DATA MINING LAB MANUAL

MR22-1CS0148

Index

Experiment 1: Installing Weka on Windows

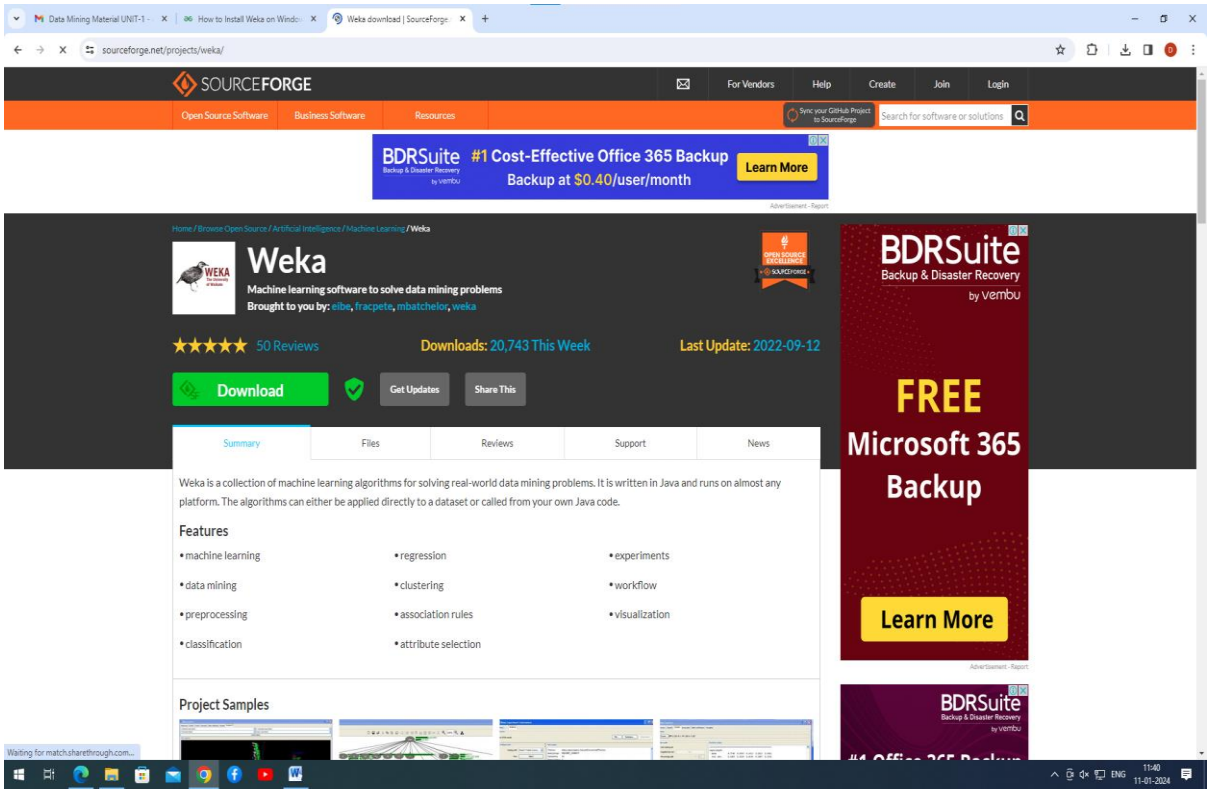
Weka stands for Waikato Environment for Knowledge Analysis, it is software that is used in the data science field for data mining. It is free software. It is written in Java hence it can be run on any system supporting Java, so weka can be run on different operating systems like Windows, Linux, Mac, etc. Weka provides a collection of visualization tools that can be used for data analysis, cleaning, and predictive modeling. Weka can perform a number of tasks like data preprocessing, clustering, classification, regression, visualization, and feature selection.

Installing Weka on Windows:

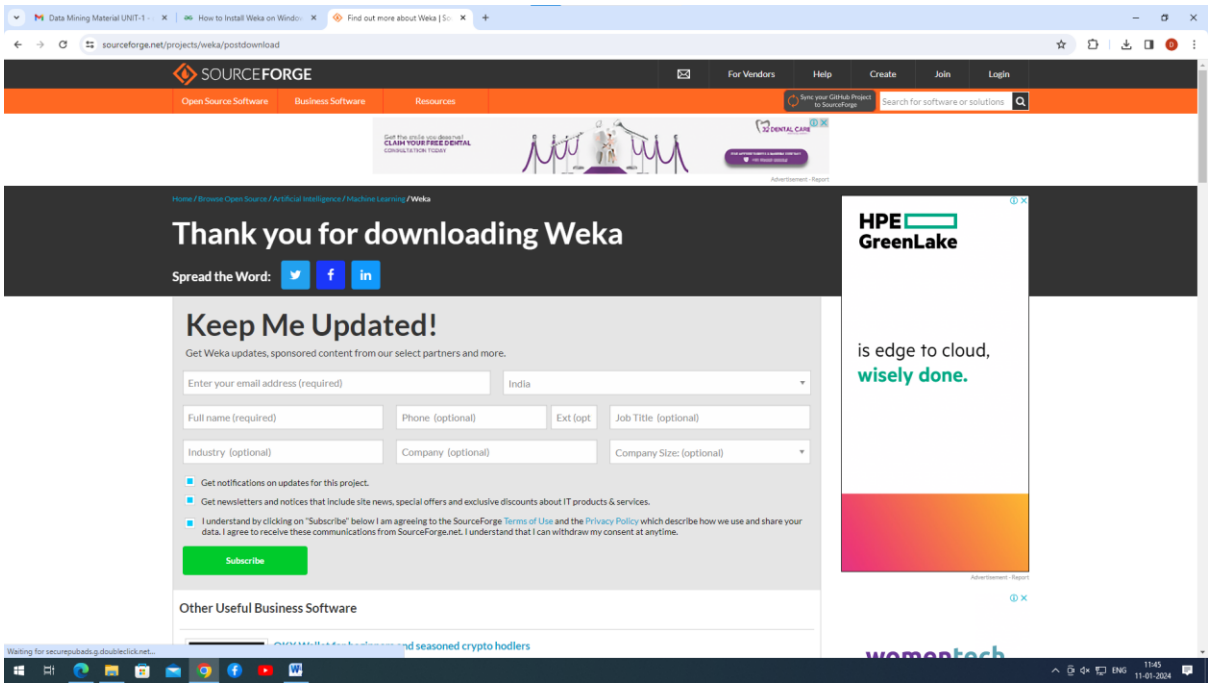
Follow the below steps to install Weka on Windows:

Step 1: Visit this website using any web browser. Click on Free Download.

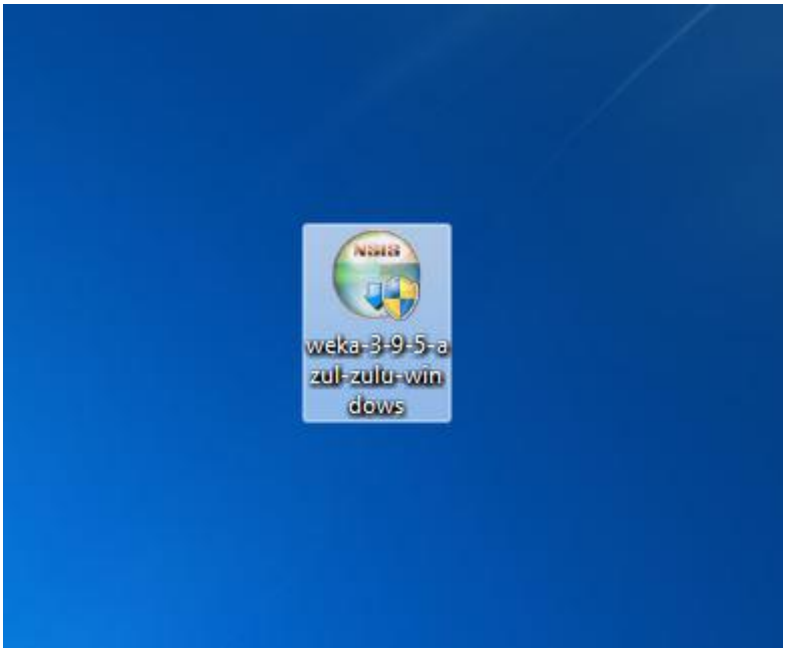
https://waikato.github.io/weka-wiki/downloading_weka/



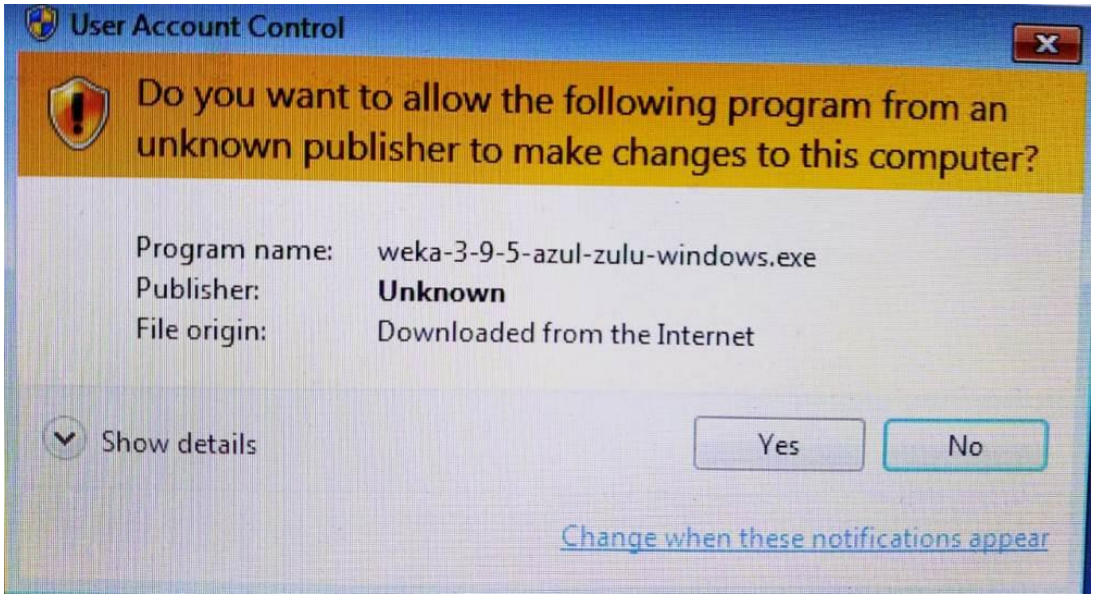
Step 2: It will redirect to a new webpage, click on Start Download. Downloading of the executable file will start shortly. It is a big 120 MB file that will take some minutes.



Step 3: Now check for the executable file in downloads in your system and run it.

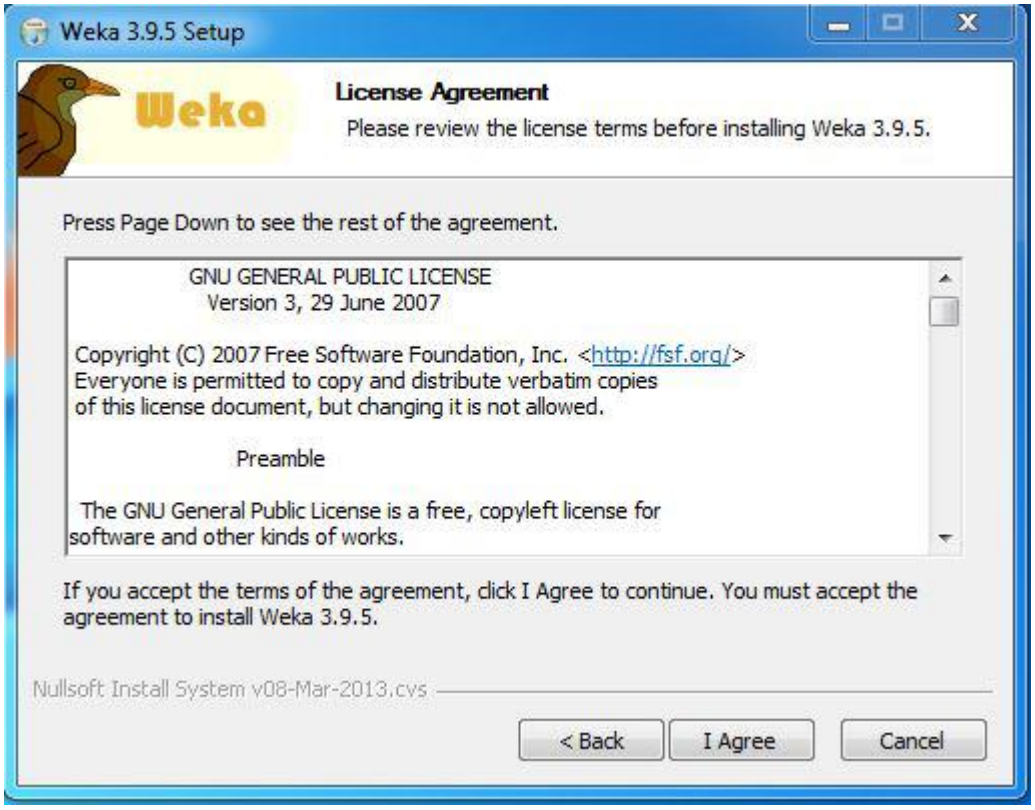


Step 4: It will prompt confirmation to make changes to your system. Click on Yes.

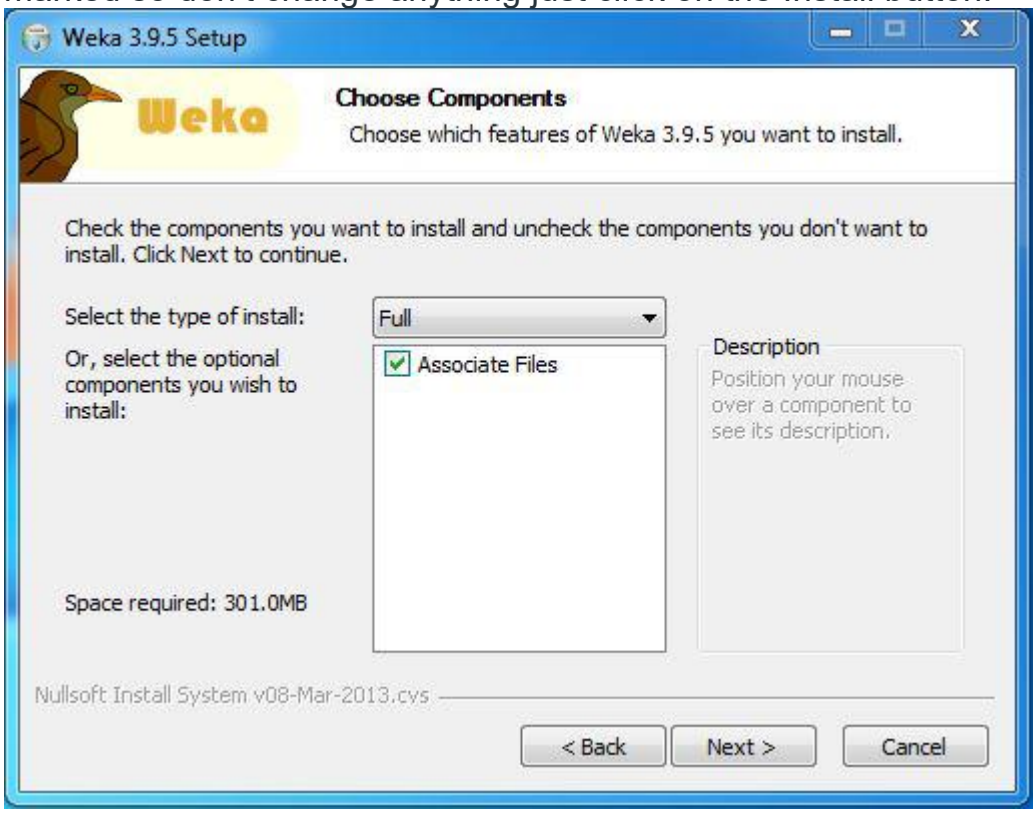


Step 5: Setup screen will appear, click on Next.

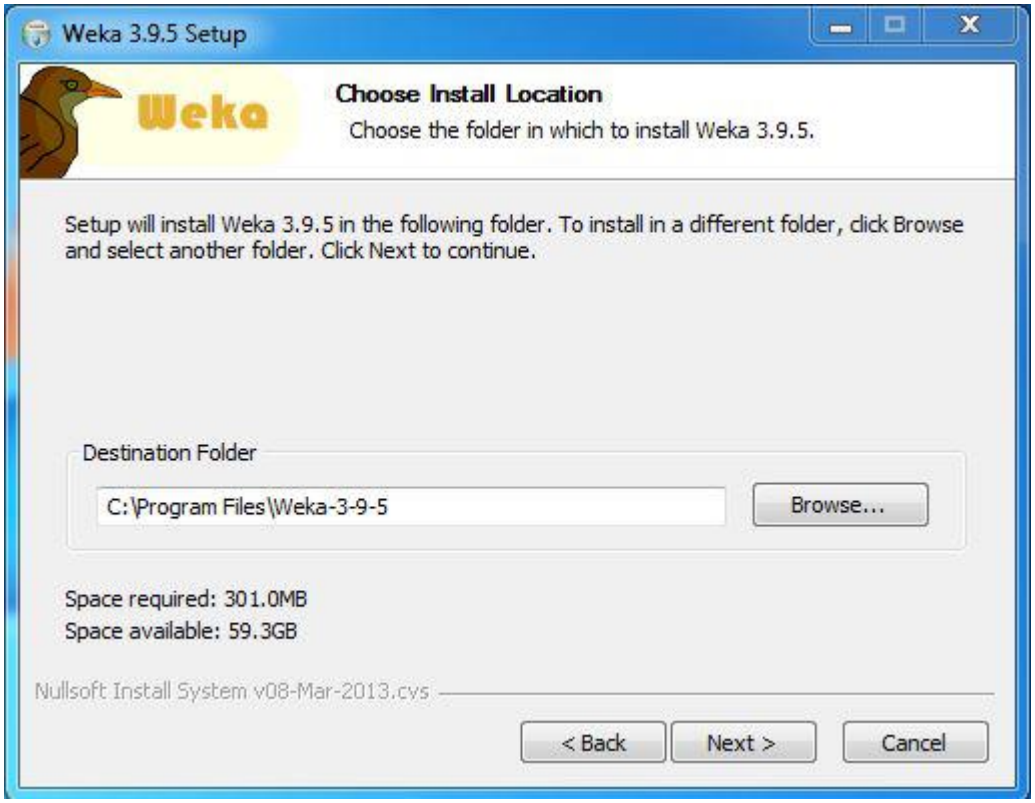
Step 6: The next screen will be of License Agreement, click on I Agree.



Step 7: Next screen is of choosing components, all components are already marked so don't change anything just click on the Install button.

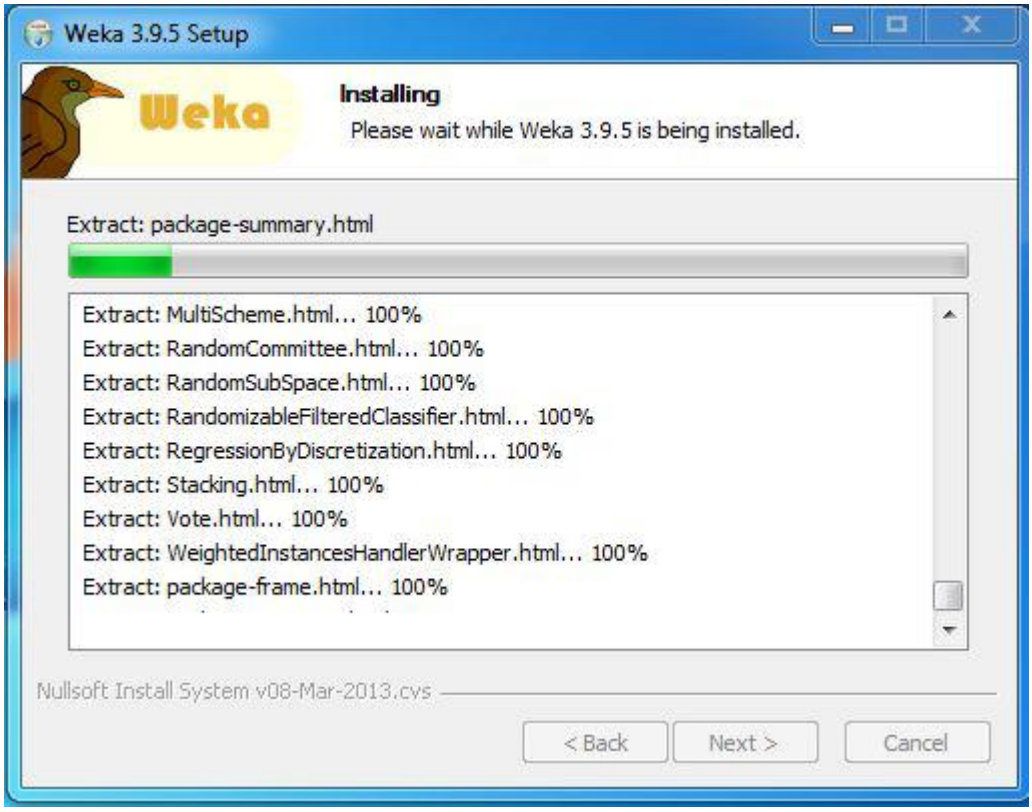


Step 8: The next screen will be of installing location so choose the drive which will have sufficient memory space for installation. It needed a memory space of 301 MB.

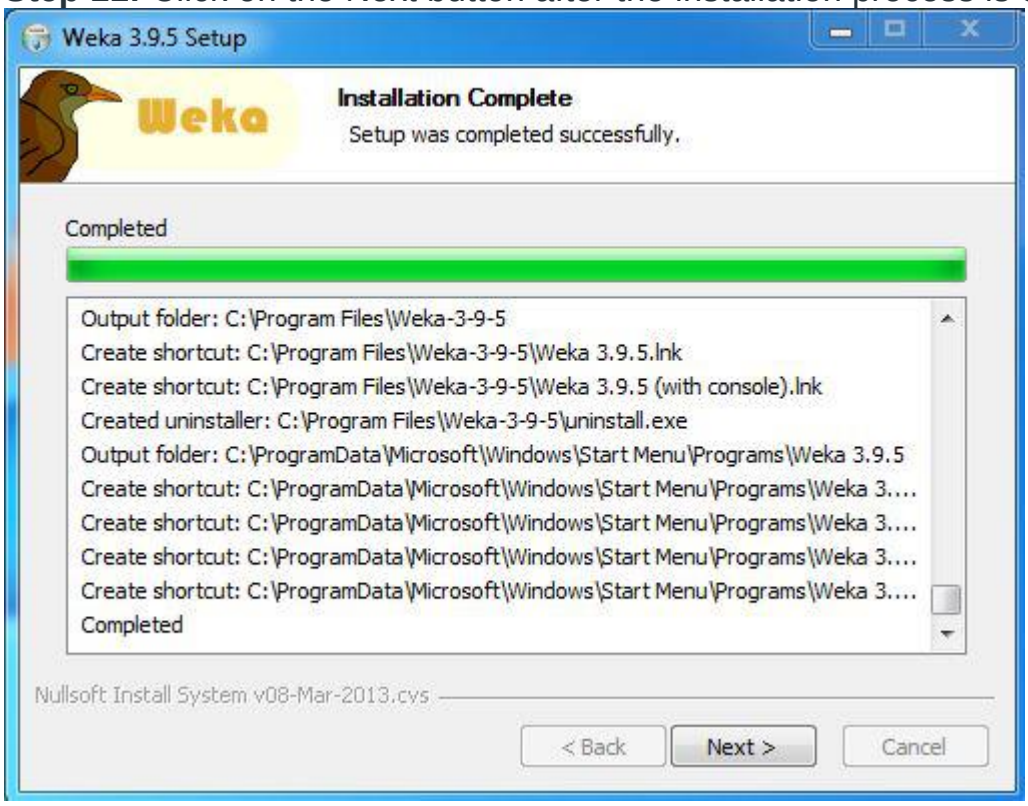


Step 9: Next screen will be of choosing the Start menu folder so don't do anything just click on Install Button.

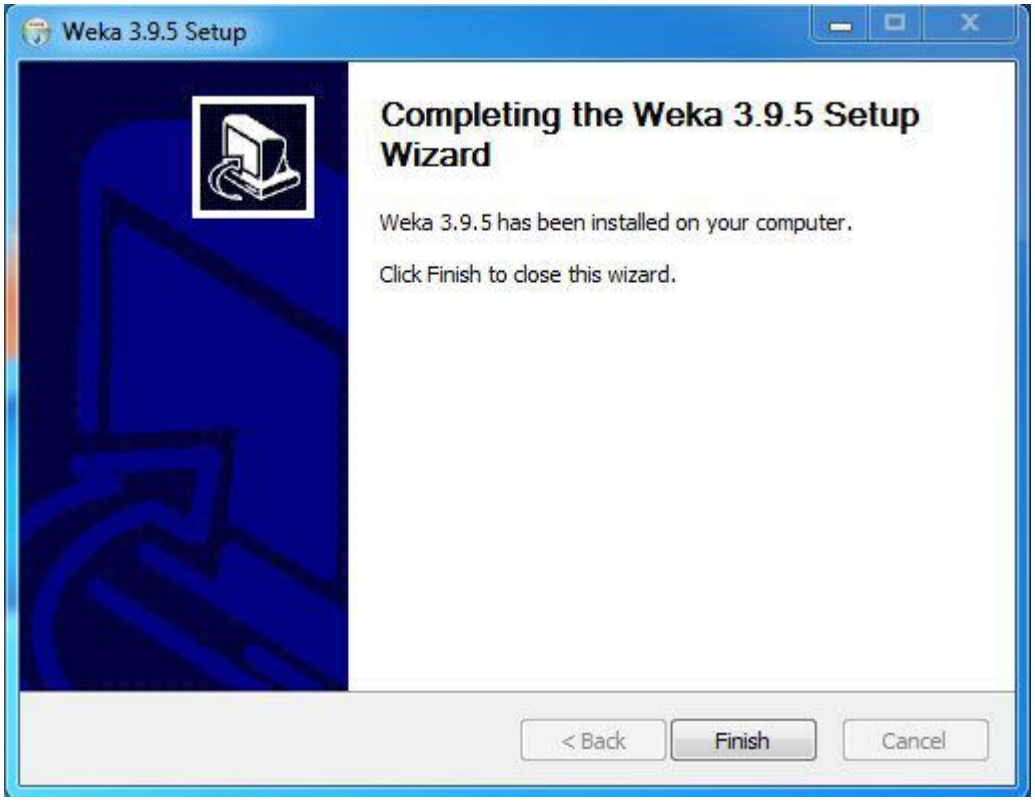
Step 10: After this installation process will start and will hardly take a minute to complete the installation.



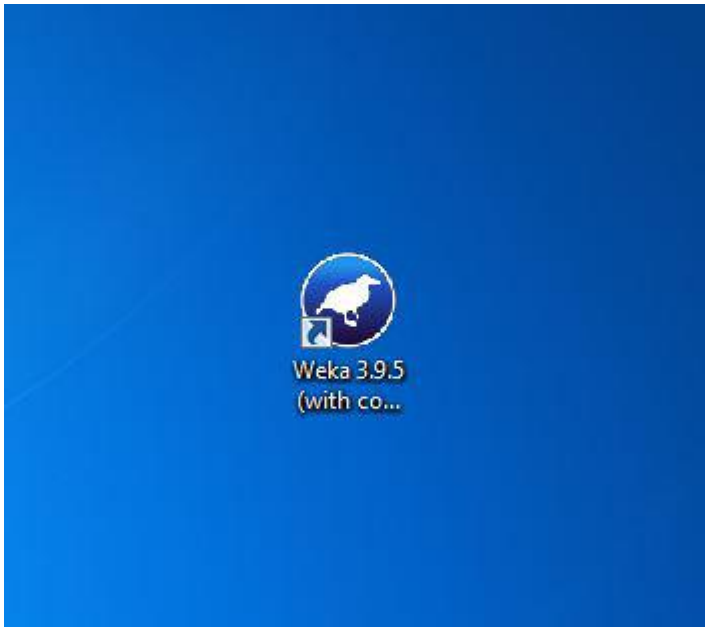
Step 11: Click on the Next button after the installation process is complete.



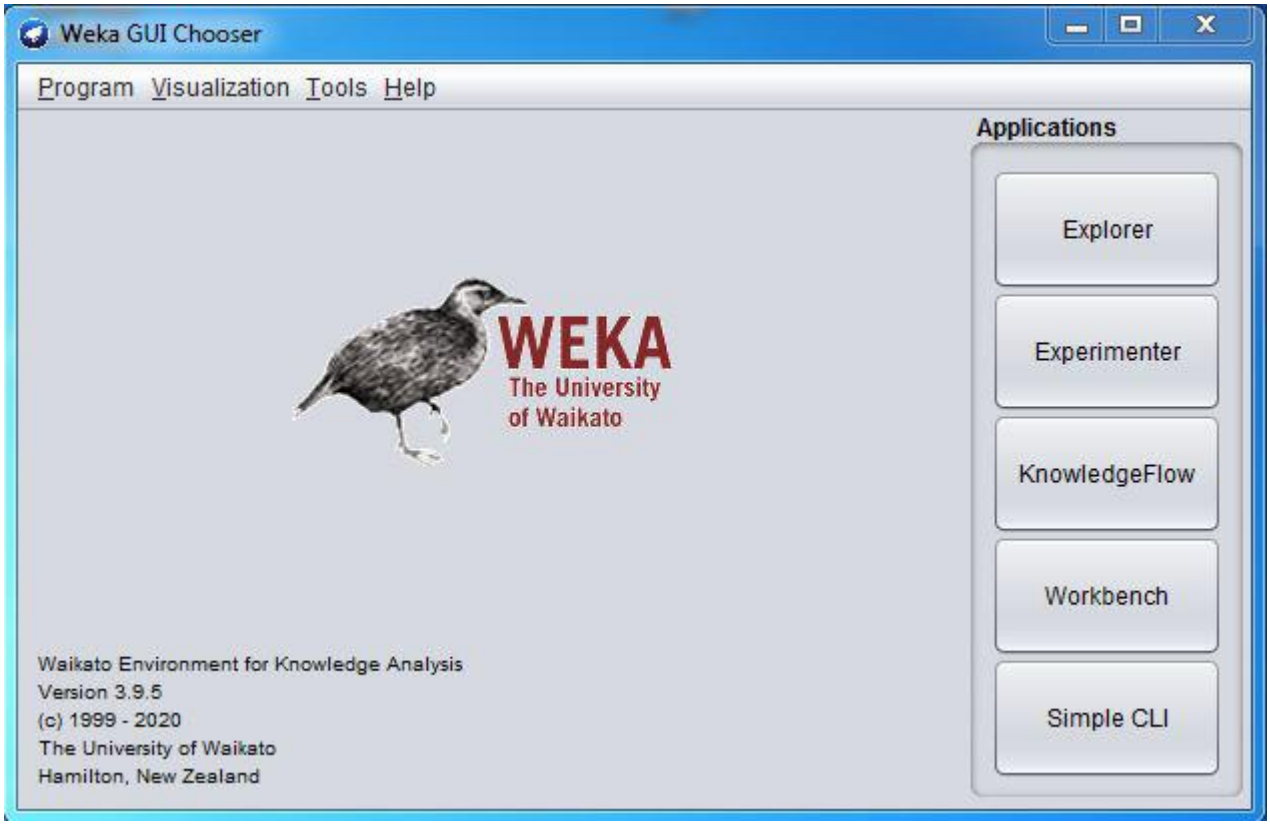
Step 12: Click on Finish to finish the installation process.



Step 13: Weka is successfully installed on the system and an icon is created on the desktop.



Step 14: Run the software and see the interface.



Congratulations!! At this point, you have successfully installed Weka on your windows system.

Experiment 2: Start working with WEKA tool kit and understand the features of WEKA tool kit. Loading Data from different sources in WEKA. Various File Formats supported by WEKA. And Study the ARFF file format.

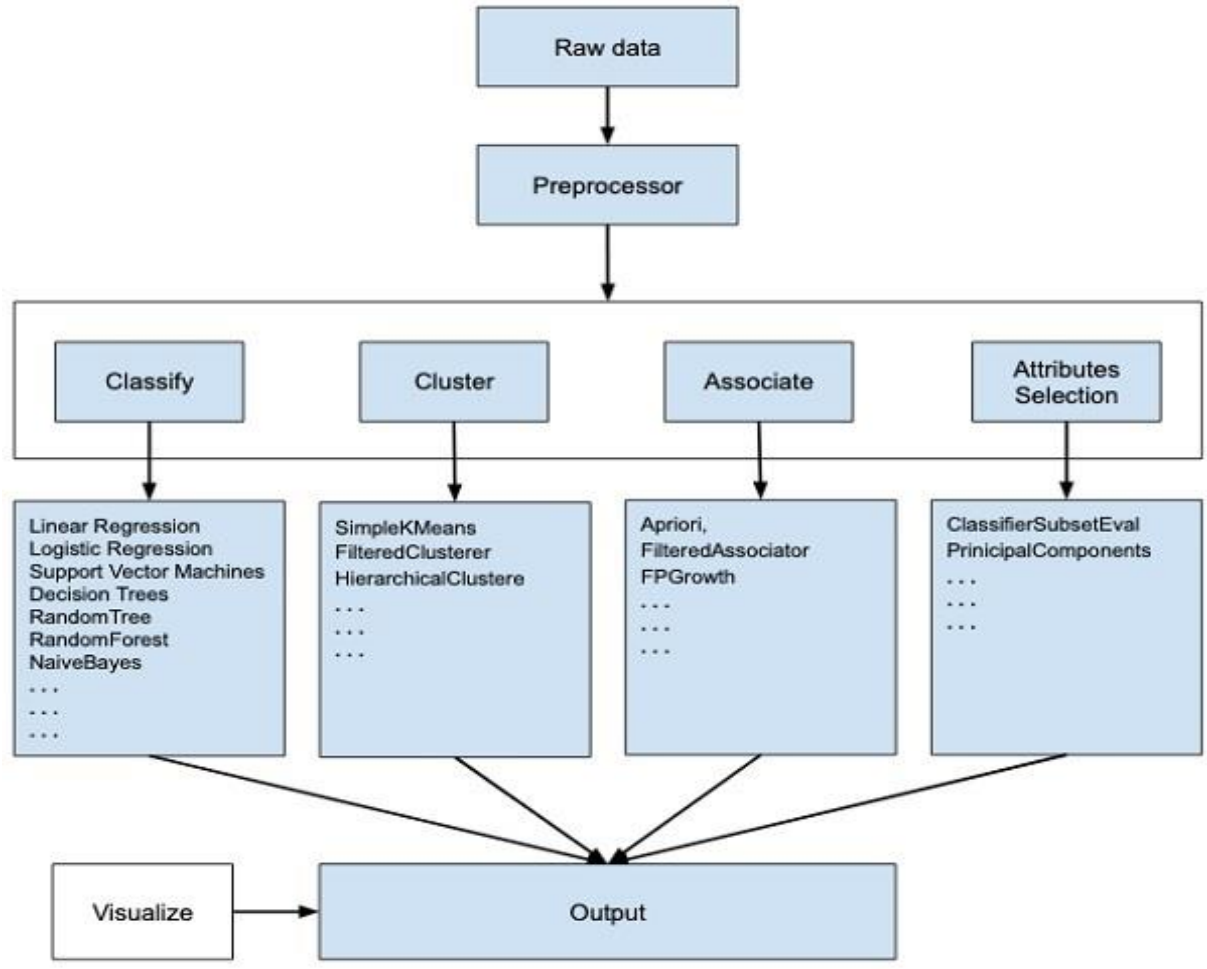
Start working with WEKA tool kit and understand the features of WEKA tool kit.

Solution :

What is WEKA:

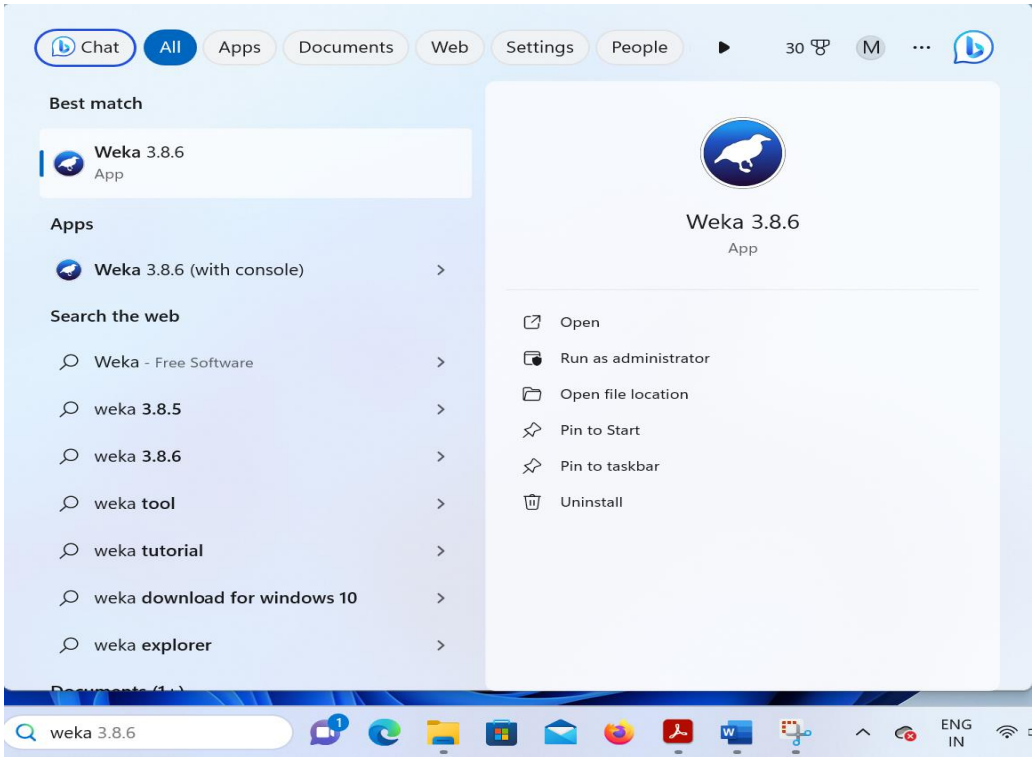
WEKA, an open-source software, offers a range of tools for data preprocessing, implementation of various Data Mining algorithms, and visualization tools. These resources enable users to develop data mining techniques and effectively apply them to real-world data mining problems.

The diagram presented below provides a concise summary of the offerings provided by WEKA.



To start Weka:

Search for Weka 3.8.6 and click on **Weka 3.8.6 app.**

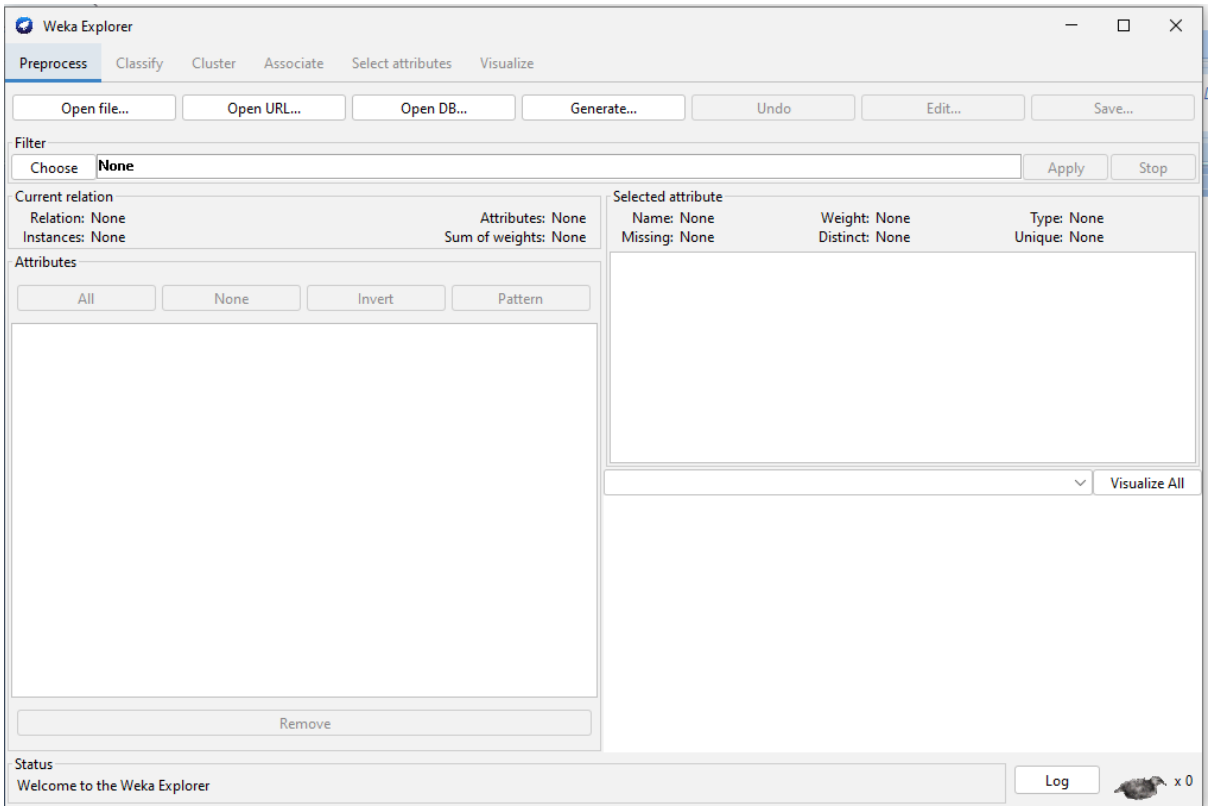


The following **Graphical User Interface** Of WEKA you get when you click on **Weka 3.8.6 app**.

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

1. Explorer

It is an environment for exploring data with WEKA. And it apply the various data mining algorithms. When you click on the **Explorer** button in the **Applications** selector, it displays the following window.



Located at the uppermost section of the window, positioned just below the title bar, is a series of tabs. Upon launching the Explorer, only the first tab is enabled, while the remaining tabs are displayed in an unresponsive manner. This is due to the prerequisite of opening and pre-processing a data set before data exploration.

The **tabs** are as follows:

Preprocess:

The first step in Data Mining is to preprocess the data. You will select the data file in the Preprocess option. Then, you will process the data and make it suitable for applying the different Data Mining algorithms.

Classify:

The Classify tab offers a range of Data Mining algorithms for the classification of your data. Some of the algorithms that can be applied include Linear Regression, Logistic Regression, Support Vector Machines, Decision Trees, Random Tree, Random Forest, Naive Bayes, and others.

Cluster:

The Cluster tab contains a variety of clustering algorithms, including Simple K-Means, Filtered Clusterer, Hierarchical Clusterer, and many more.

Associate:

The Associate tab contains Apriori, Filtered Associator and FPGrowth. These are used to learn / discover association rules in the data.

Select attributes:

This tab contains various methods to select the most relevant attributes in the data.

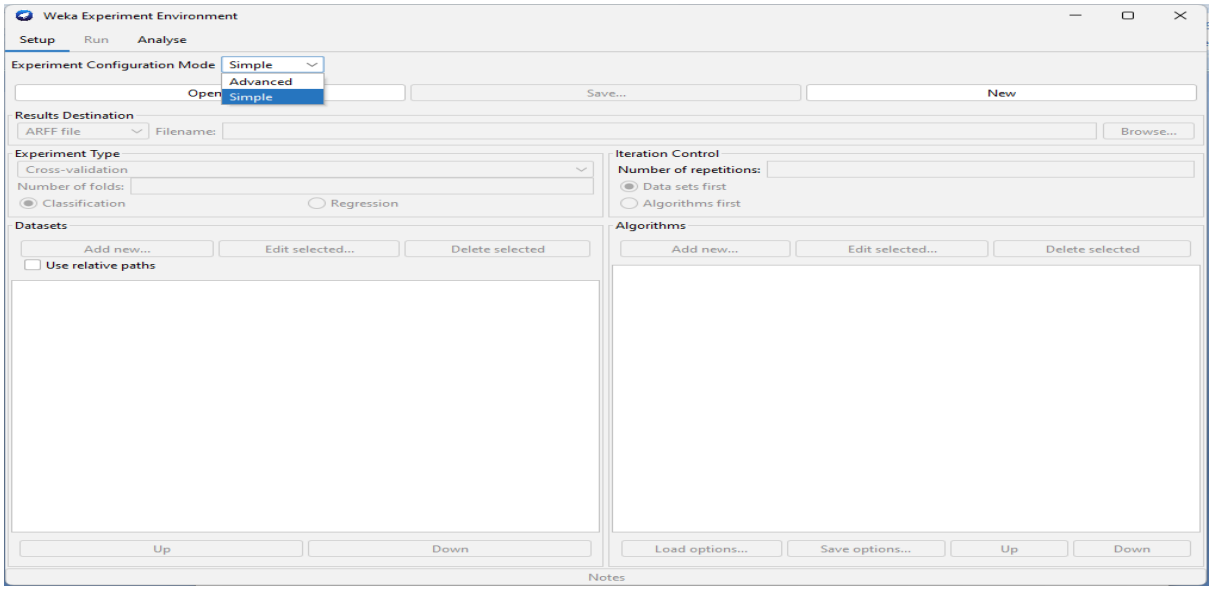
Visualize:

In this tab, various plots and graphs are available to show the trends identified by the model. I.e. it displays an interactive 2D plot of the data.

2. Experimenter

The Experimenter Environment allows users to easily create, run, modify, and analyze experiments. Users can create experiments that test multiple schemes on different datasets and analyze the results to determine statistical differences between the schemes.

When you click on the **Experimenter** button in the **Applications selector**, it displays the following window.



The Experimenter is available in two variants, those are

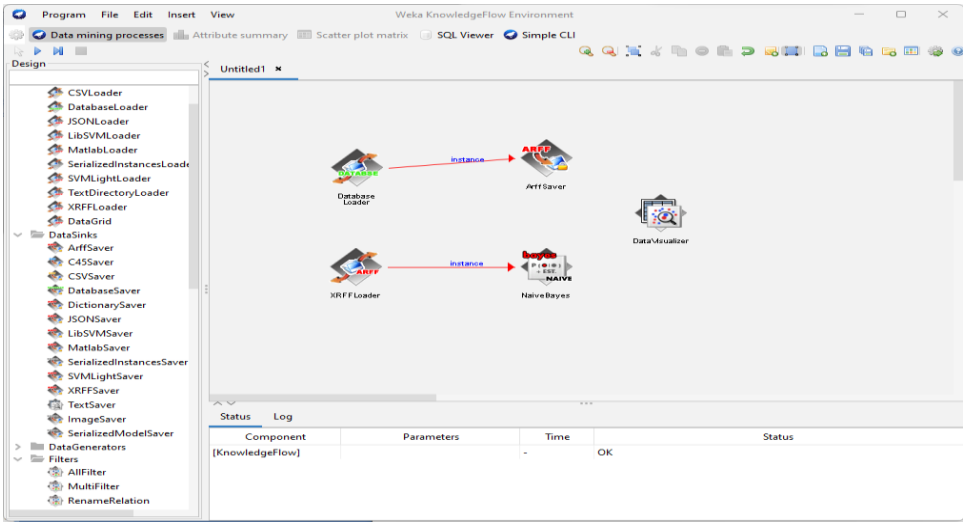
- **Simple**
  This variant provides most of the functionality one needs for experiments
- **Advanced**
  This is an interface with full access to the Experimenter's capabilities.

3. Knowledge flow

The Knowledge Flow offers an alternative to the Explorer as a graphical user interface for accessing the core algorithms of WEKA.

The Knowledge Flow platform offers an interface that draws inspiration from data-flow principles, specifically designed for WEKA. Users are able to choose components from a selection of WEKA tools, position them on a layout canvas, and establish connections between them. This facilitates the creation of a knowledge flow, enabling efficient processing and analysis of data.

When you click on the **Knowledge flow** button in the **Applications selector**, it displays the following window.



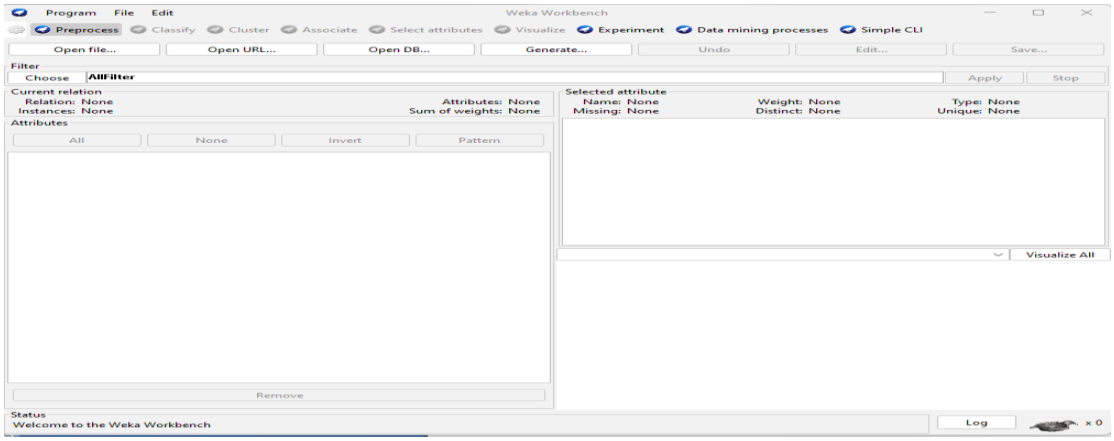
Currently, all classifiers, filters, clusterers, associators, loaders, and savers provided by WEKA are accessible within the Knowledge Flow platform, along with extra tools.

4. Workbench

The Workbench is an integrated environment that combines all graphical user interfaces into a unified or single interface.

If you frequently switch between multiple interfaces, such as the Explorer and the Experiment Environment, it can be beneficial. This is often the case when testing various scenarios in the Explorer and promptly implementing acquired knowledge into controlled experiments.

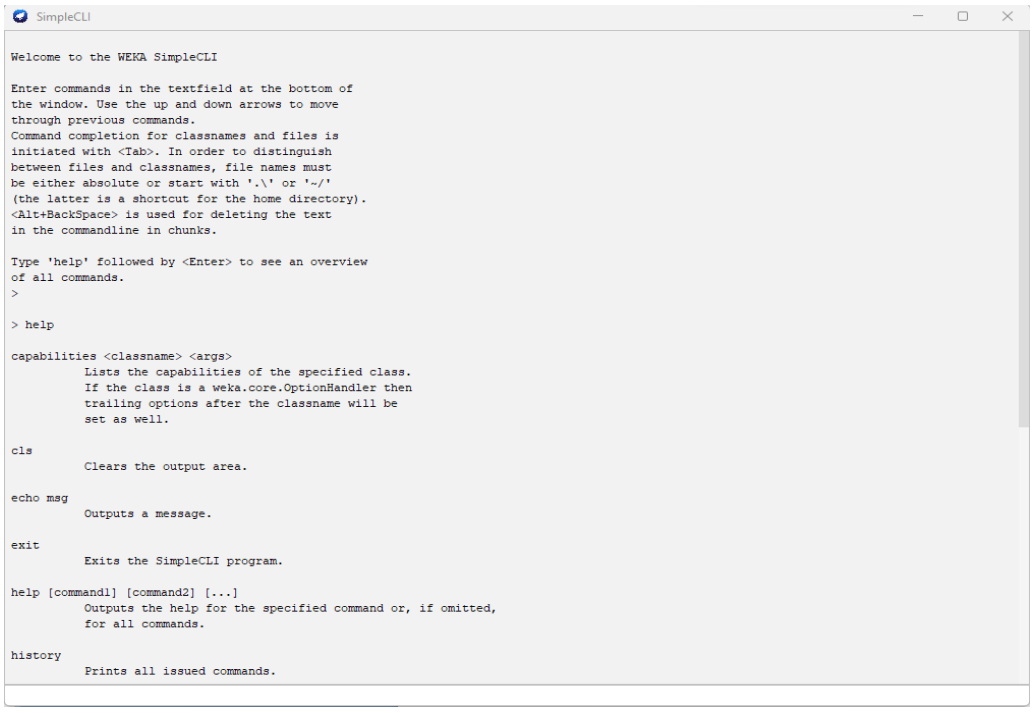
When you click on the Workbench button in the Applications selector, it displays the following window.



5. Simple CLI

The Simple Command Line Interface (CLI) grants comprehensive access to all Weka classes, including classifiers, filters, clusterers, and more, while eliminating the inconvenience of managing the CLASSPATH (it simplifies the one used during Weka's initialization). It presents a straightforward Weka shell with distinct command line and output sections.

When you click on the **Simple CLI** button in the **Applications selector**, it displays the following window.



Loading Data from different sources in WEKA.

Solution :

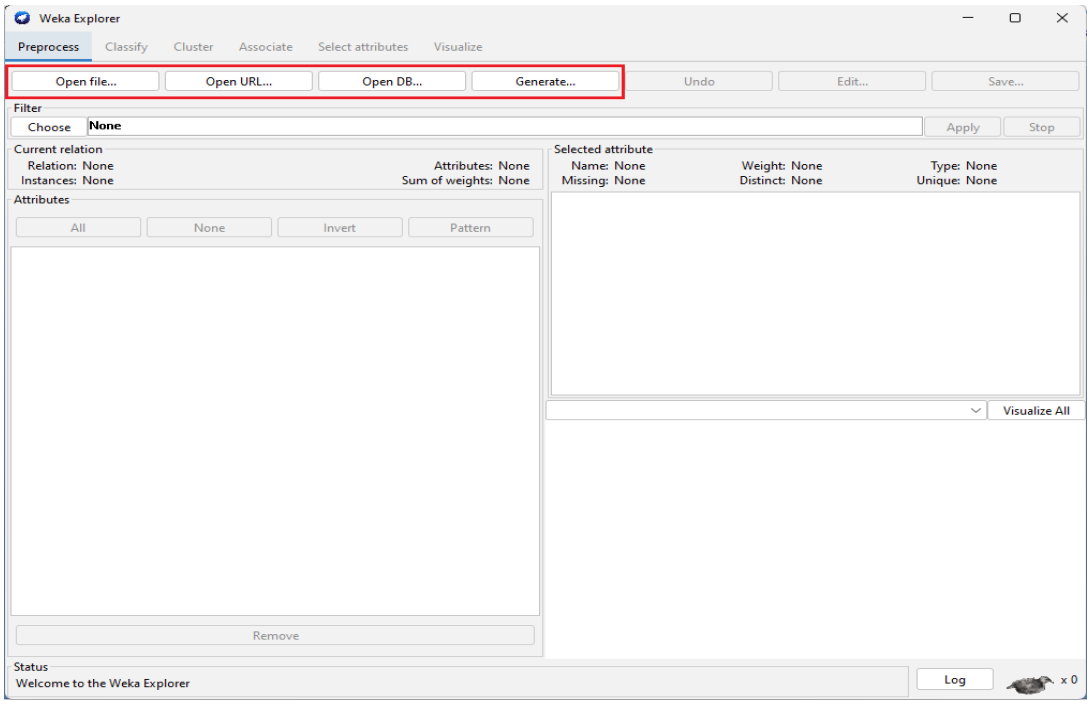
Loading Data from different sources in WEKA :

In order to utilize Weka Explorer, it is essential to start by loading the data into the application.

Multiple sources are available for data loading within Weka Explorer.Those are,

1. **Local file system**
2. **Web**
3. **Database**
4. **Generate Artificial Data**

The diagram presented below provides a concise summary of the offerings provided by WEKA.

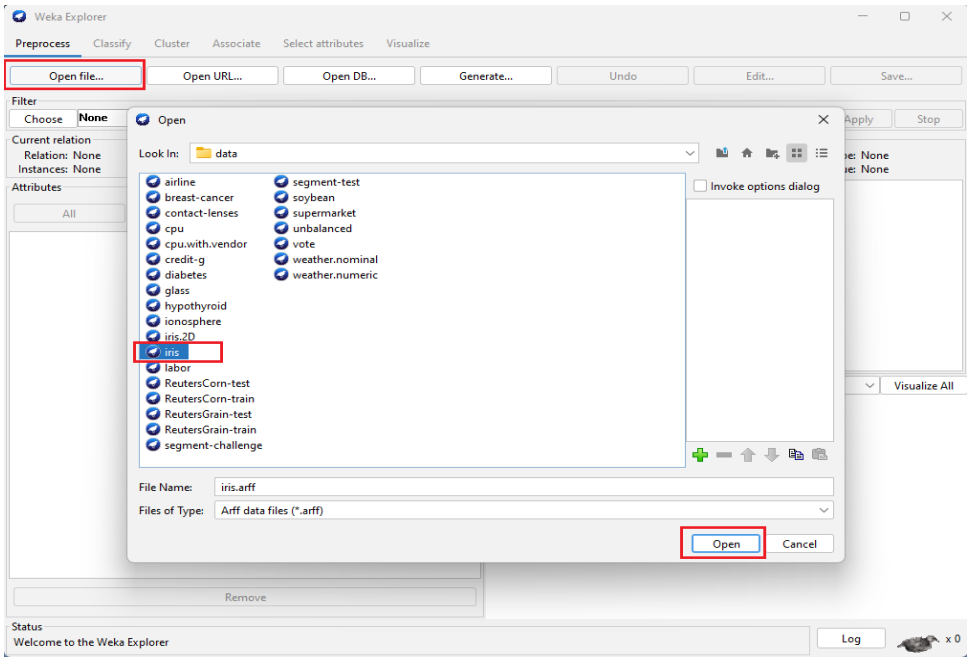


1. Loading data from Local file System:

In this, we are going to load data from the Local File System by clicking on the **Open file...** button.

Steps:

- Click on **Open file...** button.
- **Navigate the folder,** where the data files are stored.
- Select the **required data file.**
- Click on **Open** button.

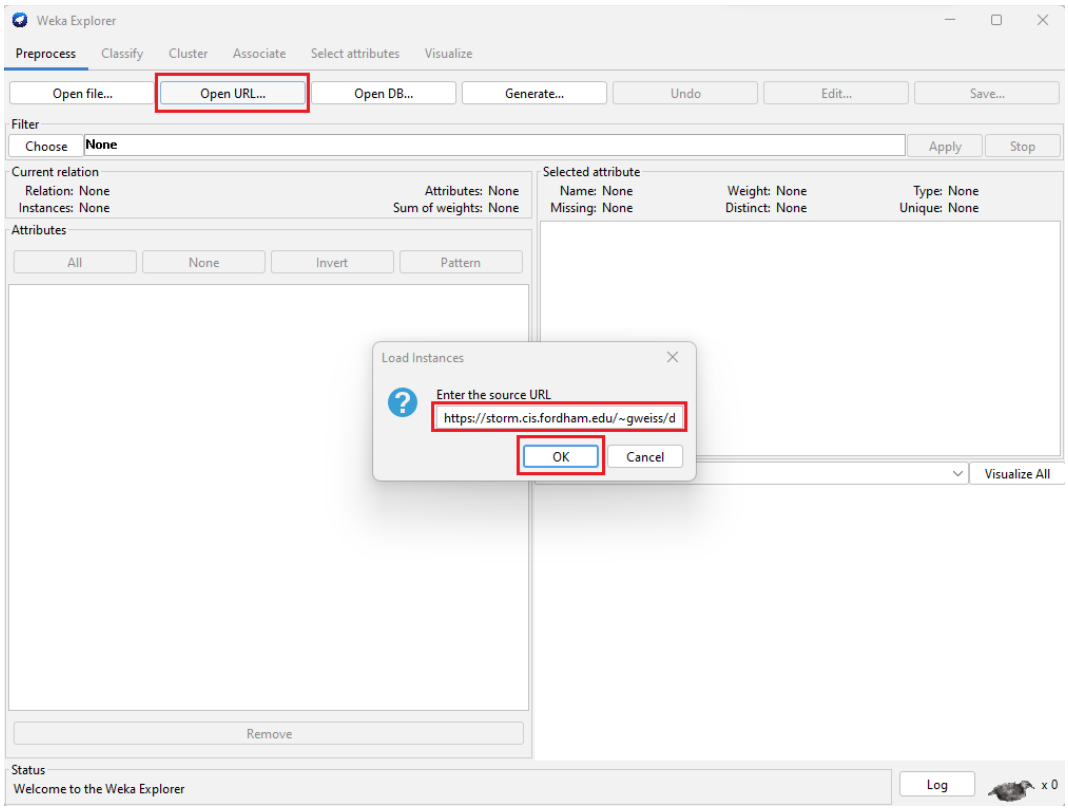


2. Loading data from Web:

In this, we are going to load data from the Web by clicking on the **Open URL...** button.

Steps:

- Click on **Open URL...** button.
- **Enter the URL** of data source in the popup box.
- Click on **Ok** button.



Example public **Data source URLs** are:

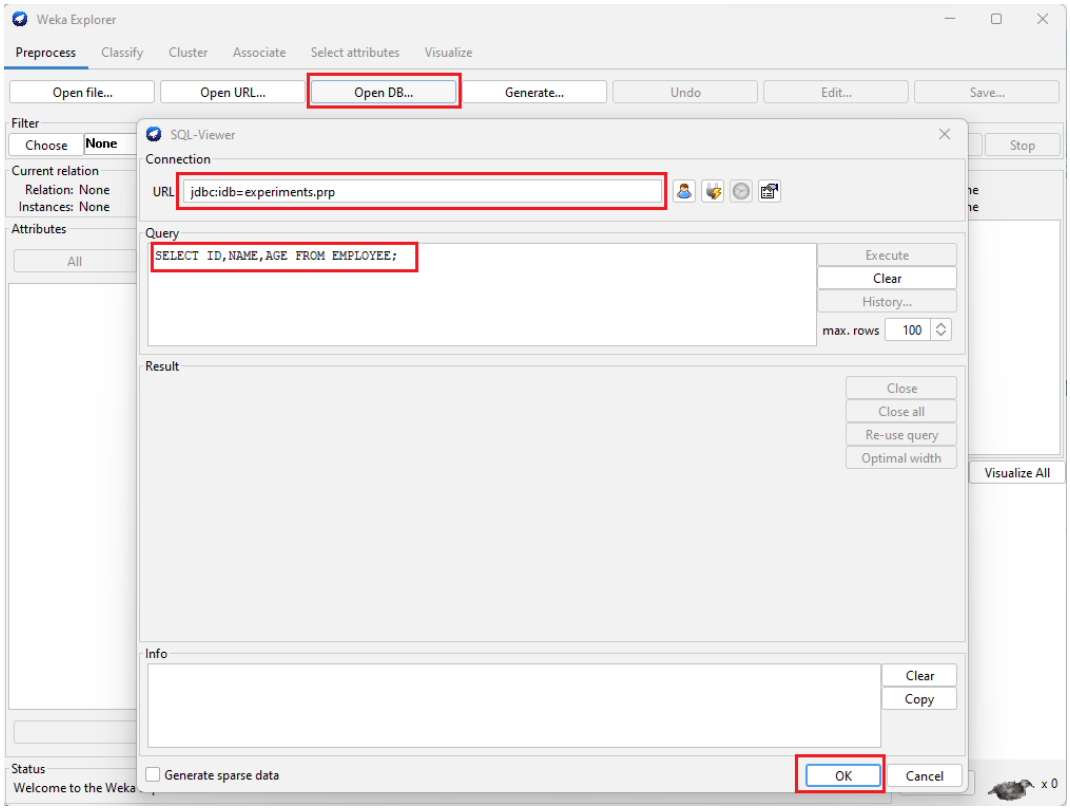
- https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/weather.nominal.arff
- https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/iris.arff
- https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/vote.arff

3. Loading data from Database:

In this, we are going to load data from the Database by clicking on the **Open DB...** button.

Steps:

- Click on **Open DB...** button.
- Enter the **Connection URL** of database.
- Type **SQL Query** to get data from Database table.
- Click on **Ok** button.

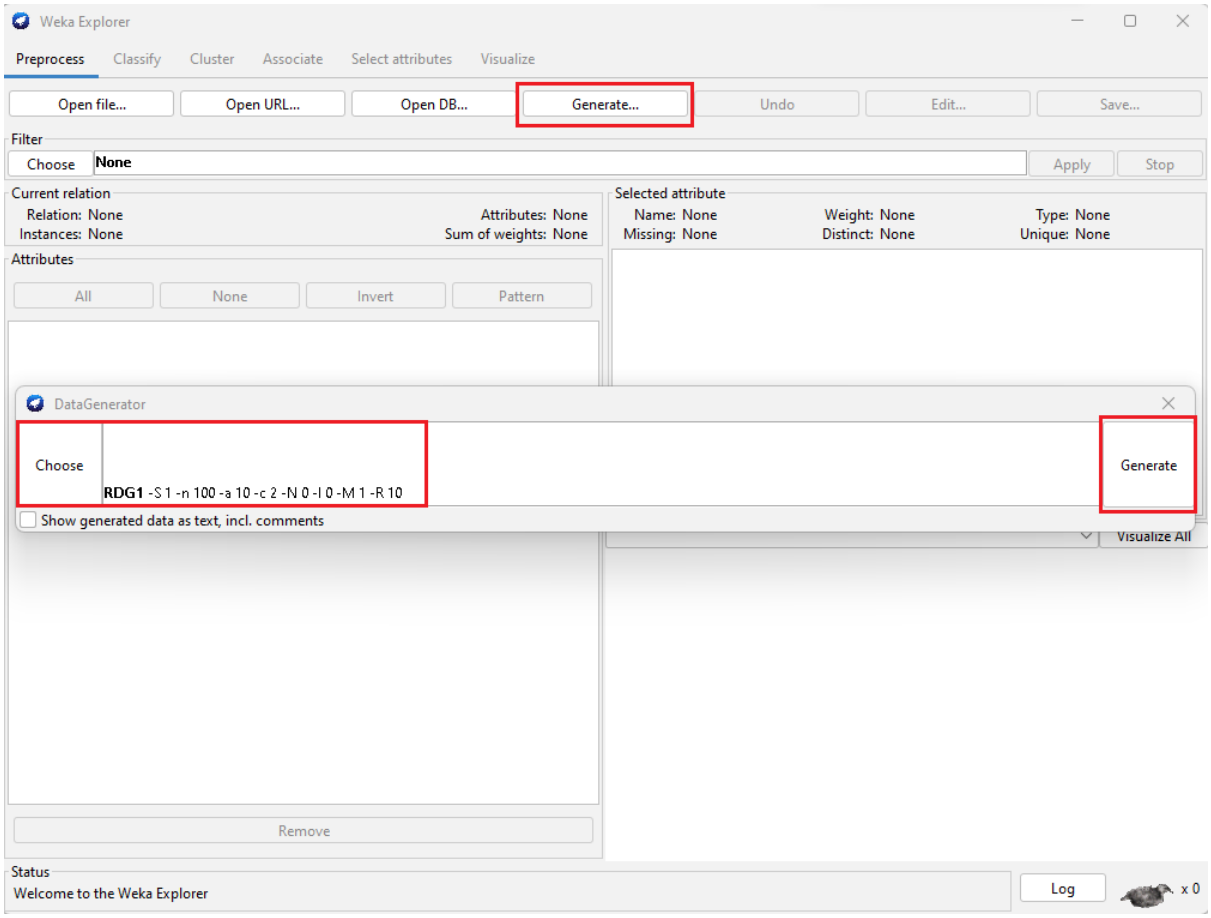


4. Generate Artificial Data:

In this, the artificial data (random data) will be generated by clicking on the **Generate...** button.

Steps:

- Click on **Generate...** button.
- **Choose Data Generator**.
- Click on **Generate** button.



Various File Formats supported by WEKA. And Study the ARFF file format.

Solution :

Various File Formats supported by WEKA :

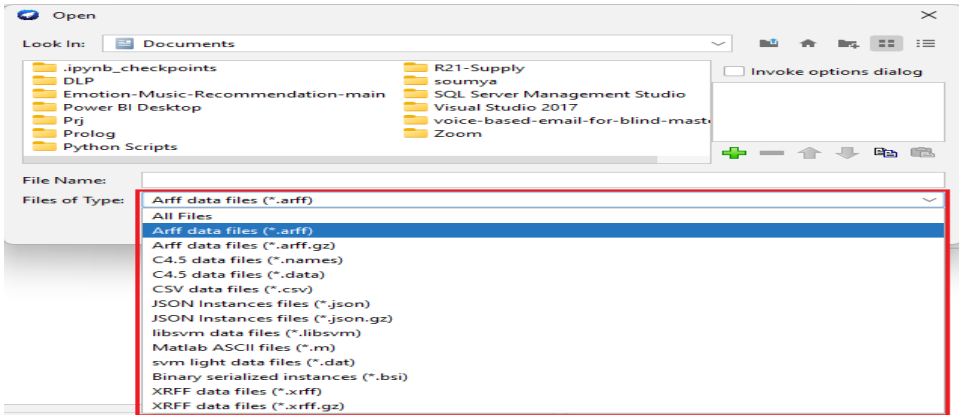
In order to work with Weka Explorer, it is essential to **load the data** into the application. The data may in different formats such as CSV, Text, JSON and so on.

WEKA supports a **wide range of file formats** to load the data.

The following is the complete list of file formats

- ☛ Arff data files(*.arff)
- ☛ Arff data files(*arff.gz)
- ☛ C4.5 data files(*.names)
- ☛ C4.5 data files(*.data)
- ☛ CSV data files(*.csv)
- ☛ JSON instance files(*.json)
- ☛ JSON instance files(*.json.gz)
- ☛ libsvm data files(*.libsvm)
- ☛ Matlab ASCII files(*.m)
- ☛ svm light data files(*.dat)
- ☛ Binary Serialized instances(*.bsi)
- ☛ XRFF data files(*.xrff)
- ☛ XRFF data files(*.xrff.gz)

The following screen displays all supported file formats in dropdown list at the bottom of window.



As we see the WEKA supports various formats of data to load, among those formats the most commonly used data formats are **Arff data files(*.arff)** and **CSV data files(*.csv)**.

NOTE : The default data format of WEKA is **Arff data files (*.arff)**.

Study the 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.

The ARFF file format has mainly **two sections**, those are

• **Header** section

• **Data** section

Header section:

The Header section of the ARFF file contains the **name of the relation**, a **list of the attributes** and their **types**.

@RELATION Declaration

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

format:

@RELATION <relation-name>

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

@ATTRIBUTE Declaration

The attribute specifies name of the attribute along with type.

format:

@ATTRIBUTE <attribute-name> <datatype>

- where the <attribute-name> must start with an alphabet. The attribute name must be quoted if the name includes spaces.

Weka supports the following four datatypes:

1. Numeric attributes:

Numeric attributes can be real or integer numbers.

2. Nominal attributes:

Nominal values are defined by providing the possible values: { nominal-value1, nominal-value2, nominal-value3,… }

3. String attributes:

String attributes allow us to define attributes holding textual values.

4. Date attributes:

Date attribute defined as follows

@ATTRIBUTE <name> date [<date-format>]

- where <name> is the name for the attribute and <date-format> is an optional string. The default date-format string is yyyy-MM-dd'T'HH:mm:ss.

Example of Header Section:

```
% Title: Student Database
%
% Sources:
% (a) Creator: Mr.T.M
% (b) Date: Oct, 2023
%

@RELATION student

@ATTRIBUTE sid NUMERIC
@ATTRIBUTE age NUMERIC
@ATTRIBUTE gender {male, female}
```

In the above example,

The lines which start with **%** are treated as comments.

@**RELATION** specifies the name of the relation.

@**ATTRIBUTE** specifies name of the attribute along with type and possible values.

Data section:

The Data section of the ARFF file contains the **list of data values (instance data)** separated by comma.

Example of Body Section:

```
@DATA
101,20,male
102,19,female
103,?,male
```

In the above, there are 3 instances with numeric and nominal values. And the symbol ? indicates missing values.

✍**NOTE :** The **@RELATION, @ATTRIBUTE** and **@DATA** declarations are **case insensitive**. i.e @RELATION and @relation are treated as same in ARFF file format.

The following is the **complete ARFF file**

Filename: student. arff

```
% Title: student Dataset
%
% Sources:
%      (a) Creator: Mr.T.M
%      (b) Date: Oct, 2023
%

@RELATION student

@ATTRIBUTE sid NUMERIC
@ATTRIBUTE age NUMERIC
@ATTRIBUTE gender {male, female}

@DATA
101,20,male
102,19,female
103,21,male
```

Comments

Relation Name

Attributes

Data Instances

Experiment 3: Demonstration of creating a Student dataset (student.arff) using WEKA tool in Data Mining.

Aim:

Demonstration of creating a Student dataset (student.arff) using WEKA tool in Data Mining.

Solution :

Creating a student dataset (student.arff):

Description:

We need to create a Student Table with training data set which includes attributes like sid, name, various subject marks, total and result.

Procedure:

Steps:

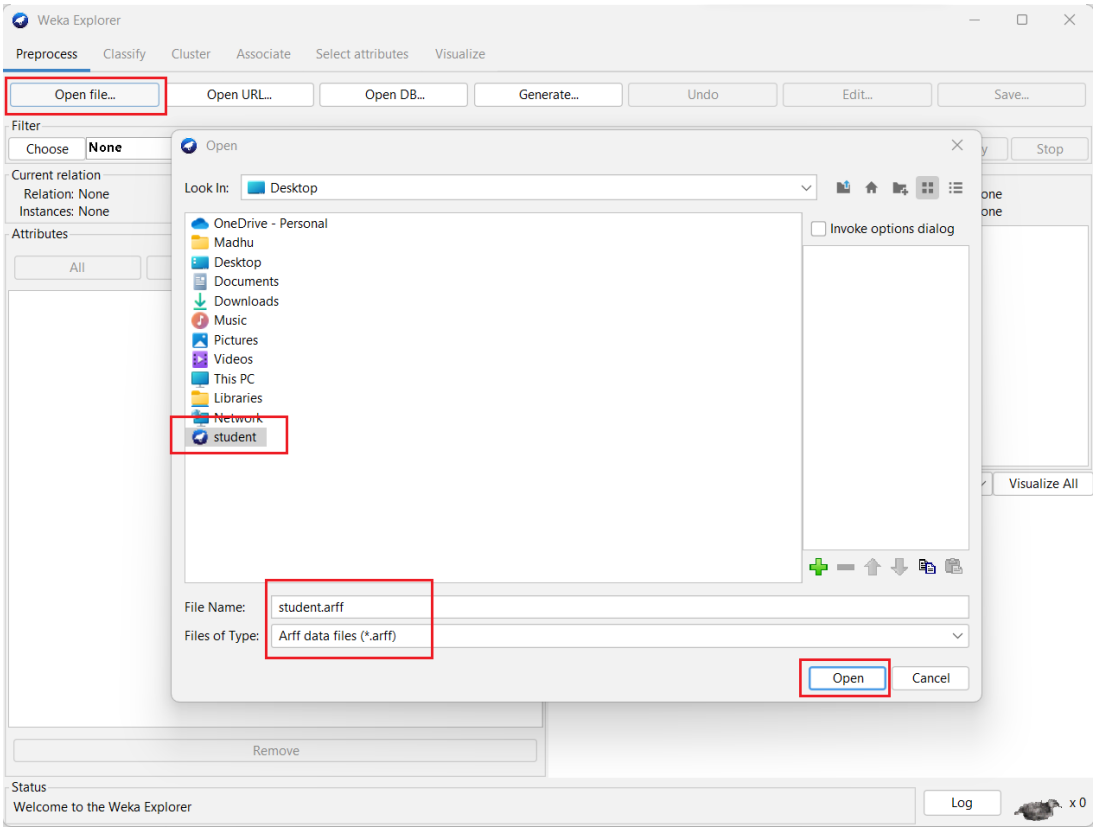
1) Open any **text editor** (e.g. Notepad)

2) Type the following training **data set** in the Notepad.

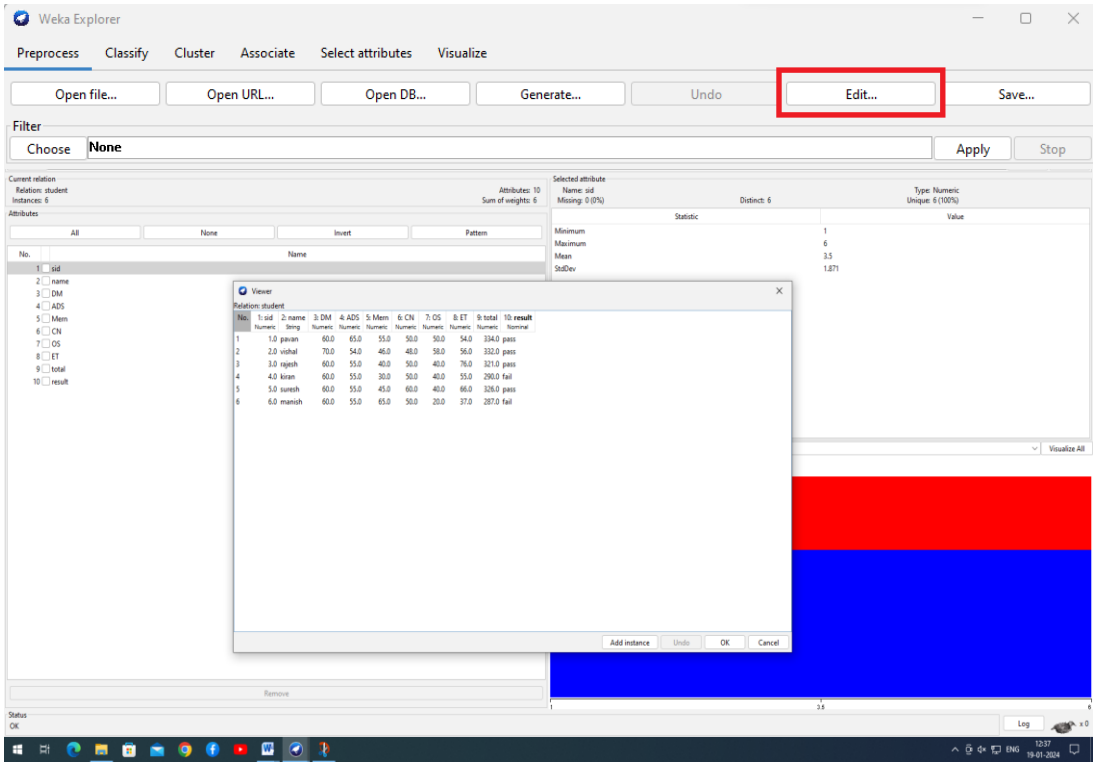
```
@relation student

@attribute sid numeric
@attribute name string
@attribute DM numeric
@attribute ADS numeric
@attribute MERN numeric
@attribute CN numeric
@attribute OS numeric
@attribute ET numeric
@attribute total numeric
@attribute result {pass, fail}

@data
1, pavan, 60, 65, 55, 50, 50, 54, 334, pass
2, vishal, 70, 54, 46, 48, 58, 56, 332, pass
3, rajesh, 60, 55, 40, 50, 40, 76, 321, pass
4, kiran, 60, 55, 30, 50, 40, 55, 290, fail
5, suresh, 60, 55, 45, 60, 40, 66, 326, pass
6, manish, 60, 55, 65, 50, 20, 37, 287, fail
```

3) After that save the file with **.arff file** format.
4) Minimize the arff file and then open **Start → Programs → weka-3.8.6.**
5) Click on **weka-3.8.6**, then **Weka GUI chooser** is displayed on the screen.
6) In that **Weka GUI chooser** there are five applications, click on **Explorer**.

7) Explorer shows many options. In that click on **'Open file...'** button and select the **.arff file** (e.g. student.arff).



8) Click on **edit** button which shows student table on weka.



Result:

The **student dataset** (student.arff) was created successfully using WEKA tool kit.

Experiment 4:

Perform data preprocessing tasks

Demonstration of preprocessing on dataset labor.arff

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

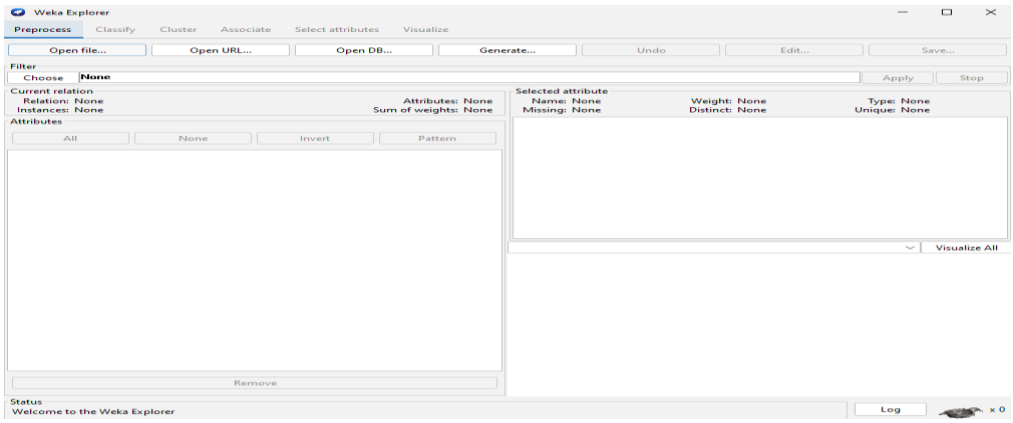
1. All. All boxes are ticked.

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

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

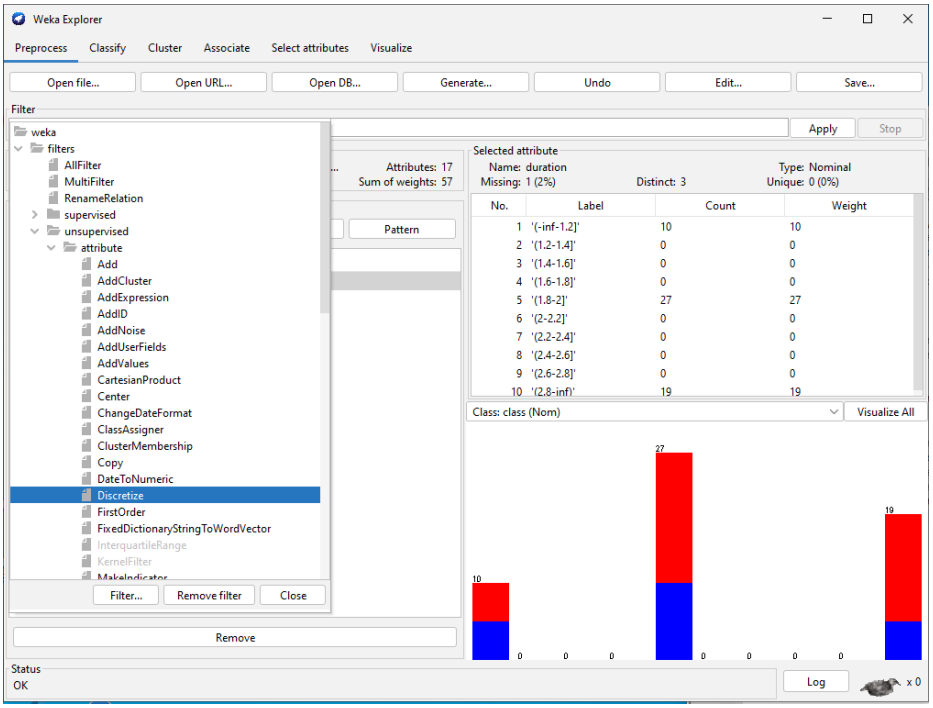
4. Pattern. Enables the user to select attributes based on a Perl 5 Regular Expression. E.g., .* id selects all attributes which name ends with id.

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 clusters 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.

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.

Steps for run preprocessing tab in WEKA

1. Open WEKA Tool.

2. Click on WEKA Explorer.

3. Click on Preprocessing tab button.

4. Click on open file button.

5. Choose WEKA folder in C drive.

6. Select and Click on data option button.

7. Choose labor data set and open file.

8. Choose filter button and select the Unsupervised-Discritize option and apply

Demonstration of preprocessing on dataset labor.arff

Aim: This experiment illustrates some of the basic data preprocessing operations that can be performed using WEKA-Explorer.

The sample dataset used for this example is the labor data available in arff format.

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

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

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

Step4: The visualization in the right button panel in the form of cross-tabulation across two attributes. Note:we can select another attribute using the dropdown list.

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

Step 6:

a)Next click the textbox immediately to the right of the choose button.In the resulting dialog box enter the index of the attribute to be filtered out.
b) Make sure that invert selection option is set to false.The click OK now in the filter box.you will see “Remove-R-7”.
c)Click the apply button to apply filter to this data.This will remove the attribute and create new working relation.
d)Save the new working relation as an arff file by clicking save button on the top(button)panel.(labor.arff)

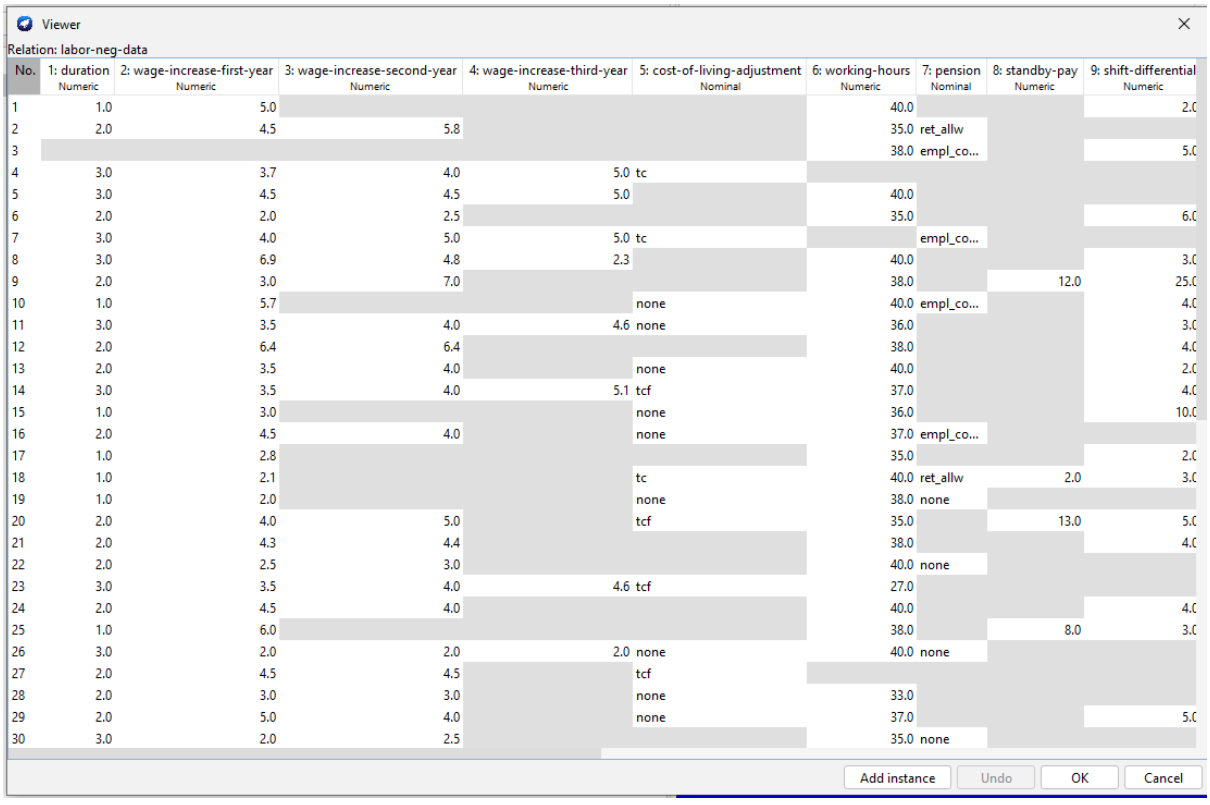
Discretization

1)Sometimes association rule mining can only be performed on categorical data. This requires performing discretization on numeric or continuous attributes.

In the following example let us discretize duration attribute.

- Let us divide the values of duration attribute into three bins(intervals).
- First load the dataset into weka(labor.arff)
- Select the duration attribute.
- Activate filter-dialog box and select “WEKA.filters.unsupervised.attribute.discretize”from the list.
- To change the defaults for the filters,click on the box immediately to the right of the choose button.
- We enter the index for the attribute to be discretized.In this case the attribute is duration So we must enter ‘1’ corresponding to the duration attribute.
- Enter ‘1’ as the number of bins.Leave the remaining field values as they are.
- Click OK button.
- Click apply in the filter panel.This will result in a new working relation with the selected attribute partition into 1 bin.
- Save the new working relation in a file called labor-data-discretized.arff

Dataset labor.arff

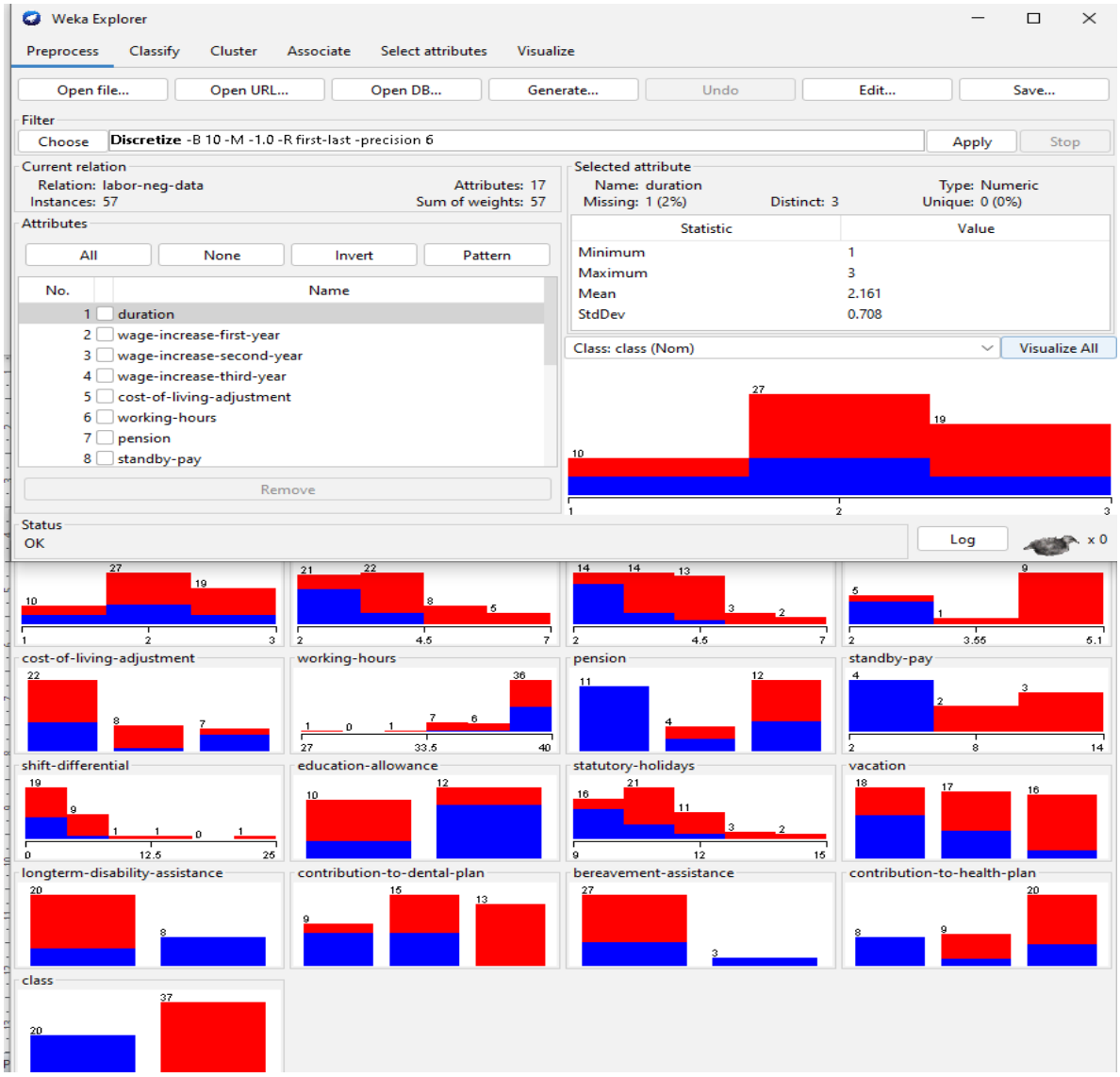


Viewer

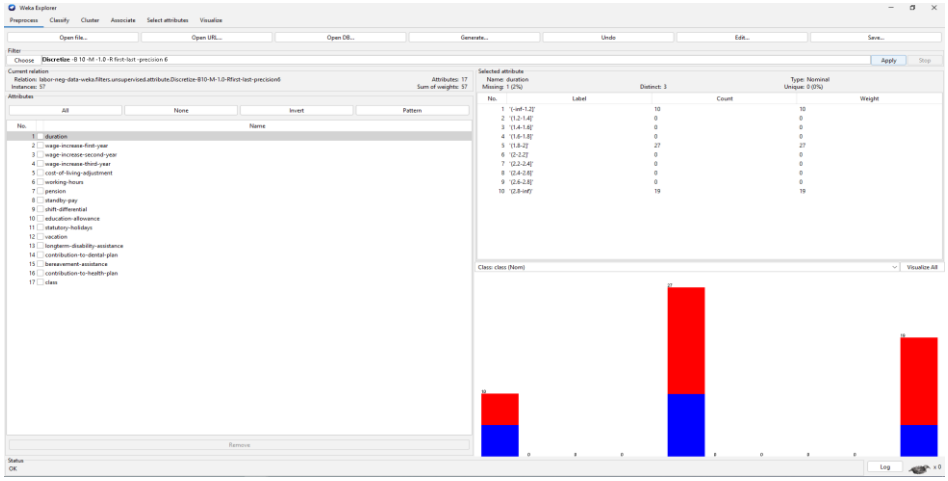
Relation: labor-neg-data

No.	1: duration Numeric	2: wage-increase-first-year Numeric	3: wage-increase-second-year Numeric	4: wage-increase-third-year Numeric	5: cost-of-living-adjustment Nominal	6: working-hours Numeric	7: pension Nominal	8: standby-pay Numeric	9: shift-differential Numeric
1	1.0	5.0				40.0			2.0
2	2.0	4.5	5.8			35.0	ret_allw		
3						38.0	empl_co...		5.0
4	3.0	3.7	4.0	5.0	tc				
5	3.0	4.5	4.5	5.0		40.0			
6	2.0	2.0	2.5			35.0			6.0
7	3.0	4.0	5.0	5.0	tc		empl_co...		
8	3.0	6.9	4.8	2.3		40.0			3.0
9	2.0	3.0	7.0			38.0		12.0	25.0
10	1.0	5.7			none	40.0	empl_co...		4.0
11	3.0	3.5	4.0	4.6	none	36.0			3.0
12	2.0	6.4	6.4			38.0			4.0
13	2.0	3.5	4.0		none	40.0			2.0
14	3.0	3.5	4.0	5.1	tcf	37.0			4.0
15	1.0	3.0			none	36.0			10.0
16	2.0	4.5	4.0		none	37.0	empl_co...		
17	1.0	2.8				35.0			2.0
18	1.0	2.1			tc	40.0	ret_allw	2.0	3.0
19	1.0	2.0			none	38.0	none		
20	2.0	4.0	5.0		tcf	35.0		13.0	5.0
21	2.0	4.3	4.4			38.0			4.0
22	2.0	2.5	3.0			40.0	none		
23	3.0	3.5	4.0	4.6	tcf	27.0			
24	2.0	4.5	4.0			40.0			4.0
25	1.0	6.0				38.0		8.0	3.0
26	3.0	2.0	2.0	2.0	none	40.0	none		
27	2.0	4.5	4.5		tcf				
28	2.0	3.0	3.0		none	33.0			
29	2.0	5.0	4.0		none	37.0			5.0
30	3.0	2.0	2.5			35.0	none		

Add instance | Undo | OK | Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
Discretize -B 10 -M -1.0 -R first-last -precision 6
Apply
Stop
Current relation
Relation: labor-neg-data
Instances: 57
Attributes: 17
Sum of weights: 57
Selected attribute
Name: duration
Missing: 1 (2%)
Distinct: 3
Type: Numeric
Unique: 0 (0%)
Statistic
Value
Minimum 1
Maximum 3
Mean 2.161
StdDev 0.708
Attributes
All
None
Invert
Pattern
No.
Name
1 duration
2 wage-increase-first-year
3 wage-increase-second-year
4 wage-increase-third-year
5 cost-of-living-adjustment
6 working-hours
7 pension
8 standby-pay
Remove
Class: class (Nom)
Visualize All
Status
OK
Log
x 0
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

The following screenshot shows the effect of discretization



5. How to handle missing values using dataset diabetes.arff

Data is rarely clean and often you can have corrupt or missing values.

It is important to identify, mark and handle missing data when developing machine learning models in order to get the very best performance.

1) How to mark missing values from the data set?
2) How to delete records with missing values from the data set?
3) How to assign missing values?

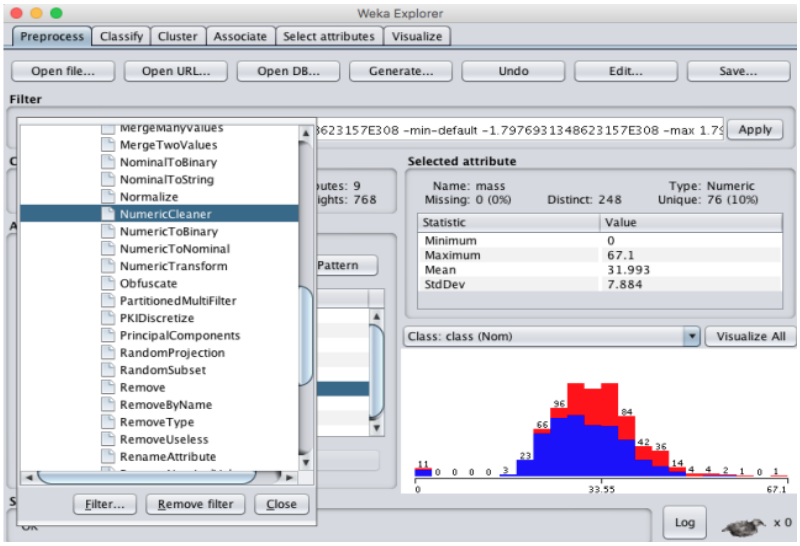
1) How to mark missing values from the data set?

The diabetes.arff is a good example of searching for missing data. Some attributes such as blood pressure (pres) and Body Mass Index (mass) have zero values that are biologically impossible. These are examples of damaged or missing data that must be marked manually. The WEKA allows you to mark missing values using the NumericalCleaner filter. The following instructions show how to use this filter, mark the 11 missing values in the Body Mass Index attribute (mass).

1. Start Weka and choose Explorer

2. Load diabetes.arff data set.

3. Click “Choose” in tab Preprocess -> Filter and choose NumericalCleaner, which you will find in unsupervized.attribute.NumericalCleaner.

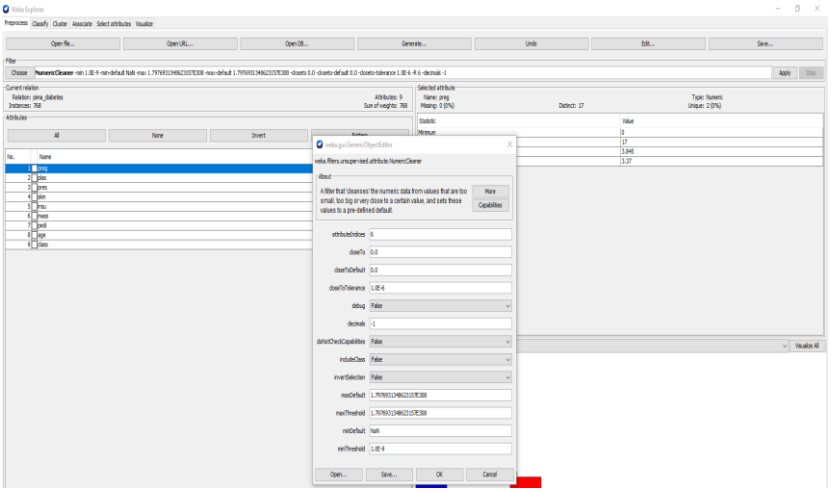


4. Select filter in order to move to its configuration

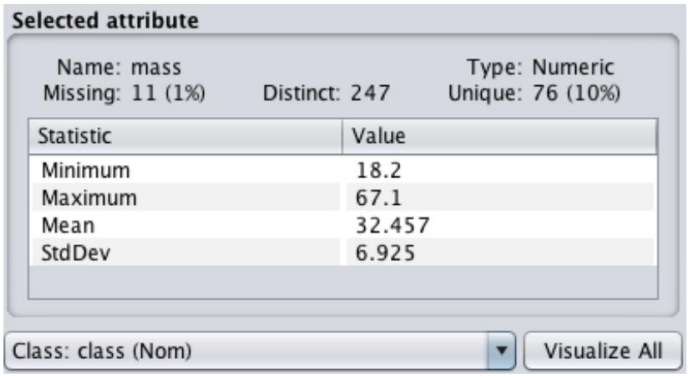
5. Set attributeIndicies to 6 (index for attribute “mass”)

6. Set minThreshold to be 0.1E-8 (close to zero), which is the minimum possible value of this attribute.

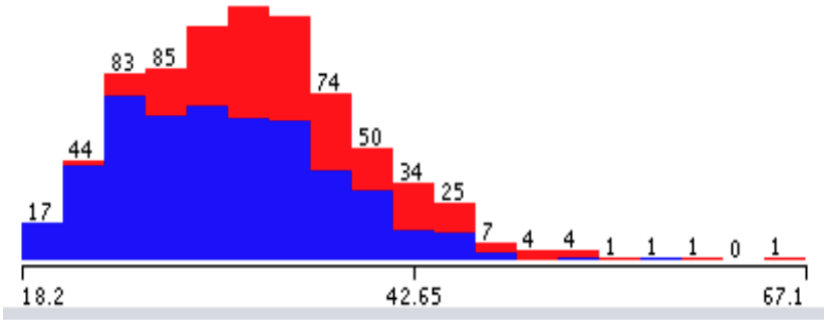
7. Set minDefault to be NaN, which represents an unknown value and will replace all values below the threshold value, and click "OK" and then “Apply” in order to apply the filter to the dataset at hand.



Select the "mass" attribute in the attribute panel and familiarize yourself with the details of the selected attribute. Note that the 11 attribute values that have been set to 0 are now marked as missing.



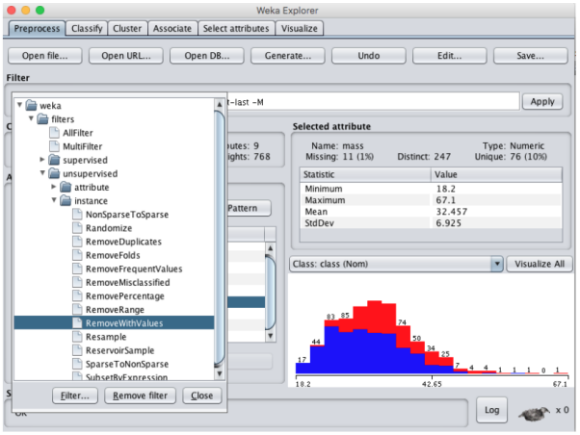
In this example, we have marked values below the threshold value as missing. You can also easily mark them by assigning a different numeric value. You can also mark values in the numerical range as missing, i.e. when they are between the upper and lower threshold value.



II. How to delete records with missing values from the data set?

A simple way to deal with missing data is to delete those occurrences that have one or more missing values. In Weka this can be done using the RemoveWithValues filter. Following with the example above, you can delete the missing values as follows:

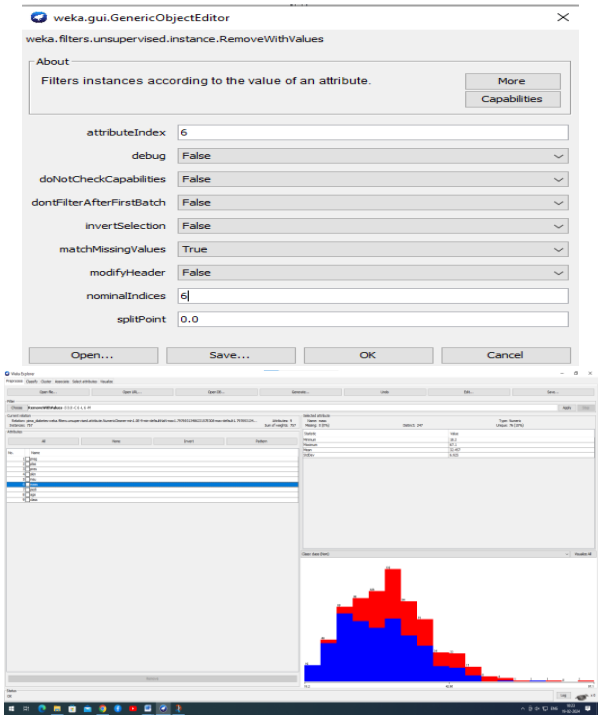
1. In the Filter selection window, select the RemoveWithValues filter and click „Choose” button.



2. Select a filter to configure it.
3. Set attributeIndicies to 6 (index for attribute “mass”)
4. Set matchMissingValues to True
5. Press "OK", in order to use filter configuration.
6. Press "Apply", in order to use selected filter.

Select the "mass" attribute from the attributes section and view the details of the selected attribute.

Please note that the 11 attribute values that were marked as "Missing" in the previous step have now been removed from the data set.



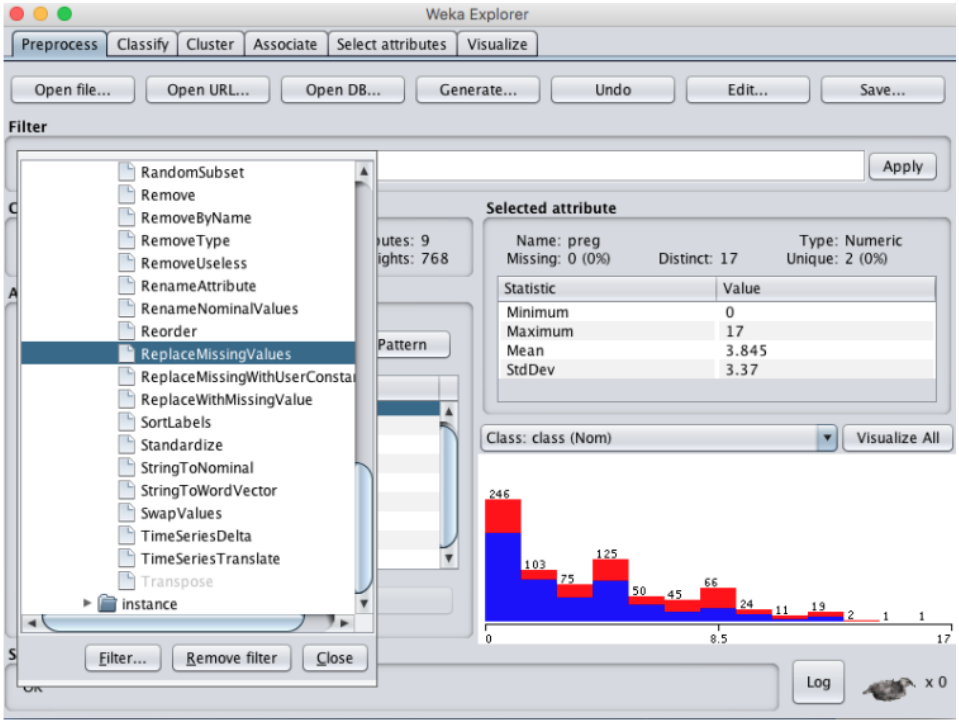
III. How to assign missing values?

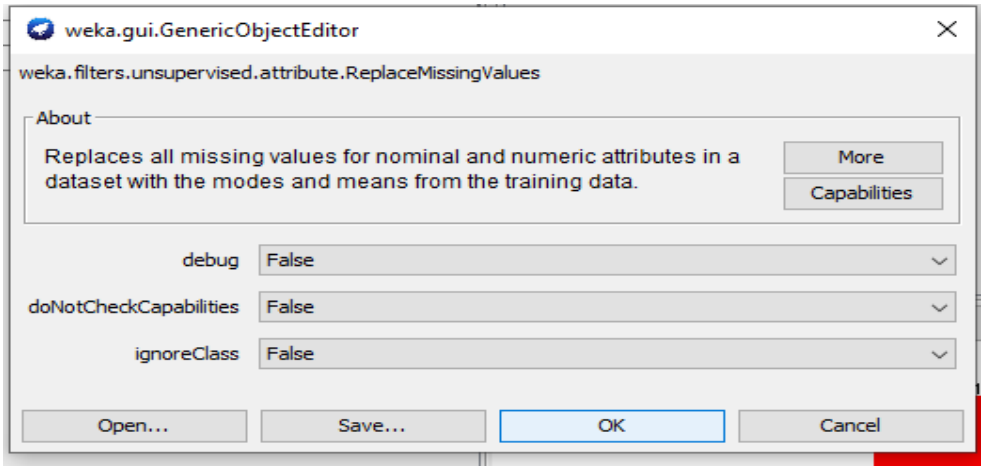
However, records with missing values do not have to be deleted, and it is possible to replace the missing values with some other predetermined value. Such solution to the problem of missing values is called an assignment.

The frequently used method of assigning missing values uses the average values of the distribution of a given variable. This can be easily done in Weka using the ReplaceMissingValues/ ReplaceMissingWithUserConstant filter.

Following with the first example, you can assign the missing values in the following way:

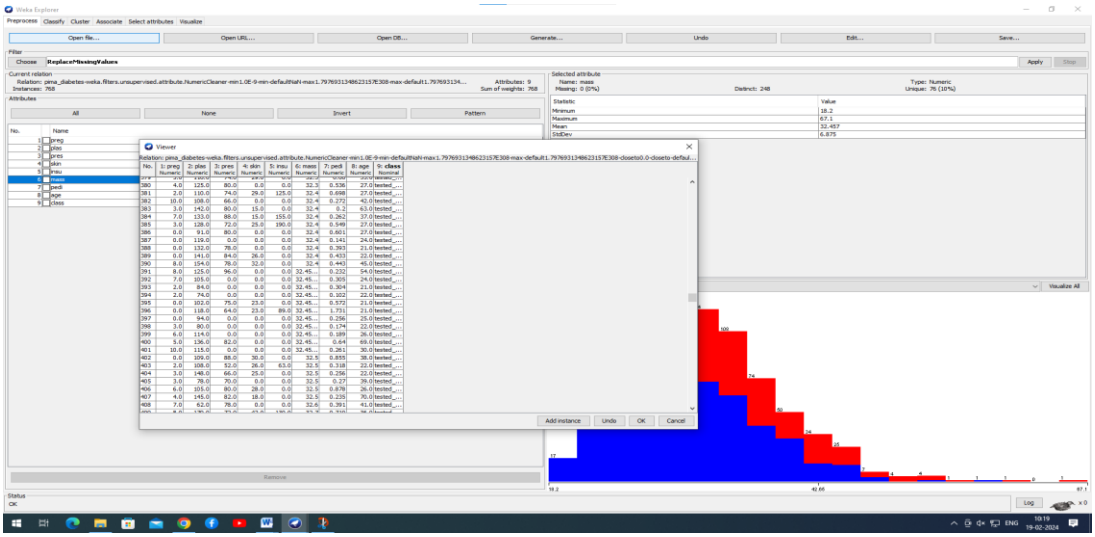
1. In the filter selection window, select ReplaceMissingValues filter and click "Choose".



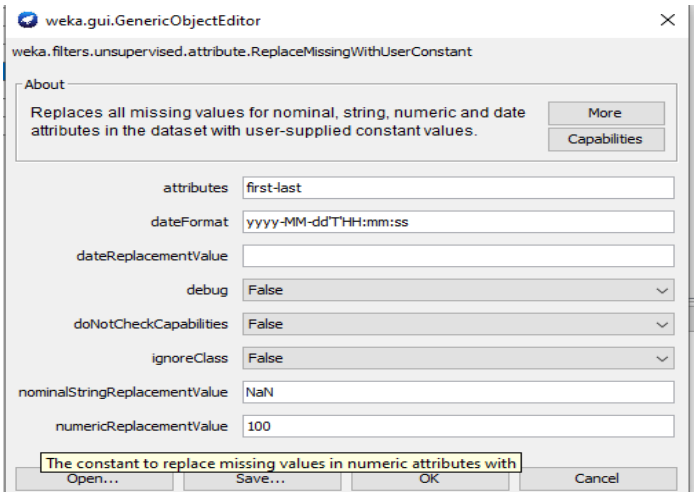


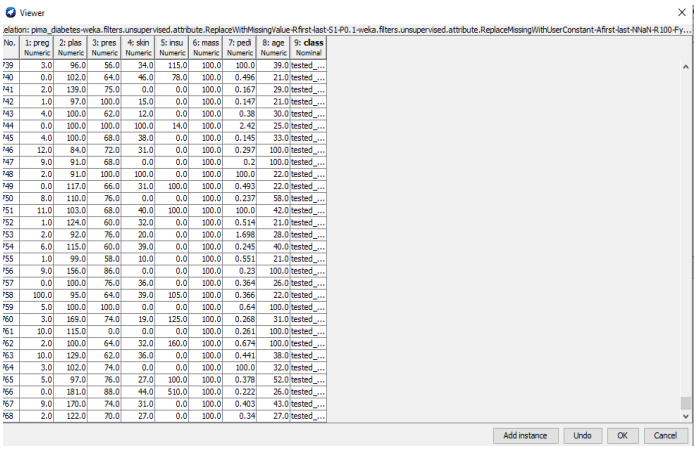
2. Click 'Apply' to apply the filter to the data set.

Click on the "mass" attribute in the attributes section and view the details of the selected attribute. Note that the 11 attribute values, which in the first example were marked as "Missing", were set to the average values of the distribution.



ReplaceMissingWithUserConstant





6. Load wheather.numeric dataset into weka and run Aprior algorithm..

Association Rule:

An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

Association rules are created by analyzing data for frequent if/then patterns and using the criteria support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database. Confidence indicates the number of times the if/then statements have been found to be true.

In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in shopping basket data analysis, product clustering, catalog design and store layout.

Support and Confidence values:

Support count: The support count of an itemset X, denoted by X.count, in a data set T is the number of transactions in T that contain X. Assume T has n transactions.

Then,

$$\text{support} = \frac{(X \cup Y).\text{count}}{n}$$

$$\text{confidence} = \frac{(X \cup Y).\text{count}}{X.\text{count}}$$

support = support({A U C})

confidence = support({A U C})/support({A})

Steps for run Aprior algorithm in WEKA

1. Open WEKA Tool.

2. Click on WEKA Explorer.

3. Click on Preprocessing tab button.

4. Click on open file button.

5. Choose WEKA folder in C drive.

6. Select and Click on data option button.

7. Choose Weather data set and open file.

8. Click on Associate tab and Choose Aprior algorithm

9. Click on start button.

Output :

=== Run information ===

Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c - 1

Relation: weather.symbolic

Instances: 14
Attributes: 5
outlook
temperature
humidity
windy
play

=== Associator model (full training set) === Apriori =======

Minimum support: 0.15 (2 instances)
Minimum metric : 0.9
Number of cycles performed: 17

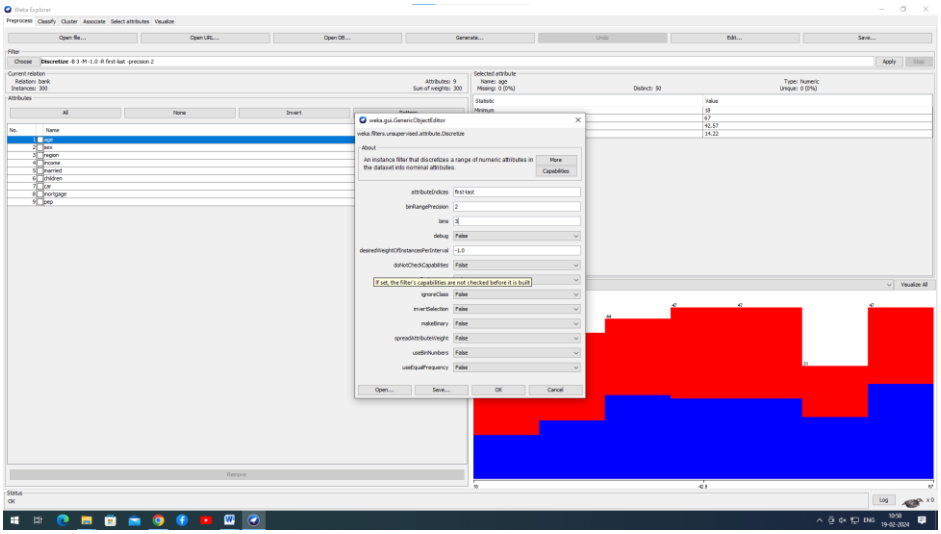
Generated sets of large itemsets:
Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

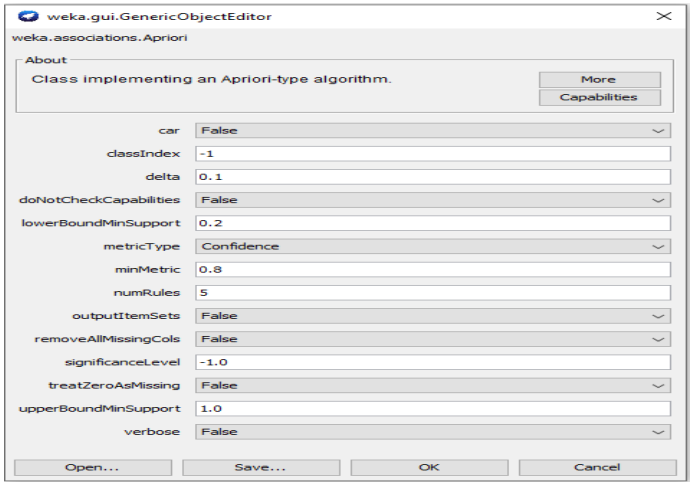
Best rules found:

1. outlook=overcast 4 ==> play=yes 4 conf:(1)
2. temperature=cool 4 ==> humidity=normal 4 conf:(1)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 conf:(1)
4. outlook=sunny play=no 3 ==> humidity=high 3 conf:(1)
5. outlook=sunny humidity=high 3 ==> play=no 3 conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 conf:(1)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 conf:(1)
8. temperature=cool play=yes 3 ==> humidity=normal 3 conf:(1)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 conf:(1)

7. Generate Association rules using apriori algorithm with bank.arff data set. Set minimum support range as 20% to 100% with incremental decrease factor as 10% as confidence factor as 80% generate 5 rules.

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. In order to apply apriori algorithm we need to discretize the attribute.
5. Let us divide attributes into three bins(intervals).
6. first load the dataset into weka(bank.arff)
7. Activate filter-dialog box and select “WEKA.filters.unsupervised.attribute.discretize”from the list.
8. To change the defaults for the filters,click on the box immediately to the right of the choose button.
9. We enter the index for the attribute ‘first-last’ to be discretized.
10. Enter ‘3’ as the number of bins.Leave the remaining field values as they are.
11. Click OK button.
12. Click apply in the filter panel.
13. This will result in a new working relation with the attributes partition into 3 bin.
14. Save the new working relation in a file called bank-discretized.arff
15. Click on Associate tab and Choose Aprior algorithm
16. Click on start button.





Output :

=== Run information ===

Scheme: weka.associations.Apriori -N 5 -T 0 -C 0.8 -D 0.1 -U 1.0 -M 0.2 -S -1.0 -c -1
Relation: bank-weka.filters.unsupervised.attribute.Discretize-B3-M-1.0-Rfirst-last-precision2-weka.filters.unsupervised.attribute.Discretize-B3-M-1.0-Rfirst-last-precision2
Instances: 300
Attributes: 9
age
sex
region
income
married
children
car
mortgage
pep
=== Associator model (full training set) ===

Apriori
=======

Minimum support: 0.2 (60 instances)
Minimum metric <confidence>: 0.8
Number of cycles performed: 9

Generated sets of large itemsets:

Size of set of large itemsets L(1): 19

Size of set of large itemsets L(2): 76

Size of set of large itemsets L(3): 17

Best rules found:

1. age='(-inf-34.33]' 95 ==> income='(-inf-24386.17]' 84 <conf:(0.88)> lift:(1.83) lev:(0.13) [38] conv:(4.09)
2. mortgage=NO pep=NO 100 ==> married=YES 83 <conf:(0.83)> lift:(1.23) lev:(0.05) [15] conv:(1.81)

8. Demonstrate performing classification on data sets.

Classification Tab

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 thatare available in WEKA.

i. **Use training set.** The classifier is evaluated on how well it predicts the class of the instances it was trained on.
ii. **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.
iii. **Cross-validation.** The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.
iv. **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.

Classifier Evaluation Options:

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 theoutput. This option is not selected by default.
4. Output confusion matrix. The confusion matrix of the classifier's predictions isincluded in the output. This option is selected by default.
5. Store predictions for visualization. The **classifier's predictions are** remembered sothat 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.**
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 tobring up a drop-down list of attributes to choose from.

Training a Classifier

Once the classifier, test options and class have all been set, the learning process is started byclicking 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.

Load IRIS dataset into Weka and run j48 classification algorithm, study the classifier output. Compute entropy values, Kappa statistics.

Ans:

Steps for running J48 Classification algorithms in WEKA

Open WEKA Tool.

1. Click on WEKA Explorer.
2. Click on Preprocessing tab button.
3. Click on open file button.
4. Choose WEKA folder in C drive.
5. Select and Click on data option button.
6. Choose iris data set and open file.
7. Click on classify tab and Choose J48 algorithm and select use training set test option.
8. Click on start button.

Output:

=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2Relation:iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class

Test mode:evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|petalwidth > 1.7: Iris-irginica (46.0/1.0)

Number of Leaves : 5
Size of the tree : 9
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===

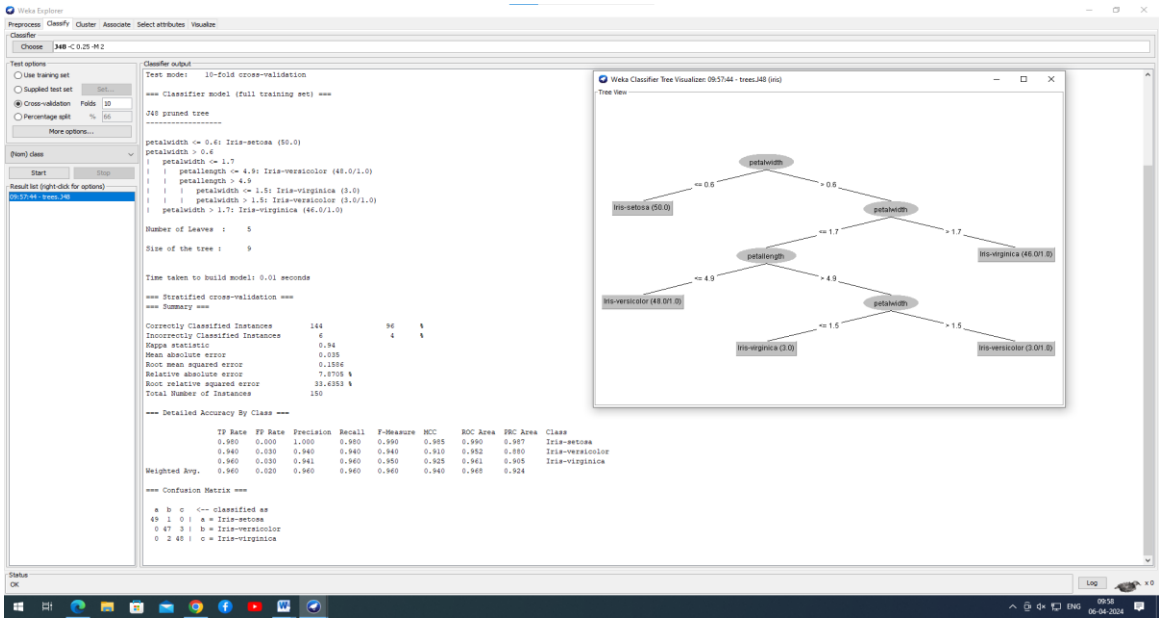
Correctly Classified Instances	147	98	%
Incorrectly Classified Instances	3	2	%
	0.97		
Kappa statistic			
K&B Relative Info Score	14376.1925 %		
K&B Information Score	227.8573 bits	1.519 bits/instance	
Class complexity | order 0	237.7444 bits	1.585bits/instance	
Class complexity | scheme	16.7179 bits	0.1115 bits/instance	
Complexity improvement (Sf)	221.0265 bits	1.4735 bits/instance	
Mean absolute error	0.0233		
Root mean squared error	0.108		
Relative absolute error	5.2482 %		
Root relative squared error	22.9089 %		
Total Number of Instances	150		

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0	1	1	1	1	Iris-setosa
0.98	0.02	0.961	0.98	0.97	0.99	Iris-versicolor
0.96	0.01	0.98	0.96	0.97	0.99	Iris-virginica
Weighted Avg.	0.98	0.01	0.98	0.98	0.98	0.993

=== Confusion Matrix ===

a	b	c	<-- classified as
50	0	0	| a = Iris-setosa
0	49	1	| b = Iris-versicolor
0	2	48	| c = Iris-virginica

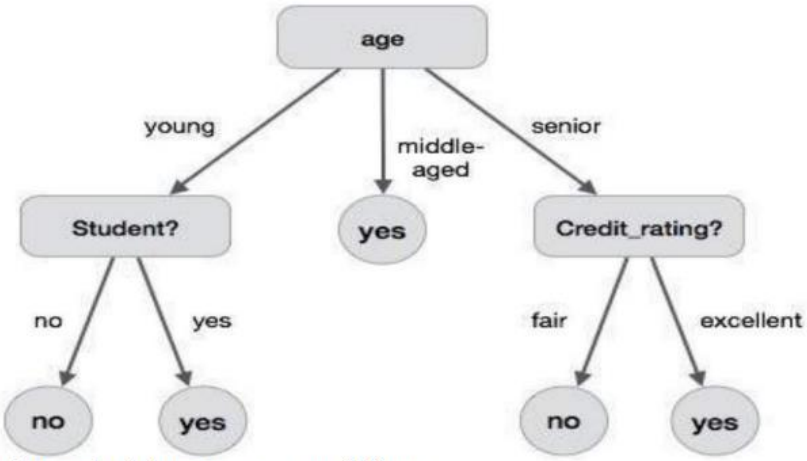

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) class
Start Stop
Result list (right-click for options)
09:57:44 - trees.J48
Classifier output
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)
Number of Leaves : 5
Size of the tree : 9
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 144 96 %
Incorrectly Classified Instances 6 4 %
Kappa statistic 0.94
Mean absolute error 0.035
Root mean squared error 0.1586
Relative absolute error 7.8705 %
Root relative squared error 33.6353 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.980 0.000 1.000 0.980 0.990 0.985 0.990 0.987 Iris-setosa
0.940 0.030 0.940 0.940 0.940 0.910 0.952 0.880 Iris-versicolor
0.960 0.030 0.941 0.960 0.950 0.925 0.961 0.905 Iris-virginica
Weighted Avg. 0.960 0.020 0.960 0.960 0.960 0.940 0.968 0.924
=== Confusion Matrix ===
a b c <-- classified as
49 1 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 2 48 | c = Iris-virginica
Weka Classifier Tree Visualizer: 09:57:44 - trees.J48 (iris)
Tree View
petalwidth
<= 0.6
> 0.6
Iris-setosa (50.0)
petalwidth
<= 1.7
> 1.7
petallength
Iris-virginica (46.0/1.0)
<= 4.9
> 4.9
Iris-versicolor (48.0/1.0)
petalwidth
<= 1.5
> 1.5
Iris-virginica (3.0)
Iris-versicolor (3.0/1.0)
Status
OK
Log
x 0
ENG
09:58
06-04-2024

9. Extract if-then rues from decision tree generated by classifier, Observe the confusion matrix and derive Accuracy, F- measure, TPrate, FPrate , Precision and recall values.

Ans:

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept buy_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class.



The benefits of having a decision tree are as follows −

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

IF-THEN Rules:

Rule-based classifier makes use of a set of IF-THEN rules for classification. We can express a rulein the following from −

IF condition THEN conclusion
Let us consider a rule R1,
R1: IF age=youth AND student=yes
THEN buy_computer=yes

Points to remember −

- The IF part of the rule is called **rule antecedent** or**precondition**.
- The THEN part of the rule is called **rule consequent**.
- The antecedent part the condition consist of one or more attribute tests and these tests arelogically ANDed.
- The consequent part consists of class prediction.

Note − We can also write rule R1 as follows:

R1: (age = youth) ^ (student = yes))(buys computer = yes)

If the condition holds true for a given tuple, then the

antecedent is satisfied.

Rule Extraction

Here we will learn how to build a rule-based classifier by extracting IF-THEN rules from adecision tree.

Points to remember −

- One rule is created for each path from the root to the leaf node.
- To form a rule antecedent, each splitting criterion is logically ANDed.
- The leaf node holds the class prediction, forming the rule consequent.

Rule Induction Using Sequential Covering Algorithm

Sequential Covering Algorithm can be used to extract IF-THEN rules form the training data. We do not require to generate a decision tree first. In this algorithm, each rule for a given class covers many of the tuples of that class.

Some of the sequential Covering Algorithms are AQ, CN2, and RIPPER. As per the general strategy the rules are learned one at a time. For each time rules are learned, a tuple covered by the rule is removed and the process continues for the rest of the tuples. This is because the path to each leaf in a decision tree corresponds to a rule.

Note − The Decision tree induction can be considered as learning a set of rules simultaneously.

The Following is the sequential learning Algorithm where rules are learned for one class at a time. When learning a rule from a class Ci, we want the rule to cover all the tuples from class C onlyand no tuple form any other class.

Algorithm: Sequential Covering

Input:
D, a data set class-labeled tuples,
Att_vals, the set of all attributes and their possible values.
Output: A Set of IF-THEN rules.
Method:
Rule_set={ }; // initial set of rules learned is empty
for each class c do
repeat
Rule = Learn_One_Rule(D, Att_valls, c);
remove tuples covered by Rule form D;
until termination condition;
Rule_set=Rule_set+Rule; // add a new rule to rule-set
end for
return Rule_Set;

- Steps for run decision tree algorithms in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on classify tab and Choose decision table algorithm and select cross-validation folds value-10 test option.
9. Click on start button.

=== Run information ===

Scheme: weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

Decision Table:

Number of training instances: 150
Number of Rules : 3

Non matches covered by Majority class.
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 12
Merit of best subset found: 96
Evaluation (for feature selection): CV (leave one out)
Feature set: 4,5

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

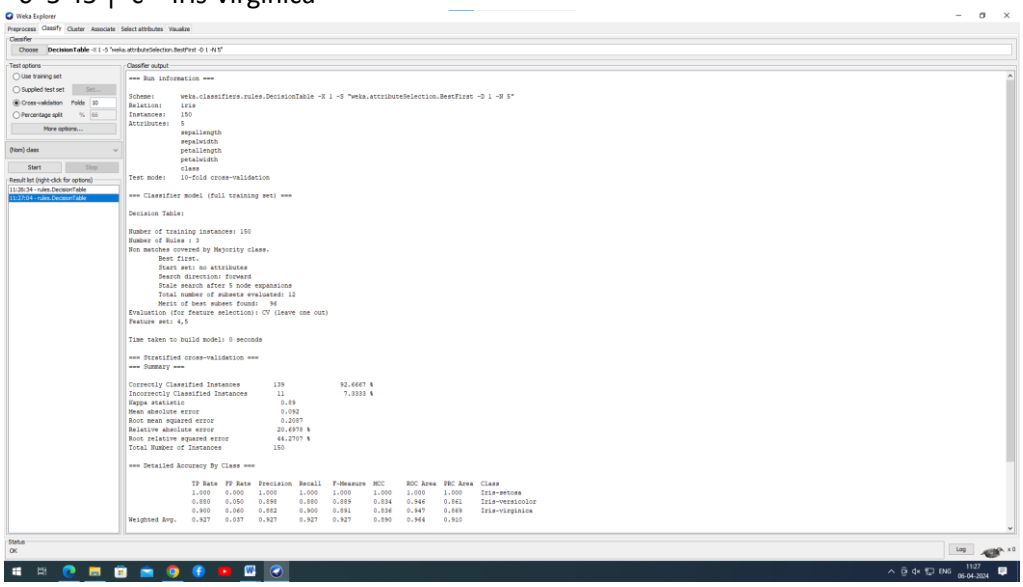
Correctly Classified Instances	139	92.6667 %
Incorrectly Classified Instances	11	7.3333 %
Kappa statistic	0.89	
Mean absolute error	0.092	
Root mean squared error	0.2087	
Relative absolute error	20.6978 %	
Root relative squared error	44.2707 %	
Total Number of Instances	150	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
	0.880	0.050	0.898	0.880	0.889	0.834	0.946	0.861	Iris-versicolor
	0.900	0.060	0.882	0.900	0.891	0.836	0.947	0.869	Iris-virginica
Weighted Avg.	0.927	0.037	0.927	0.927	0.927	0.890	0.964	0.910	

=== Confusion Matrix ===

a	b	c	<-- classified as
50	0	0	a = Iris-setosa
0	44	6	b = Iris-versicolor
0	5	45	c = Iris-virginica



10. Load IRIS dataset into Weka and perform Naive-bayes classification and Interpret the results obtained.

➊ Steps for run Naïve-bayes and k-nearest neighbor Classification algorithms in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on classify tab and Choose Naïve-bayes algorithm and select use training set testoption.
9. Click on start button.

=== Run information ===

Scheme: weka.classifiers.bayes.NaiveBayes
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

	Class		
Attribute	Iris-setosa	Iris-versicolor	Iris-virginica
	(0.33)	(0.33)	(0.33)

==========

sepallength			
mean	4.9913	5.9379	6.5795

std. dev.	0.355	0.5042	0.6353
weight sum	50	50	50
precision	0.1059	0.1059	0.1059

sepalwidth

mean	3.4015	2.7687	2.9629
std. dev.	0.3925	0.3038	0.3088
weight sum	50	50	50
precision	0.1091	0.1091	0.1091

petallength

mean	1.4694	4.2452	5.5516
std. dev.	0.1782	0.4712	0.5529
weight sum	50	50	50
precision	0.1405	0.1405	0.1405

petalwidth

mean	0.2743	1.3097	2.0343
std. dev.	0.1096	0.1915	0.2646
weight sum	50	50	50
precision	0.1143	0.1143	0.1143

Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	144	96 %
Incorrectly Classified Instances	6	4 %
Kappa statistic	0.94	
Mean absolute error	0.0342	
Root mean squared error	0.155	
Relative absolute error	7.6997 %	
Root relative squared error	32.8794 %	
Total Number of Instances	150	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
	0.960	0.040	0.923	0.960	0.941	0.911	0.992	0.983	Iris-versicolor
	0.920	0.020	0.958	0.920	0.939	0.910	0.992	0.986	Iris-virginica
Weighted Avg.	0.960	0.020	0.960	0.960	0.960	0.940	0.994	0.989	

=== Confusion Matrix ===

a	b	c	<-- classified as
50	0	0	a = Iris-setosa
0	48	2	b = Iris-versicolor
0	4	46	c = Iris-virginica