot shows the Weka Explorer interface with the **Preprocess** tab selected. The **Current relation** is 'weather...' with 14 instances and 3 attributes. The **Selected attribute** is 'temperature', which is of type 'Nom...' with 0 missing values, 3 distinct values, and 0 unique values (0%).

The **Attributes** list shows three attributes: 'temperature', 'humidity', and 'play'. The 'temperature' attribute is selected.


The **Selected attribute** table shows the following data:

No.	Label	Count	Weight
1	hot	4	4.0
2	mild	6	6.0
3	cool	4	4.0

Below the table is a bar chart showing the distribution of the 'temperature' attribute. The bars are stacked with blue at the bottom and red at the top. The counts are 4 for 'hot', 6 for 'mild', and 4 for 'cool'.

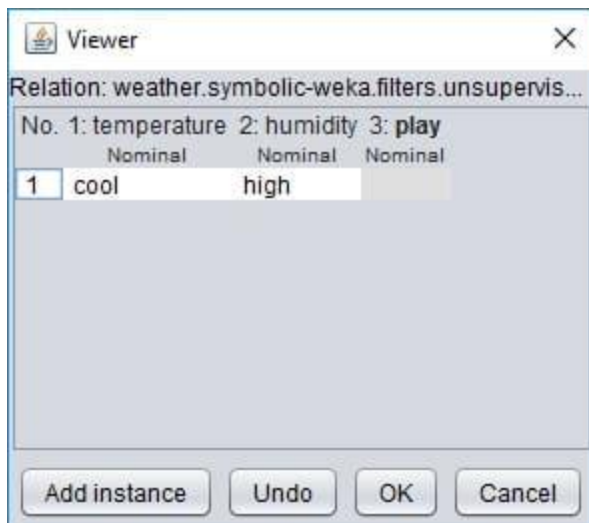
The **Status** bar at the bottom shows 'OK' and a 'Log' button.

Here we can see the basic statistics of attributes. If you click *the Edit* button, the new Viewer window with the data table will be loaded.



No.	1: temperature	2: humidity	3: play
	Nominal	Nominal	Nominal
1	hot	high	no
2	hot	high	no
3	hot	high	yes
4	mild	high	yes
5	cool	normal	yes
6	cool	normal	no
7	cool	normal	yes
8	mild	high	no
9	cool	normal	yes
10	mild	normal	yes
11	mild	normal	yes
12	mild	high	yes
13	hot	normal	yes
14	mild	high	no

You can edit data as you like in the viewer, and then you can permanently save new data set with the Save button in explorer. We will do so when we create a test set with cool and high parameter values. For this, we delete all lines of data except the first one and edit values to look like this:



No.	1: temperature	2: humidity	3: play
	Nominal	Nominal	Nominal
1	cool	high	

Select nothing on play attribute because we don't know it yet.

Click OK and then Save data as a separate file. The file should look like this:

```
@relation 'weather.symbolic-weka.filters.unsupervised.attribute.Remove-R1,4'
```

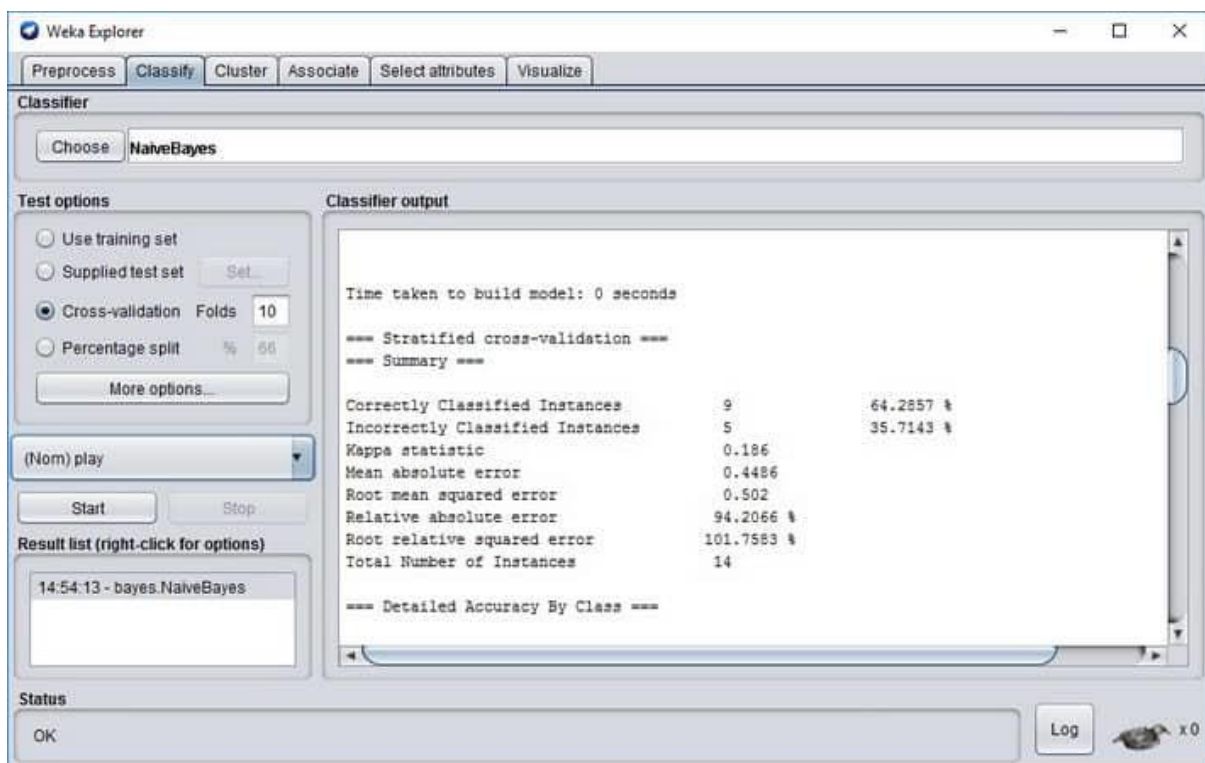
@attribute temperature {hot,mild,cool}
@attribute humidity {high,normal}
@attribute play {yes,no}
@data
cool,high,?

The question “?” mark is a standard way of representing the missing values in WEKA.

Building a Naive Bayes model

Now that we have data prepared, we can proceed with building the model. Load complete weather data set again in explorer and then go to *Classify* tab.

Here you need to press *the* Choose Classifier button, and from the tree menu, select NaiveBayes. Be sure that the Play attribute is selected as a class selector, and then press **the Start** button to build a model.



Model outputs some information on how accurate it classifies and other parameters.

Correctly Classified Instances 9 64.2857 %

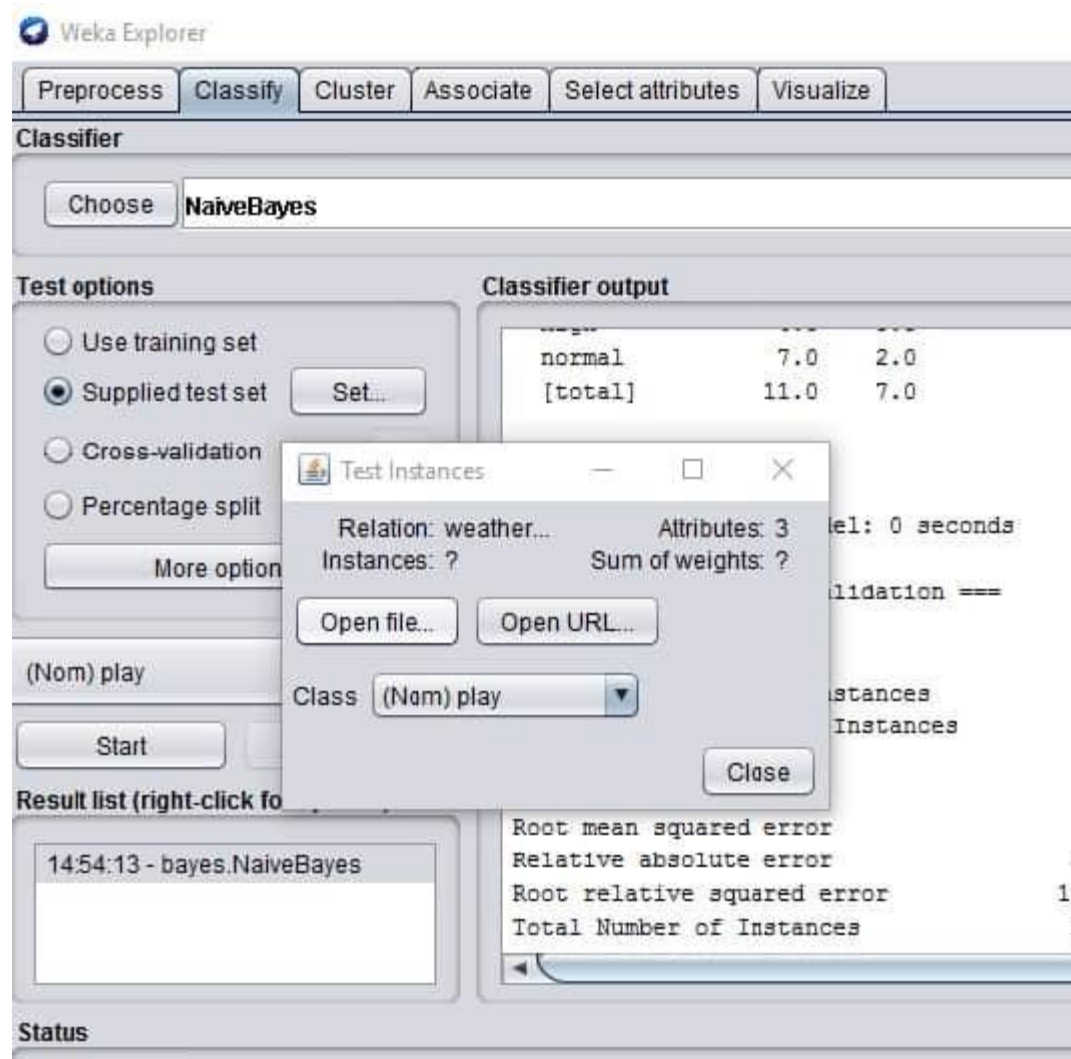
Incorrectly Classified Instances 5 35.7143 %

You can see that on a given data set, the classifier's accuracy is about 64%. So remember that you shouldn't always take the results for granted. To get better results, you might want to try different classifiers or preprocess data even further. We won't get into this right now. We need to demonstrate the usage of the model on new upcoming data.

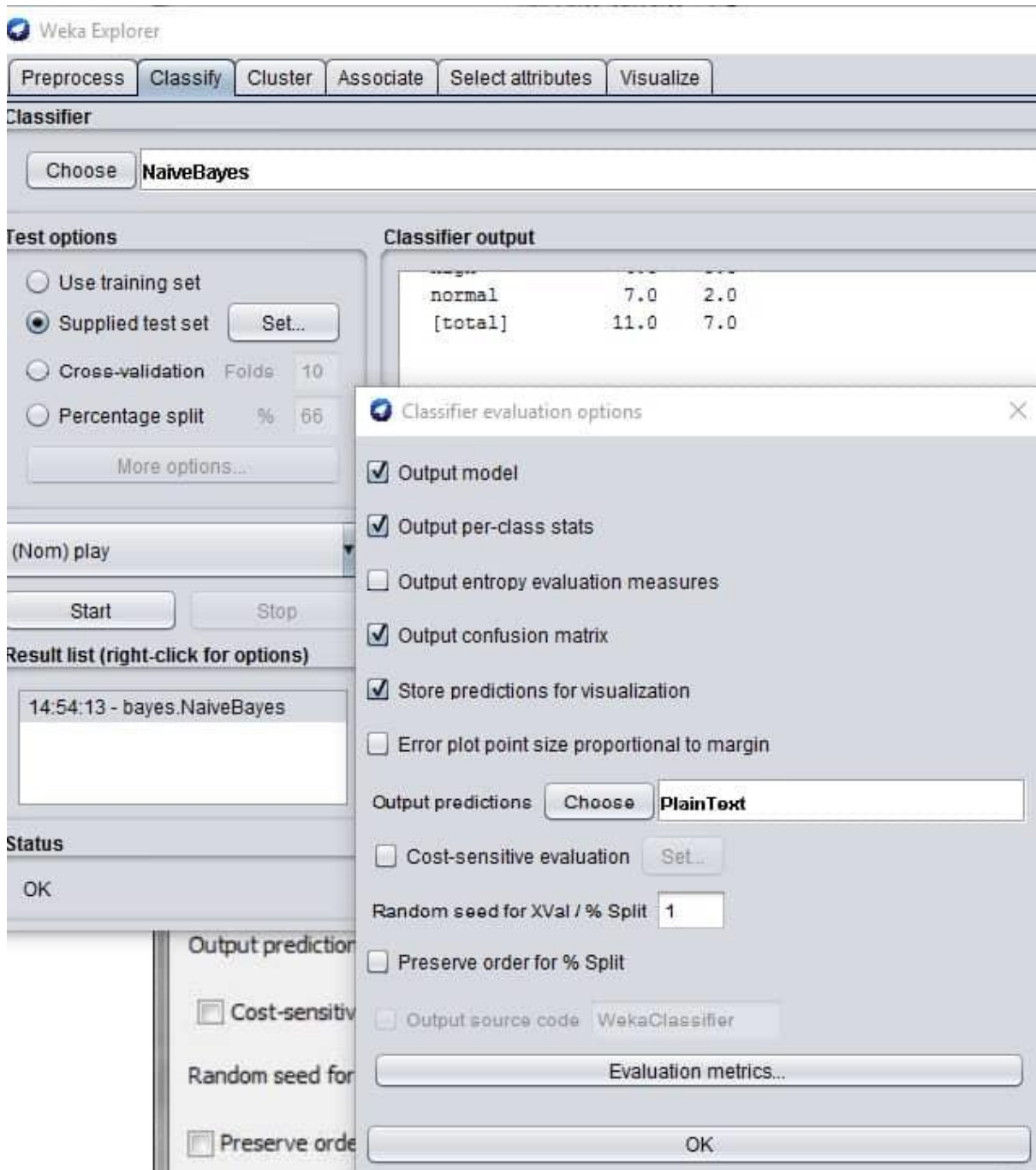
Evaluating classifier with the test set

Now when we have a model, we need to load the test data we've created before.

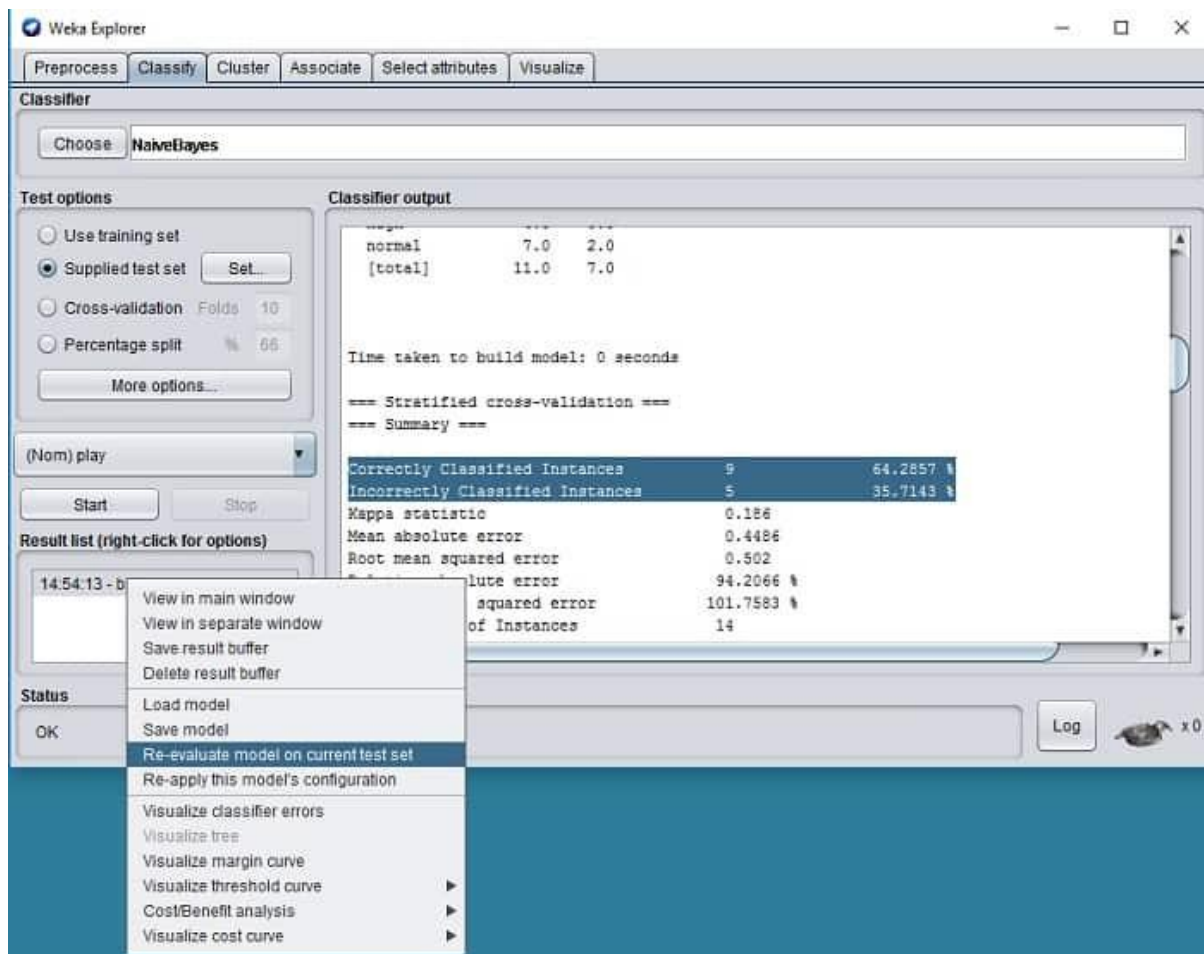
For this, select Supplied test set and click the button Set.



Click *More Options* wherein new window choose **PlainText** from *Output predictions* as follows:



Then click the left mouse button on a recently created model on the result list and select *Re-evaluate model on the current test set*.



And you should see the prediction for your given data cool and hot like this:

```

=== Predictions on user test set ===
inst#  actual  predicted error prediction
1      1:?    1:yes    0.531

```

RESULT

It has been predicted as yes, with an error of 53.1%. In the previous analytical example, we've got a 50% error on prediction.

EX : 8 Applying k-means clustering on a given data set

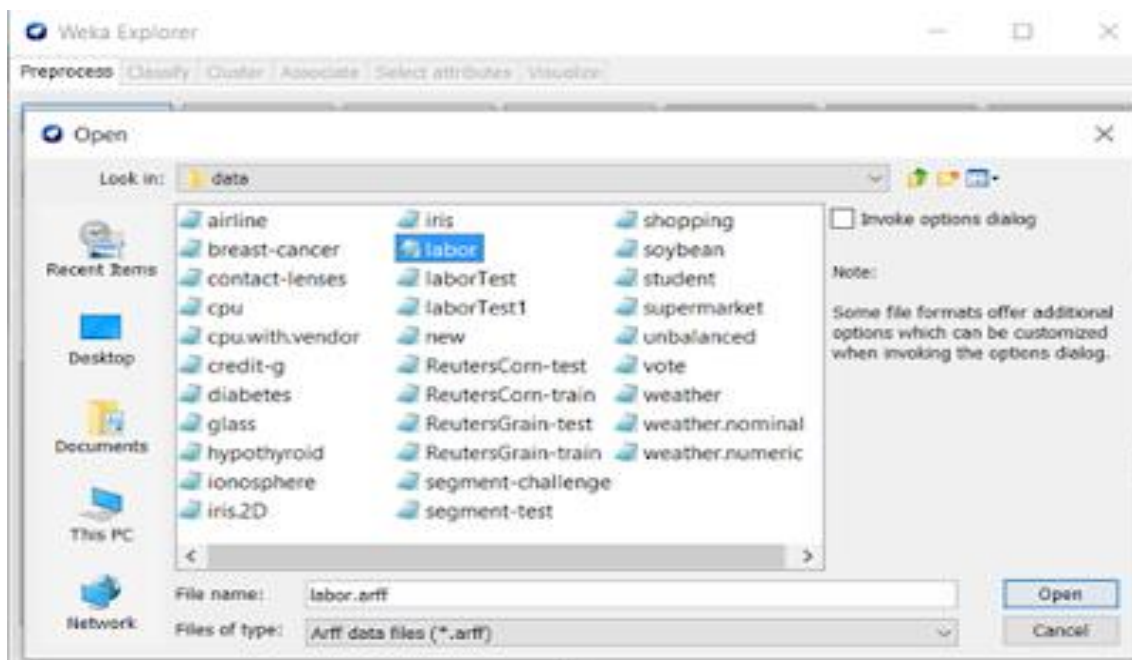
Procedure:

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 K-means 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.



Viewer ✕

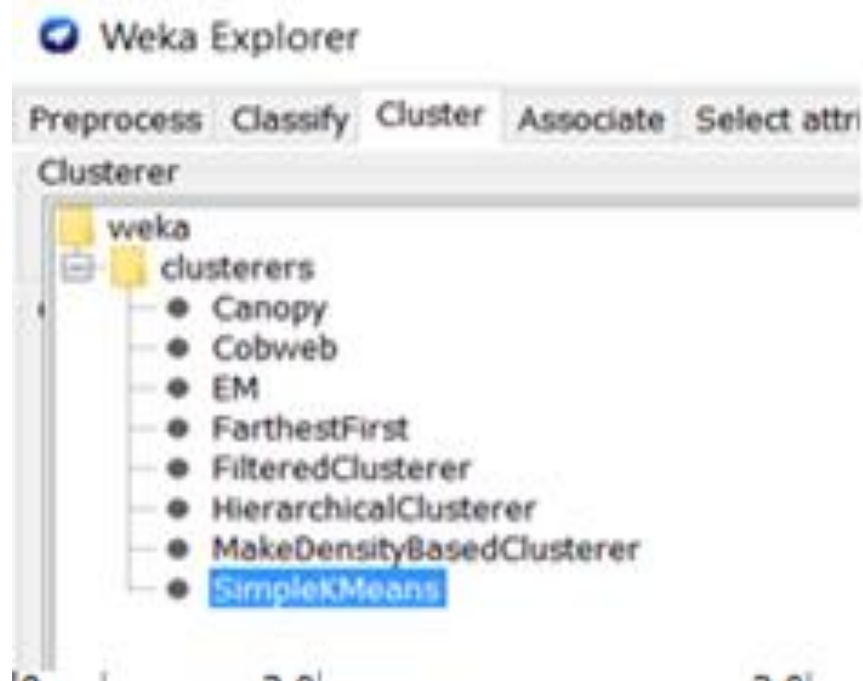
Relation: labor-neg-data

No.	1: duration Numeric	2: wage-increase-first-year Numeric	3: wage-increase-second- Numeric
1	1.0	5.0	
2	2.0	4.5	
3			
4	3.0	3.7	
5	3.0	4.5	
6	2.0	2.0	
7	3.0	4.0	
8	3.0	6.9	
9	2.0	3.0	
10	1.0	5.7	
11	3.0	3.5	
12	2.0	6.4	
13	2.0	3.5	
14	3.0	3.5	
15	1.0	3.0	
16	2.0	4.5	
17	1.0	2.8	
18	1.0	2.1	
19	1.0	2.0	
20	2.0	4.0	
21	2.0	4.3	
22	2.0	2.5	
23	3.0	3.5	

^
v

< >

Undo OK Cancel



weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

[More
Capabilities](#)

canopyMaximumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance +
doNotCheckCapabilities	False
doNotReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	3
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

Open...

Save...

OK

Cancel

Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation: labor-neg-data
Instances: 57
Attributes: 17

duration
wage-increase-first-year
wage-increase-second-year
wage-increase-third-year
cost-of-living-adjustment
working-hours
pension
standby-pay
shift-differential
education-allowance
statutory-holidays
vacation
longterm-disability-assistance
contribution-to-dental-plan
bereavement-assistance
contribution-to-health-plan
class

Test mode: evaluate on training data
=== Clustering model (full training set) ===

kMeans
=====

Number of iterations: 3
Within cluster sum of squared errors: 119.5224194214812

Initial starting points (random):

Cluster 0: 1,5,7,3,971739,3.913333,none,40,empl_contr,7.444444,4,no,11,generous,yes,full,yes,full,good
Cluster 1: 1,2,3,971739,3.913333,tc,40,ret_allw,4,0,no,11,generous,no,none,no,none,bad
Cluster 2: 2,2,5,3,3,913333,tcf,40,none,7.444444,4.870968,no,11,below_average,yes,half,yes,full,bad

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Cluster#			
	Full Data	0	1	2
	(57.0)	(36.0)	(5.0)	(16.0)
=====				
duration	2.1607	2.2267	1.4	2.25
wage-increase-first-year	3.8036	4.4695	3.2	2.4938
wage-increase-second-year	3.9717	4.4175	4.183	2.9027
wage-increase-third-year	3.9133	4.1093	3.9133	3.4725
cost-of-living-adjustment	none	none	none	none
working-hours	38.0392	37.4766	39.2078	38.94
pension	empl_contr	empl_contr	none	empl_contr
standby-pay	7.4444	7.9938	6.7556	6.4236
shift-differential	4.871	5.4776	3.1484	4.0444
education-allowance	no	no	no	no
statutory-holidays	11.0943	11.4801	10.6	10.3809
vacation	below_average	generous	below_average	below_average
longterm-disability-assistance	yes	yes	no	yes
contribution-to-dental-plan	half	half	none	half
bereavement-assistance	yes	yes	no	yes
contribution-to-health-plan	full	full	none	full
class	good	good	bad	bad

Time taken to build model (full training data) : 0.01 seconds

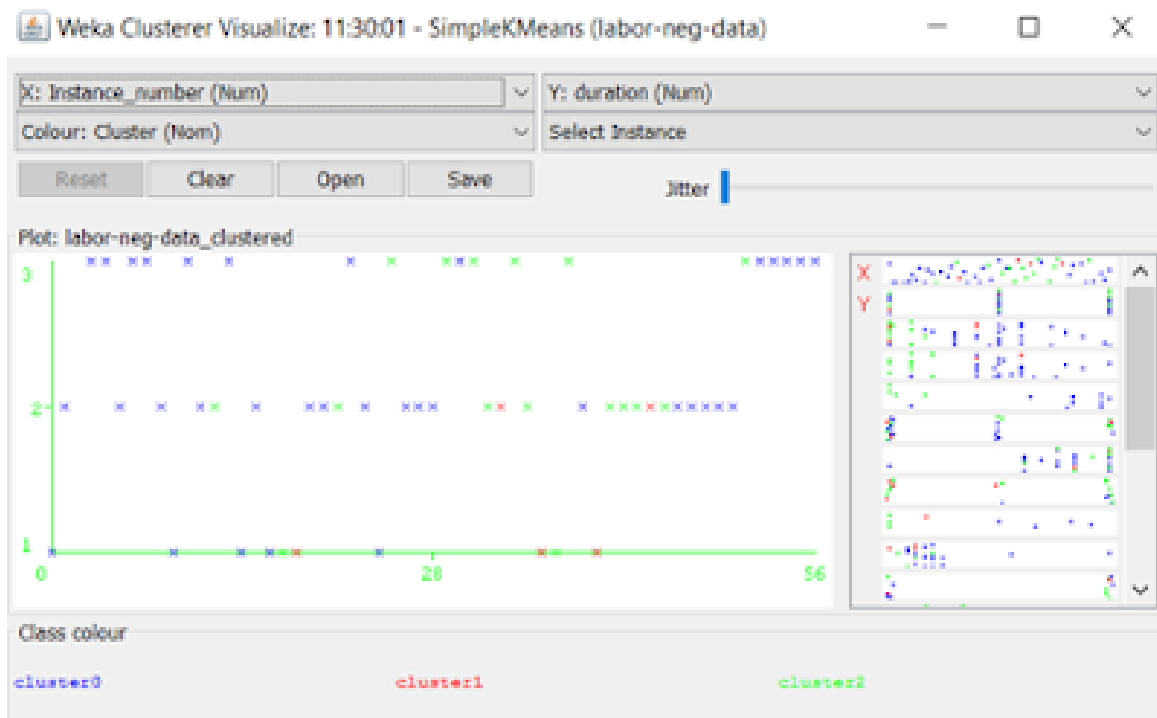
=== Model and evaluation on training set ===

Clustered Instances

```

0    36 ( 63%)
1     5 (  9%)
2    16 ( 28%)

```



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning

Cluster mode

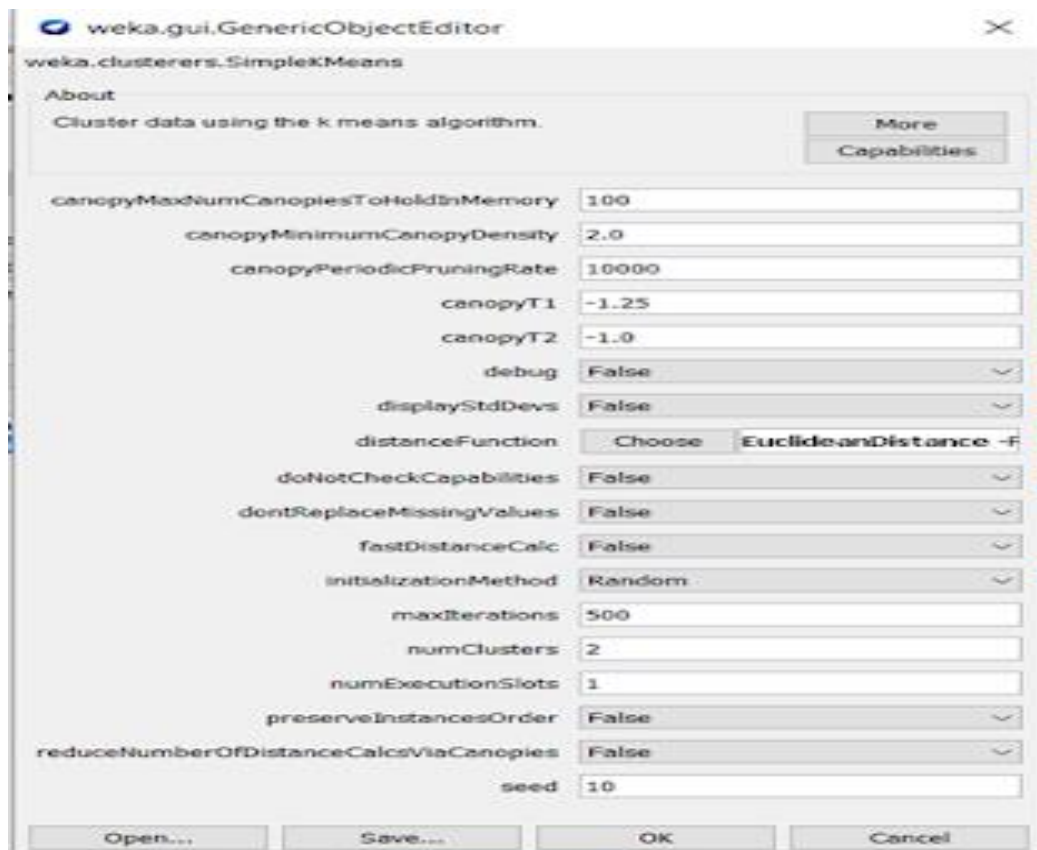
☐ Use training set
☐ Supplied test set Get...
☐ Percentage split % 56
☒ Classes to clusters evaluation

(Nom) class

(Nom) education-allowance
(Num) statutory-holidays
(Nom) vacation
(Nom) longterm-disability-assistance
(Nom) contribution-to-dental-plan
(Nom) bereavement-assistance
(Nom) contribution-to-health-plan
(Nom) class

Clusterer output

Number of iterations
Within cluster
Initial starting
Cluster 0: 1,5,1
Cluster 1: 1,2,1
Cluster 2: 2,2,1
Missing values
Final cluster of
Attribute



Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Relation: labor-neg-data

Instances: 57

Attributes: 17

- duration
- wage-increase-first-year
- wage-increase-second-year
- wage-increase-third-year
- cost-of-living-adjustment
- working-hours
- pension
- standby-pay
- shift-differential
- education-allowance
- statutory-holidays
- vacation
- longterm-disability-assistance
- contribution-to-dental-plan
- bereavement-assistance
- class

Ignored:

contribution-to-health-plan

Test mode: Classes to clusters evaluation on training data

=== Clustering model (full training set) ===

kMeans

=====

Number of iterations: 5

Within cluster sum of squared errors: 122.05464734126849

Initial starting points (random):

Cluster 0: 1,5,7,3.971739,3.913333,none,40,empl_contr,7.444444,4,no,11,generous,yes,full,yes,good

Cluster 1: 1,2,3.971739,3.913333,tc,40,ret_allw,4,0,no,11,generous,no,none,no,bad

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Cluster#		
	Full Data	0	1
	(57.0)	(43.0)	(14.0)
=====			
duration	2.1607	2.213	2
wage-increase-first-year	3.8036	4.2024	2.5786
wage-increase-second-year	3.9717	4.221	3.2062
wage-increase-third-year	3.9133	4.0329	3.5462
cost-of-living-adjustment	none	none	none
working-hours	38.0392	37.6557	39.2171
pension	empl_contr	empl_contr	none
standby-pay	7.4444	7.7778	6.4206
shift-differential	4.871	5.2018	3.8548
education-allowance	no	no	no
statutory-holidays	11.0943	11.2878	10.5
vacation	below_average	below_average	below_average
longterm-disability-assistance	yes	yes	yes
contribution-to-dental-plan	half	half	none
bereavement-assistance	yes	yes	yes
class	good	good	bad

Time taken to build model (full training data) : 0 seconds

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

Clustered Instances

0 43 (75%)

1 14 (25%)

Class attribute: contribution-to-health-plan

Classes to Clusters:

0 1 <-- assigned to cluster

20 8 | none

9 0 | half

14 6 | full

Cluster 0 <-- none

Cluster 1 <-- full

Incorrectly clustered instances : 31.0 54.386 %

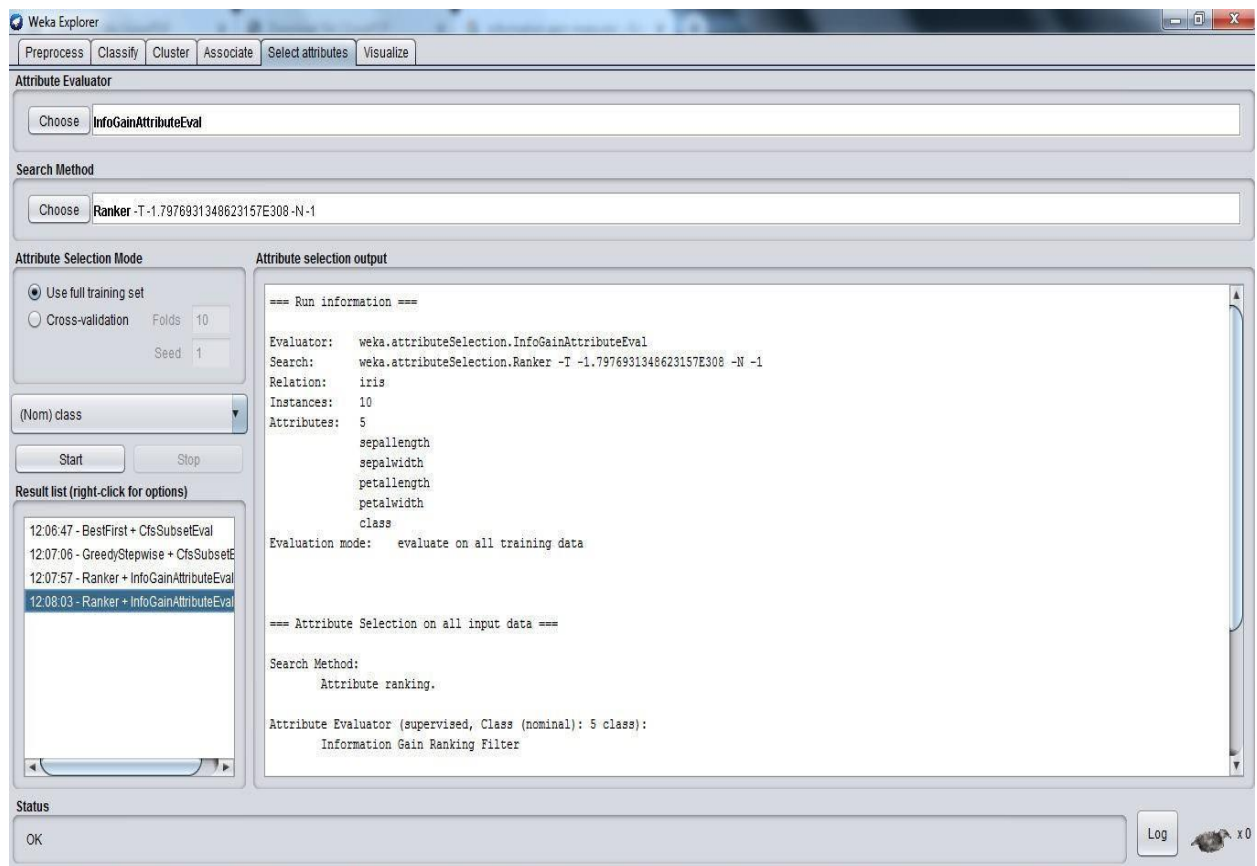


EX : 9 Calculating Information gains measures

Information gain (IG) measures how much “information” a feature gives us about the class. – Features that perfectly partition should give maximal information. – Unrelated features should give no information. It measures the reduction in entropy. CfsSubsetEval aims to identify a subset of attributes that are highly correlated with the target while not being strongly correlated with one another. It searches through the space of possible attribute subsets for the “best” one using the BestFirst search method by default, although other methods can be chosen. To use the wrapper method rather than a filter method, such as CfsSubsetEval, first select WrapperSubsetEval and then configure it by choosing a learning algorithm to apply and setting the number of cross-validation folds to use when evaluating it on each attribute subset.

Steps:

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Select and Click on data option button.
6. Choose a data set and open file.
7. Click on select attribute tab and Choose attribute evaluator, search method algorithm



Click on start button.

