Software Lab III
(DMW & AI for Cyber Security)

LABORATORY MANUAL

Third Year (SEM - II)

(2024-2025)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

GENBA SOPANRAO MOZE COLLEGE OF ENGINEERING,
Balewadi, Pune.

INDEX

Assignment: 1

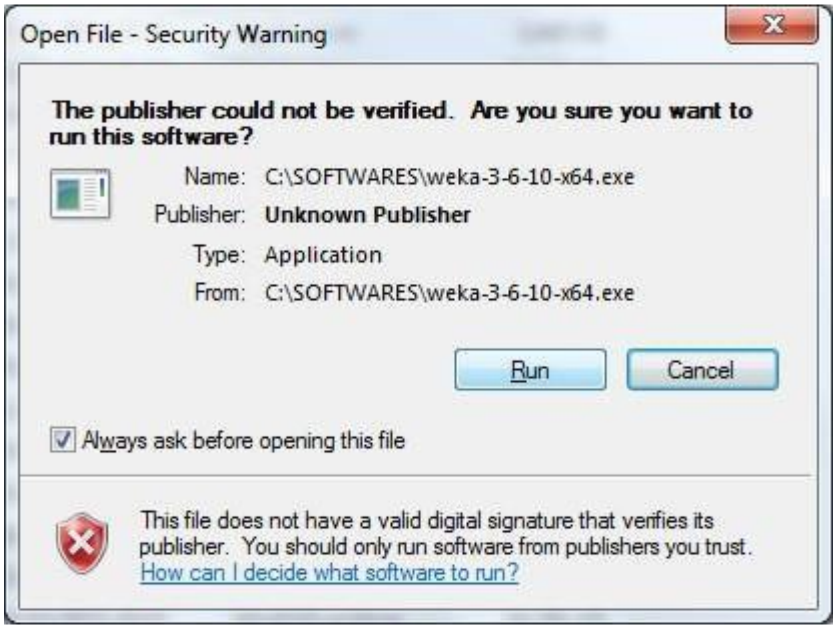
Title:- Build Data Warehouse and Explore WEKA

Aim:- To Study Data Warehousing and Explore WEKA.

Theory:

<u>Install Steps for WEKA a Data Mining Tool</u>

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

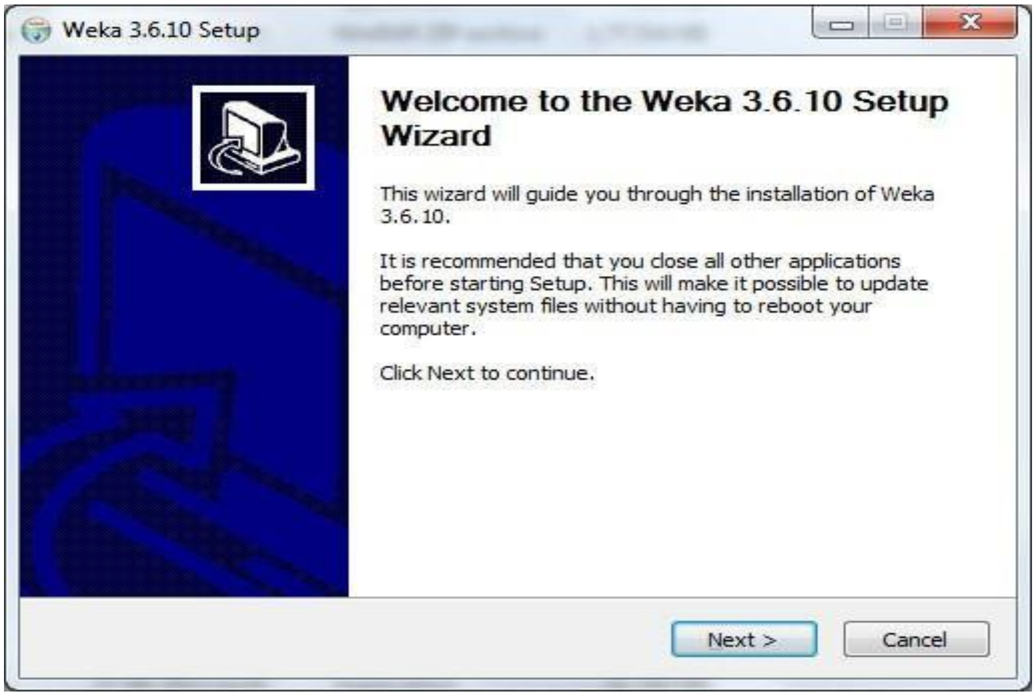

Open File - Security Warning

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

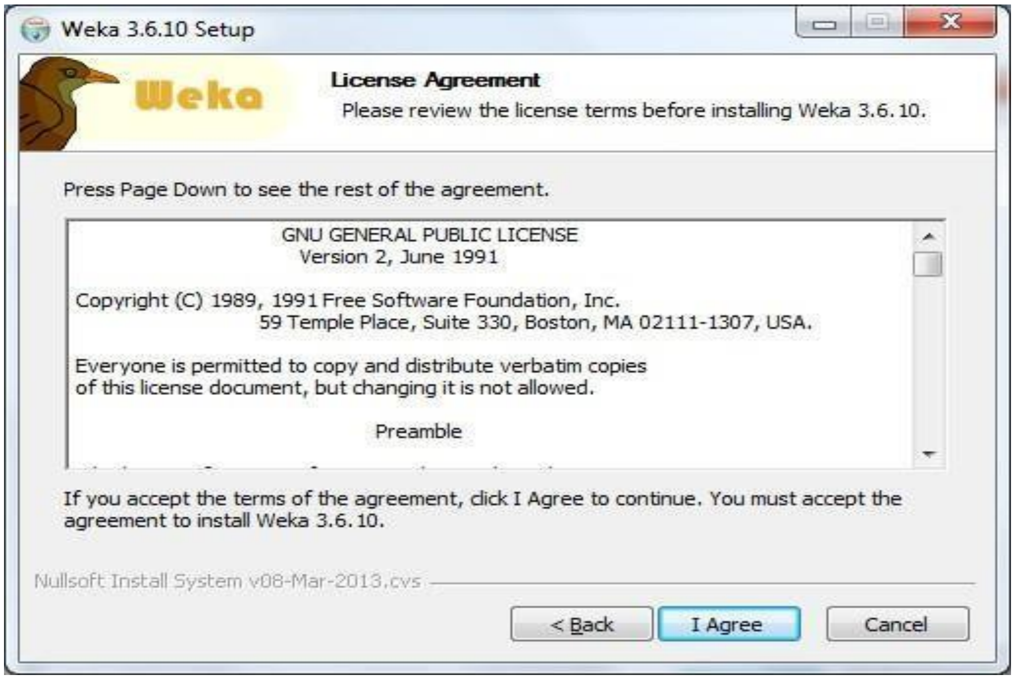
Welcome to the Weka 3.6.10 Setup Wizard

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.

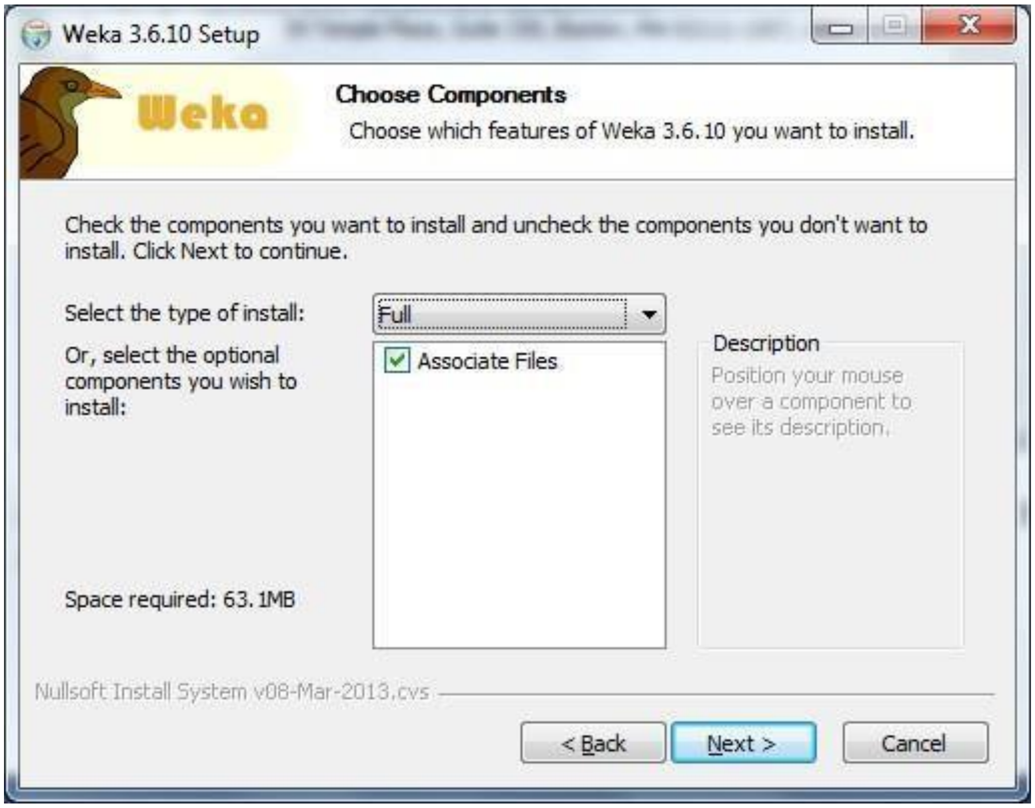
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

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.



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.

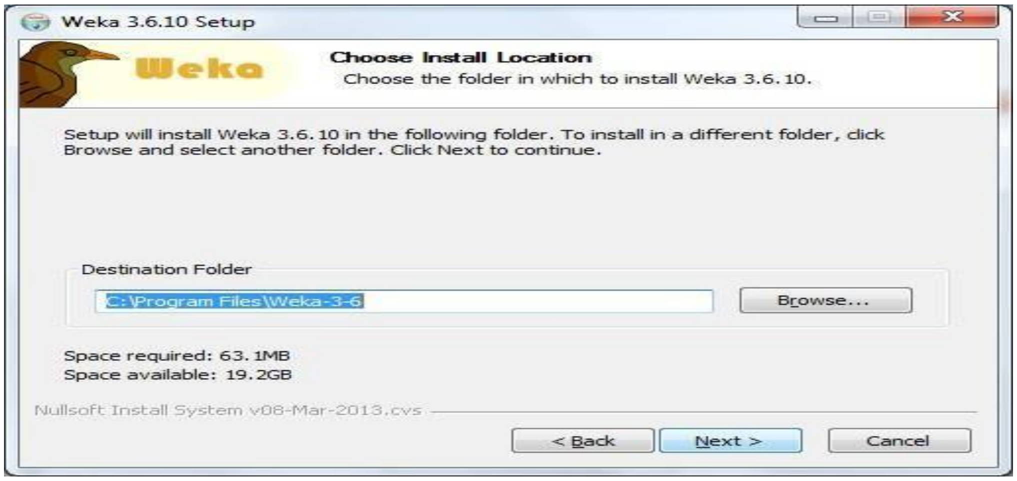
Select the type of install: Full

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

7. Change to your desire installation location.



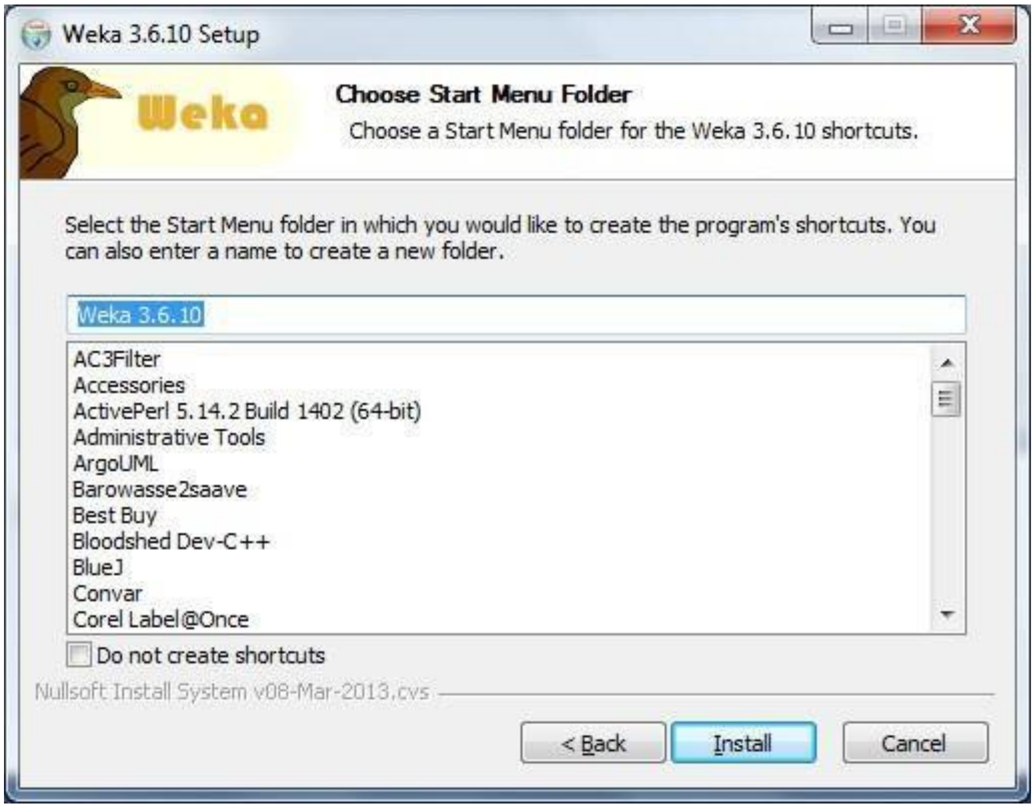
Choose the folder in which to install Weka 3.6.10.

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

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



Weka 3.6.10 Setup

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.

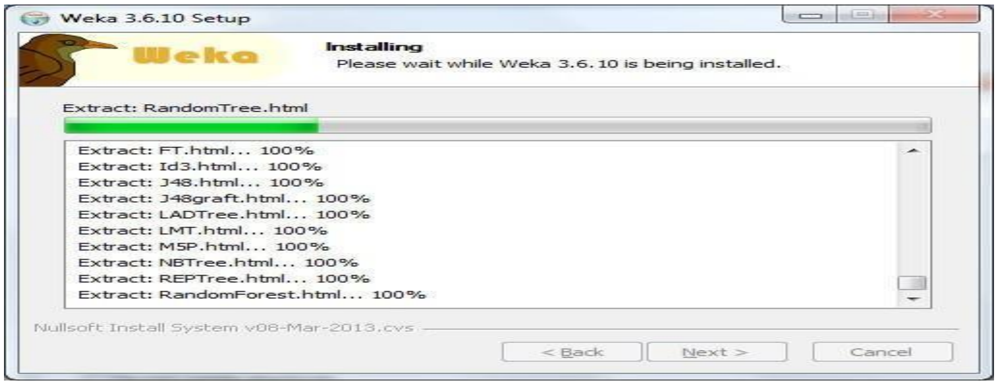
Weka 3.6.10

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.



Weka 3.6.10 Setup

Installing

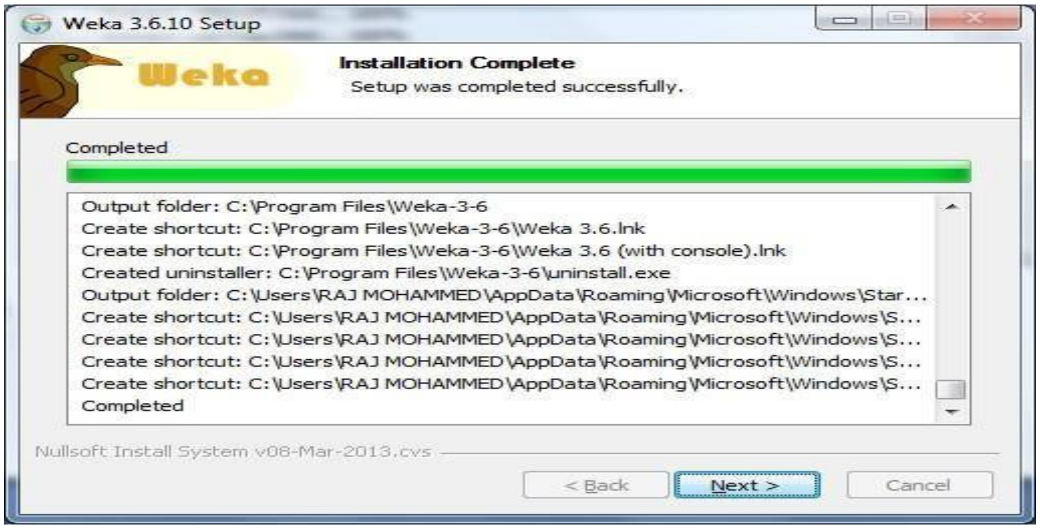
Please wait while Weka 3.6.10 is being installed.

```
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.



Weka 3.6.10 Setup

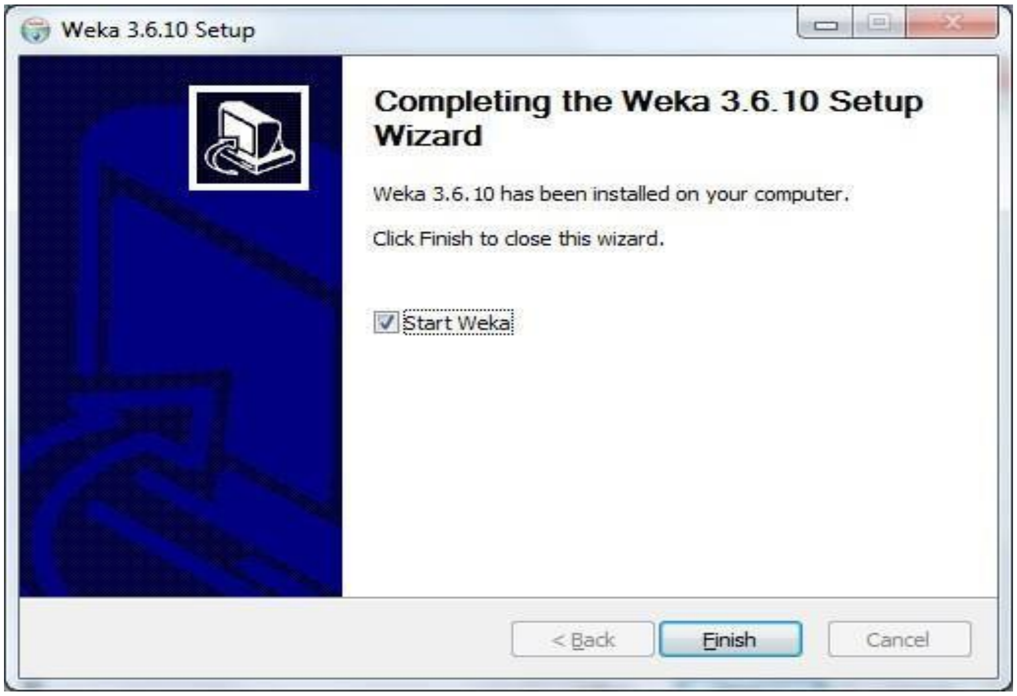
Installation Complete
Setup was completed successfully.

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.



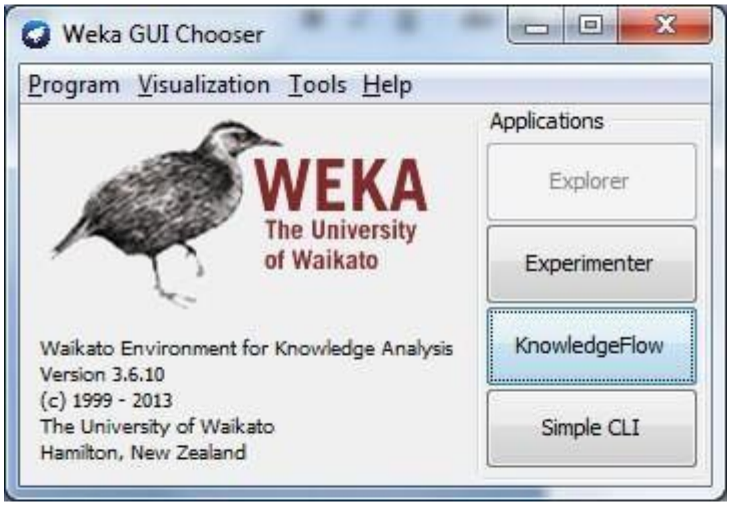
Weka 3.6.10 Setup

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.

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

Ans: **WEKA**

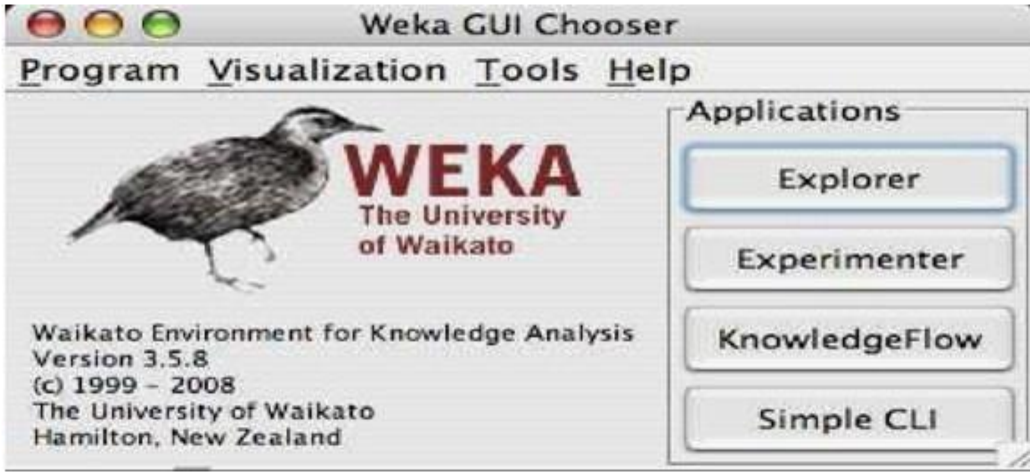
Weka is created by researchers at the university WIKATO in New Zealand. University of Waikato, Hamilton, New Zealand Alex Seewald (original Command-line primer) David Scuse (original Experimenter tutorial)

- 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. Explorer

The Graphical user interface

1.1 Section Tabs

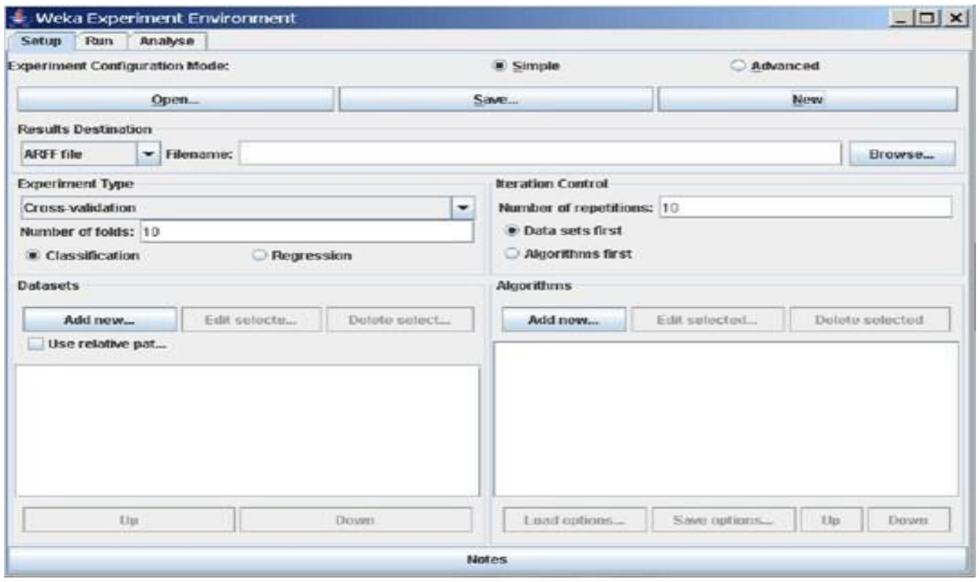
At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are grayed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data. The tabs are as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

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

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

2. Weka Experimenter:-

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



Experiment Configuration Mode: (●) Simple ( ) Advanced

Results Destination: ARFF file Filename:

Experiment Type: Cross-validation

Number of folds: 10

(●) Classification ( ) Regression

Iteration Control

Number of repetitions: 10

(●) Data sets first

( ) Algorithms first

Datasets

Algorithms

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

- Simple
- Advanced

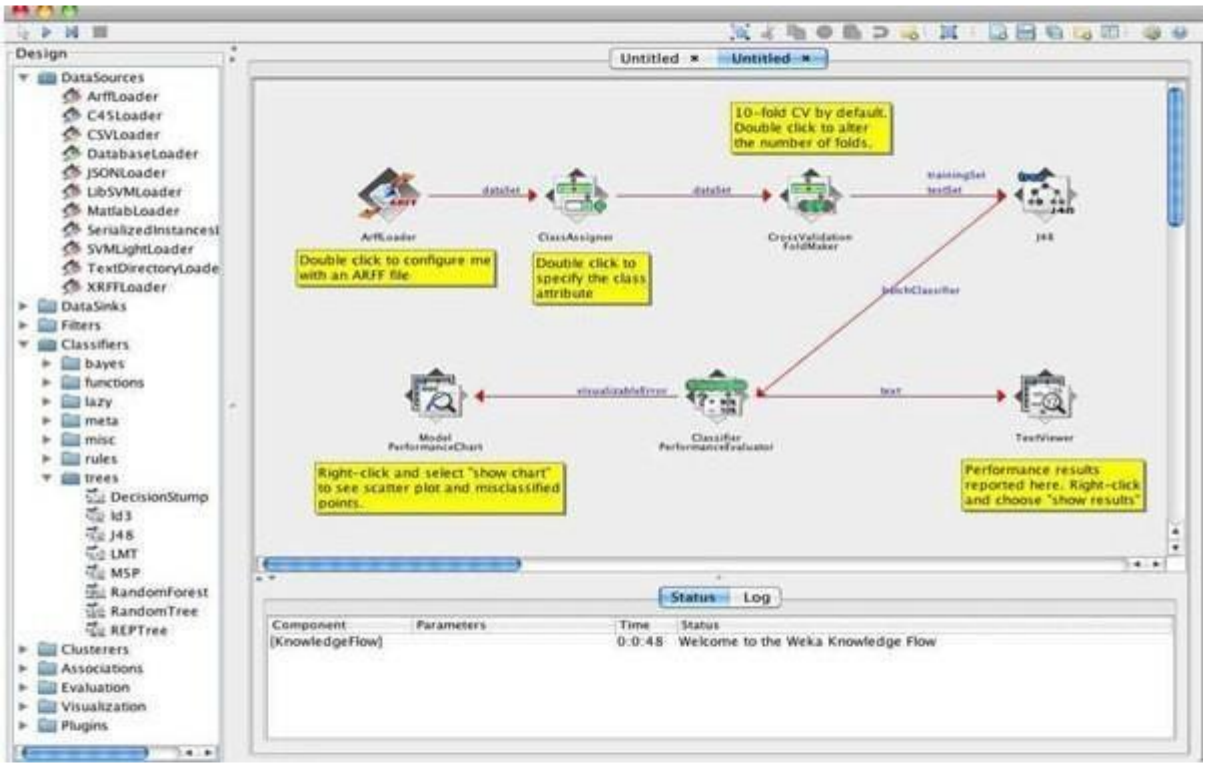
Both setups allow you to setup standard experiments, that are run locally on a single machine, or remote experiments, which are distributed between several hosts. The distribution of experiments cuts down the time the experiments will take until completion, but on the other hand the setup takes more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

Knowledge Flow

Introduction

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

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



10-fold CV by default. Double click to alter the number of folds.

Double click to configure me with an ARFF file

Double click to specify the class attribute

Right-click and select "show chart" to see scatter plot and misclassified points.

Performance results reported here. Right-click and choose "show results"

```
Component          Parameters          Time     Status
[KnowledgeFlow]                        0:0:48   Welcome to the Weka Knowledge Flow
```

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

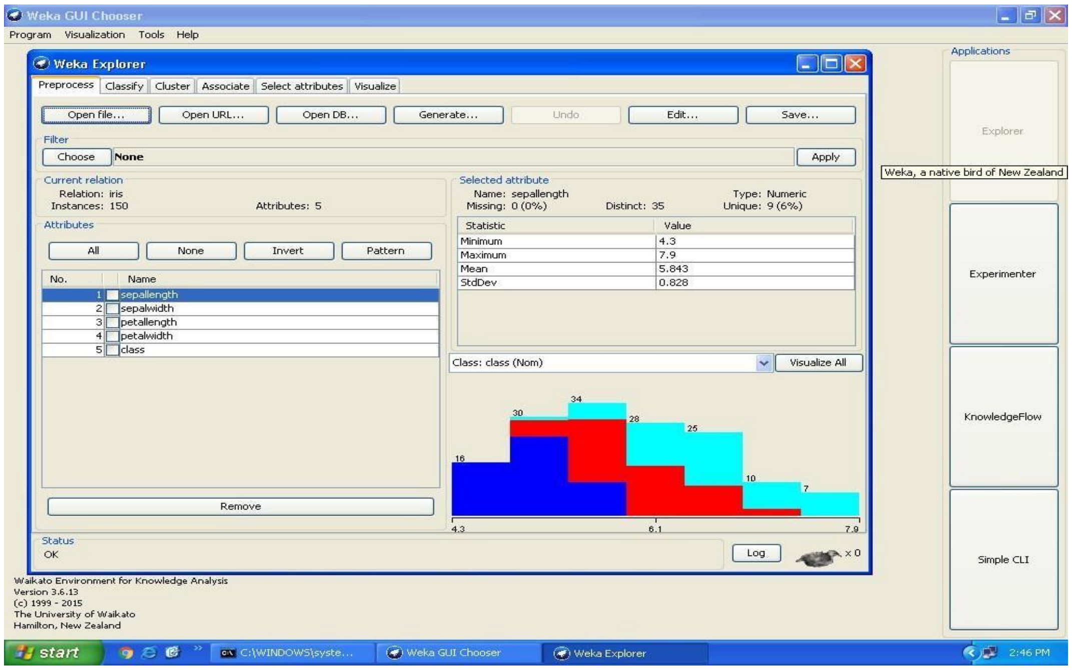
The Knowledge Flow offers the following features:

- **Intuitive** data flow style layout.
- **Process** data in batches or incrementally.
- **Process multiple batches** or streams in parallel (each separate flow executes in its own thread) .
- **Process multiple streams sequentially** via a user-specified order of execution.
- **Chain filters** together.
- **View models** produced by classifiers for each fold in a cross validation.
- **Visualize performance** of incremental classifiers during processing (scrolling plots of classification accuracy, RMS error, predictions etc.).
- **Plugin "perspectives" that add major new functionality (e.g. 3D data** visualization, time series forecasting environment etc.).

D.(iii)Navigate the options available in the WEKA(ex.select attributes panel,preprocess panel,classify panel,cluster panel,associate panel and visualize)

Ans: Steps for identify options 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. All tabs available in WEKA home page.



Filter: None

Current relation
Relation: iris
Instances: 150 Attributes: 5

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

Attributes

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

Class: class (Nom)

Status
OK

Waikato Environment for Knowledge Analysis
Version 3.6.13
(c) 1999 - 2015
The University of Waikato
Hamilton, New Zealand

Conclusion:- Hence we successfully explore WEKA tool.

Assignment: 2

Title: Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets.

Objective: To learn the Preprocessing and Association rule.

Theory:

A. Explore various options in Weka for Preprocessing data and apply (like DiscretizationFilters, 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.

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. 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.

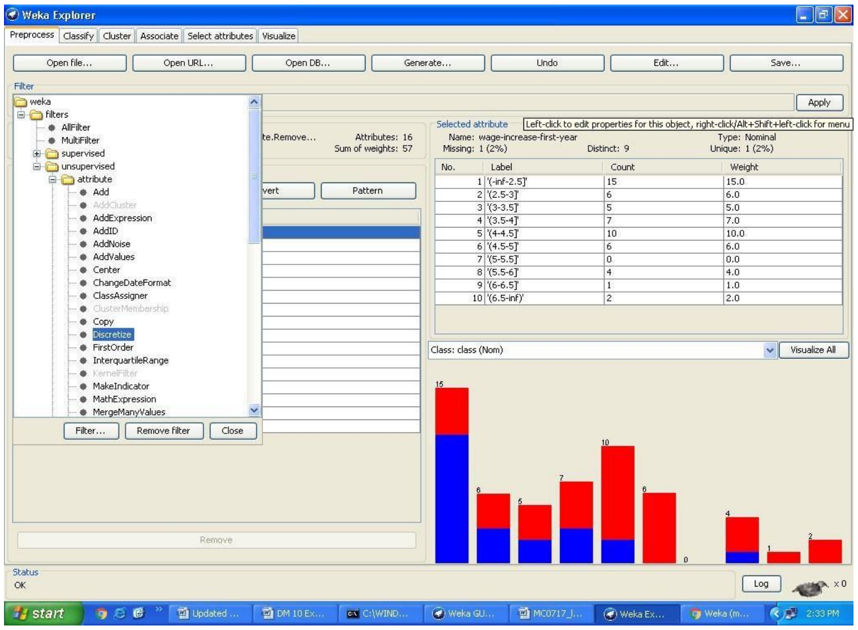
4. **Pattern.** Enables the user to select attributes based on a Perl 5 Regular Expression. E.g., .* id selects all attributes which name ends with id.

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.



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

Attributes: 16
Sum of weights: 57

Selected attribute
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)

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 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.

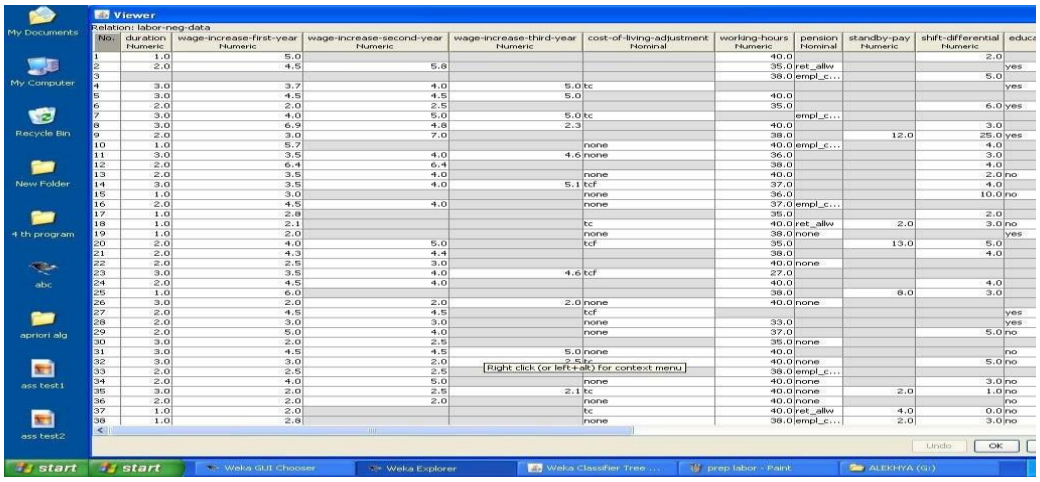
Left-Clicking on any of these gives an opportunity to alter the filters settings. For example, the setting may take a text string, in which case you type the string into the text field provided. Or it may give a drop-down box listing several states to choose from. Or it may do something else, depending 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

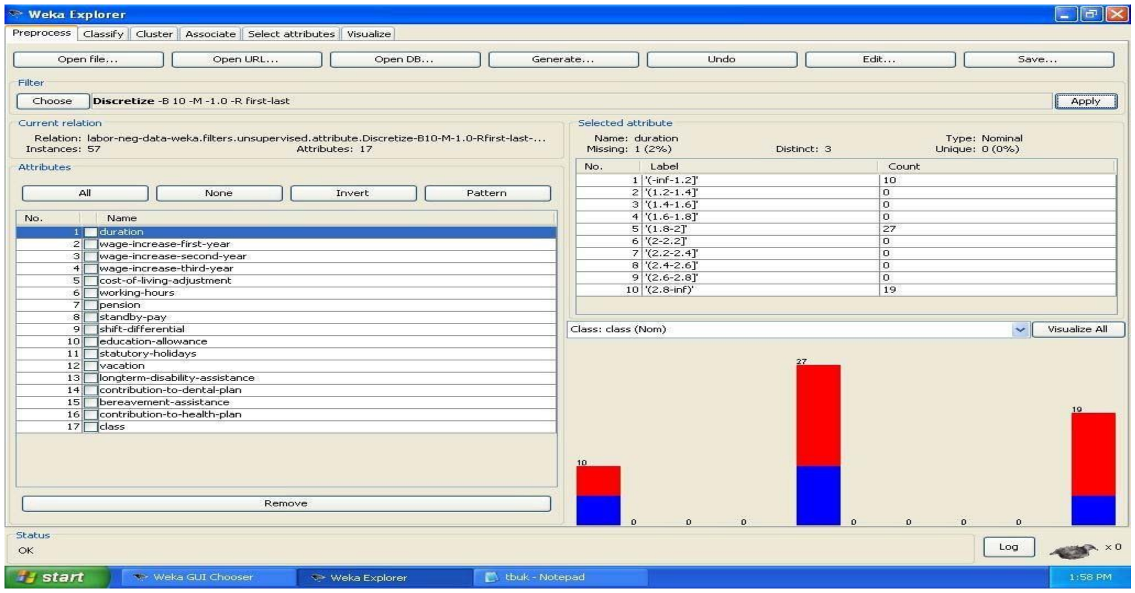
- Open WEKA Tool.
- Click on WEKA Explorer.
- Click on Preprocessing tab button.
- Click on open file button.
- Choose WEKA folder in C drive.
- Select and Click on data option button.
- Choose labor data set and open file.
- Choose filter button and select the Unsupervised-Discritize option and apply
- **Dataset labor.arff**


Viewer

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

The following screenshot shows the effect of discretization


Weka Explorer

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

Conclusion:- Hence we successfully learned data preprocessing tasks and Demonstrate performing association rule mining on data sets.

Assignment:3

Problem Statement: -

Aim: Demonstration of classification rule process on WEKA data-set using Naive Bayes algorithm.

Objectives:
To Learn The Demonstration of classification rule process on WEKA data-set using Naive Bayes algorithm.

Theory:-

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks. A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time.

In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.

Classifications are discrete and do not imply order. Continuous, floating point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm. The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating. In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown

Different Classification Algorithms: Oracle Data Mining provides the following algorithms for classification:

- Decision Tree - Decision trees automatically generate rules, which are conditional statements that reveal the logic used to build the tree.

- Naive Bayes - Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

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) you can once again copy the setup string to the clipboard or display the properties in a GenericObjectEditor dialog box. The Choose button allows you to choose one of the classifiers that are available in WEKA.

Test Options

The result of applying the chosen classifier will be tested according to the options that are set byclicking 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 wastrained on.

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

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

4. Percentage split. The classifier is evaluated on how well it predicts a certain percentage of the datawhich 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 databefore it is divided up for evaluation purposes.

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

11. Output source code. If the classifier can output the built model as Java source code, you canspecify 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 canonly 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 topredict a different attribute, click on the box below the Test options box to bring up a drop-downlist 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 perform Naïve-bayes classification and k- Nearest Neighbor classification, Interpret the results obtained.

AIM: Determining and classifying the credit good or bad in the dataset with an Accuracy.

THEORY:

Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even though these features depend on the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix The naive Bayes probabilistic model :

The probability model for a classifier is a conditional model.

P(C|F1 .... Fn) over a dependent class variable *C* with a small number of outcomes or *classes*, conditional on several feature variables *F*1 through *Fn*. The problem is that if the number of features *n* is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes‟ theorem, we write

P(C|F1...............Fn)=[{p(C)p(F1..................Fn|C)}/p(F1, Fn)]

In plain English the above equation can be written as

Posterior= [(prior *likehood)/evidence]

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on *C* and the values of the features *Fi* are given, so that the denominator is effectively constant.

Now the "naive" conditional independence assumptions come into play: assume that each feature *Fi* is conditionally independent of every other feature *Fj* .

This means that p(Fi|C,Fj)=p(Fi|C) and so the joint model can beexpressed as p(C,F1,.......Fn)=p(C)p(F1|C)p(F2|C) ...... = p(C)π p(Fi|C).

This means that under the above independence assumptions, the conditional distribution over the class variable *C* can be expressed like this:

p(C|F1. .... Fn)= p(C) πp(Fi|C) Z

where *Z* is a scaling factor dependent only on F1.........Fn, i.e., a constant if the values of the feature variables are known.

Models of this form are much more manageable, since they factor into a so called *class prior p*(*C*) and independent probability distributions p(Fi|C). If there are *k* classes and if a model for each p(Fi|C=c) can be expressed in terms of *r* parameters, then the corresponding naive Bayes model has $(k - 1) + n\ r\ k$ parameters. In practice, often $k = 2$ (binary classification) and $r = 1$ (Bernoulli variables as features) are common, and so the total number of parameters of the naive Bayes model is$2n + 1$, where *n* is the number of binary features used for prediction

P(h/D)= P(D/h) P(h) P(D)

- P(h) : Prior probability of hypothesis h
- P(D) : Prior probability of training data D
- P(h/D) : Probability of h given D
- P(D/h) : Probability of D given h

Naïve Bayes Classifier : Derivation

• D : Set of tuples

– Each Tuple is an „n" dimensional attribute vector

– X : (x1,x2,x3,…. xn)

- Let there me „m" Classes : C1,C2,C3…Cm
- NB classifier predicts X belongs to Class Ci iff

– P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i

- Maximum Posteriori Hypothesis

– P(Ci/X) = P(X/Ci) P(Ci) / P(X)

– Maximize P(X/Ci) P(Ci) as P(X) is

- With many attributes, it is computationally expensive to evaluate P(X/Ci)

- Naïve Assumption of “class conditional independence”

- P(X/Ci) = n P(xk/ Ci)

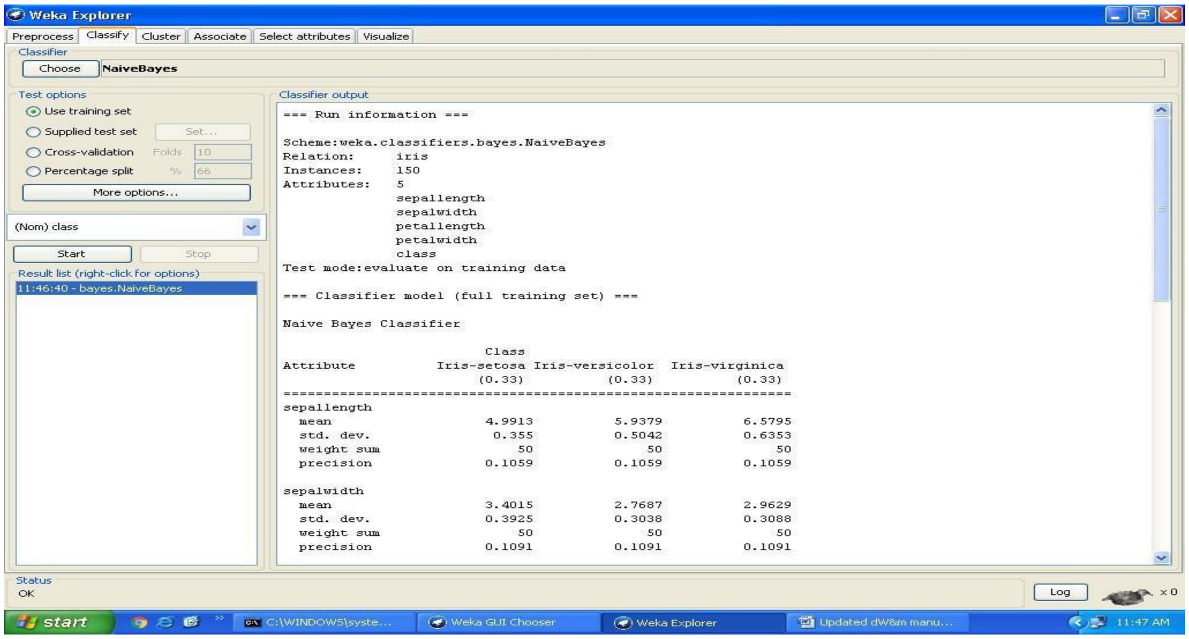
• P(X/Ci) = P(x1/Ci) * P(x2/Ci) *…* P(xn/ Ci).

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

- Open WEKA Tool.
- Click on WEKA Explorer.
  - Click on Preprocessing tab button.
  - Click on open file button.
  - Choose WEKA folder in C drive.
- Select and Click on data option button.
- Choose iris data set and open file.
- Click on classify tab and Choose Naïve-bayes algorithm and select use training set testoption.

- Click on start button.
- Click on classify tab and Choose k-nearest neighbor and select use training set test option.
- Click on start button.

OUTPUT:



Classifier: NaiveBayes

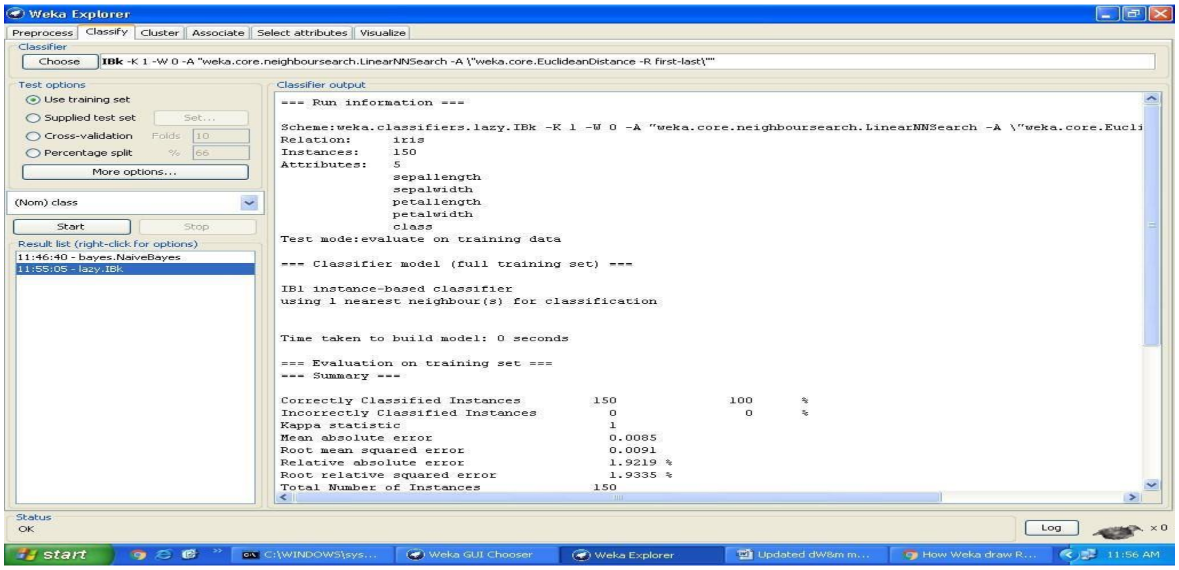
Test options: [✓ Use training set] Supplied test set / Cross-validation Folds 10 / Percentage split % 66

(Nom) class

Result list (right-click for options): 11:46:40 - bayes.NaiveBayes

Status: OK

```
=== 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
```



Classifier: IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options: [✓ Use training set] Supplied test set / Cross-validation Folds 10 / Percentage split % 66

(Nom) class

Result list (right-click for options): 11:46:40 - bayes.NaiveBayes; 11:55:05 - lazy.IBk

Status: OK

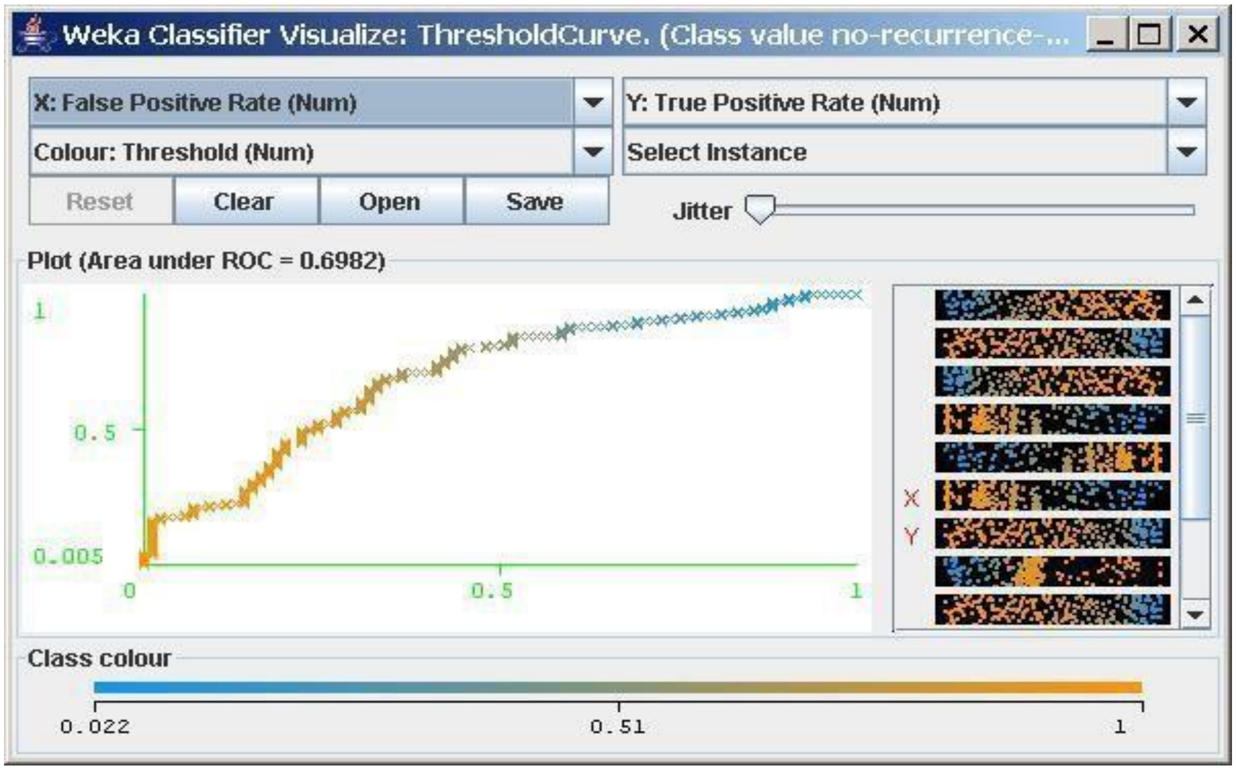
```
=== Run information ===

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.Eucli
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
```

Plot RoC Curves.

Ans: Steps for identify the plot RoC Curves.

1. Open WEKA Tool.

2. Click on WEKA Explorer.
3. Click on Visualize button.
4. Click on right click button.
5. Select and Click on polyline option button.


Weka Classifier Visualize: ThresholdCurve. (Class value no-recurrence-...
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.6982)
1
0.5
0.005
0
0.5
1
X
Y
Class colour
0.022
0.51
1

Viva voice questions

1. **What is K-nearest neighbor algorithm?**
   It is one of the lazy learner algorithm used in classification. It finds the k-nearestneighbor of the point of interest.

2. What are the issues regarding classification and prediction?

   Preparing data for classification and prediction Comparing classification and prediction

3. **What is decision tree classifier?**

A decision tree is an hierarchically based classifier which compares data with a range of properly selected features.

4. **What is multimedia data mining?**

Multimedia Data Mining is a subfield of data mining that deals with an extraction of implicit knowledge, multimedia data relationships, or other patterns not explicitly stored in multimedia databases.

5. **What is text mining?**

Text mining is the procedure of synthesizing information, by analyzing relations, patterns, and rules among textual data. These procedures contains text summarization, text categorization, and text clustering.

6. **What is Naïve Bayes Algorithm?**

Naïve Bayes Algorithm is used to generate mining models. These models help to identify relationships between input columns and the predictable columns. This algorithm can be used in the initial stage of exploration. The algorithm calculates the probability of every state of each input column given predictable columns possible states. After the model is made, the results can be used for exploration and making predictions.

7. **What is distributed data warehouse?**

Distributed data warehouse shares data across multiple data repositories for the purpose of OLAP operation.

8. **What is are different data warehouse model?**

    Enterprise data ware houseData marts
    Virtual Data warehouse

9. **What are issues in data mining?**

    Issues in mining methodology, performance issues, user interactive issues, different source of data types issues etc

10. **What are frequent pattern?**
    a. A set of items that appear frequently together in a transaction data set.
    b. eg milk, bread, sugar

Conclusion:- Hence we learned classification rule process on WEKA data-set using Naive Bayes algorithm.

Assignment: 4

Problem Statement:- Demonstrate performing Regression on data sets

Aim :- To Study Regression On Data Sets.

Theory:-
Regression refers to a data mining technique that is used to predict the numeric values in a given data set. For example, regression might be used to predict the product or service cost or other variables. It is also used in various industries for business and marketing behavior, trend analysis, and financial forecast. In this tutorial, we will understand the concept of regression, types of regression with certain examples.

- **What is regression?**

Regression refers to a type of supervised machine learning technique that is used to predict any continuous-valued attribute. Regression helps any business organization to analyze the target variable and predictor variable relationships. It is a most significant tool to analyze the data that can be used for financial forecasting and time series modeling.

Regression involves the technique of fitting a straight line or a curve on numerous data points. It happens in such a way that the distance between the data points and cure comes out to be the lowest.

The most popular types of regression are linear and logistic regressions. Other than that, many other types of regression can be performed depending on their performance on an individual data set.

Regression can predict all the dependent data sets, expressed in the expression of independent variables, and the trend is available for a finite period. Regression provides a good way to predict variables, but there are certain restrictions and assumptions like the independence of the variables, inherent normal distributions of the variables. For example, suppose one considers two variables, A and B, and their joint distribution is a bivariate distribution, then by that nature. In that case, these two variables might be independent, but they are also correlated. The marginal distributions of A and B need to be derived and used. Before applying Regression analysis, the data needs to be studied carefully and perform certain preliminary tests to ensure the Regression is applicable. There are non-Parametric tests that are available in such cases.

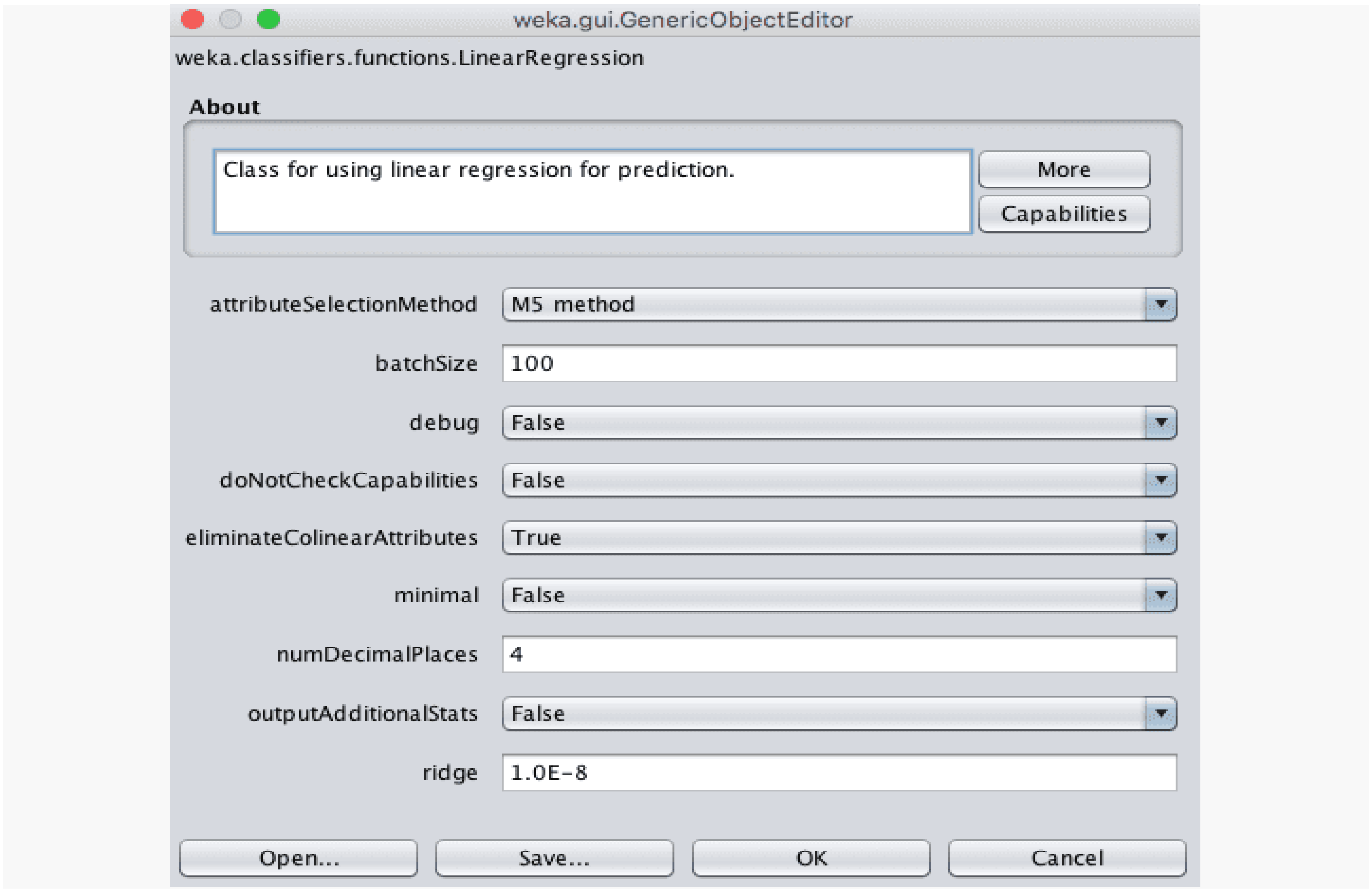
Linear regression only supports regression type problems.

It works by estimating coefficients for a line or hyperplane that best fits the training data. It is a very simple regression algorithm, fast to train and can have great performance if the output variable for your data is a linear combination of your inputs.

It is good idea to evaluate linear regression on your problem before moving onto more complex algorithms in case it performs well.

Choose the linear regression algorithm:

1. Click the “Choose” button and select “LinearRegression” under the “functions” group.
2. Click on the name of the algorithm to review the algorithm configuration.



weka.gui.GenericObjectEditor

weka.classifiers.functions.LinearRegression

About

Class for using linear regression for prediction.

attributeSelectionMethod	M5 method
batchSize	100
debug	False
doNotCheckCapabilities	False
eliminateColinearAttributes	True
minimal	False
numDecimalPlaces	4
outputAdditionalStats	False
ridge	1.0E-8

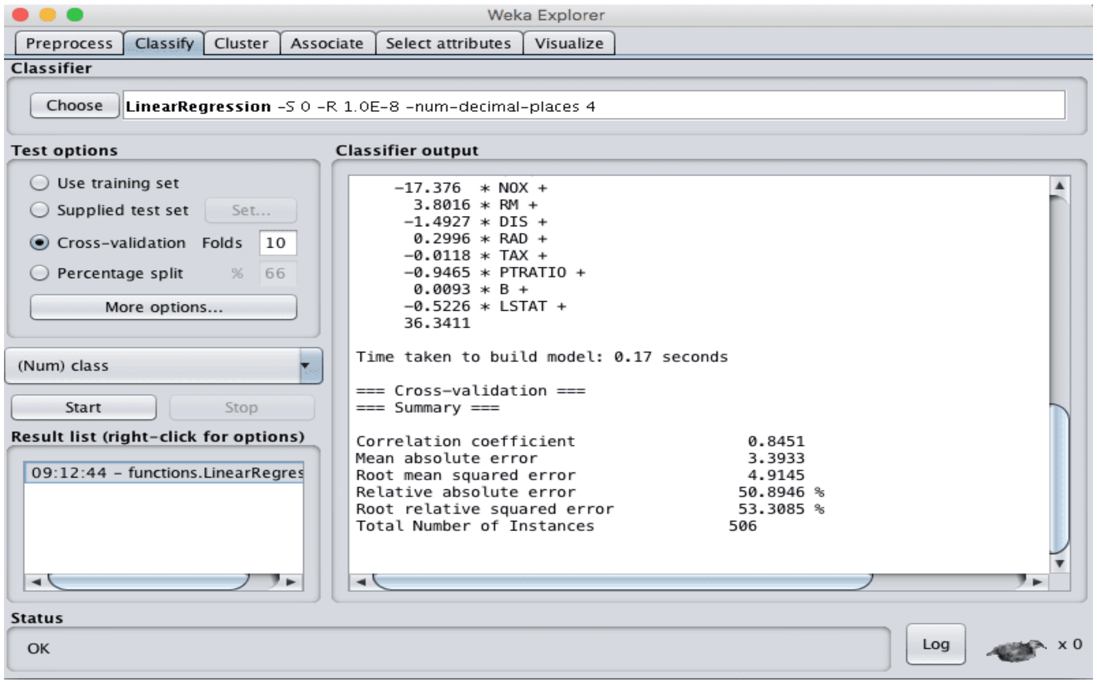
The performance of linear regression can be reduced if your training data has input attributes that are highly correlated. Weka can detect and remove highly correlated input attributes automatically by setting eliminateColinearAttributes to True, which is the default.

Additionally, attributes that are unrelated to the output variable can also negatively impact performance. Weka can automatically perform feature selection to only select those relevant attributes by setting the attributeSelectionMethod. This is enabled by default and can be disabled.

Finally, the Weka implementation uses a ridge regularization technique in order to reduce the complexity of the learned model. It does this by minimizing the square of the absolute sum of the learned coefficients, which will prevent any specific coefficient from becoming too large (a sign of complexity in regression models).

1. Click "OK" to close the algorithm configuration.
2. Click the "Start" button to run the algorithm on the Boston house price dataset.

You can see that with the default configuration that linear regression achieves an RMSE of 4.9.



LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4

Cross-validation Folds 10

Percentage split % 66

09:12:44 - functions.LinearRegres

```
     -17.376  * NOX +
       3.8016 * RM +
      -1.4927 * DIS +
       0.2996 * RAD +
      -0.0118 * TAX +
      -0.9465 * PTRATIO +
       0.0093 * B +
      -0.5226 * LSTAT +
      36.3411

Time taken to build model: 0.17 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.8451
Mean absolute error                      3.3933
Root mean squared error                  4.9145
Relative absolute error                 50.8946 %
Root relative squared error             53.3085 %
Total Number of Instances              506
```

OK

Conclusion:- Hence we understood demonstrate performing Regression on data sets

Assignment: 5

Problem Statement:- Demonstration of clustering rule process on data-set iris.arff using simple k-means

Aim :- To Study clustering rule Process.

Theory:-

length and breadth of rectangleSelecting 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 andPercentage 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 attributein 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 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.

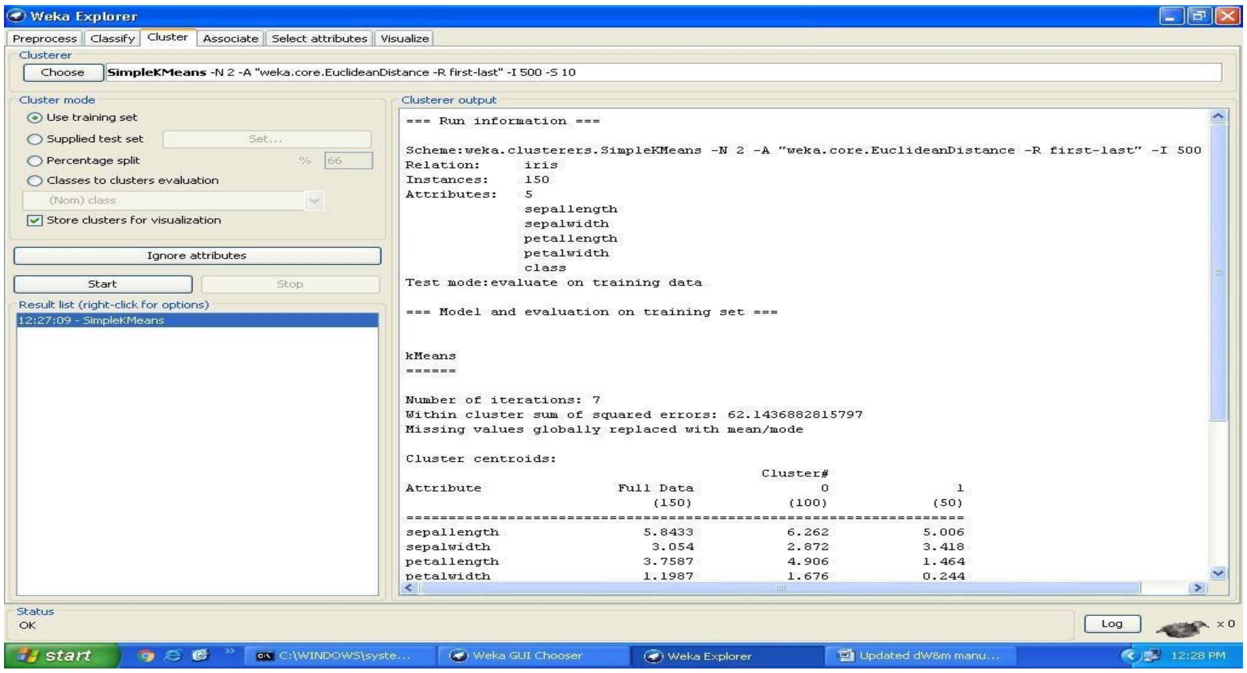
A. Load each dataset into Weka and run simple k-means clustering algorithm with differentvalues of k(number of desired clusters). Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.

Ans:

 Steps for run K-mean Clustering algorithms in WEKA

- Open WEKA Tool.
- Click on WEKA Explorer.
- Click on Preprocessing tab button.
- Click on open file button.
- Choose WEKA folder in C drive.
- Select and Click on data option button.
- Choose iris data set and open file.
- Click on cluster tab and Choose k-mean and select use training set test option.
- Click on start button.

OUTPUT:



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

Conclusion:- Hence we understood demonstration of clustering rule process on data-set iris.arff using simple k-means

Assignment: 6

Problem Statement:

Write a program of Apriori algorithm using any programming language.

Aim :

To understand Apriori Algorithm.

Theory

Apriori Algorithm is a Machine Learning algorithm utilized to understand the patterns of relationships among the various products involved. The most popular use of the algorithm is to suggest products based on the items already in the user's shopping cart. Walmart specifically has utilized the algorithm in recommending items to its users.

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example, the items customers but at a Big Bazar.

Apriori algorithm helps the customers to buy their products with ease and increases the sales performance of the particular store.

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Let's take an example to understand this concept.

We have already discussed above; you need a huge database containing a large no of transactions. Suppose you have 4000 customers transactions in a Big Bazar. You have to calculate the Support, Confidence, and Lift for two products, and you may say Biscuits and Chocolate. This is because customers frequently buy these two items together.

Out of 4000 transactions, 400 contain Biscuits, whereas 600 contain Chocolate, and these 600 transactions include a 200 that includes Biscuits and chocolates. Using this data, we will find out the support, confidence, and lift.

Support

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)

= 400/4000 = 10 percent.

Confidence

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

Hence,

Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

It means that 50 percent of customers who bought biscuits bought chocolates also.

Lift

Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

= 50/10 = 5

It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both the items together. Larger the value, the better is the combination.

Implementing Apriori Algorithm in Python

1. Step 1: Import the required libraries. import numpy as np. ...
2. Step 2: Load and explore the data. ...
3. Step 3: Clean the Data. ...
4. Step 4: Split the data according to the region of transaction. ...
5. Step 5: Hot encoding the Data. ...
6. Step 6: Build the models and analyse the results.

Step 1: Import the required libraries

```
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

Step 2: Load and explore the data

```
# Now, we will load the Data
data1 = pnd.read_excel('Online_Retail.xlsx')
data1.head()
```

Output:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID	8	2010-12-01	2.75	17850.0	United Kingdo

			HEARTS COAT HANGER		08:26:00			m
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Input:

1. # here, we will explore the columns of the data
2. data1.columns

Output:

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      Dtype = 'object')
```

Input:

1. # Now, we will explore the different regions of transactions
2. data1.Country.unique()

Output:

```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype = object)
```

Step 3: Clean the Data

```
# here, we will strip the extra spaces in the description
data1['Description'] = data1['Description'].str.strip()

# Now, drop the rows which does not have any invoice number
data1.dropna(axis = 0, subset = ['InvoiceNo'], inplace = True)
data1['InvoiceNo'] = data1['InvoiceNo'].astype('str')

# Now, we will drop all transactions which were done on credit
data1 = data1[~data1['InvoiceNo'].str.contains('C')]
```

Step 4: Split the data according to the region of transaction

```
# Transactions done in France
basket1_France = (data1[data1['Country'] == "France"]
        .groupby(['InvoiceNo', 'Description'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))

# Transactions done in the United Kingdom
basket1_UK = (data1[data1['Country'] == "United Kingdom"]
        .groupby(['InvoiceNo', 'Description'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))

# Transactions done in Portugal
basket1_Por = (data1[data1['Country'] == "Portugal"]
        .groupby(['InvoiceNo', 'Description'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))
```

```

basket1_Sweden = (data1[data1['Country'] == "Sweden"]
        .groupby(['InvoiceNo', 'Description'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('InvoiceNo'))
```

Step 5: Hot encoding the Data

```
# Here, we will define the hot encoding function
# for making the data suitable
# for the concerned libraries
def hot_encode1(P):
    if(P<= 0):
        return 0
    if(P>= 1):
        return 1

# Here, we will encode the datasets
basket1_encoded = basket1_France.applymap(hot_encode1)
basket1_France = basket1_encoded

basket1_encoded = basket1_UK.applymap(hot_encode1)
basket1_UK = basket1_encoded

basket1_encoded = basket1_Por.applymap(hot_encode1)
basket1_Por = basket1_encoded

basket1_encoded = basket1_Sweden.applymap(hot_encode1)
basket1_Sweden = basket1_encoded
```

Step 6: Build the models and analyse the results

a) France:

```
# Build the model
```

```
frq_items1 = AP(basket1_France, min_support = 0.05, use_colnames = True)

# Collect the inferred rules in a dataframe
rules1 = AR(frq_items1, metric = "lift", min_threshold = 1)
rules1 = rules1.sort_values(['confidence', 'lift'], ascending = [False, False])
print(rules1.head())
```

Output:

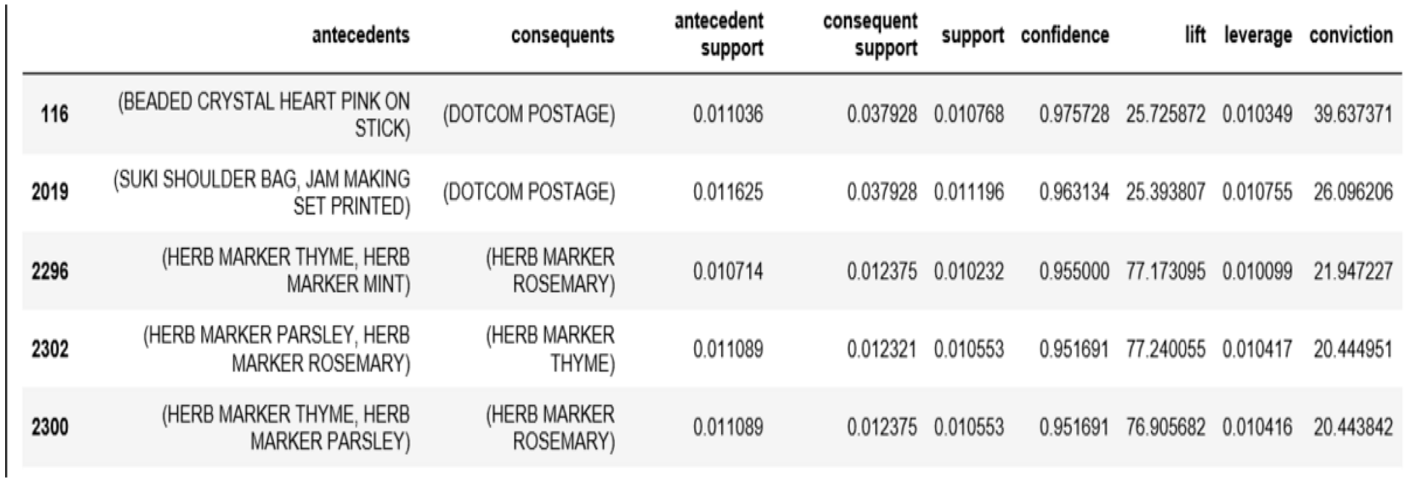
```
antecedents \
45 (JUMBO BAG WOODLAND ANIMALS)
260 (PLASTERS IN TIN CIRCUS PARADE, RED TOADSTOOL ...
272 (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
302 (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...
301 (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...

consequents antecedent support consequent support \
45 (POSTAGE) 0.076531 0.765306
260 (POSTAGE) 0.051020 0.765306
272 (POSTAGE) 0.053571 0.765306
302 (SET/6 RED SPOTTY PAPER PLATES) 0.102041s 0.127551
301 (SET/6 RED SPOTTY PAPER CUPS) 0.102041 0.137755

support confidence lift leverage conviction
45 0.076531 1.000 1.306667 0.017961 inf
260 0.051020 1.000 1.306667 0.011974 inf
272 0.053571 1.000 1.306667 0.012573 inf
```

302 0.099490 0.975 7.644000 0.086474 34.897959
301 0.099490 0.975 7.077778 0.085433 34.489796

From the above output, it can be seen that paper cups, paper and plates are bought together in France. This is because the French has a culture of having a get-together with their friends and family at least once a week. Also, since the French government has banned the use of plastic in the country, people have to purchase paper-based alternatives.

b) United Kingdom:

```
frq_items = apriori(basket_UK, min_support = 0.01, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```



	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
116	(BEADED CRYSTAL HEART PINK ON STICK)	(DOTCOM POSTAGE)	0.011036	0.037928	0.010768	0.975728	25.725872	0.010349	39.637371
2019	(SUKI SHOULDER BAG, JAM MAKING SET PRINTED)	(DOTCOM POSTAGE)	0.011625	0.037928	0.011196	0.963134	25.393807	0.010755	26.096206
2296	(HERB MARKER THYME, HERB MARKER MINT)	(HERB MARKER ROSEMARY)	0.010714	0.012375	0.010232	0.955000	77.173095	0.010099	21.947227
2302	(HERB MARKER PARSLEY, HERB MARKER ROSEMARY)	(HERB MARKER THYME)	0.011089	0.012321	0.010553	0.951691	77.240055	0.010417	20.444951
2300	(HERB MARKER THYME, HERB MARKER PARSLEY)	(HERB MARKER ROSEMARY)	0.011089	0.012375	0.010553	0.951691	76.905682	0.010416	20.443842

If the guidelines for British transactions are examined in greater detail, it is discovered that British consumers purchase various colored tea plates. The reason could be due to the fact that the British love tea and tend to collect different colors of tea plates to suit different occasions.

c) Portugal:

```
frq_items1 = AP(basket1_Sweden, min_support = 0.05, use_colnames = True)
rules1 = AR(frq_items1, metric ="lift", min_threshold = 1)
```

3. rules1 = rules1.sort_values(['confidence', 'lift'], ascending =[False, False])
4. **print**(rules1.head())
   Output:

ntecedents consequents \
1170 (SET 12 COLOUR PENCILS DOLLY GIRL) (SET 12 COLOUR PENCILS SPACEBOY)
1171 (SET 12 COLOUR PENCILS SPACEBOY) (SET 12 COLOUR PENCILS DOLLY GIRL)
1172 (SET OF 4 KNICK KNACK TINS LONDON) (SET 12 COLOUR PENCILS DOLLY GIRL)
1173 (SET 12 COLOUR PENCILS DOLLY GIRL) (SET OF 4 KNICK KNACK TINS LONDON)
1174 (SET 12 COLOUR PENCILS DOLLY GIRL) (SET OF 4 KNICK KNACK TINS POPPIES)
antecedent support consequent support support confidence lift \
1170 0.051724 0.051724 0.051724 1.0 19.333333
1171 0.051724 0.051724 0.051724 1.0 19.333333
1172 0.051724 0.051724 0.051724 1.0 19.333333
1173 0.051724 0.051724 0.051724 1.0 19.333333
1174 0.051724 0.051724 0.051724 1.0 19.333333
leverage conviction
1170 0.049049 inf
1171 0.049049 inf
1172 0.049049 inf
1173 0.049049 inf

```
1174 0.049049 inf
```

In analysing the association regulations to Portuguese transactions, the use of Tiffin set (Knick Knack Tins) and colour pencils can be found. These two items are typically belonging to a primary school child. Both of these items are needed by students at school to carry their lunches as well as for work that requires creativity, and therefore it is logical to pair them together.

d) Sweden:

```
frq_items1 = AP(basket1_Sweden, min_support = 0.05, use_colnames = True)
rules1 = AR(frq_items1, metric ="lift", min_threshold = 1)
rules1 = rules1.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules1.head())
```

Output:

```
antecedents consequents \
0 (PACK OF 72 SKULL CAKE CASES) (12 PENCILS SMALL TUBE SKULL)
1 (12 PENCILS SMALL TUBE SKULL) (PACK OF 72 SKULL CAKE CASES)
4 (36 DOILIES DOLLY GIRL) (ASSORTED BOTTLE TOP MAGNETS)
5 (ASSORTED BOTTLE TOP MAGNETS) (36 DOILIES DOLLY GIRL)
180 (CHILDRENS CUTLERY DOLLY GIRL) (CHILDRENS CUTLERY CIRCUS PARADE)

antecedent support consequent support support confidence lift \
0 0.055556 0.055556 0.055556 1.0 18.0
1 0.055556 0.055556 0.055556 1.0 18.0
4 0.055556 0.055556 0.055556 1.0 18.0
5 0.055556 0.055556 0.055556 1.0 18.0
180 0.055556 0.055556 0.055556 1.0 18.0
```

leverage conviction
0 0.052469 inf
1 0.052469 inf
4 0.052469 inf
5 0.052469 inf
180 0.052469 inf

Analysing the above guidelines and the above rules, we find that both girls' and boys' cutlery are placed together. This makes sense since when a parent shops for cutlery items for their children, they would like the item to be specific to the child's desires.

Conclusion:- Hence we clear concept of a Apriori algorithm.

Assignment: 8

Title:- Build a spam filter using Python and the Naive Bayes algorithm.

Aim:- To Study Naïve Bayes algorithm.

Theory:

Naive Bayes is a probabilistic algorithm based on the Bayes Theorem used for email spam filtering in data analytics. If you have an email account, we are sure that you have seen emails being categorised into different buckets and automatically being marked important, spam, promotions, etc. Isn't it wonderful to see machines being so smart and doing the work for you?

More often than not, these labels added by the system are right. So does this mean our email software is reading through every communication and now understands what you as a user would have done? Absolutely right! In this age and time of data analytics & machine learning, automated filtering of emails happens via algorithms like Naive Bayes Classifier, which apply the basic Bayes Theorem on the data.

One of the most simple yet powerful classifier algorithms, Naive Bayes is based on Bayes' Theorem Formula with an assumption of independence among predictors. Given a Hypothesis A and evidence B, Bayes' Theorem calculator states that the relationship between the probability of Hypothesis before getting the evidence P(A) and the probability of the hypothesis after getting the evidence P(A|B) is:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Here:

- A, B = events
- P(A|B) = probability of A given B is true
- P(B|A) = probability of B given A is true
- P(A), P(B) = the independent probabilities of A and B

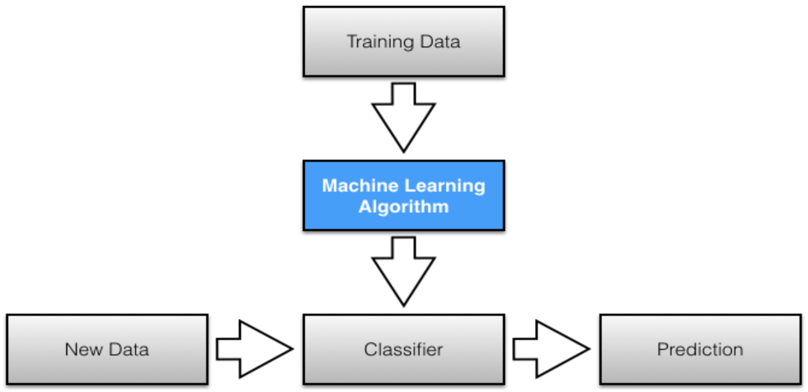
This theorem, as explained in one of our previous articles, is mainly used for classification techniques in data analytics. The Naive Bayes theorem calculator pays an important role in spam detection of emails.

Detecting Email Spam

Modern spam filtering software continuously struggles to categorise the emails correctly. Unwanted spam & promotional communication is the toughest of them all. Spam communication algorithms must be iterated continuously since there is an ongoing battle between spam filtering software and anonymous spam & promotional mail senders. Naive Bayes Algorithm in data analytics forms the base for text filtering in Gmail, Yahoo Mail, Hotmail & all other platforms.

Like Naive Bayes, other classifier algorithms like Support Vector Machine, or Neural Network also get the job done! Before we begin, here is the dataset for you to download:

Email Spam Filtering Using Naive Bayes Algorithm
This would be a zipped file, attached in the email. Please allow users to download this data.



Conclusion:- Hence understand that how to apply naïve bayes classification for spam filter detection.

Assignment: 9

Title:- Classify DDoS attacks with Artificial Intelligence

Aim:- To Study DDos Attacks.

Theory:

In a DDoS attack, the attacker tries to make a particular service unavailable by directing continuous and huge traffic from multiple end systems. Due to this enormous traffic, the network resources get utilized in serving requests of those false end systems such that, a legitimate user is unable to access the resources for themselves.

How Does a DDoS Attack Work?

A denial-of-service (DDoS) attack is essentially an excessive use of a valid online service. For instance, a website might be able to process a specific amount of requests per minute. The website may become completely unusable if that number is surpassed, or its functionality may be negatively impacted. An attack or even a legitimate use, like an e-commerce site experiencing overflow on Black Friday or a ticket sales platform experiencing a glitch when sales for a big event begin, could be the cause of this overload.

A DDoS attack is launched from multiple compromised devices which are present in different places over the globe, this such devices are referred to as botnets. It differs from other denial of service (DoS) attacks in that it floods a target with malicious traffic using a single network connection or Internet-connected device.

Types of DDoS Attacks

DDoS attacks can be divided into three major categories:

1. **Application Layer Attacks:** These attacks focus on attacking layer 7 of the OSI model where the webpages are generated in response to the request initiated by the end-user. For a client, generating a request does not take any heavy load and it can easily generate multiple requests to the server. On the other hand, responding to a request takes a considerable load for the server as it has to build all the pages, compute any queries, and load the results from the database according to the request.

Examples: HTTP Flood attack and attack on DNS Services.

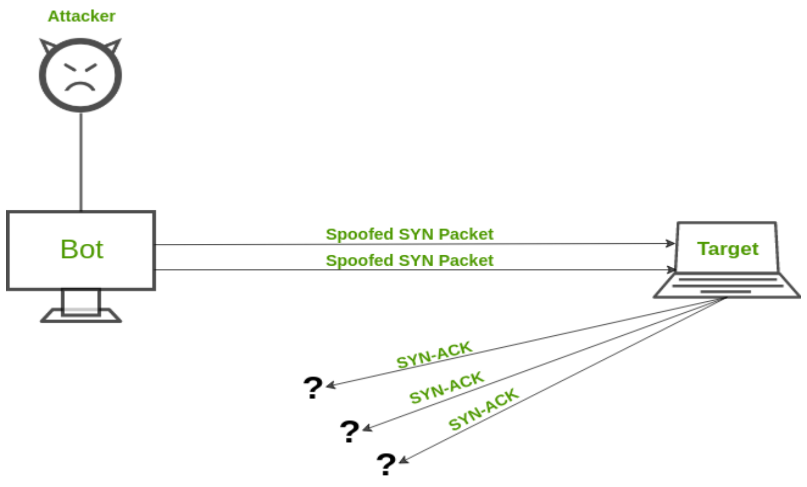
2. **Protocol Attacks:** They are also known as state-exhaustion attacks. These attacks focus on vulnerabilities in layer 3 and layer 4 of the protocol stack. These types of attacks consume resources like servers, firewalls, and load balancers.

Examples: SYN Flood attack and Ping of Death.

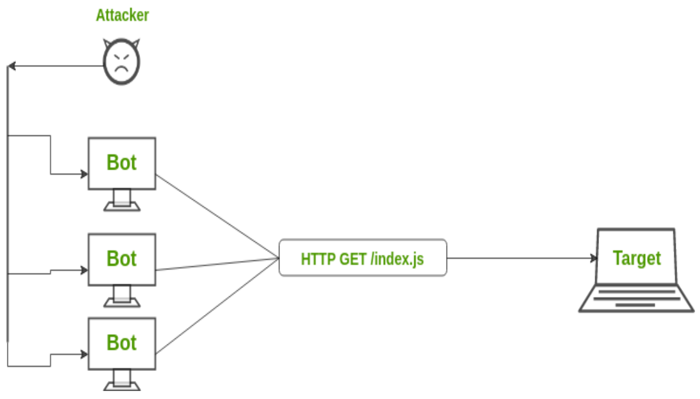
3. **Volumetric Attacks:** Volumetric attacks focus on consuming the network bandwidth and saturating it by amplification or botnet to hinder its availability to the users. They are easy to generate by directing a massive amount of traffic to the target server. **Examples:** NTP Amplification, DNS Amplification, UDP Flood attack, and TCP Flood attack.

Common DDoS Attacks

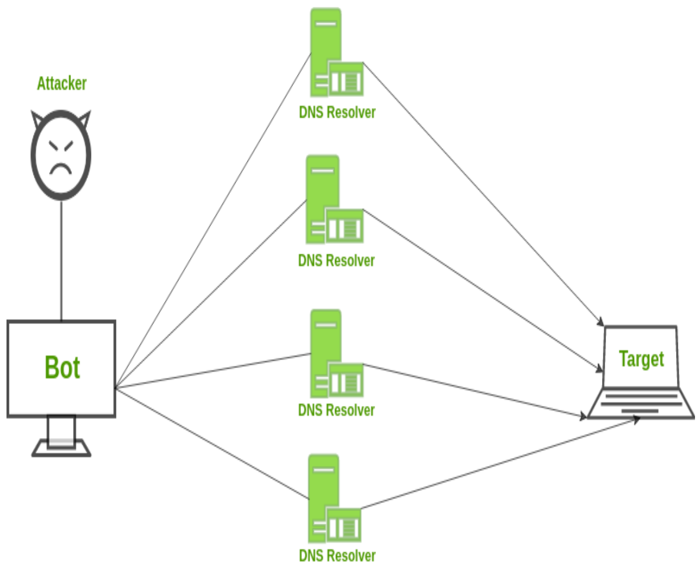
- **SYN Flood Attack:** An SYN Flood attack works similarly a mischievous child keeps on ringing the doorbell (request) and running away. The old person inside comes out, opens the door, and does not see anyone (no response). Ultimately, after frequent such scenarios, the old person gets exhausted and does not answer even genuine people. An SYN attack exploits TCP Handshake by sending out SYN messages with a spoofed IP address. The victim server keeps on responding but does not receive a final acknowledgment.



- **HTTP Flood Attack:** In an HTTP Flood attack, multiple HTTP requests are generated simultaneously against a target server. This leads to the exhaustion of network resources of that server and thus fails to serve actual users' requests. The variations of HTTP Flood attacks are – HTTP GET attacks and HTTP POST attacks.



- **DNS Amplification:** Assume a scenario where you call Pizza Hut and ask them to call you back on a number and tell all the combinations of pizzas they have along with the toppings and desserts. You generated a large output with a very small input. But, the catch is the number you gave them is not yours. Similarly, DNS Amplification works by requesting a DNS server from a spoofed IP address and structuring your request so that the DNS server responds with a large amount of data to the target victim.



Conclusion:- Hence we learned ai base approach are becoming an essential tool for modern security.

Assignment: 10

Title:- Split sample data into training and test sets. (Use suitable data set).

Aim:- To Study Sample Data Spliting.

Theory:

The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results. By default, the Test set is split into 30 % of actual data and the training set is split into 70% of the actual data.

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

Dataset Splitting:

Scikit-learn alias sklearn is the most useful and robust library for machine learning in Python. The scikit-learn library provides us with the model_selection module in which we have the splitter function train_test_split().
Syntax:

```
rain_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)
```

Parameters:

1. *arrays: inputs such as lists, arrays, data frames, or matrices
2. test_size: this is a float value whose value ranges between 0.0 and 1.0. it represents the proportion of our test size. its default value is none.
3. train_size: this is a float value whose value ranges between 0.0 and 1.0. it represents the proportion of our train size. its default value is none.
4. random_state: this parameter is used to control the shuffling applied to the data before applying the split. it acts as a seed.
5. shuffle: This parameter is used to shuffle the data before splitting. Its default value is true.
6. stratify: This parameter is used to split the data in a stratified fashion.

Conclusion:- In this practical we have demonstrated how to split data into training and test sets.

Assignment: 11

Title:- Perform feature engineering operations on raw data. (Use suitable data set).

Aim:- To Study Feature Engineering Operations.

Theory:

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.

What is Feature Engineering?

Feature engineering is the process of **transforming raw data into features that are suitable for machine learning models**. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.
The success of machine learning models heavily depends on the quality of the features used to train them. Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.

Processes Involved in Feature Engineering

Feature engineering in Machine learning consists of mainly 5 processes: Feature Creation, Feature Transformation, Feature Extraction, Feature Selection, and Feature Scaling. It is an iterative process that requires experimentation and testing to find the best combination of features for a given problem. The success of a machine learning model largely depends on the quality of the features used in the model.

1. Feature Creation
Feature Creation is the process of generating new features based on domain knowledge or by observing patterns in the data. It is a form of feature engineering that can significantly improve the performance of a machine-learning model.

Types of Feature Creation:
1. Domain-Specific: Creating new features based on domain knowledge, such as creating features based on business rules or industry standards.
2. Data-Driven: Creating new features by observing patterns in the data, such as calculating aggregations or creating interaction features.

3. Synthetic: Generating new features by combining existing features or synthesizing new data points.

Feature Transformation

Feature Transformation is the process of transforming the features into a more suitable representation for the machine learning model. This is done to ensure that the model can effectively learn from the data.
Types of Feature Transformation:
1. Normalization: Rescaling the features to have a similar range, such as between 0 and 1, to prevent some features from dominating others.
2. Scaling: Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. Rescaling the features to have a similar scale, such as having a standard deviation of 1, to make sure the model considers all features equally.
3. Encoding: Transforming categorical features into a numerical representation. Examples are one-hot encoding and label encoding.
4. Transformation: Transforming the features using mathematical operations to change the distribution or scale of the features. Examples are logarithmic, square root, and reciprocal transformations.

Feature Extraction

Feature Extraction is the process of creating new features from existing ones to provide more relevant information to the machine learning model. This is done by transforming, combining, or aggregating existing features.
Types of Feature Extraction:
1. Dimensionality Reduction: Reducing the number of features by transforming the data into a lower-dimensional space while retaining important information. Examples are PCA and t-SNE.
2. Feature Combination: Combining two or more existing features to create a new one. For example, the interaction between two features.
3. Feature Aggregation: Aggregating features to create a new one. For example, calculating the mean, sum, or count of a set of features.
4. Feature Transformation: Transforming existing features into a new representation. For example, log transformation of a feature with a skewed distribution.

Feature Selection

Feature Selection is the process of selecting a subset of relevant features from the dataset to be used in a machine-learning model. It is an important step in the feature engineering process as it can have a significant impact on the model's performance.
Types of Feature Selection:
1. Filter Method: Based on the statistical measure of the relationship between the feature and the target variable. Features with a high correlation are selected.
2. Wrapper Method: Based on the evaluation of the feature subset using a specific machine learning algorithm. The feature subset that results in the best performance is selected.
3. Embedded Method: Based on the feature selection as part of the training process of the machine learning algorithm.

Feature Scaling

Feature Scaling is the process of transforming the features so that they have a similar scale. This is important in machine learning because the scale of the features can affect the performance of the model.
Types of Feature Scaling:

1. Min-Max Scaling: Rescaling the features to a specific range, such as between 0 and 1, by subtracting the minimum value and dividing by the range.
2. Standard Scaling: Rescaling the features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation.
3. Robust Scaling: Rescaling the features to be robust to outliers by dividing them by the interquartile range.

Conclusion:- Hence we learned feature engineering helps in increasing the accuracy and performance of the ML model.