COURSE OBJECTIVES:

1. Learn how to build a data warehouse and query it (using open source tools like Pentaho Data Integration Tool, Pentaho Business Analytics).
2. Learn to perform data mining tasks using a data mining toolkit (such as open source WEKA).
3. Understand the data sets and data preprocessing.
4. Demonstrate the working of algorithms for data mining tasks such association rule mining, classification, clustering and regression.
5. Exercise the data mining techniques with varied input values for different parameters.

COURSE OUTCOMES:

1. Ability to understand the various kinds of tools.
2. Demonstrate the classification, clustering and etc. in large data sets.
3. Ability to add mining algorithms as a component to the exiting tools.
4. Ability to apply mining techniques for realistic data.

DATA WAREHOUSING AND MINING LAB- INDEX

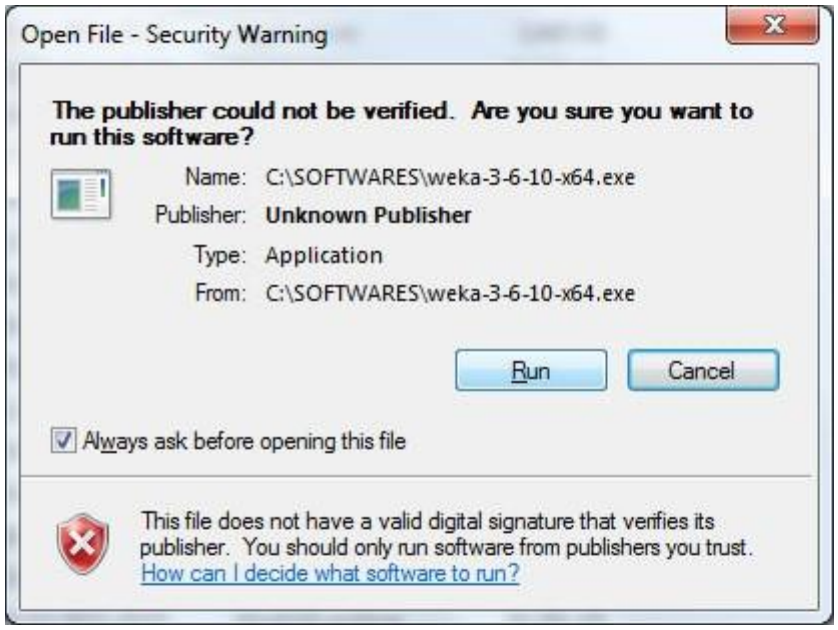
	each dataset, and deduce which classifier is performing best and poor for each dataset and justify.	
4	Demonstrate performing clustering on data sets • Load each dataset into Weka and run simple k-means clustering algorithm with different values of k (number of desired clusters). • Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights. • Explore other clustering techniques available in Weka. • Explore visualization features of Weka to visualize the clusters.	42-45
5	Write a java program to prepare a simulated data set with unique instances.	46
6	Write a Python program to generate frequent item sets / association rules using Apriori algorithm	47-48
7	Write a program to calculate chi-square value using Python. Report your observation.	49
8	Write a program of Naive Bayesian classification using Python programming language.	50
9	Write a program to cluster your choice of data using simple k-means algorithm using JDK	51-53
10	Write a program of cluster analysis using simple k-means algorithm Python programming language.	53-55
11	Write a program to compute/display dissimilarity matrix (for your own dataset containing at least four instances with two attributes) using Python	56
12	Visualize the datasets using matplotlib in python.(Histogram, Box plot, Bar chart, Pie chart etc.,)	57-58

1.Explore WEKA Data Mining/Machine Learning Toolkit

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

Ans: **Install Steps for WEKA a Data Mining Tool**

1. Download the software as your requirements from the below given link. http://www.cs.waikato.ac.nz/ml/weka/downloading.html
2. The Java is mandatory for installation of WEKA so if you have already Java on your machine then download only WEKA else download the software with JVM.
3. Then open the file location and double click on the file

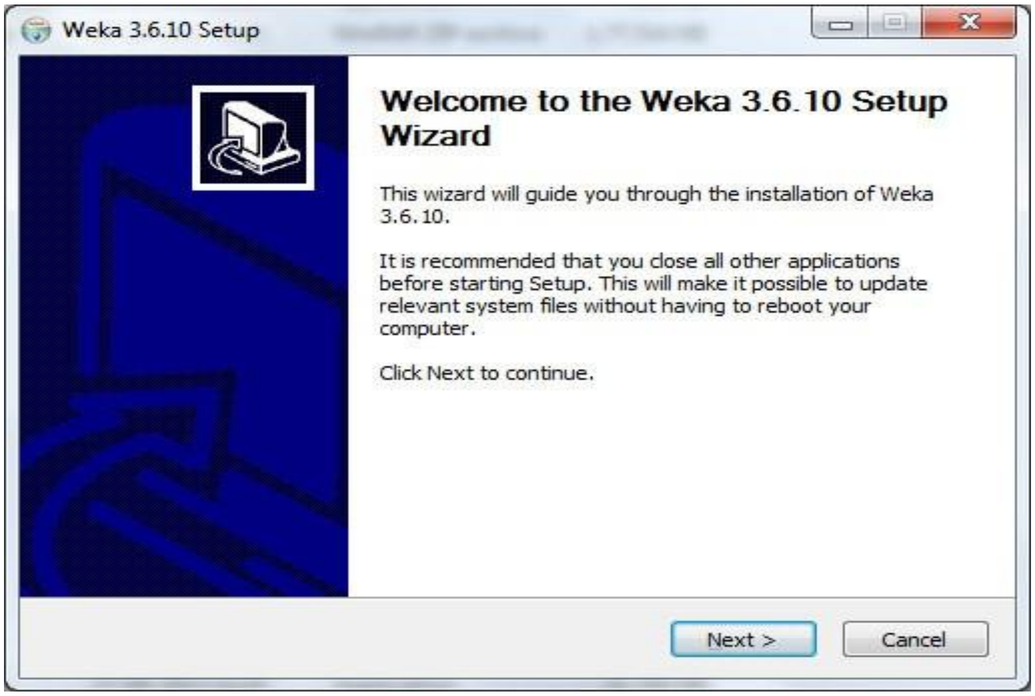

Open File - Security Warning

The publisher could not be verified. Are you sure you want to run this software?

Name: C:\SOFTWARES\weka-3-6-10-x64.exe
Publisher: Unknown Publisher
Type: Application
From: C:\SOFTWARES\weka-3-6-10-x64.exe

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
How can I decide what software to run?

4. Click Next



Weka 3.6.10 Setup

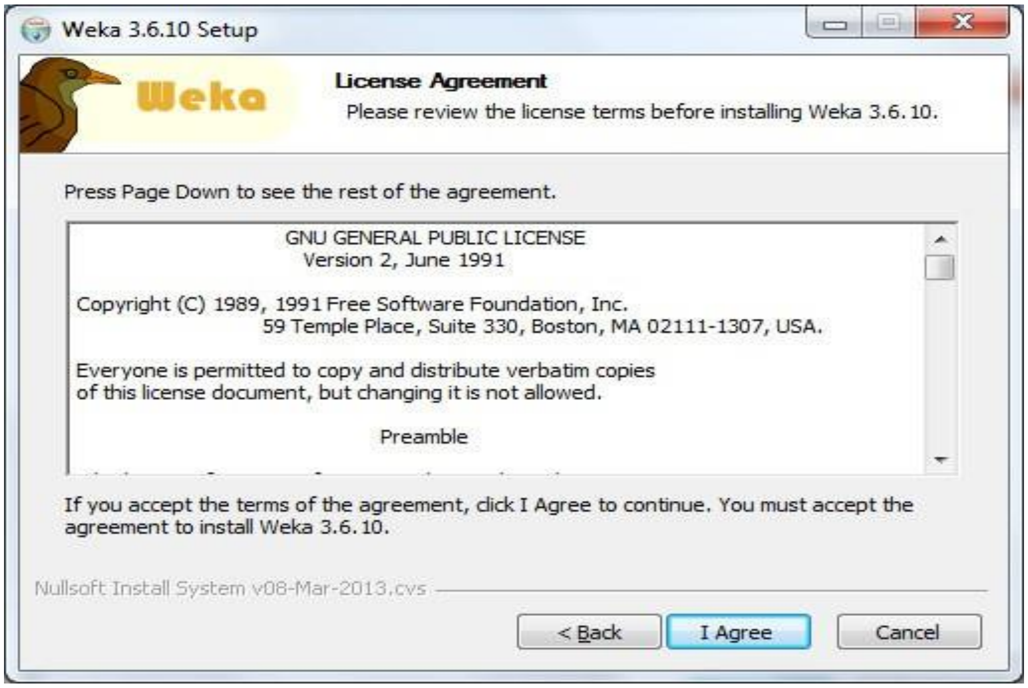
Welcome to the Weka 3.6.10 Setup Wizard

This wizard will guide you through the installation of Weka 3.6.10.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

5. Click I Agree.



Weka 3.6.10 Setup

License Agreement

Please review the license terms before installing Weka 3.6.10.

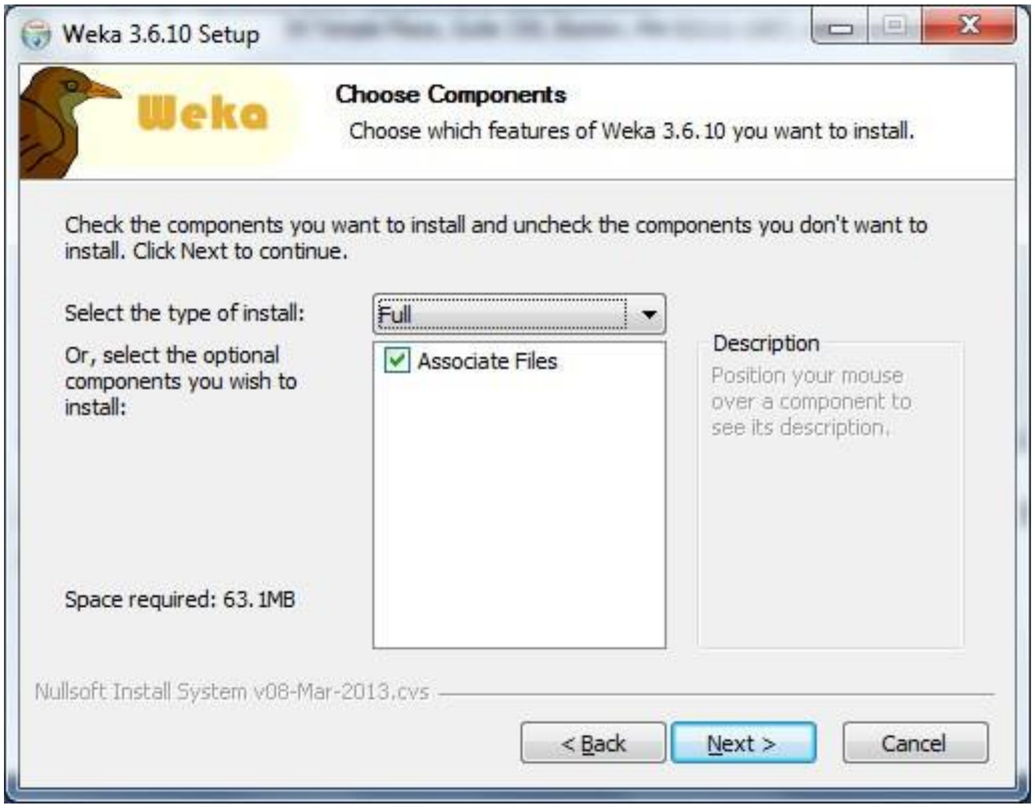
Press Page Down to see the rest of the agreement.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
               59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.

 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Weka 3.6.10.

Nullsoft Install System v08-Mar-2013.cvs

6. As your requirement do the necessary changes of settings and click Next. Full and Associate files are the recommended settings.



Weka 3.6.10 Setup

Choose Components

Choose which features of Weka 3.6.10 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Full

Or, select the optional components you wish to install: [✓] Associate Files

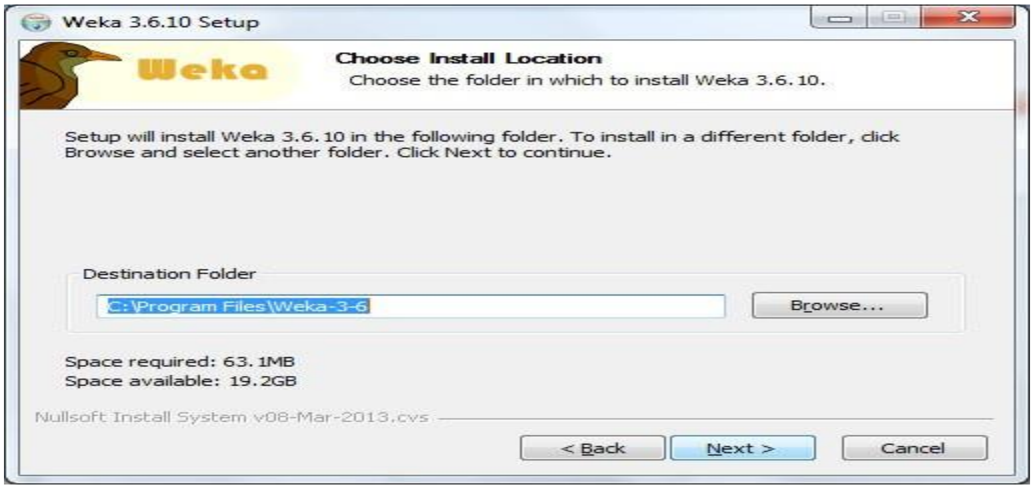
Description

Position your mouse over a component to see its description.

Space required: 63.1MB

Nullsoft Install System v08-Mar-2013.cvs

7. Change to your desire installation location.



Weka 3.6.10 Setup

Choose Install Location

Choose the folder in which to install Weka 3.6.10.

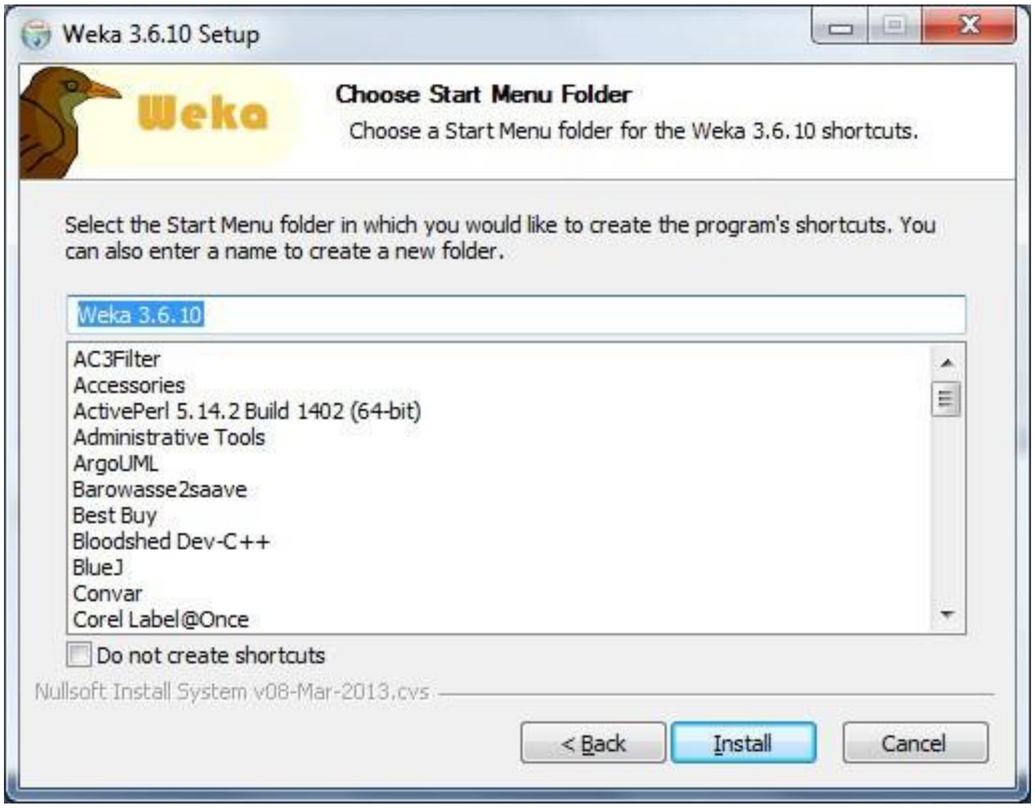
Setup will install Weka 3.6.10 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Program Files\Weka-3-6

Space required: 63.1MB
Space available: 19.2GB

Nullsoft Install System v08-Mar-2013.cvs

8. If you want a shortcut then check the box and click Install.



Choose Start Menu Folder

Choose a Start Menu folder for the Weka 3.6.10 shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

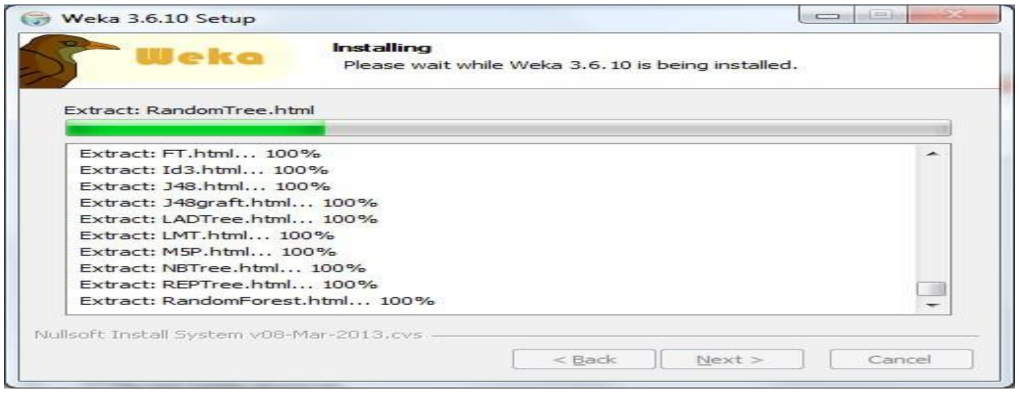
Weka 3.6.10

AC3Filter
Accessories
ActivePerl 5.14.2 Build 1402 (64-bit)
Administrative Tools
ArgoUML
Barowasse2saave
Best Buy
Bloodshed Dev-C++
BlueJ
Convar
Corel Label@Once

Do not create shortcuts

Nullsoft Install System v08-Mar-2013.cvs

9. The Installation will start wait for a while it will finish within a minute.



Installing

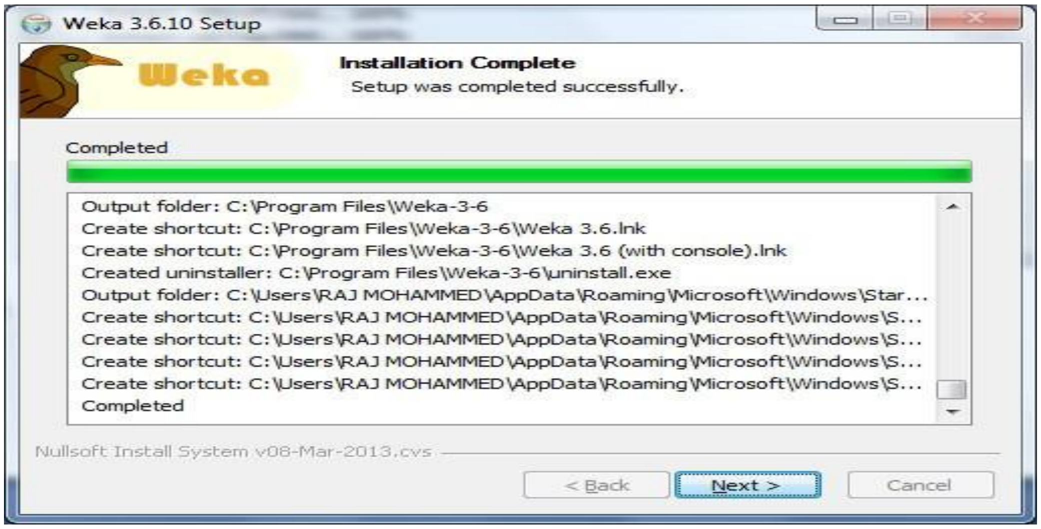
Please wait while Weka 3.6.10 is being installed.

```
Extract: RandomTree.html
```

```
Extract: FT.html... 100%
Extract: Id3.html... 100%
Extract: J48.html... 100%
Extract: J48graft.html... 100%
Extract: LADTree.html... 100%
Extract: LMT.html... 100%
Extract: M5P.html... 100%
Extract: NBTree.html... 100%
Extract: REPTree.html... 100%
Extract: RandomForest.html... 100%
```

Nullsoft Install System v08-Mar-2013.cvs

10. After complete installation click on Next.



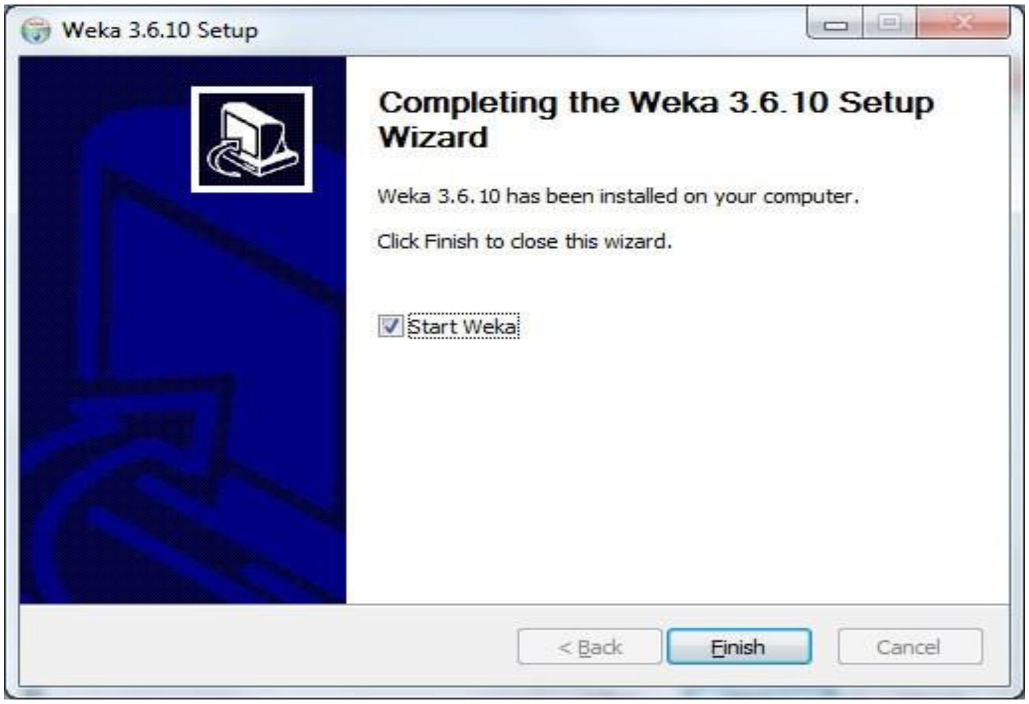
Installation Complete
Setup was completed successfully.

Completed

```
Output folder: C:\Program Files\Weka-3-6
Create shortcut: C:\Program Files\Weka-3-6\Weka 3.6.lnk
Create shortcut: C:\Program Files\Weka-3-6\Weka 3.6 (with console).lnk
Created uninstaller: C:\Program Files\Weka-3-6\uninstall.exe
Output folder: C:\Users\RAJ MOHAMMED\AppData\Roaming\Microsoft\Windows\Star...
Create shortcut: C:\Users\RAJ MOHAMMED\AppData\Roaming\Microsoft\Windows\S...
Create shortcut: C:\Users\RAJ MOHAMMED\AppData\Roaming\Microsoft\Windows\S...
Create shortcut: C:\Users\RAJ MOHAMMED\AppData\Roaming\Microsoft\Windows\S...
Create shortcut: C:\Users\RAJ MOHAMMED\AppData\Roaming\Microsoft\Windows\S...
Completed
```

Nullsoft Install System v08-Mar-2013.cvs

11. Hurray !!!!!!! That’s all click on the Finish and take a shovel and start Mining. Best of Luck.

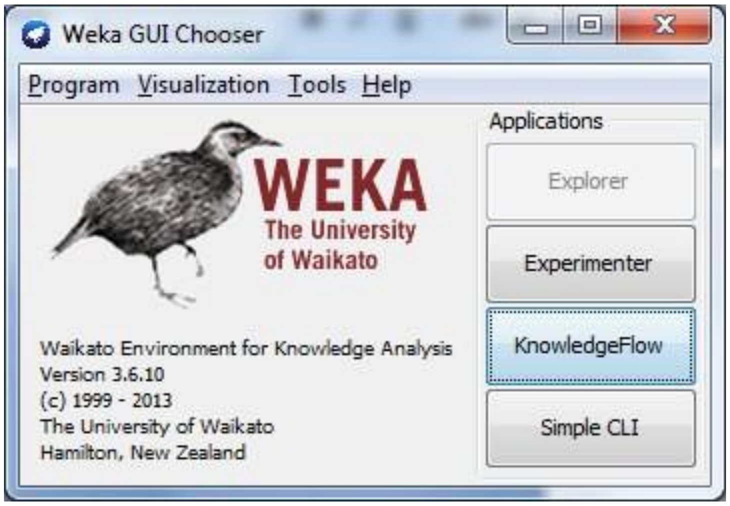


Completing the Weka 3.6.10 Setup Wizard

Weka 3.6.10 has been installed on your computer.

Click Finish to close this wizard.

[✓] Start Weka



Waikato Environment for Knowledge Analysis
Version 3.6.10
(c) 1999 - 2013
The University of Waikato
Hamilton, New Zealand

This is the GUI you get when started. You have 4 options Explorer, Experimenter, KnowledgeFlow and Simple CLI.

1.(iii)Understand the features of WEKA tool kit such as Explorer, Knowledge flow interface, Experimenter, command-line interface.

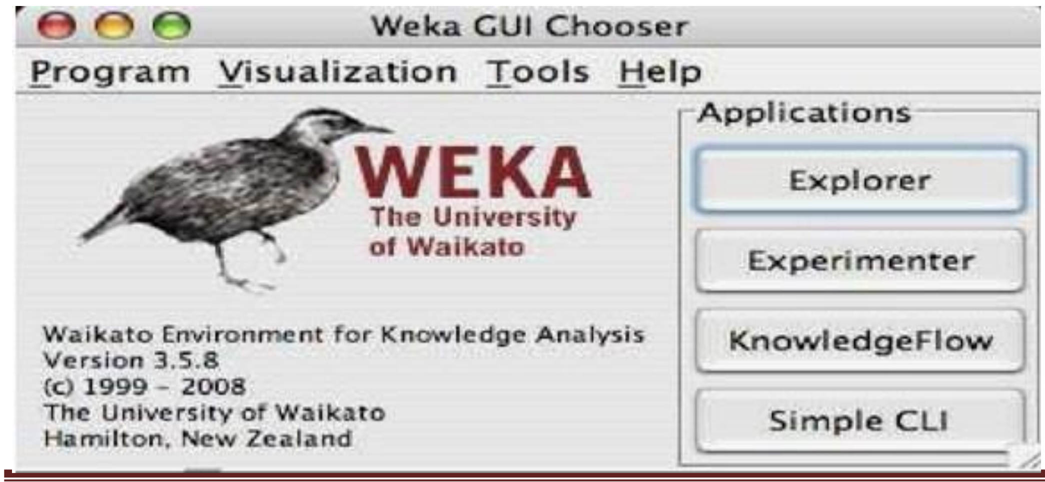
Ans: **<u>WEKA</u>**

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

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

The Graphical User Interface;-

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



Waikato Environment for Knowledge Analysis
Version 3.5.8
(c) 1999 - 2008
The University of Waikato
Hamilton, New Zealand

The buttons can be used to start the following applications:

- **Explorer An environment** for exploring data with WEKA (the rest of this Documentation deals with this application in more detail).
- **Experimenter** An environment for performing experiments and conducting statistical tests between learning schemes.

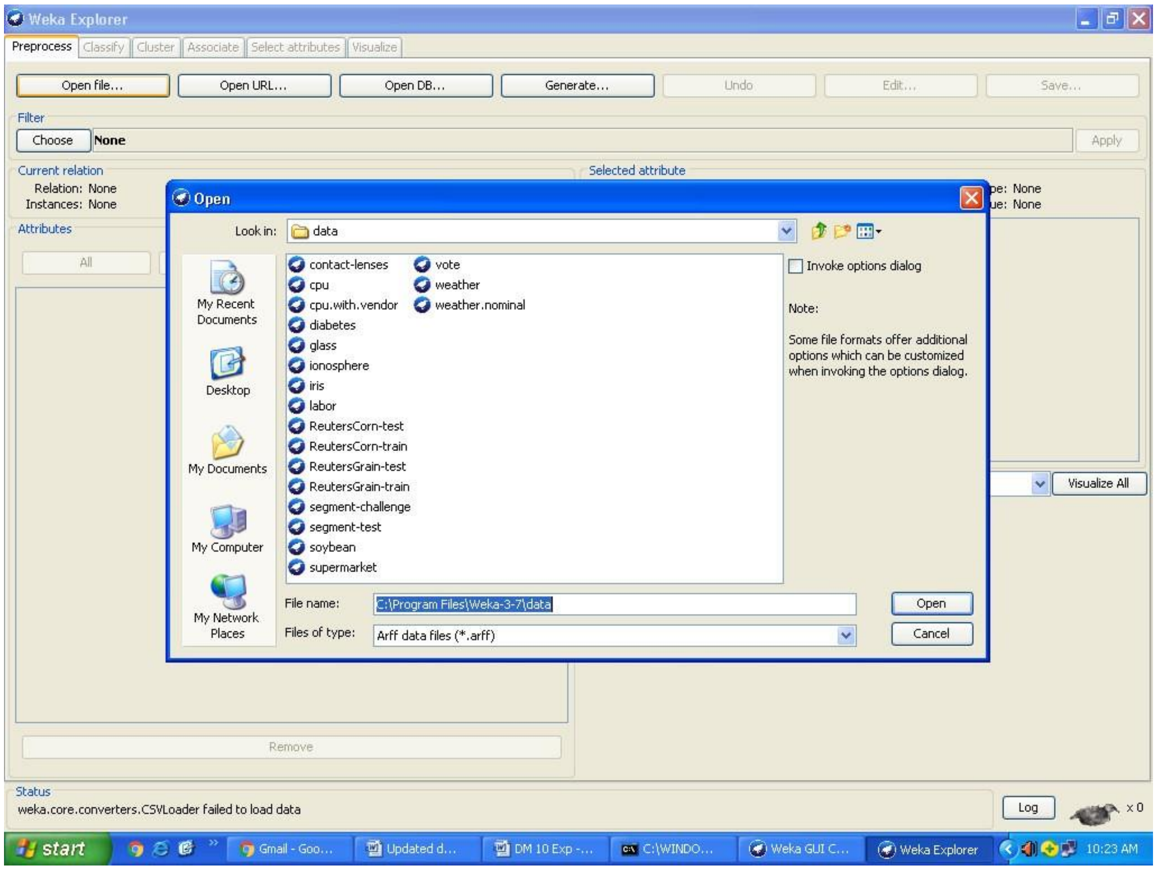
- **Knowledge Flow** This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.

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

1.(iv) Explore the available data sets in WEKA.

Ans: Steps for identifying data sets in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on open file button.
4. Choose WEKA folder in C drive.
5. Select and Click on data option button.



Filter: None

Relation: None
Instances: None

Look in: data

contact-lenses, cpu, cpu.with.vendor, diabetes, glass, ionosphere, iris, labor, ReutersCorn-test, ReutersCorn-train, ReutersGrain-test, ReutersGrain-train, segment-challenge, segment-test, soybean, supermarket, vote, weather, weather.nominal

Note:

Some file formats offer additional options which can be customized when invoking the options dialog.

File name: C:\Program Files\Weka-3-7\data

Files of type: Arff data files (*.arff)

Status

weka.core.converters.CSVLoader failed to load data

.

Sample Weka Data Sets

Below are some sample WEKA data sets, in arff format.

- contact-lens.arff
- cpu.arff
- cpu.with-vendor.arff
- diabetes.arff
- glass.arff
- ionospehre.arff
- iris.arff
- labor.arff
- ReutersCorn-train.arff

- ReutersCorn-test.arff
- ReutersGrain-train.arff
- ReutersGrain-test.arff
- segment-challenge.arff
- segment-test.arff
- soybean.arff
- supermarket.arff
- vote.arff
- weather.arff
- weather.nominal.arff

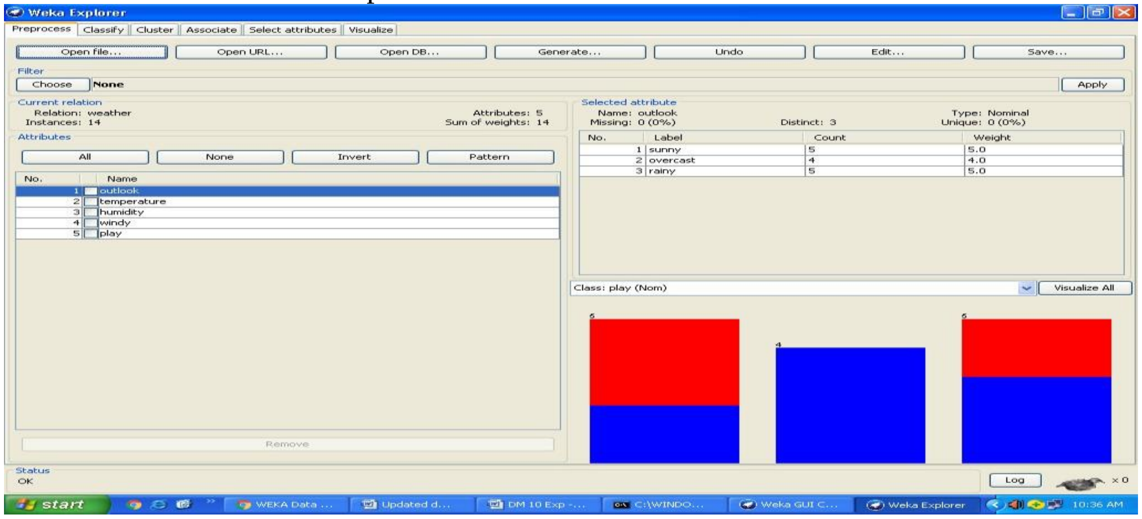
1. (vi) Load a data set (ex.Weather dataset,Iris dataset,etc.)

Ans: Steps for load the Weather data set.

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on open file button.
4. Choose WEKA folder in C drive.
5. Select and Click on data option button.
6. Choose Weather.arff file and Open the file.

Steps for load the Iris data set.

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on open file button.
4. Choose WEKA folder in C drive.
5. Select and Click on data option button.
6. Choose Iris.arff file and Open the file.



Filter: None

Current relation
Relation: weather Attributes: 5
Instances: 14 Sum of weights: 14

Attributes

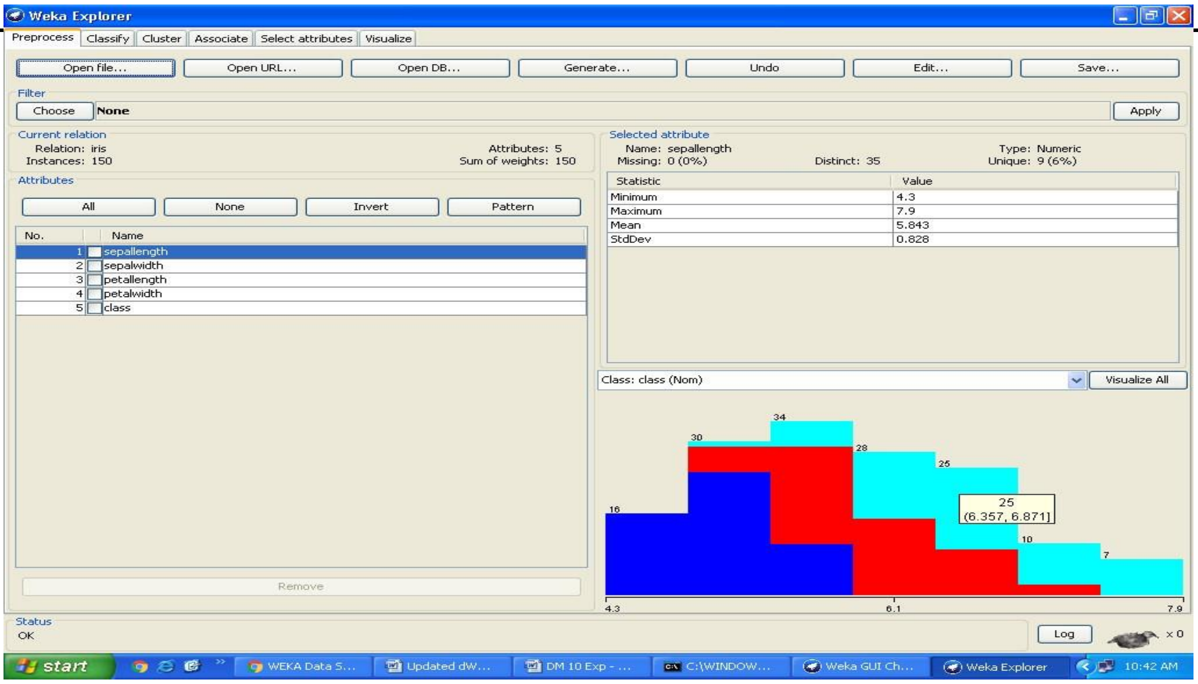
No.	Name
1	outlook
2	temperature
3	humidity
4	windy
5	play

Selected attribute
Name: outlook Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

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

Class: play (Nom)

Status
OK



Filter: None

Current relation
Relation: iris Attributes: 5
Instances: 150 Sum of weights: 150

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute
Name: sepallength Type: Numeric
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Class: class (Nom)

Status
OK

2.Explore various options in Weka for Preprocessing data and apply (like Discretization Filters, Resample filter, etc.) n each dataset.

Ans:

<u>Preprocess Tab</u>

1. Loading Data

The first four buttons at the top of the preprocess section enable you to load data into WEKA:

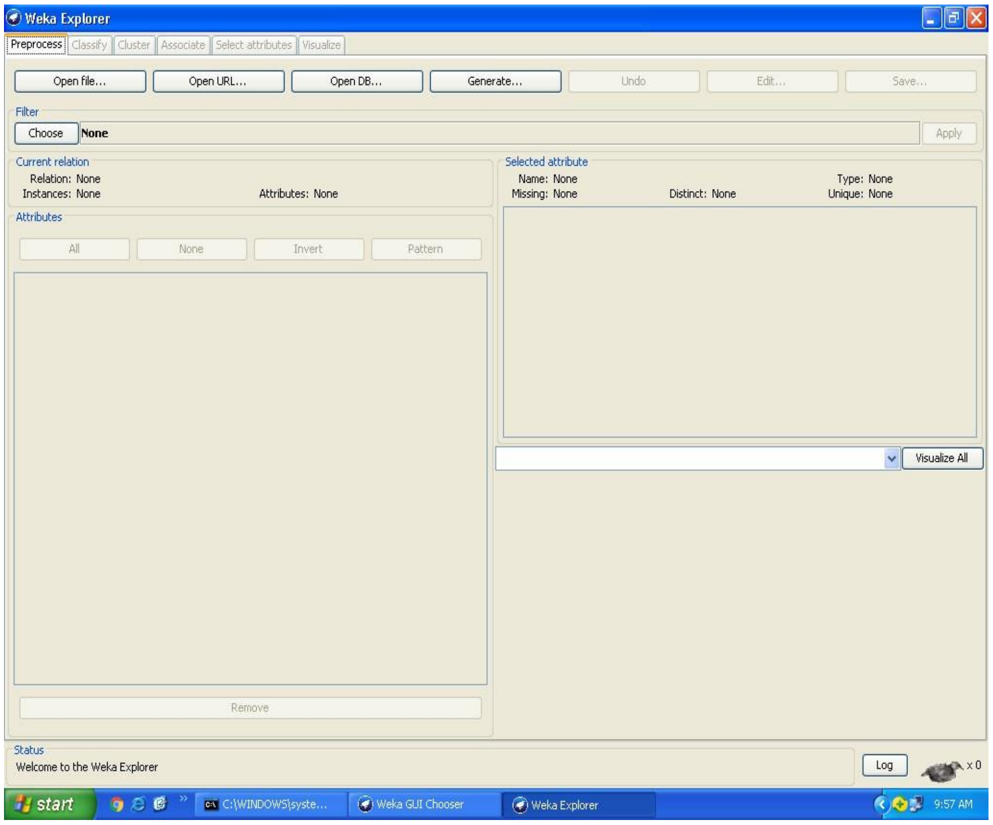
1. Open file.... Brings up a dialog box allowing you to browse for the data file on the local file system.

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

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

4. Generate .... Enables you to generate artificial data from a variety of Data Generators. Using the Open file ...button you can read files in a variety of formats: **WEKA's ARFF format, CSV**

format, C4.5 format, or serialized Instances format. ARFF files typically have a .arff extension, CSV files a .csv extension, C4.5 files a .data and .names extension, and serialized Instances objects a .bsi extension.



None

Relation: None
Instances: None Attributes: None

Name: None Type: None
Missing: None Distinct: None Unique: None

Welcome to the Weka Explorer

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

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

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

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

Working With Attributes

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

The list has three columns:

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

2. Selection tick boxes. These allow you select which attributes are present in the relation.
3. Name. The name of the attribute, as it was declared in the data file. When you click on different rows in the list of attributes, the fields change in the box to the right titled Selected attribute.

This box displays the characteristics of the currently highlighted attribute in the list:

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

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

3. Missing. The number (and percentage) of instances in the data for which this attribute is missing (unspecified).
4. Distinct. The number of different values that the data contains for this attribute.

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

Below these statistics is a list showing more information about the values stored in this attribute, which differ depending on its type. If the attribute is nominal, the list consists of each possible value for the attribute along with the number of instances that have that value. If the attribute is numeric, the list gives four statistics describing the distribution of values in the data— the minimum, maximum, mean and standard deviation. And below these statistics there is acoloured 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 whenclicked.) 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

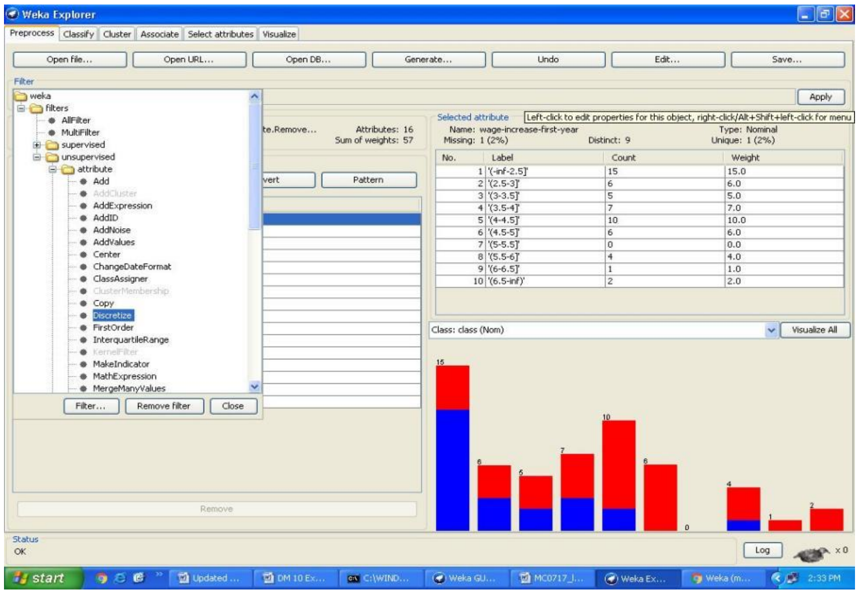
1. **All.** All boxes are ticked.
2. **None.** All boxes are cleared (unticked).
3. **Invert.** Boxes that are ticked become unticked and vice versa.

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.



Attributes: 16
Sum of weights: 57

Selected attribute
Left-click to edit properties for this object, right-click/Alt+Shift+left-click for menu

Name: wage-increase-first-year Type: Nominal
Missing: 1 (2%) Distinct: 9 Unique: 1 (2%)

No.	Label	Count	Weight
1	'(-inf-2.5]'	15	15.0
2	'(2.5-3]'	6	6.0
3	'(3-3.5]'	5	5.0
4	'(3.5-4]'	7	7.0
5	'(4-4.5]'	10	10.0
6	'(4.5-5]'	6	6.0
7	'(5-5.5]'	0	0.0
8	'(5.5-6]'	4	4.0
9	'(6-6.5]'	1	1.0
10	'(6.5-inf)'	2	2.0

Class: class (Nom)

Status
OK

The GenericObjectEditor Dialog Box

The GenericObjectEditor dialog box lets you configure a filter. The same kind of dialog box is used to configure other objects, such as classifiers and clusterers

(see below). The fields in the window reflect the available options.

Right-clicking (or Alt+Shift+Left-Click) on such a field will bring up a popup menu, listing the following options:

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

2. Copy configuration to clipboard copies the currently displayed configuration string to the **system's clipboar**d 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 theclipboard 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.

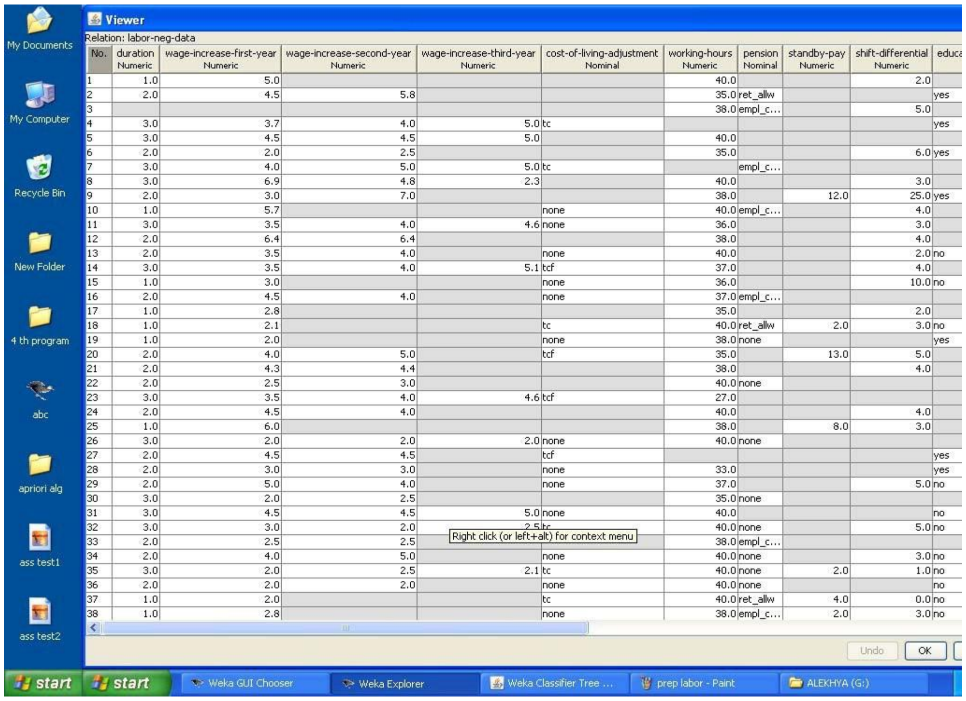
Applying Filters

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

Dataset labor.arff

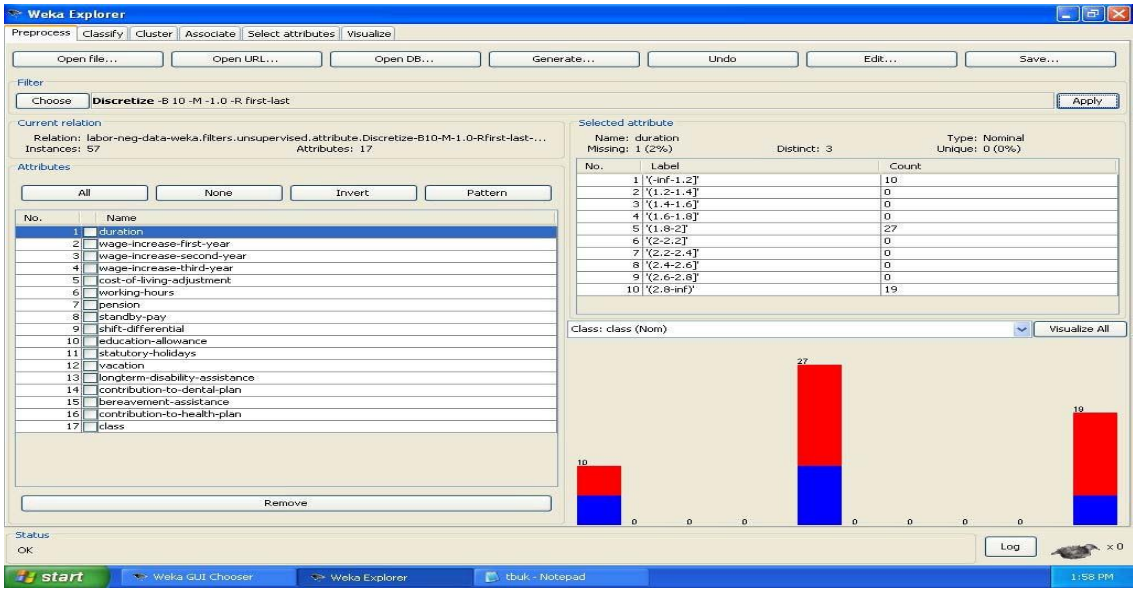


Relation: labor-neg-data

No.	duration Numeric	wage-increase-first-year Numeric	wage-increase-second-year Numeric	wage-increase-third-year Numeric	cost-of-living-adjustment Nominal	working-hours Numeric	pension Nominal	standby-pay Numeric	shift-differential Numeric	educa
1	1.0	5.0				40.0			2.0	
2	2.0	4.5	5.8			35.0	ret_allw			yes
3						38.0	empl_c...		5.0	
4	3.0	3.7	4.0	5.0	tc					yes
5	3.0	4.5	4.5	5.0		40.0				
6	2.0	2.0	2.5			35.0			6.0	yes
7	3.0	4.0	5.0	5.0	tc		empl_c...			
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	yes
10	1.0	5.7			none	40.0	empl_c...		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	no
14	3.0	3.5	4.0	5.1	tcf	37.0			4.0	
15	1.0	3.0			none	36.0			10.0	no
16	2.0	4.5	4.0		none	37.0	empl_c...			
17	1.0	2.8				35.0			2.0	
18	1.0	2.1			tc	40.0	ret_allw	2.0	3.0	no
19	1.0	2.0			none	38.0	none			yes
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					yes
28	2.0	3.0	3.0		none	33.0				yes
29	2.0	5.0	4.0		none	37.0			5.0	no
30	3.0	2.0	2.5			35.0	none			
31	3.0	4.5	4.5	5.0	none	40.0				no
32	3.0	3.0	2.0	2.5	tc	40.0	none		5.0	no
33	2.0	2.5	2.5	[illegible]	[illegible]	38.0	empl_c...			
34	2.0	4.0	5.0		none	40.0	none		3.0	no
35	3.0	2.0	2.5	2.1	tc	40.0	none	2.0	1.0	no
36	2.0	2.0	2.0		none	40.0	none			no
37	1.0	2.0			tc	40.0	ret_allw	4.0	0.0	no
38	1.0	2.8			none	38.0	empl_c...	2.0	3.0	no

Right click (or left+alt) for context menu

The following screenshot shows the effect of discretization



Filter: **Discretize** -B 10 -M -1.0 -R first-last

Current relation
Relation: labor-neg-data-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last-...
Instances: 57 Attributes: 17

Attributes

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
9	shift-differential
10	education-allowance
11	statutory-holidays
12	vacation
13	longterm-disability-assistance
14	contribution-to-dental-plan
15	bereavement-assistance
16	contribution-to-health-plan
17	class

Selected attribute
Name: duration Type: Nominal
Missing: 1 (2%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count
1	'(-inf-1.2]'	10
2	'(1.2-1.4]'	0
3	'(1.4-1.6]'	0
4	'(1.6-1.8]'	0
5	'(1.8-2]'	27
6	'(2-2.2]'	0
7	'(2.2-2.4]'	0
8	'(2.4-2.6]'	0
9	'(2.6-2.8]'	0
10	'(2.8-inf)'	19

Class: class (Nom)

Status
OK

A. Load each dataset into Weka and run Aprior algorithm with different support and confidence values. Study the rules generated.

Ans:

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 <confidence>: 0.9
Number of cycles performed: 17
```

Generated sets of large itemsets:

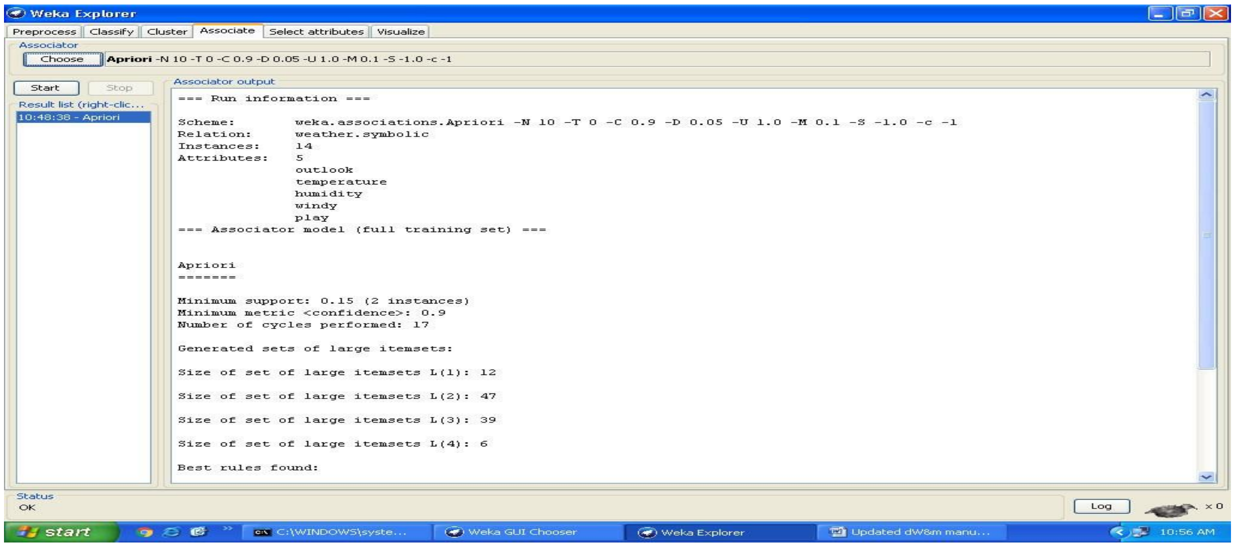
Size of set of large itemsets L(1): 12

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)



Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

10:48:38 - Apriori

```
=== 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 <confidence>: 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:
```

OK

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 criteriasupport 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,

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

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

support = support({*A* U *C*})

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

3.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 fieldthat 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) youcan 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.

Test Options

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

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

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

3. Cross-validation. The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.
4. Percentage split. The classifier is evaluated on how well it predicts a certain percentage of the data which is held out for testing. The amount of data held out depends on the value entered in the % field.

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 the output. This option is not selected by default.
4. Output confusion matrix. The confusion matrix of the classifier's predictions is included in the output. This option is selected by default.

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

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

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

7. Output additional attributes. If additional attributes need to be output alongside the

predictions, e.g., an ID attribute for tracking misclassifications, then the index of this attribute can be specified here. The usual **Weka ranges are supported,"first" and "last" are therefore valid** indices as **well (example: "first**-3,6,8,12-**last").**

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

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

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

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

The Class Attribute

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

By default, the class is taken to be the last attribute in the data. If you want

to train a classifier to predict a different attribute, click on the box below the Test options box to bring up a drop-down list of attributes to choose from.

Training a Classifier

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

A. Load each dataset into Weka and run id3, j48 classification algorithm, study the classifier output. Compute entropy values, Kappa ststistic.

Ans:

- Steps for run ID3 and J48 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 J48 algorithm and select use training set test option.
9. Click on start button.
10. Click on classify tab and Choose ID3 algorithm and select use training set test option.
11. Click on start button.

Output:

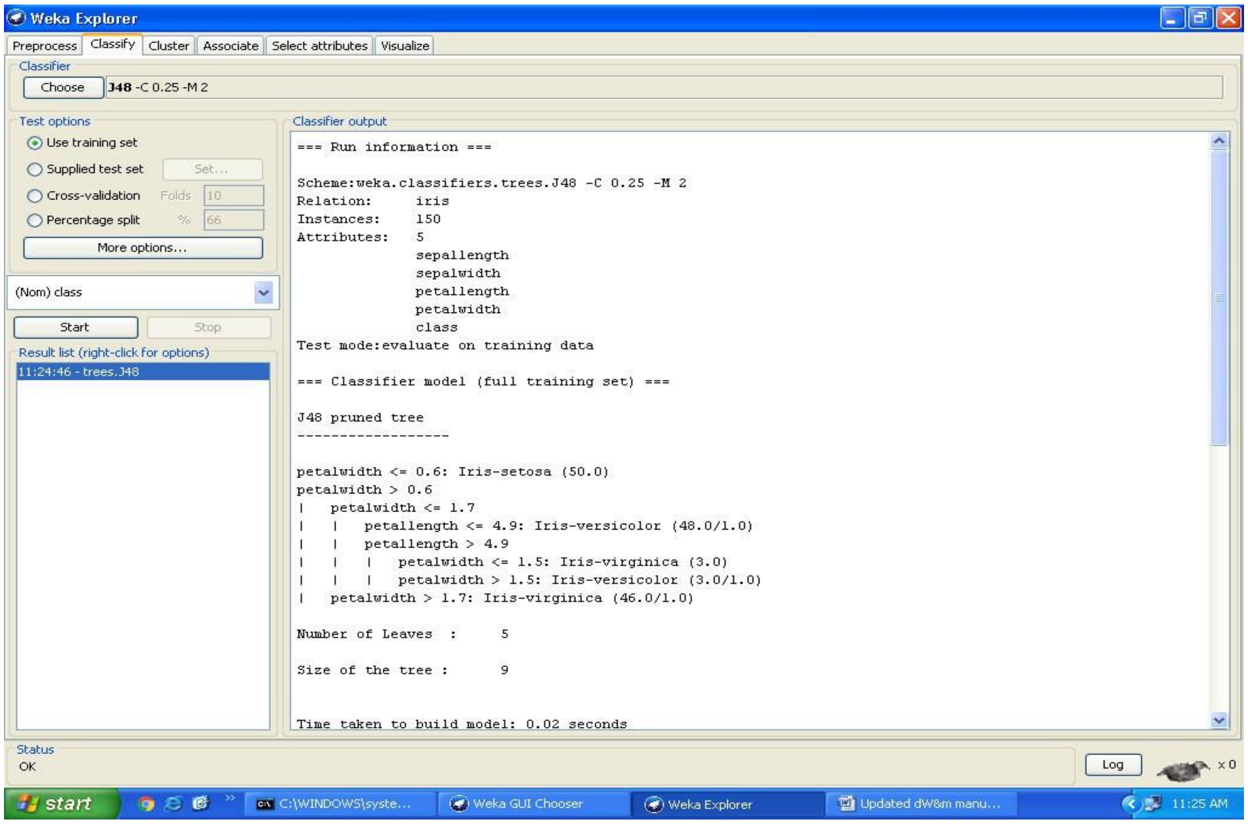
```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     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-virginica (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      %
Kappa statistic                        0.97
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      1.585 bits/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  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.980    0.020    0.961      0.980    0.970      0.955    0.990     0.969     Iris-versicolor
                 0.960    0.010    0.980      0.960    0.970      0.955    0.990     0.970     Iris-virginica
Weighted Avg.    0.980    0.010    0.980      0.980    0.980      0.970    0.993     0.980
```

=== 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
```



J48 -C 0.25 -M 2

Test options: (● Use training set) Cross-validation Folds 10; Percentage split % 66

(Nom) class

11:24:46 - trees.J48

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     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-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9


Time taken to build model: 0.02 seconds
```

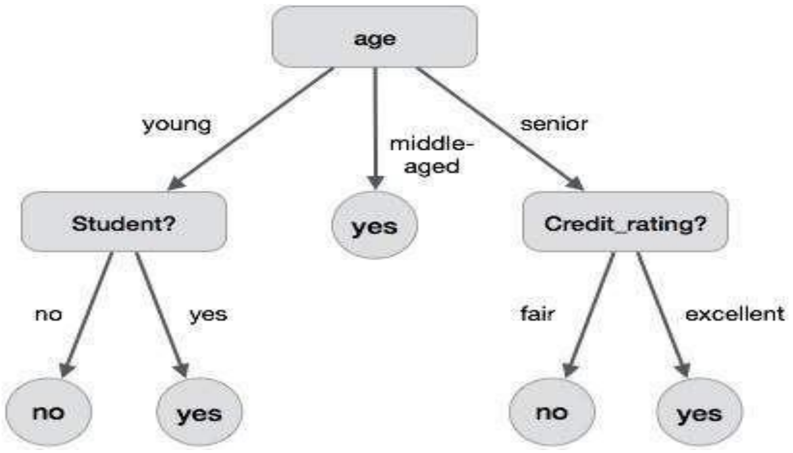
OK

B.Extract if-then rues from decision tree gentrated by classifier, Observe the confusionmatrix and derive Accuracy, F- measure, TPrate, FPrate , Precision and recall values. Apply cross-validation strategy with various fold levels and compare the accuracy results.

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 rule in 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 are logically 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 a decision 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 toeach 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 only and 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;
```

Rule Pruning

The rule is pruned is due to the following reason −

- The Assessment of quality is made on the original set of training data. The rule may perform well on training data but less well on subsequent data. That's why the rule pruningis required.
- The rule is pruned by removing conjunct. The rule R is pruned, if pruned version of R has greater quality than what was assessed on an independent set of tuples.

FOIL is one of the simple and effective method for rule pruning. For a given rule R,

FOIL_Prune = pos - neg / pos + neg
where pos and neg is the number of positive tuples covered by R, respectively.

Note − This value will increase with the accuracy of R on the pruning set. Hence, if the FOIL_Prune value is higher for the pruned version of R, then we prune R.

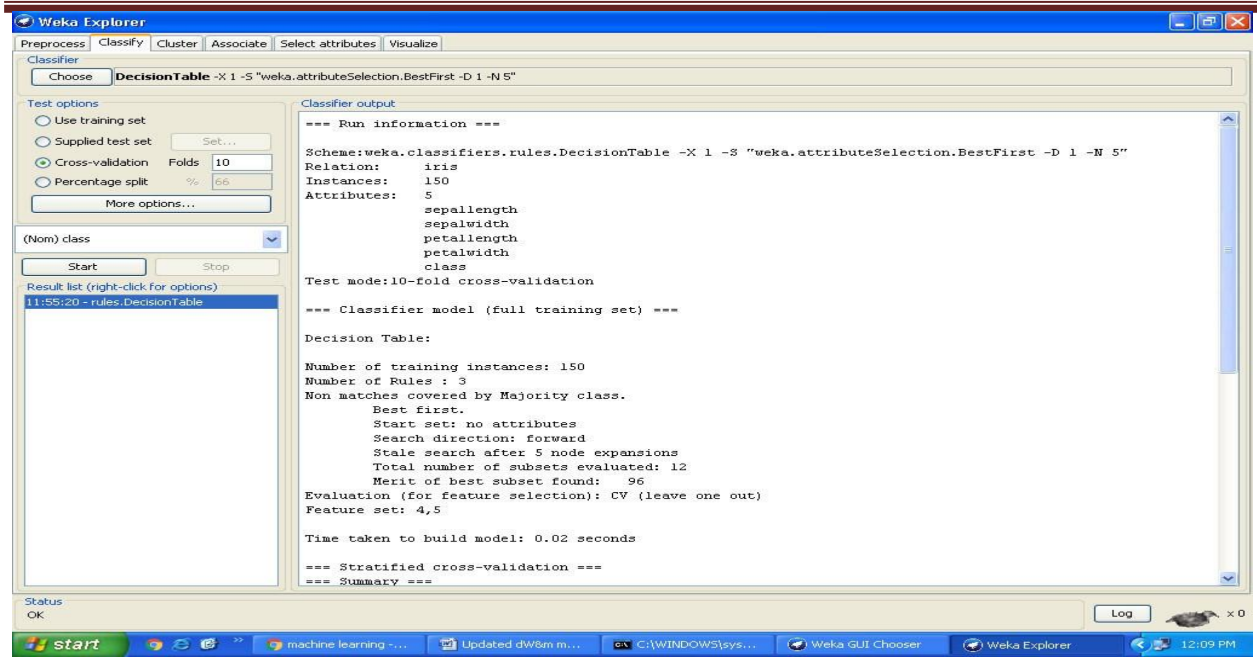
- Steps for run decision tree algorithms in WEKA

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.

Output:

```
=== 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   ROC Area  Class



Weighted Avg.    0.927     0.037        0.927        0.927         0.927      0.964

                 1            0             1            1              1         1             Iris-setosa
                 0.88      0.05          0.898        0.88          0.889      0.946          Iris-versicolor
                 0.9       0.06          0.882        0.9           0.891      0.947         Iris-virginica


=== 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
```



Classifier: DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"

Test options: Cross-validation, Folds 10

(Nom) class

Result list: 11:55:20 - rules.DecisionTable

```
=== 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 ===
```

Status: OK

C.Compare classification results of ID3,J48, Naïve-Bayes and k-NN classifiers for each dataset , and reduce which classifier is performing best and poor for each dataset and justify.

Ans:

- Steps for run ID3 and J48 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 J48 algorithm and select use training set test option.

9. Click on start button.
10. Click on classify tab and Choose ID3 algorithm and select use training set test option.
11. Click on start button.
12. Click on classify tab and Choose Naïve-bayes algorithm and select use training set test option.
13. Click on start button.
14. Click on classify tab and Choose k-nearest neighbor and select use training set test option.
15. Click on start button.

J48:

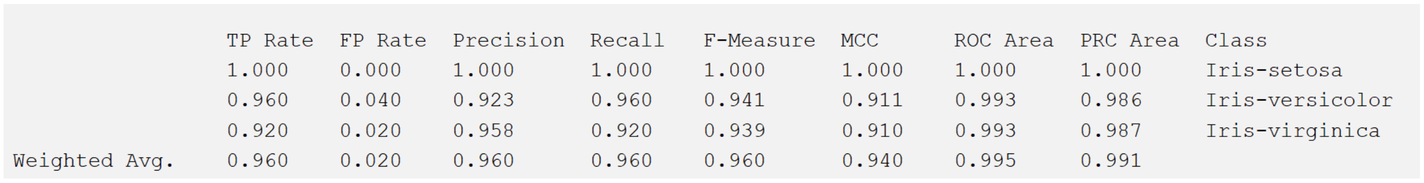
```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     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-virginica (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      %
Kappa statistic                          0.97
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  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.980    0.020    0.961      0.980    0.970      0.955    0.990     0.969     Iris-versicolor
                 0.960    0.010    0.980      0.960    0.970      0.955    0.990     0.970     Iris-virginica
Weighted Avg.    0.980    0.010    0.980      0.980    0.980      0.970    0.993     0.980

=== 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
```

Naïve-bayes:

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayes
Relation:     iris
Instances:    150
Attributes:   5
sepallength
```

```
sepalwidth
petallength
petalwidth
class
Test mode:evaluate on training data
=== 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

=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances        144              96      %
Incorrectly Classified Instances        6               4      %
Kappa statistic                    0.94
Mean absolute error                    0.0324
Root mean squared error                   0.1495
Relative absolute error                7.2883 %
Root relative squared error            31.7089 %
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    0.993     0.986     Iris-versicolor
                 0.920    0.020    0.958      0.920    0.939      0.910    0.993     0.987     Iris-virginica
Weighted Avg.    0.960    0.020    0.960      0.960    0.960      0.940    0.995     0.991
```

```
=== 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
```

<u>K-Nearest Neighbor (IBK):</u>

```
=== Run information ===
Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A
\"weka.core.EuclideanDistance -R first-last\""
Relation:     iris
Instances:    150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode:evaluate on training data

=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification

Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         150               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0085
Root mean squared error                  0.0091
Relative absolute error                  1.9219 %
Root relative squared error              1.9335 %
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
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-versicolor
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-virginica
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 50  0 |  b = Iris-versicolor
  0  0 50 |  c = Iris-virginica
```

4.demonstrate performing clustering on data sets Clustering Tab

Selecting a Clusterer

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

window brings up a GenericObjectEditor dialog with which to choose a new clustering scheme.

Cluster Modes

The Cluster mode box is used to choose what to cluster and how to evaluate

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

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

Ignoring Attributes

Often, some attributes in the data should be ignored when clustering. The Ignore attributes button brings up a small window that allows you to select which attributes are ignored. Clicking on an attribute in the window highlights it, holding down the SHIFT key selects a range

of consecutive attributes, and holding down CTRL toggles individual attributes on and off. To cancel the selection, back out with the Cancel button. To activate it, click the Select button. The next time clustering is invoked, the selected attributes are ignored.

Working with Filters

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

Learning Clusters

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

A.Load each dataset into Weka and run simple k-means clustering algorithm with different values of k(number of desired clusters). Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.

Ans:

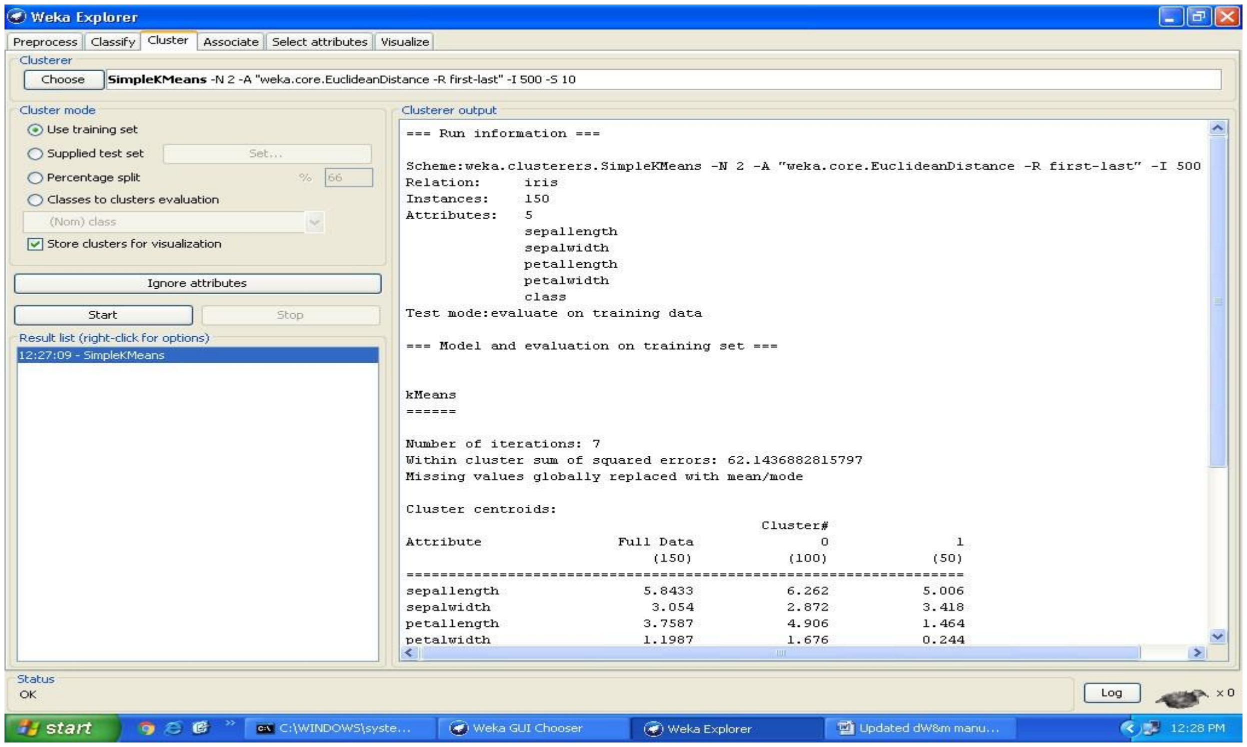
- Steps for run K-mean Clustering algorithms in WEKA

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 cluster tab and Choose k-mean and select use training set test option.
9. Click on start button.

Output:

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500
-S 10
Relation:     iris
```

```
Instances:    150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode:evaluate on training data

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

kMeans
======
Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797
Missing values globally replaced with mean/mode

Cluster centroids:
Cluster#
Attribute            Full Data          0            1
(150)        (100)          (50)
==============================================================
sepallength              5.8433       6.262        5.006
sepalwidth                3.054       2.872        3.418
petallength              3.7587       4.906        1.464
petalwidth               1.1987       1.676        0.244
class               Iris-setosa  Iris-versicolor  Iris-setosa

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

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

Clustered Instances

0      100 ( 67%)
1       50 ( 33%)
```



Clusterer: SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Cluster mode: [✓ Use training set] Supplied test set Percentage split % 66 Classes to clusters evaluation (Nom) class [✓ Store clusters for visualization]

Result list (right-click for options): 12:27:09 - SimpleKMeans

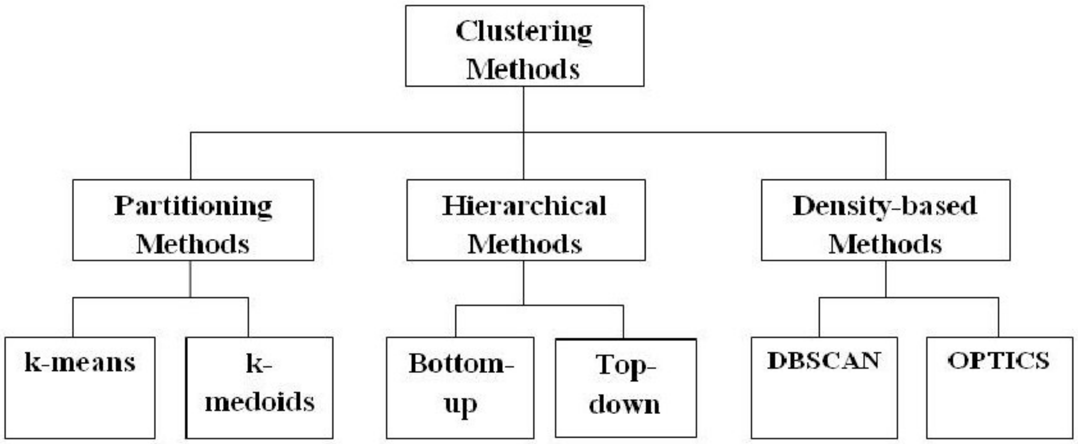
Clusterer output

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:evaluate on training data

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


kMeans
======

Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797
Missing values globally replaced with mean/mode

Cluster centroids:
                                        Cluster#
Attribute               Full Data              0               1
                            (150)          (100)            (50)
==================================================================
sepallength                5.8433          6.262           5.006
sepalwidth                  3.054          2.872           3.418
petallength                3.7587          4.906           1.464
petalwidth                 1.1987          1.676           0.244
```

Status: OK

B.Explore other clustering techniques available in Weka.

Ans: Clustering Algorithms And Techniques in WEKA, They are



5.Write a java program to prepare a simulated data set with unique instances.

Ans:

Description:

A dataset in java is mostly used for providing a type of safe view to the data present as part of the SQL queries. It is part of a library called java.util.list which holds for the mostly parameterized types of data. When a select annotation for the method is selected then in that case the query parameter is used for any data class which have other access modifiers like the public to make the accessibility from the queries to the methods present within the class. A dataset in java can behave in either a connected or disconnected way.

Program:

```
        import java.util.HashSet;
        import java.util.Random;
        import java.util.Set;
        public class SimulatedDataSet{
        public static void main(String[] args) {
        Set<Integer> dataSet = new HashSet<>();
        Random random = new Random();
        while (dataSet.size() < 10) {
        int instance=random.nextInt(100);
        dataSet.add(instance);
}
        System.out.println(dataSet);
}
}
```

Output:

```
java -cp /tmp/VICb6w1FIF SimulatedDataSet
[0, 97, 18, 34, 35, 23, 41, 58, 13, 15]
```

6. Write a Python program to generate frequent item sets / association rules using Apriori algorithm

Ans:

Description:

This program uses the pandas library to create a DataFrame from the transaction data. The mixtend library is used to apply the Apriori algorithm and generate the frequent item sets and association rules. The program sets the minimum support to 0.6 and the minimum confidence to 0.8, which can be adjusted as needed. The output shows the frequent item sets and association rules.

Program:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
#example transaction data
dataset=[["Milk","Onion","Nutmeg","Kidney Beans","Eggs","Yogurt"],
        ["Dill","Onion","Nutmeg","Kidney Beans","Eggs","Yogurt"],
        ["Milk","Apple","Kidney Beans","Eggs"],
        ["Milk","Unicorn","Corn","Kidney Beans","Yogurt"],
        ["Corn","Onion","Onion","Kidney Beans","Ice Cream","Eggs"]]
#Transform transactions into a one-hot encoded dataframe
te=TransactionEncoder()
te_ary=te.fit(dataset).transform(dataset)
df=pd.DataFrame(te_ary,columns=te.columns_)
#Generate frequent  itemsets with minimium support of 0.6
frequent_itemsets=apriori(df,min_support=0.6,use_colnames=True)
#generate association rules with confidence of 0.8
rules=association_rules(frequent_itemsets,metric="confidence",min_threshold=0.8)
print("Frequent Itemsets:")
print(frequent_itemsets)
print("\nAssociation Rules :")
print(rules)
```

output:

```
D:\pythonProject\Scripts\python.exe "C:\Users\jaswa\PycharmProjects\pythonProject\apriori
algo.py"
Frequent Itemsets:
   support                    itemsets
0      0.8                      (Eggs)
1      1.0              (Kidney Beans)
2      0.6                      (Milk)
3      0.6                     (Onion)
4      0.6                    (Yogurt)
5      0.8        (Kidney Beans, Eggs)
6      0.6               (Onion, Eggs)
7      0.6        (Milk, Kidney Beans)
8      0.6       (Kidney Beans, Onion)
9      0.6      (Yogurt, Kidney Beans)
10     0.6  (Kidney Beans, Onion, Eggs)

Association Rules :
           antecedents          consequents  ...  conviction  zhangs_metric
0       (Kidney Beans)               (Eggs)  ...         1.0            0.0
1               (Eggs)       (Kidney Beans)  ...         inf            0.0
2              (Onion)               (Eggs)  ...         inf            0.5
3               (Milk)       (Kidney Beans)  ...         inf            0.0
4              (Onion)       (Kidney Beans)  ...         inf            0.0
5             (Yogurt)       (Kidney Beans)  ...         inf            0.0
6  (Kidney Beans, Onion)               (Eggs)  ...         inf            0.5
7        (Onion, Eggs)       (Kidney Beans)  ...         inf            0.0
8              (Onion)  (Kidney Beans, Eggs)  ...         inf            0.5

[9 rows x 10 columns]

Process finished with exit code 0
```

7. Write a program to calculate chi-square value using Python. Report your observation.

Ans:

Description:

The chi-square value is a measure of the association between two categorical variables. The chi2_contingency function from the spicy. stats library calculates the chi-square statistic and pvalue based on a contingency table. In this example, the contingency table represents the observed frequencies of two categorical variables. The chi-square statistic measures how much the observed frequencies differ from the expected frequencies under the assumption of independence. The p-value is the probability of observing the calculated chi-square statistic or a more extreme value, assuming the null hypothesis of independence. In general, if the p-value is less than a certain significance level (e.g. 0.05), then the null hypothesis is rejected and the association between the two variables is considered significant.

Program:

```
import numpy as np
from scipy.stats import chi2_contingency
observed=np.array([[10,20,30],[6,9,17]])
chi2,p,dof,expected=chi2_contingency(observed)
print("Chi-Square Statistic : ",chi2)
print("p value : ",p)
```

output:

```
Chi-Square Statistic :  0.27157465150403504
p value :  0.873028283380073
```

8. Write a program of Naive Bayesian classification using Python programming language.

Ans:

Description:

The Naive Bayes classification algorithm is a probabilistic classifier. It is based on probability models that incorporate strong independence assumptions.The independence assumptions often do not have an impact on reality. Therefore they are considered as naive.You can derive probability models by using Bayes' theorem (credited to Thomas Bayes). Depending on the nature of the probability model, you can train the Naive Bayes algorithm in a supervised learning setting.

Data mining in InfoSphere™ Warehouse is based on the maximum likelihood for parameter estimation for Naive Bayes models. The generated Naive Bayes model conforms to the Predictive Model Markup Language (PMML) standard.

A Naive Bayes model consists of a large cube that includes the following dimensions:

- Input field name
- Input field value for discrete fields, or input field value range for continuous fields. Continuous fields are divided into discrete bins by the Naive Bayes algorithm
- Target field value

This means that a Naive Bayes model records how often a target field value appears together with a value of an input field.

Program:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
iris=load_iris()
X=iris.data
y=iris.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
gnb=GaussianNB()
gnb.fit(X_train,y_train)
y_pred=gnb.predict(X_test)
accuracy=np.mean(y_pred==y_test)
print("Accuracy : ",accuracy)
```

output:

```
Accuracy :  0.9333333333333333
```

9. Write a program to cluster your choice of data using simple k-means algorithm using JDK

Ans:

Description:

The output of the program will vary depending on the initial data points and the randomly initialized centroids. However, the program should output the final clusters after the K-means algorithm has been run for a certain number of iterations. For example, if we use the same data points and number of clusters as in the example program, we might get the following output The outputoutput indicates that the data points have been clustered into two groups, with the first cluster containing the first three data points and the second cluster containing the last three data points. The centroids of these clusters should be close to [2.67, 3.33] and [5.67, 5.67], respectively, although the exact values will depend on the initial centroids and the number of iterations performed by the algorithm

Program:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;
public class KMeansClustering {
 public static void main(String[] args) {
 // Initialize the data points
 double[][] dataPoints = {{2.0, 3.0}, {3.0, 3.0}, {3.0, 4.0}, {5.0, 4.0}, {5.0, 6.0}, {7.0,
7.0}};
 // Initialize the number of clusters
 int k = 2;
 // Initialize the centroids randomly
 double[][] centroids = new double[k][dataPoints[0].length];
 Random rand = new Random();
 for (int i = 0; i < k; i++) {
 int randomIndex = rand.nextInt(dataPoints.length);
 centroids[i] = Arrays.copyOf(dataPoints[randomIndex],
dataPoints[randomIndex].length);
 }
 // Perform the K-means algorithm
 List<List<double[]>> clusters = new ArrayList<>();
 for (int i = 0; i < 10; i++) {
 // Initialize the clusters
 clusters.clear();
 for (int j = 0; j < k; j++) {
 clusters.add(new ArrayList<>());
 }
 // Assign data points to the nearest centroid
 for (double[] dataPoint : dataPoints) {
 int closestCentroidIndex = getClosestCentroidIndex(dataPoint, centroids);
 clusters.get(closestCentroidIndex).add(dataPoint);
 }
 // Update the centroids
 for (int j = 0; j < k; j++) {
 centroids[j] = getNewCentroid(clusters.get(j));
 }
 }
 // Print the clusters
 for (int i = 0; i < clusters.size(); i++) {
 System.out.println("Cluster " + i + ": " + clusters.get(i));
 }
 }
 // Helper method to get the index of the closest centroid for a data point
 private static int getClosestCentroidIndex(double[] dataPoint, double[][] centroids) {
 double minDistance = Double.MAX_VALUE;
 int closestCentroidIndex = 0;
 for (int i = 0; i < centroids.length; i++) {
 double distance = getDistance(dataPoint, centroids[i]);
 if (distance < minDistance) {
 minDistance = distance;
 closestCentroidIndex = i;
 }
 }
 return closestCentroidIndex;
 }
 // Helper method to calculate the Euclidean distance between two points
 private static double getDistance(double[] point1, double[] point2) {
 double sum = 0.0;
 for (int i = 0; i < point1.length; i++) {
 sum += Math.pow(point1[i] - point2[i], 2);
 }
 return Math.sqrt(sum);
 }
 // Helper method to calculate the new centroid for a cluster
 private static double[] getNewCentroid(List<double[]> cluster) {
 double[] newCentroid = new double[cluster.get(0).length];
 for (int i = 0; i < newCentroid.length; i++) {
 double sum = 0.0;
 for (double[] dataPoint : cluster) {
 sum += dataPoint[i];
 }
 newCentroid[i] = sum / cluster.size();
 }
 return newCentroid;
 }
}
```

Output:

```
D:\java>javac cluster.java
D:\java>java KMeansClustering
Cluster 0: [[D@65ab7765, [D@1b28cdfa, [D@eed1f14]
Cluster 1: [[D@7229724f, [D@4c873330, [D@119d7047]
```

10. Write a program of cluster analysis using simple k-means algorithm Python programming language.

Ans:

Description:

Program:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;
public class clusters{
    private static final int K=3;
    private static final int NUM_ITERATIONS=100;
    private static final double EPSILON=0.1;
    private static List<double[]> data=new ArrayList<>();
    public static void main(String args[])
    {
        data.add(new double[]{1,2});
        data.add(new double[]{2,3});
        data.add(new double[]{3,1});
        data.add(new double[]{1,1});
        data.add(new double[]{2,2});
        data.add(new double[]{3,3});
        int clusterAssignment[]=new int[data.size()];
        Arrays.fill(clusterAssignment,-1);
        List<double[]> centroids=initializeCentroids(K);
        int iteration=0;
        double[][] distances=new double[data.size()][K];
        while(iteration<NUM_ITERATIONS)
        {
            for(int i=0;i<data.size();i++)
            {
                for(int j=0;j<K;j++)
                {
                    distances[i][j]=euclideanDistance(data.get(i),centroids.get(j));
                }
            }
            int[] newClusterAssignment=getClusterAssignment(distances);
            if(Arrays.equals(clusterAssignment,newClusterAssignment))
            {
                break;
            }
            clusterAssignment=newClusterAssignment;
            for(int i=0;i<K;i++)
            {
                double[] newCentroid=getCentroid(i,clusterAssignment,data);
                double distance=euclideanDistance(centroids.get(i),newCentroid);
                if(distance>EPSILON)
                {
                    centroids.set(i,newCentroid);
                }
            }
            iteration++;
        }
        for(int i=0;i<data.size();i++)
        {
            System.out.println("DataPoint"+(i+1)+"belongs to cluster"+clusterAssignment[i]);
        }
    }
    private static List<double[]> initializeCentroids(int k)
    {
        List<double[]>centroids=new ArrayList<>();
        Random random=new Random();
        for(int i=0;i<k;i++)
        {
            int randomIndex=random.nextInt(data.size());
            centroids.add(data.get(randomIndex));
        }
        return centroids;
    }
    private static double euclideanDistance(double[] a,double[] b)
    {
        double sum=0;
        for(int i=0;i<a.length;i++)
        {
            sum+=Math.pow(a[i]-b[i],2);
        }
        return Math.sqrt(sum);
    }
// THIS TWO METHODS ARE ADDED
    private static double[] getCentroid(int clusterIndex, int[] clusterAssignment, List<double[]> data) {
        double[] centroid = new double[data.get(0).length]; // initialize centroid to 0
        int count = 0;

        for (int i = 0; i < clusterAssignment.length; i++) {
            if (clusterAssignment[i] == clusterIndex) { // if data point belongs to current cluster
                double[] dataPoint = data.get(i);
                for (int j = 0; j < dataPoint.length; j++) {
                    centroid[j] += dataPoint[j]; // add up values for each dimension
                }
                count++;
            }
        }
        for (int j = 0; j < centroid.length; j++) {
            centroid[j] /= count; // divide by number of data points to get average
        }
        return centroid;
    }
    private static int[] getClusterAssignment(double[][] distances) {
        int[] clusterAssignment = new int[data.size()]; // initialize new cluster assignment
        for (int i = 0; i < data.size(); i++) {
            double minDistance = Double.MAX_VALUE;
            int clusterIndex = -1;
            for (int j = 0; j < K; j++) {
                if (distances[i][j] < minDistance) { // if distance to centroid is smaller
                    minDistance = distances[i][j];
                    clusterIndex = j;
                }
            }
            clusterAssignment[i] = clusterIndex; // assign data point to closest centroid
        }
        return clusterAssignment;
    }

}
```

Output:

```
DataPoint1belongs to cluster0
DataPoint2belongs to cluster2
DataPoint3belongs to cluster1
DataPoint4belongs to cluster0
DataPoint5belongs to cluster2
DataPoint6belongs to cluster2
```

11. Write a program to compute/display dissimilarity matrix (for your own dataset containing at least four instances with two attributes) using Python

Ans:

```
import numpy as np
def dissimilarity_matrix(data):
    n = data.shape[0]
    dissimilarity = np.zeros((n, n))
    for i in range(n):
        for j in range(i, n):
            dissimilarity[i][j] = np.linalg.norm(data[i] - data[j])
            dissimilarity[j][i] = dissimilarity[i][j]
    return dissimilarity
# Example usage:
data = np.array([[1, 2], [2, 3], [3, 1], [1, 1], [2, 2], [3, 3]])
d = dissimilarity_matrix(data)
print(d)
```

output:

```
[[0.         1.41421356 2.23606798 1.         1.         2.23606798]
 [1.41421356 0.         2.23606798 2.23606798 1.         1.        ]
 [2.23606798 2.23606798 0.         2.         1.41421356 2.        ]
 [1.         2.23606798 2.         0.         1.41421356 2.82842712]
 [1.         1.         1.41421356 1.41421356 0.         1.41421356]
 [2.23606798 1.         2.         2.82842712 1.41421356 0.        ]]
```

12. Visualize the datasets using matplotlib in python.(Histogram, Box plot, Bar chart, Pie chart etc.,)

Ans:

Program:

```
import matplotlib.pyplot as plt
data=[1,2,3,4,2,3,4,5,2,4,5,6]
#histogram
plt.hist(data,bins=6,color='green')
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
#Box plot
plt.boxplot(data)
plt.title("BoxPlot")
plt.ylabel("value")
plt.show()
#Bar Chart
labels=['A','B','C','D','E','F','G']
data=[2,4,6,8,10,12,14]
plt.bar(labels,data,color='purple')
plt.title("Bar Chart")
plt.xlabel("Labels")
plt.ylabel("Values")
plt.show()
#pie chart
labels=['A','B','C','D']
data=[10,20,30,40]
plt.pie(data,labels=labels,startangle=90,autopct='%1.1f%%')
plt.title("pie chart")
plt.axis("equal")
plt.show()
```

output:

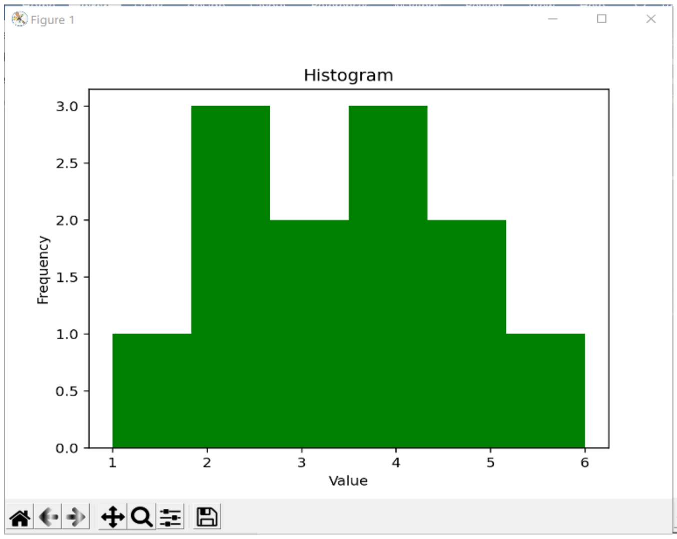


Figure 1
Histogram
Frequency
Value
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6

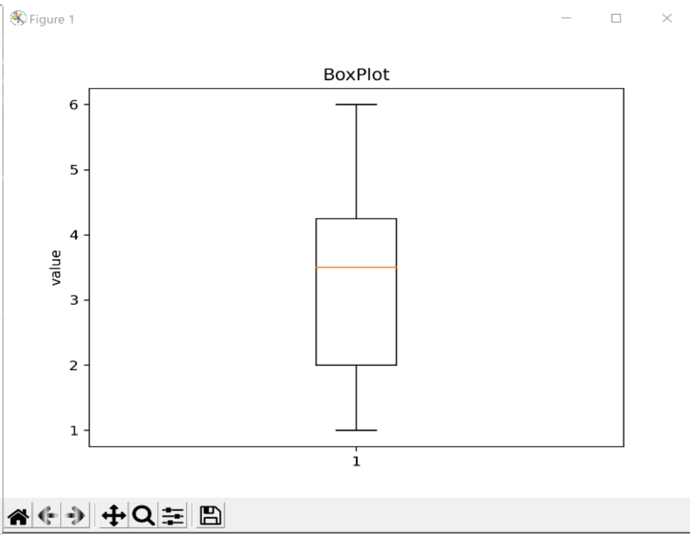


Figure 1
BoxPlot
value
6
5
4
3
2
1
1

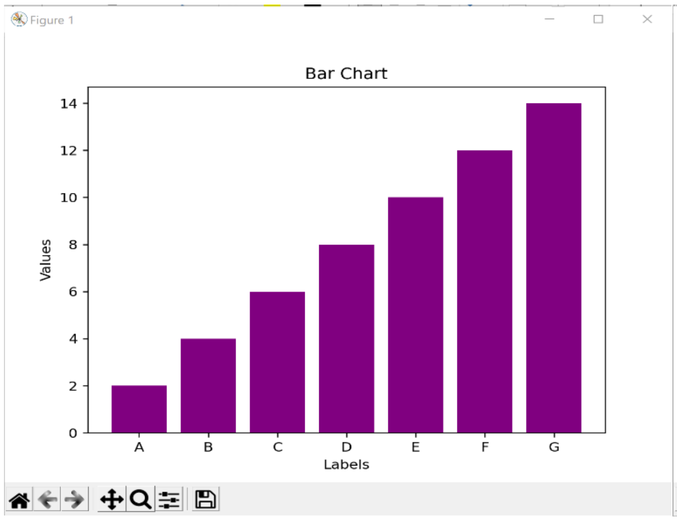


Figure 1
Bar Chart
Values
Labels
14
12
10
8
6
4
2
0
A
B
C
D
E
F
G

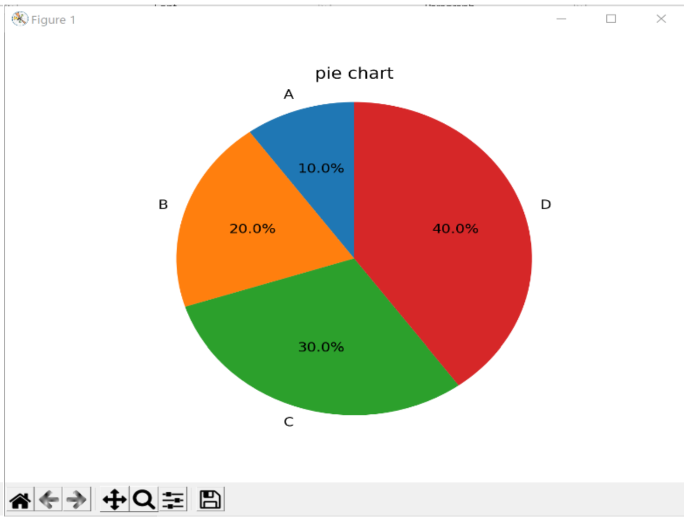


Figure 1
pie chart
A
B
C
D
10.0%
20.0%
30.0%
40.0%