Experiment-1

AIM: Creation of a Data Warehouse.

(i) Build Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration Tool, Pentaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects,etc.,)

(ii) Design multi-dimensional data models namely Star, Snowflake and Fact Constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, manufacturing, Automobiles, sales etc).

(iii) Write ETL scripts and implement using data warehouse tools.

(iv) Perform Various OLAP operations such as slice, dice, roll up, drill up and pivot

Description:

(i)The data warehouse contains 4 tables:

1. Date dimension: contains every single date from 2006 to 2016.

2. Customer dimension: contains 100 customers. To be simple we'll make it type 1 so we don't create a new row for each change.

3. Van dimension: contains 20 vans. To be simple we'll make it type 1 so we don't create a new row for each change.

4. Hire fact table: contains 1000 hire transactions since 1st Jan 2011. It is a daily snapshot fact table so that every day we insert 1000 rows into this fact table. So over time we can track the changes of total bill, van charges, satnav income, etc.

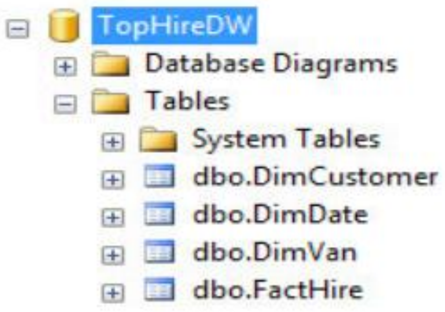
Create the Data Warehouse

Step 1: Create the 3 dimension tables and 1 fact table in the data warehouse

Named as DimDate,DimCustomer,DimVan,FactHire

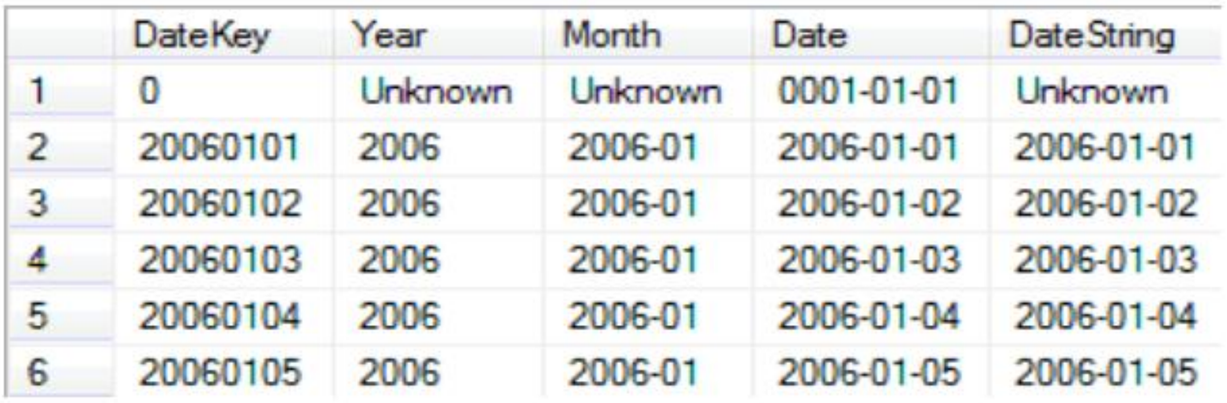
Step2: populate the 3 dimensions and leave the fact table empty.

First I'll show you how it looks when it's done:



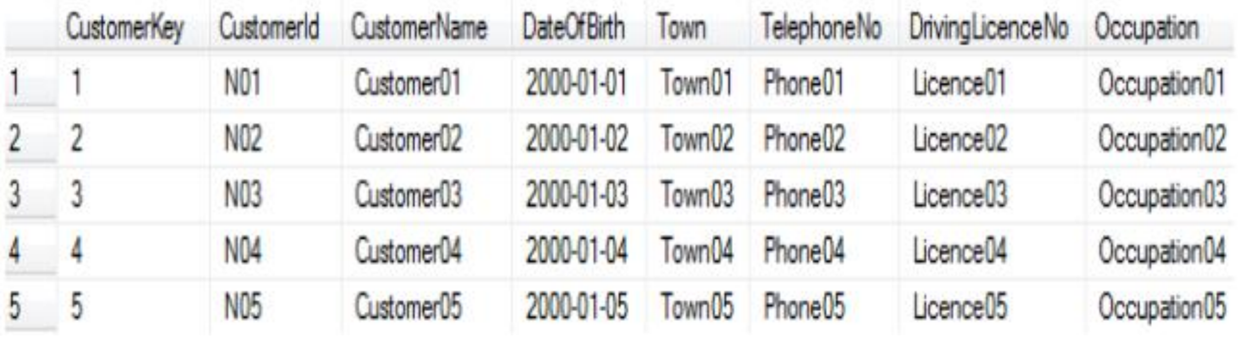
```
⊟ TopHireDW
    ⊞ Database Diagrams
    ⊟ Tables
        ⊞ System Tables
        ⊞ dbo.DimCustomer
        ⊞ dbo.DimDate
        ⊞ dbo.DimVan
        ⊞ dbo.FactHire
```

Date Dimension:



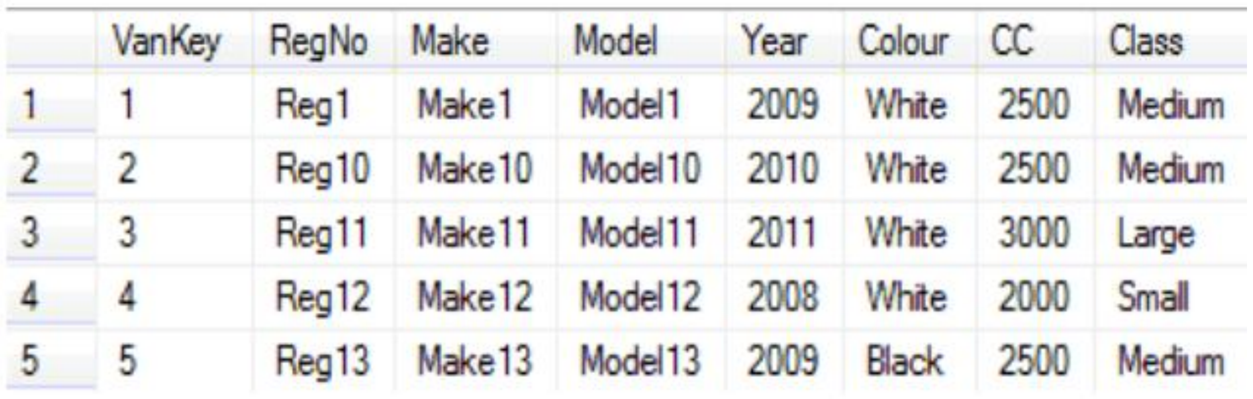
	DateKey	Year	Month	Date	DateString
1	0	Unknown	Unknown	0001-01-01	Unknown
2	20060101	2006	2006-01	2006-01-01	2006-01-01
3	20060102	2006	2006-01	2006-01-02	2006-01-02
4	20060103	2006	2006-01	2006-01-03	2006-01-03
5	20060104	2006	2006-01	2006-01-04	2006-01-04
6	20060105	2006	2006-01	2006-01-05	2006-01-05

Customer Dimension:



	CustomerKey	CustomerId	CustomerName	DateOfBirth	Town	TelephoneNo	DrivingLicenceNo	Occupation
1	1	N01	Customer01	2000-01-01	Town01	Phone01	Licence01	Occupation01
2	2	N02	Customer02	2000-01-02	Town02	Phone02	Licence02	Occupation02
3	3	N03	Customer03	2000-01-03	Town03	Phone03	Licence03	Occupation03
4	4	N04	Customer04	2000-01-04	Town04	Phone04	Licence04	Occupation04
5	5	N05	Customer05	2000-01-05	Town05	Phone05	Licence05	Occupation05

Van Dimension:



	VanKey	RegNo	Make	Model	Year	Colour	CC	Class
1	1	Reg1	Make1	Model1	2009	White	2500	Medium
2	2	Reg10	Make10	Model10	2010	White	2500	Medium
3	3	Reg11	Make11	Model11	2011	White	3000	Large
4	4	Reg12	Make12	Model12	2008	White	2000	Small
5	5	Reg13	Make13	Model13	2009	Black	2500	Medium

This is the script to create and populate those dim and fact tables:

-- Create the data warehouse

create database TopHireDW

go

use TopHireDW

go

-- Create Date Dimension

if exists (select * from sys.tables where name = 'DimDate')

drop table DimDate

go

create table DimDate

(DateKey int not null primary key,

[Year] varchar(7), [Month] varchar(7), [Date] date, DateString varchar(10))

go

-- Populate Date Dimension

truncate table DimDate

go

```
declare @i int, @Date date, @StartDate date, @EndDate date, @DateKey int,
@DateString varchar(10), @Year varchar(4),
@Month varchar(7), @Date1 varchar(20)
set @StartDate = '2006-01-01'
set @EndDate = '2016-12-31'
set @Date = @StartDate
insert into DimDate (DateKey, [Year], [Month], [Date], DateString)
values (0, 'Unknown', 'Unknown', '0001-01-01', 'Unknown') --The unknown row
while @Date <= @EndDate
begin
set @DateString = convert(varchar(10), @Date, 20)
set @DateKey = convert(int, replace(@DateString,'-',''))
set @Year = left(@DateString,4)
set @Month = left(@DateString, 7)
insert into DimDate (DateKey, [Year], [Month], [Date], DateString)
values (@DateKey, @Year, @Month, @Date, @DateString)
set @Date = dateadd(d, 1, @Date)
end
go
-- Create Customer dimension
if exists (select * from sys.tables where name = 'DimCustomer')
drop table DimCustomer
go
create table DimCustomer
( CustomerKey int not null identity(1,1) primary key,
CustomerId varchar(20) not null,
CustomerName varchar(30), DateOfBirth date, Town varchar(50),
TelephoneNo varchar(30), DrivingLicenceNo varchar(30), Occupation varchar(30)
)
go
insert into DimCustomer (CustomerId, CustomerName, DateOfBirth, Town, TelephoneNo,
DrivingLicenceNo, Occupation)
select * from HireBase.dbo.Customer
select * from DimCustomer
-- Create Van dimension
if exists (select * from sys.tables where name = 'DimVan')
drop table DimVan
go
create table DimVan
( VanKey int not null identity(1,1) primary key,
RegNo varchar(10) not null,
Make varchar(30), Model varchar(30), [Year] varchar(4),
Colour varchar(20), CC int, Class varchar(10)
)
go
insert into DimVan (RegNo, Make, Model, [Year], Colour, CC, Class)
select * from HireBase.dbo.Van
go
select * from DimVan
-- Create Hire fact table
```

```
if exists (select * from sys.tables where name = 'FactHire')

drop table FactHire

go

create table FactHire

( SnapshotDateKey int not null, --Daily periodic snapshot fact table

HireDateKey int not null, CustomerKey int not null, VanKey int not null, --Dimension Keys

HireId varchar(10) not null, --Degenerate Dimension

NoOfDays int, VanHire money, SatNavHire money,

Insurance money, DamageWaiver money, TotalBill money

)

go

select * from FactHire
```

(ii)Schema Definition

Multidimensional schema is defined using Data Mining Query Language (DMQL). The two

primitives, cube definition and dimension definition, can be used for defining the data warehouses

and data marts.

Star Schema

• Each dimension in a star schema is represented with only one-dimension table.

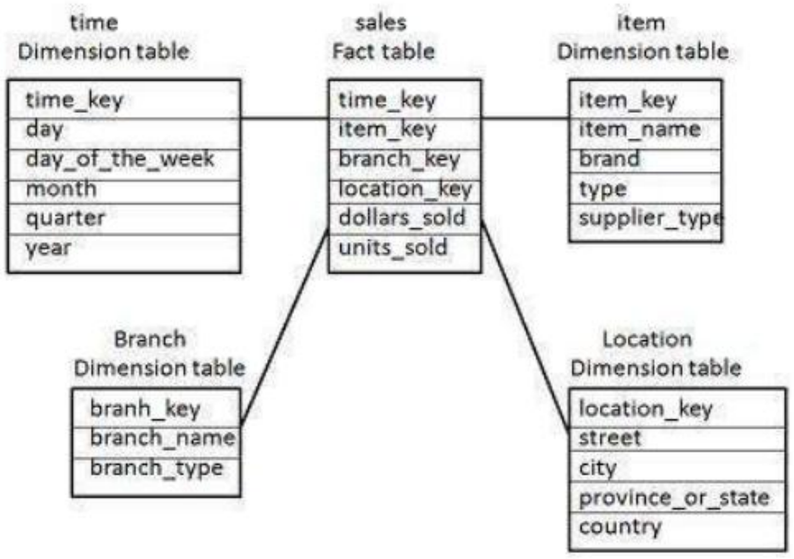
• This dimension table contains the set of attributes.

• The following diagram shows the sales data of a company with respect to the four

dimensions, namely time, item, branch, and location.

• There is a fact table at the center. It contains the keys to each of four dimensions.

• The fact table also contains the attributes, namely dollars sold and units sold.



Snowflake Schema

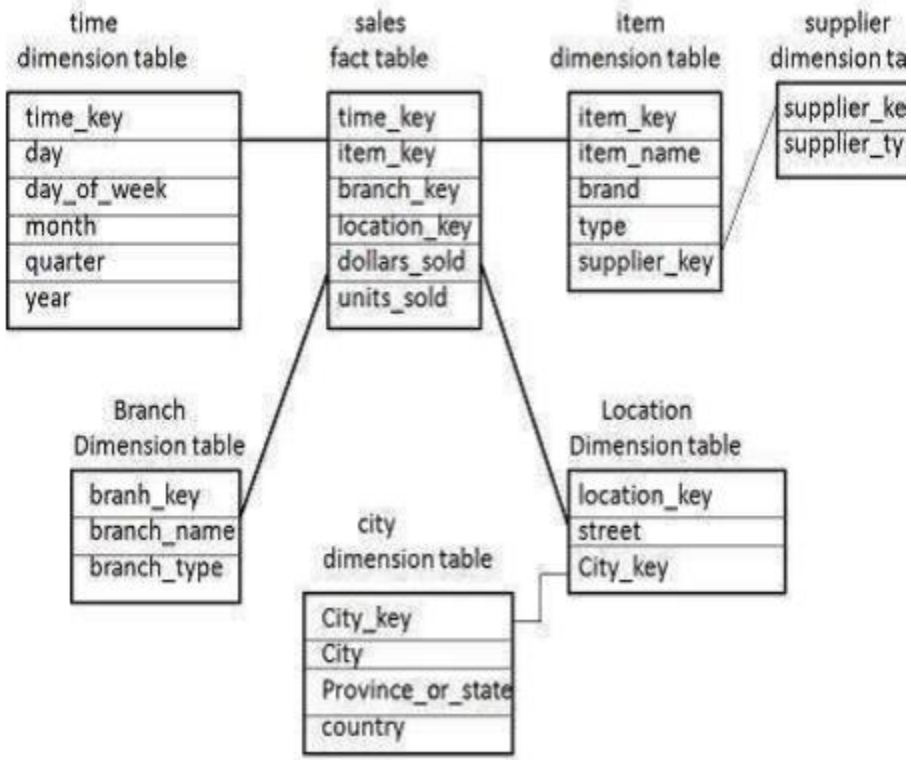
• Some dimension tables in the Snowflake schema are normalized.

• The normalization splits up the data into additional tables.

• Unlike Star schema, the dimensions table in a snowflake schema is normalized. For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.

• Now the item dimension table contains the attributes item_key, item_name, type, brand, and supplier-key.

• The supplier key is linked to the supplier dimension table. The supplier dimension table contains the attributes supplier_key and supplier_type.



Fact Constellation Schema

• A fact constellation has multiple fact tables. It is also known as galaxy schema.

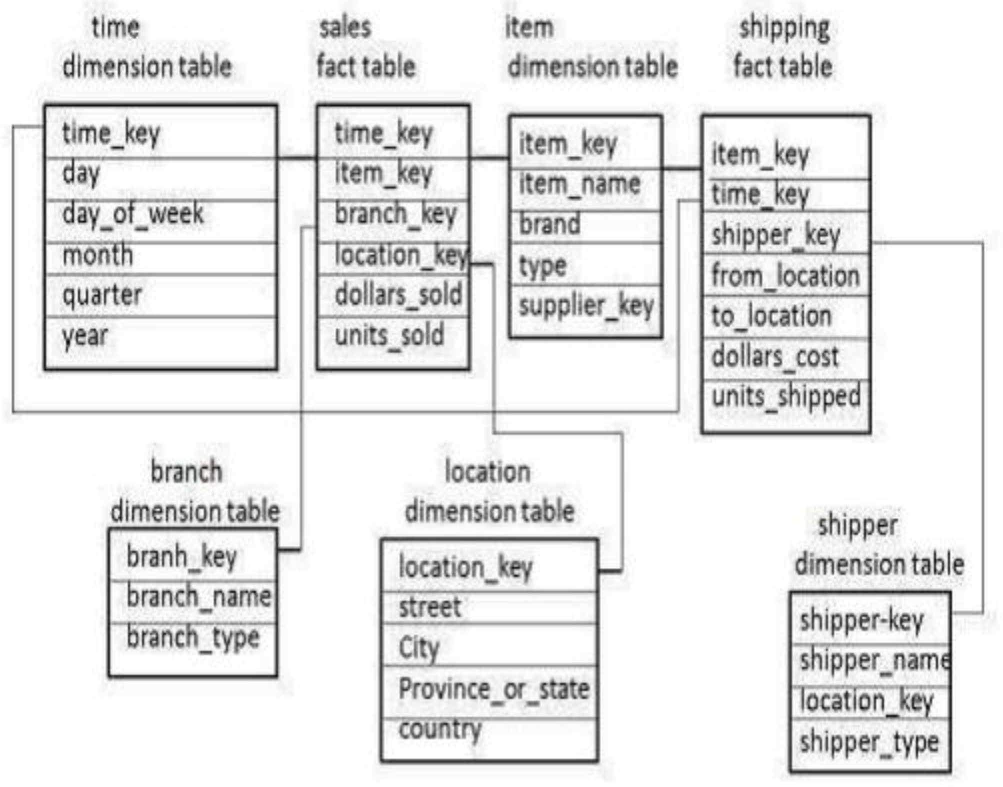
• The following diagram shows two fact tables, namely sales and shipping.

• The sales fact table is same as that in the star schema.

• The shipping fact table has the five dimensions, namely item_key, time_key, shipper_key, from_location, to_location.

• The shipping fact table also contains two measures, namely dollars sold and units sold.

• It is also possible to share dimension tables between fact tables. For example, time, item, and location dimension tables are shared between the sales and shipping fact table.



(iii)ETL Process:

Extract

The Extract step covers the data extraction from the source system and makes it accessible for further processing. The main objective of the extract step is to retrieve all the required data from the source system with as little resources as possible. The extract step should be designed in a way that it does not negatively affect the source system in terms or performance, response time or any kind of locking.

There are several ways to perform the extract:

• Update notification - if the source system is able to provide a notification that a record has been changed and describe the change, this is the easiest way to get the data.

• Incremental extract - some systems may not be able to provide notification that an update has occurred, but they are able to identify which records have been modified and provide an extract of such records. During further ETL steps, the system needs to identify changes and propagate it down. Note, that by using daily extract, we may not be able to handle deleted records properly.

• Full extract - some systems are not able to identify which data has been changed at all, so a full extract is the only way one can get the data out of the system. The full extract requires keeping a copy of the last extract in the same format in order to be able to identify changes. Full extract handles deletions as well.

Transform

The transform step applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

Load

During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

Load verification

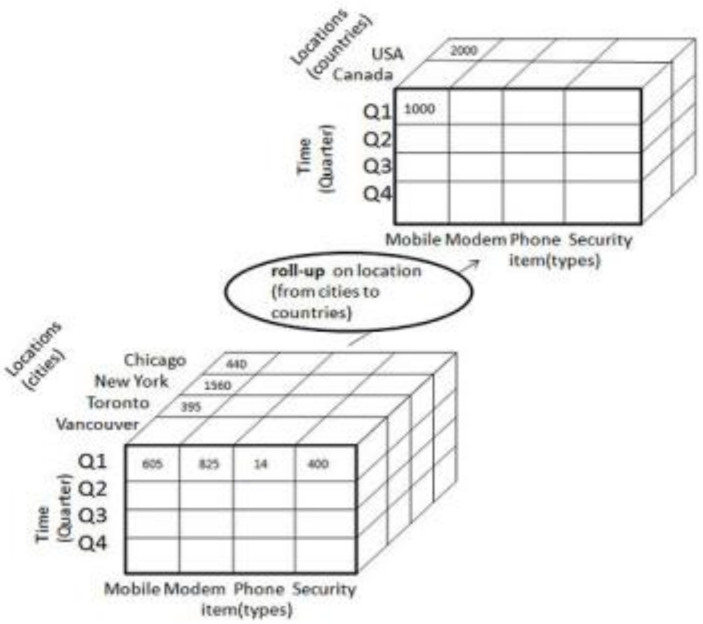
Check that the key field data is not missing or null.
Modelling views based on target tables should be tested.
Examine the combined values3 and computed measures.
Data checks in the dimension and history tables.
Examine the BI reports on the loaded fact and dimension table

Algorithm:

Step 1: Create database Data Warehouse :
Step 2: Create **Customer dimension** table in Data Warehouse which will hold customer personal details. Fill the **Customer dimension** with sample Values
Step 3: Create basic level of **Product Dimension** table without considering any Category or Subcategory Fill the **Product dimension** with sample Values
Step 4: Create **Store Dimension** table which will hold details related stores available across various places. Fill the **Store Dimension** with sample Values
Step 5: Create **Dimension Sales Person** table which will hold details related stores available across various places. Fill the Dimension **Sales Person** with sample values:

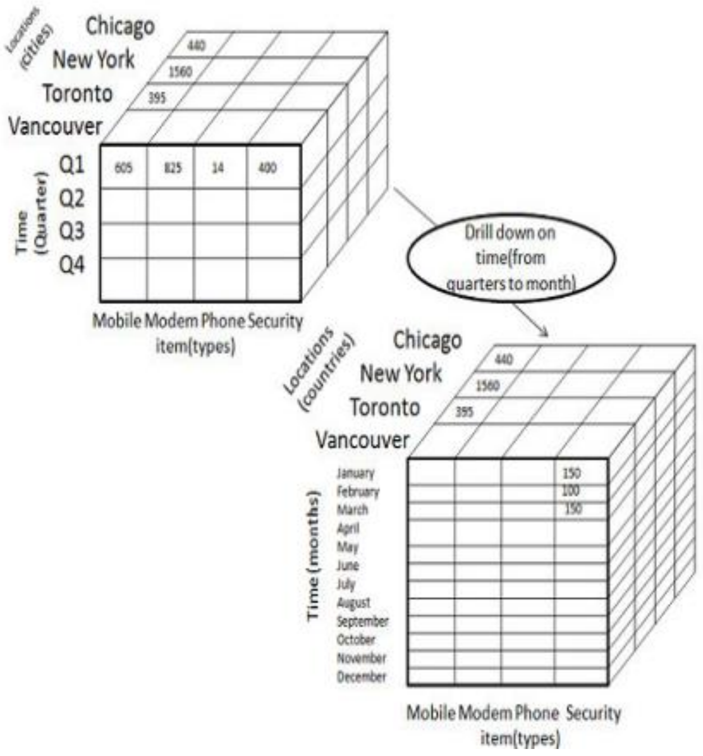
Step 6: Create **Date Dimension** table which will create and populate date data divided on various levels.
Step 7: Create **Time Dimension** table which will create and populate Time data for the entire day with various time buckets.
Step 8: Create **Fact table** to hold all your transactional entries of previous day sales with appropriate foreign key columns which refer to primary key column of your dimensions.

(iv) OLAP OPERATIONS
Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information.
OLAP operations in multidimensional data.
Here is the list of OLAP operations:
• Roll-up
• Drill-down
• Slice and dice
• Pivot (rotate)
Roll-up
Roll-up performs aggregation on a data cube in any of the following ways:
• By climbing up a concept hierarchy for a dimension
• By dimension reduction
The following diagram illustrates how roll-up works.



• Roll-up is performed by climbing up a concept hierarchy for the dimension location.
• Initially the concept hierarchy was "street < city < province < country".
• On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
• The data is grouped into cities rather than countries.
• When roll-up is performed, one or more dimensions from the data cube are removed

Drill-down
Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:
• By stepping down a concept hierarchy for a dimension
• By introducing a new dimension.
The following diagram illustrates how drill-down works:



Drill-down is performed by stepping down a concept hierarchy for the dimension time.

• Initially the concept hierarchy was "day < month < quarter < year."

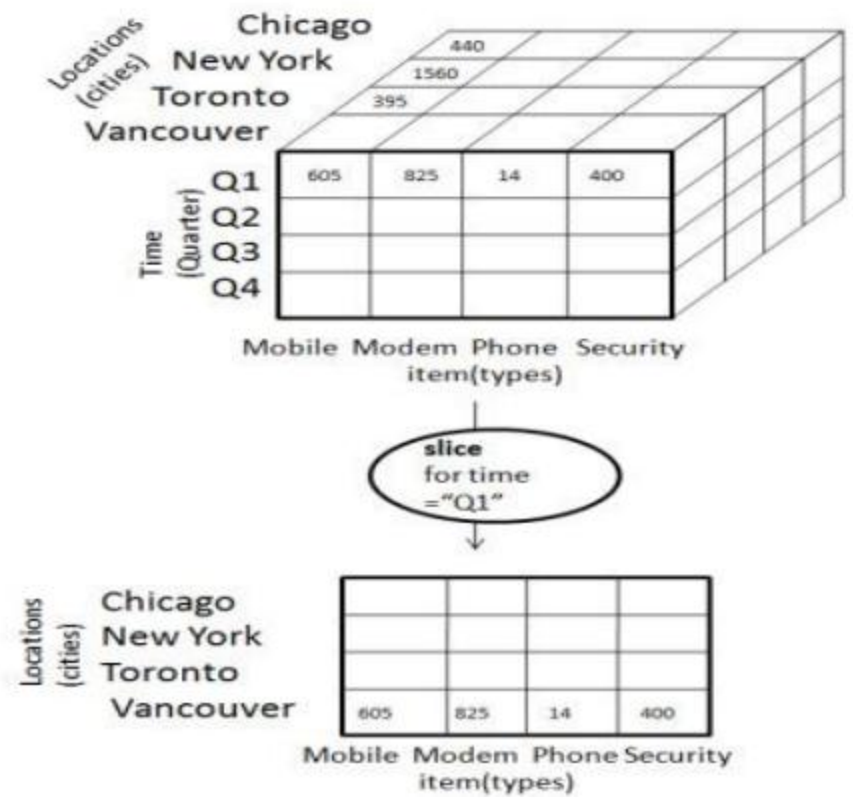
• On drilling down, the time dimension is descended from the level of quarter to the level of month.

• When drill-down is performed, one or more dimensions from the data cube are added.

• It navigates the data from less detailed data to highly detailed data.

Slice

The slice operation selects one particular dimension from a given cube and provides a new subcube. Consider the following diagram that shows how slice works.

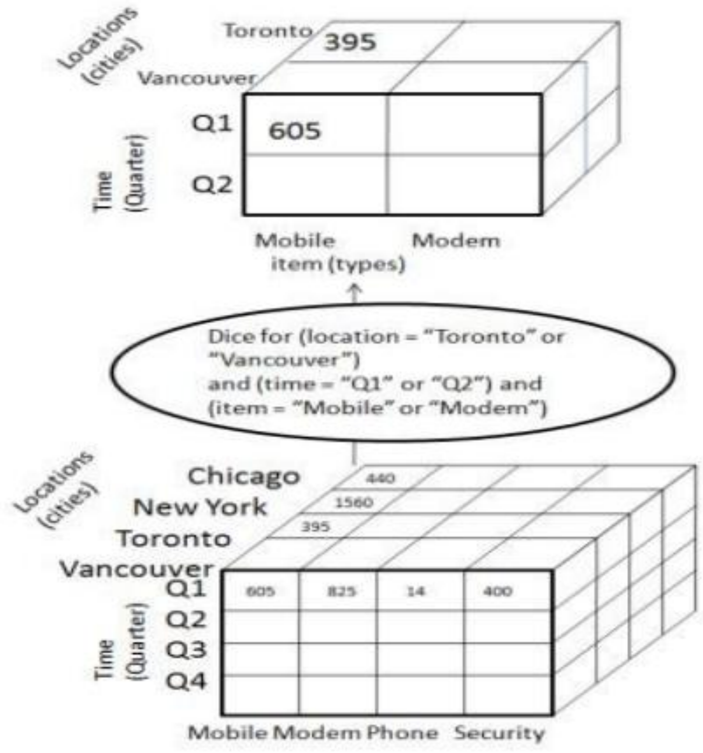


• Here Slice is performed for the dimension "time" using the criterion time = "Q1".

• It will form a new sub-cube by selecting one or more dimensions.

Dice

Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.



The dice operation on the cube based on the following selection criteria involves three dimensions.

• (location = "Toronto" or "Vancouver")

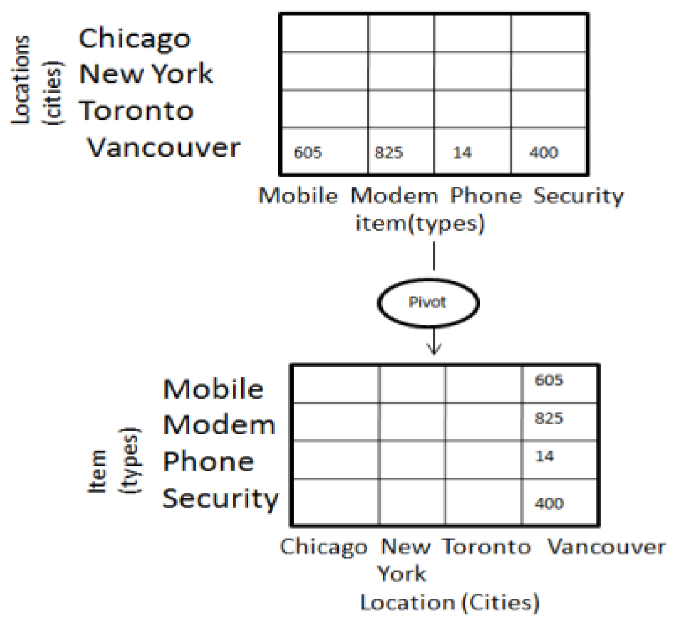
• (time = "Q1" or "Q2")

• (item =" Mobile" or "Modem")

Pivot

The pivot operation is also known as rotation. It rotates the data axes in view in order to provide

an alternative presentation of data. Consider the following diagram that shows the pivot operation.


Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
Pivot
Item (types)
Mobile
Modem
Phone
Security
605
825
14
400
Chicago
New York
Toronto
Vancouver
Location (Cities)

Experiment-2

AIM: Explore machine learning tool "WEKA"

Study the arff file format Explore the available data sets in WEKA. Load a data set (ex. Weather dataset, Iris dataset, etc.)

Load each dataset and observe the following:

1. List the attribute names and they types

2. Number of records in each dataset

3. Identify the class attribute (if any)

4. Plot Histogram

5. Determine the number of records for each class.

6. Visualize the data in various dimensions

Description:

Install Steps for WEKA a Data Mining Tool

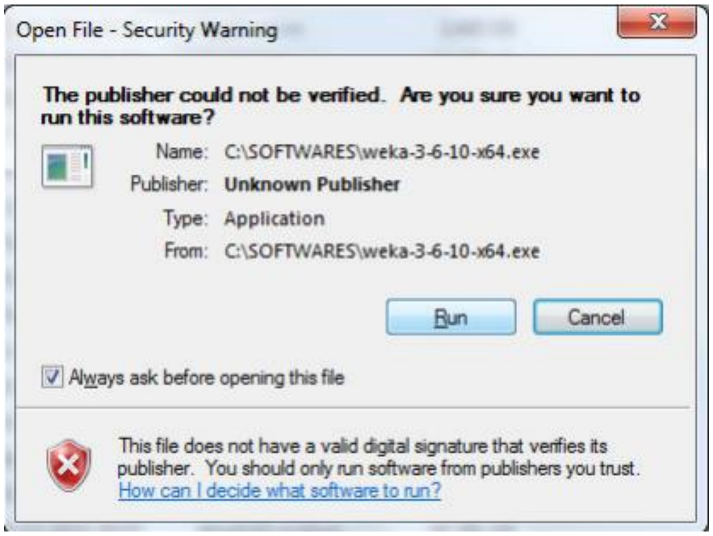
1. Download the software as your requirements from the below given link.

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

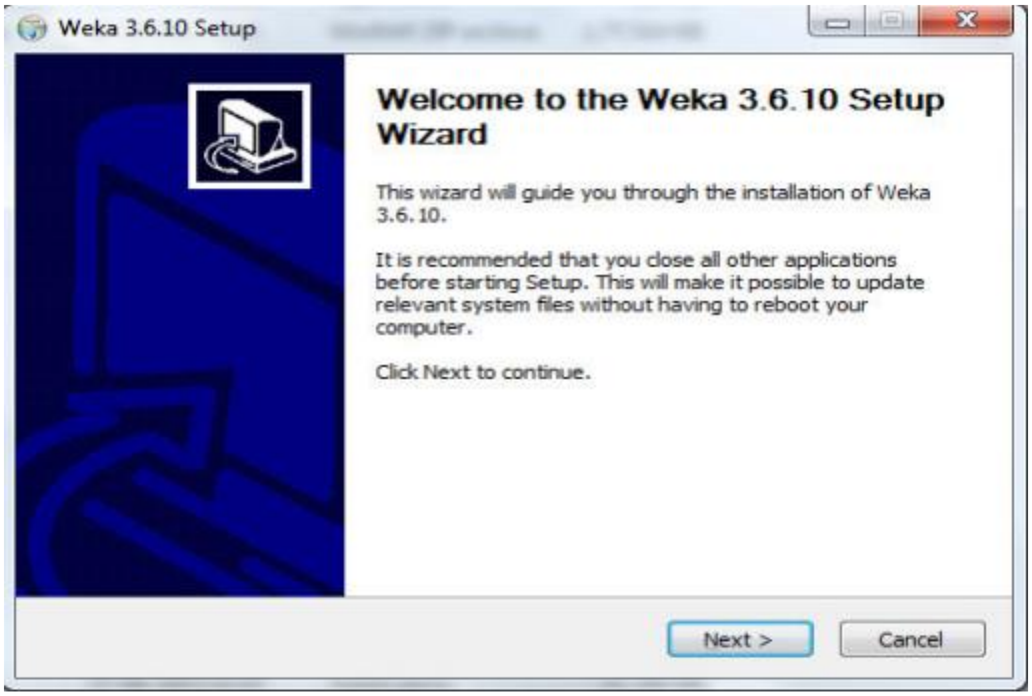
The publisher could not be verified. Are you sure you want to run this software?

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

☑ Always ask before opening this file

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

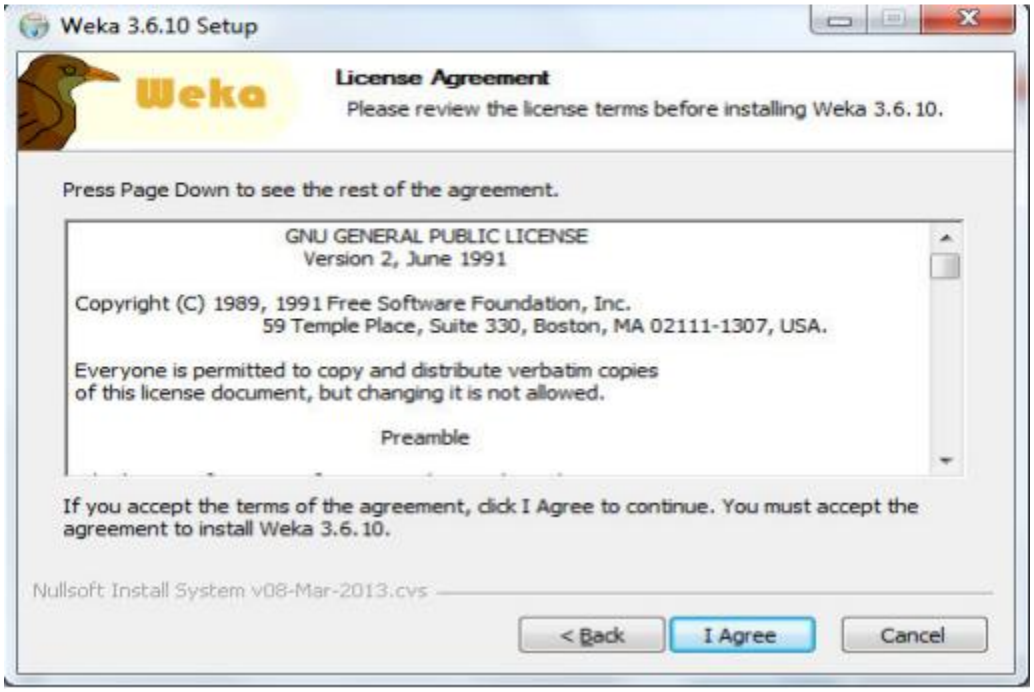
Welcome to the Weka 3.6.10 Setup Wizard

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.

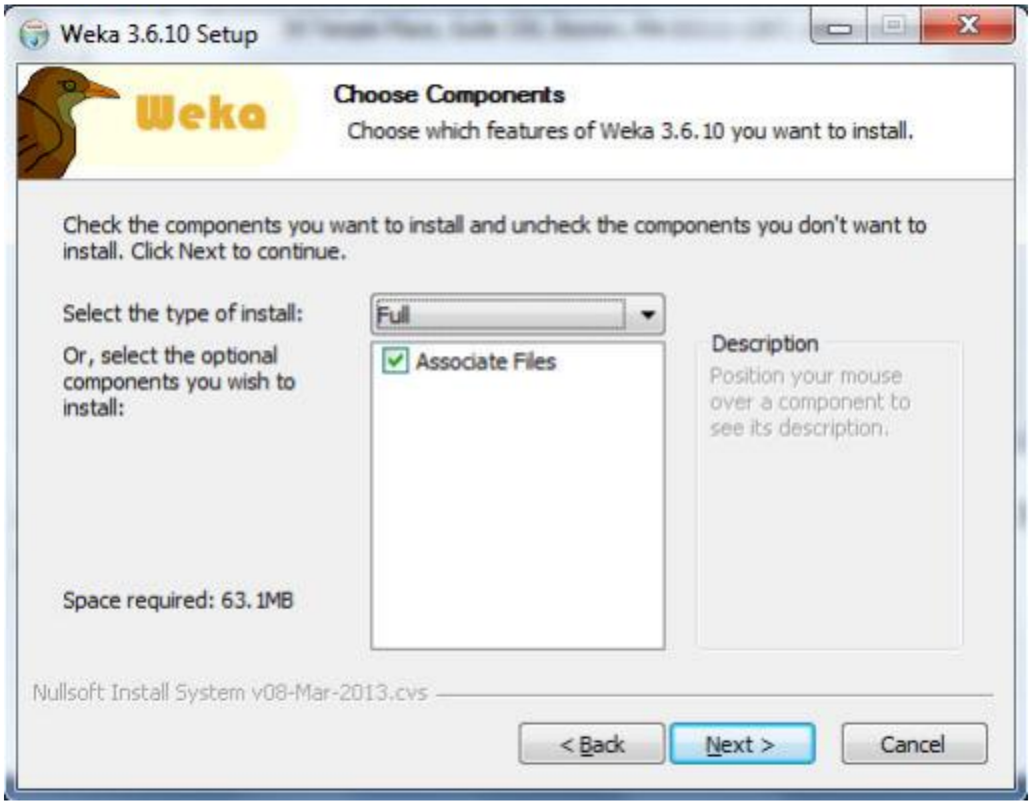
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

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

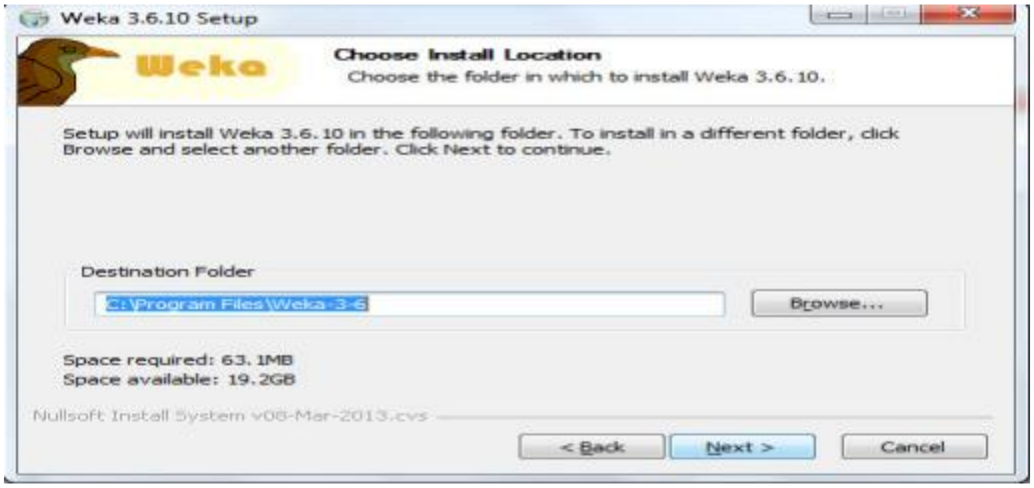
Description

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.

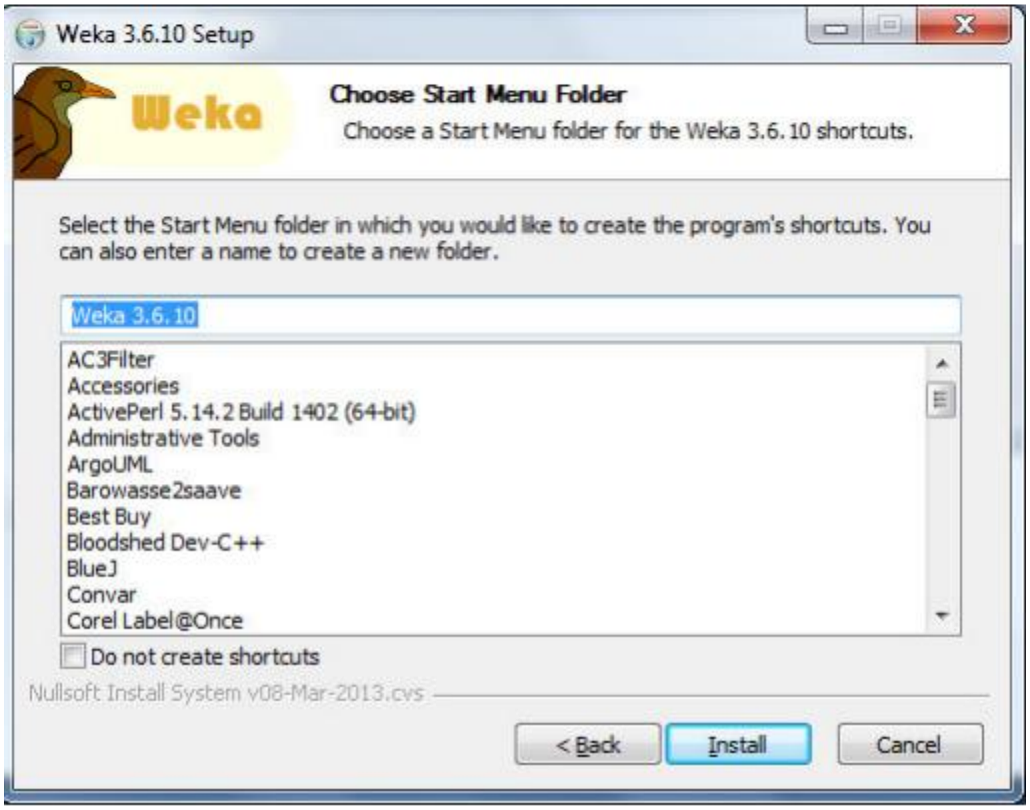
Destination Folder: C:\Program Files\Weka-3-6

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.


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.

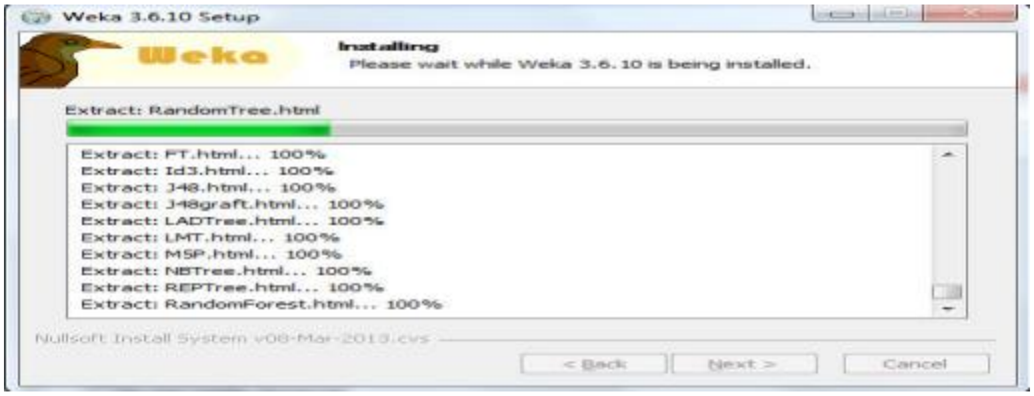
Weka 3.6.10

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

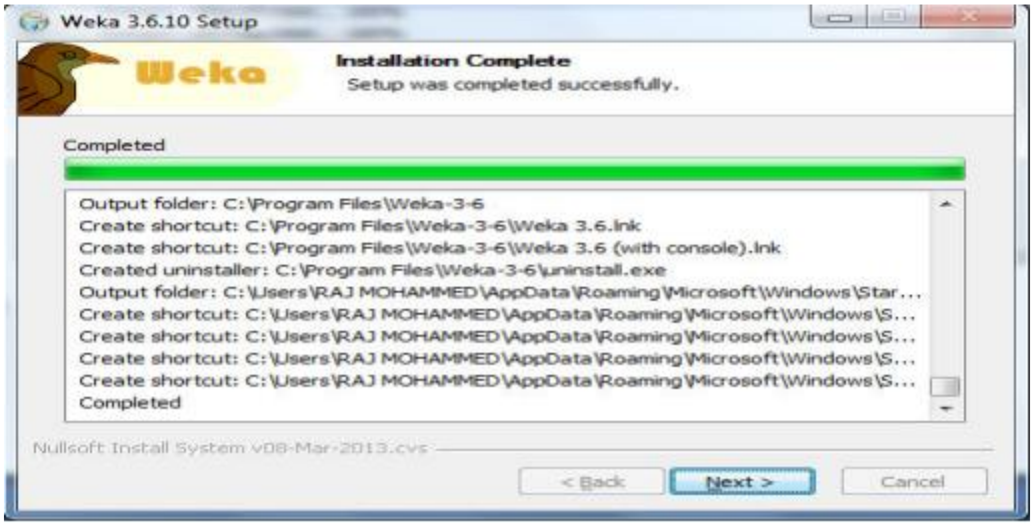
Please wait while Weka 3.6.10 is being installed.

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

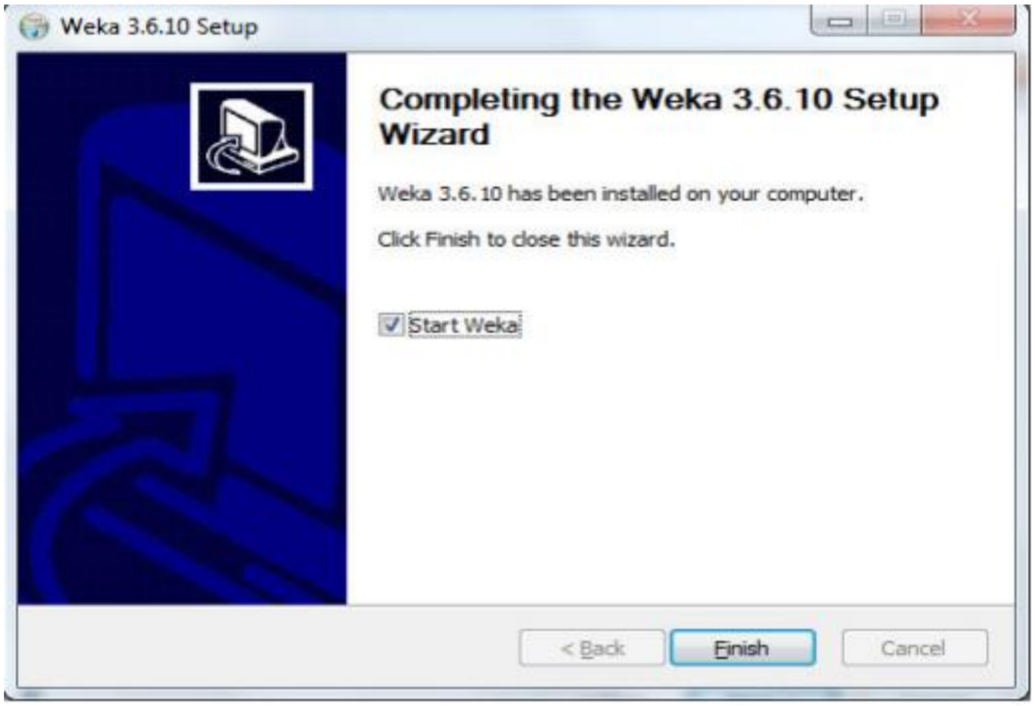
Setup was completed successfully.

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.

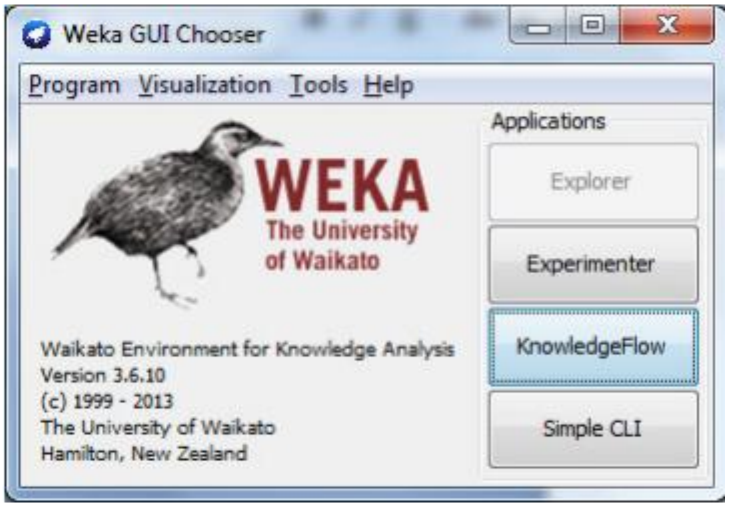

Weka 3.6.10 Setup

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]



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

(ii)ARFF File Format

An ARFF (= Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes.

ARFF files are not the only format one can load, but all files that can be converted with Weka's "core converters". The following formats are currently supported:

• ARFF (+ compressed)

• C4.5

• CSV

• libsvm

• binary serialized instances

• XRFF (+ compressed)

Algorithm for data sets in WEKA

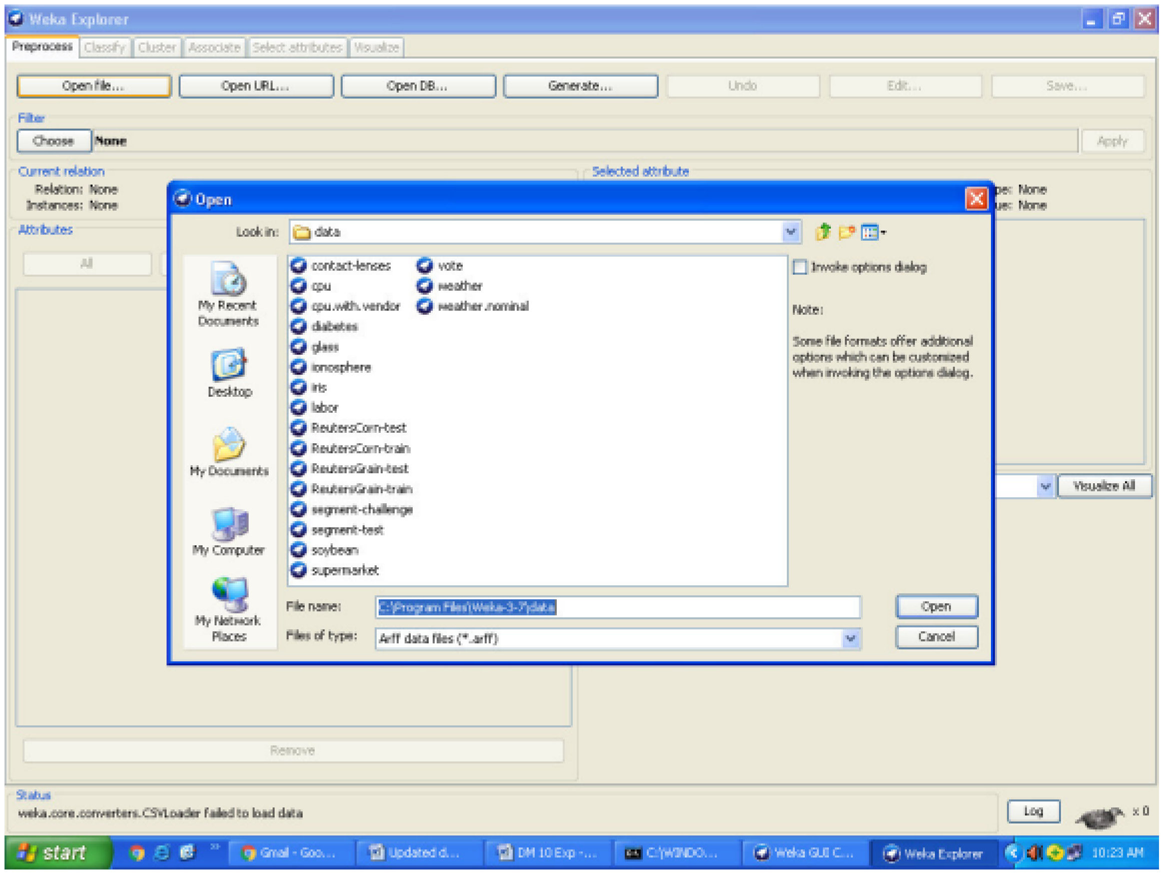
Step 1: Open WEKA Tool.

Step 2: Click on WEKA Explorer.

Step 3: Click on open file button.

Step 4: Choose WEKA folder in C drive.

Step 5: Select and Click on data option button.



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)

weka.core.converters.CSVLoader failed to load data

1 Algorithm for load the Weather data set.

Step 1: Open WEKA Tool.

Step 2: Click on WEKA Explorer.

Step 3: Click on open file button.

Step 4: Choose WEKA folder in C drive.

Step 5: Select and Click on data option button.

Step 6: Choose Weather.arff file and Open the file.

Algorithm for load the Iris data set.

Step 1: Open WEKA Tool.

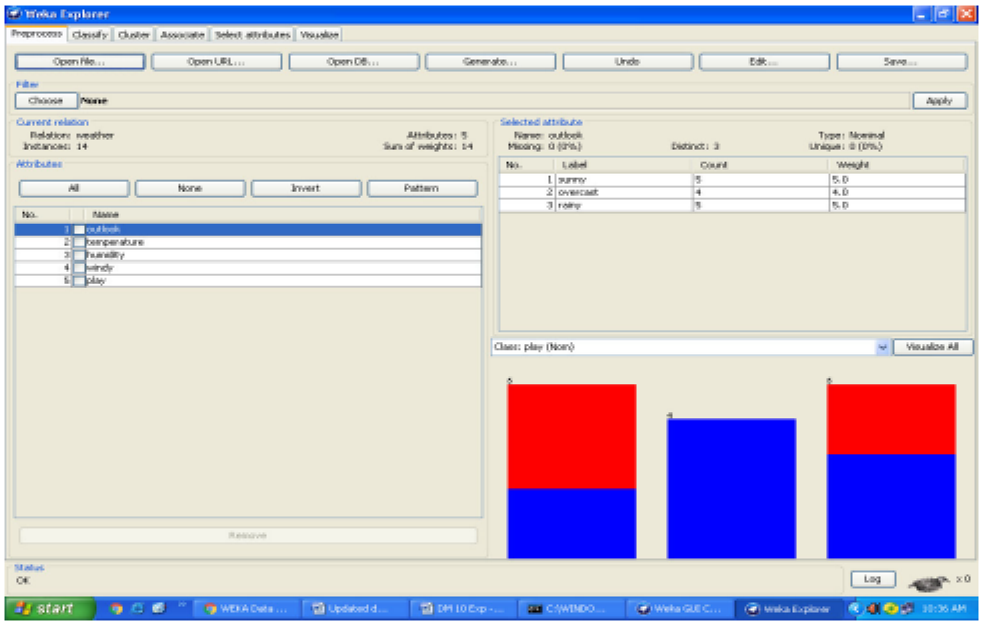
Step 2: Click on WEKA Explorer.

Step 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.



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

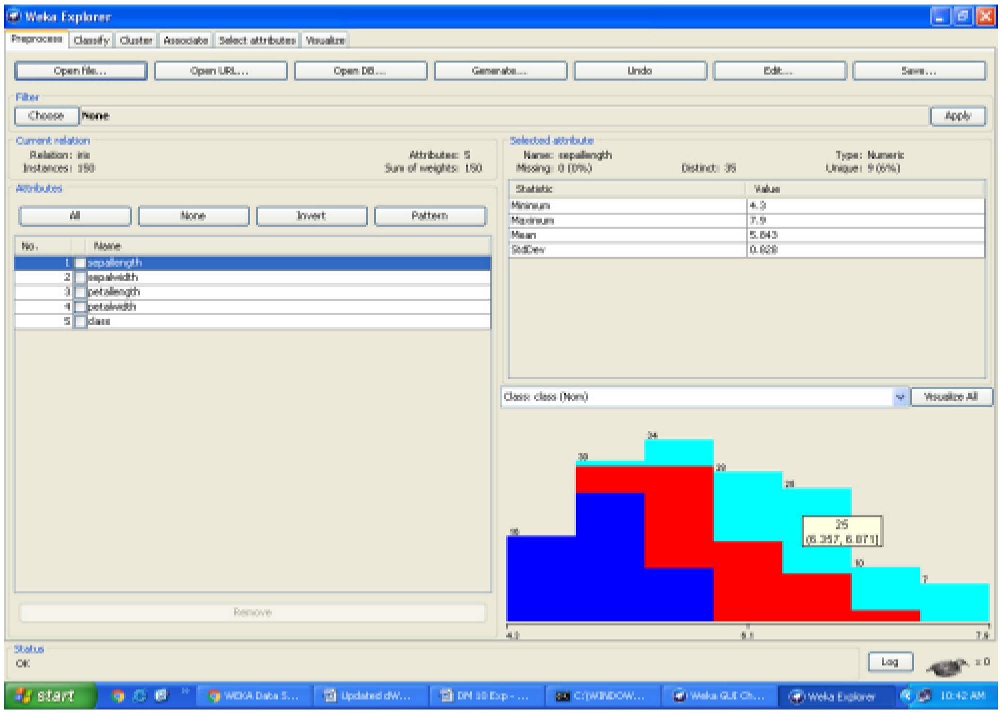
Selected attribute
Name: outlook
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

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

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

Class: play (Nom)

Status
OK



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

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

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

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

Class: class (Nom)

Status
OK

(1) List attribute names and they types

Example dataset-Weather.arff

List out the attribute names:

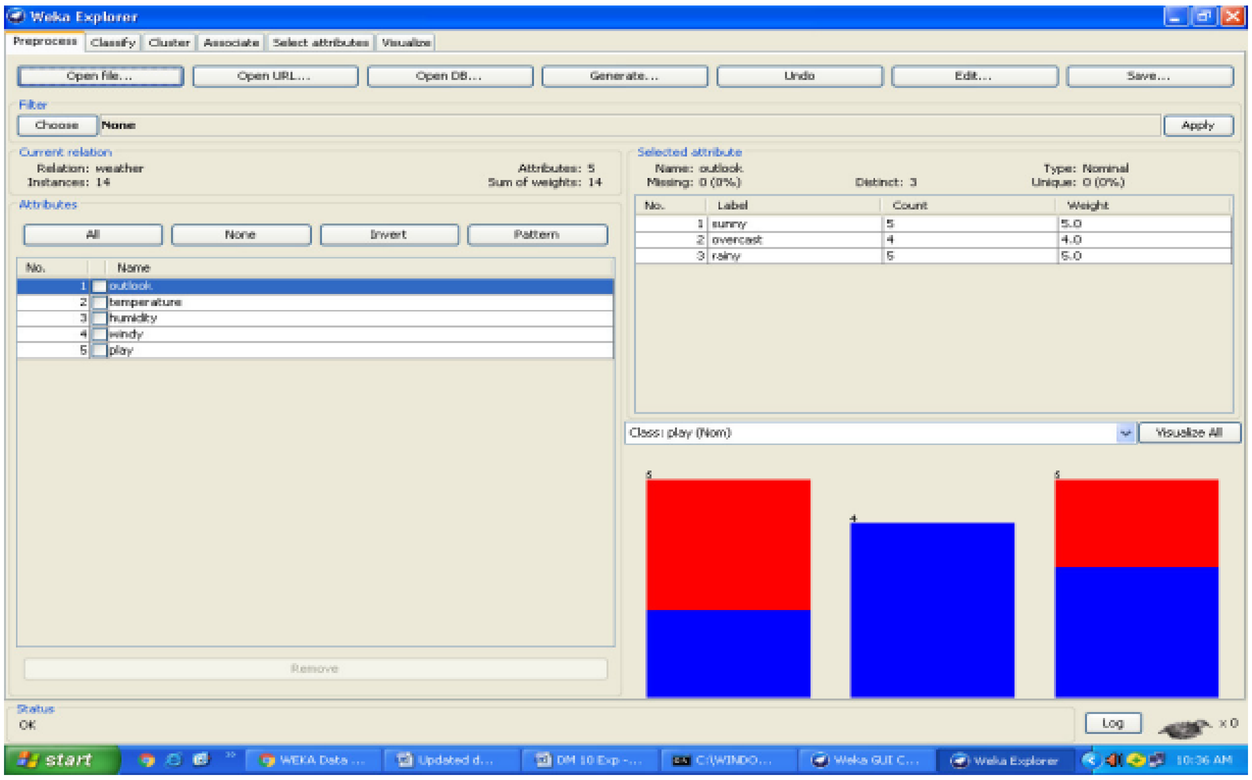
1. outlook

2. temperature

3. humidity

4. windy

5. play



Filter: None

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

Attributes

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

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

(2) Number of records in each dataset

@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}

@attribute temperature {hot, mild, cool}

@attribute humidity {high, normal}

@attribute windy {TRUE, FALSE}

```
@attribute play {yes, no}

@data

sunny,hot,high,FALSE,no

sunny,hot,high,TRUE,no

overcast,hot,high,FALSE,yes

rainy,mild,high,FALSE,yes

rainy,cool,normal,FALSE,yes

rainy,cool,normal,TRUE,no

overcast,cool,normal,TRUE,yes

sunny,mild,high,FALSE,no

sunny,cool,normal,FALSE,yes

rainy,mild,normal,FALSE,yes

sunny,mild,normal,TRUE,yes

overcast,mild,high,TRUE,yes

overcast,hot,normal,FALSE,yes

rainy,mild,high,TRUE,no
```

(3) Identify the class attribute

class attributes

1. sunny

2. overcast

3. rainy

(4) Plot Histogram

Algorithm for identify the plot histogram

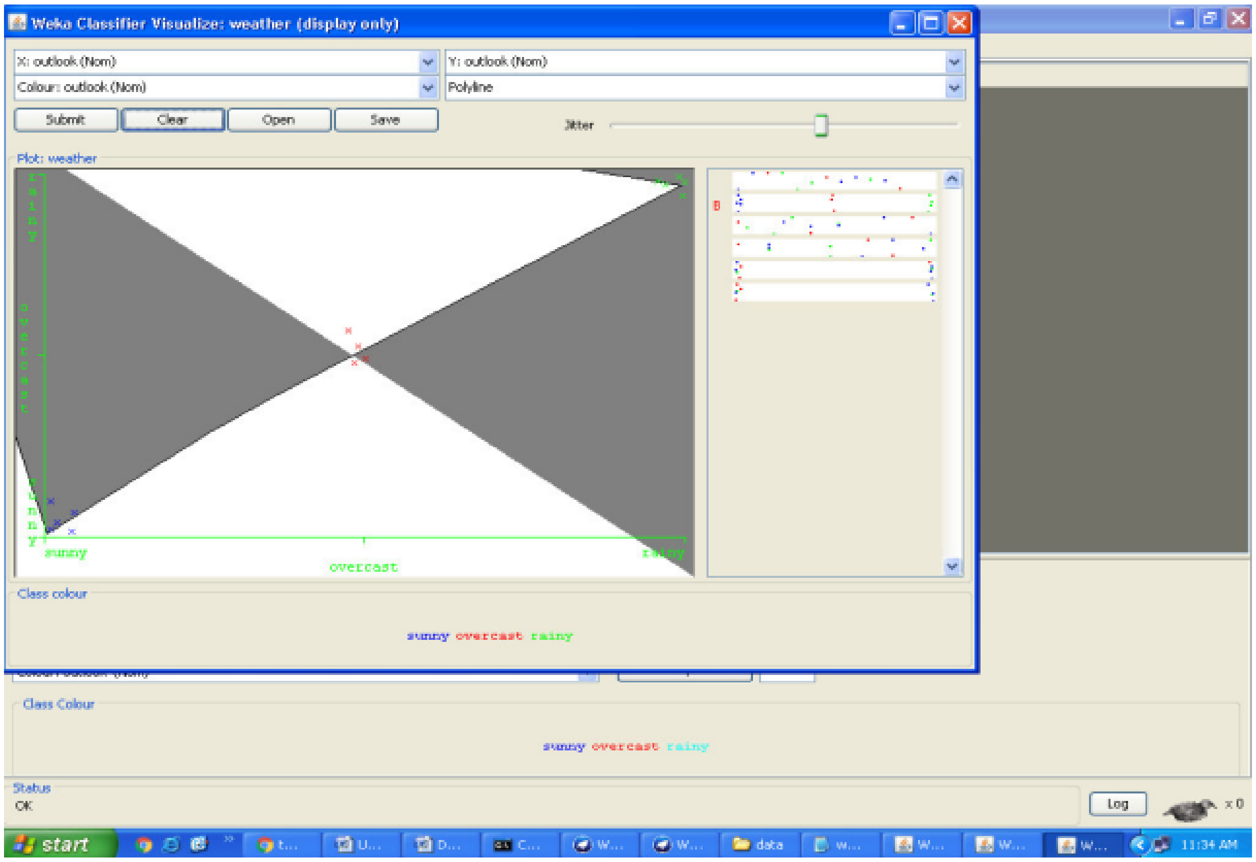
Step 1: Open WEKA Tool.

Step 2: Click on WEKA Explorer.

Step 3: Click on Visualize button.

Step 4: Click on right click button.

Step 5: Select and Click on polyline option button.



(5) Determine the number of records for each class
@relation weather.symbolic

@data

sunny,hot,high,FALSE,no

sunny,hot,high,TRUE,no

overcast,hot,high,FALSE,yes

rainy,mild,high,FALSE,yes

rainy,cool,normal,FALSE,yes

rainy,cool,normal,TRUE,no

overcast,cool,normal,TRUE,yes

sunny,mild,high,FALSE,no

sunny,cool,normal,FALSE,yes

rainy,mild,normal,FALSE,yes

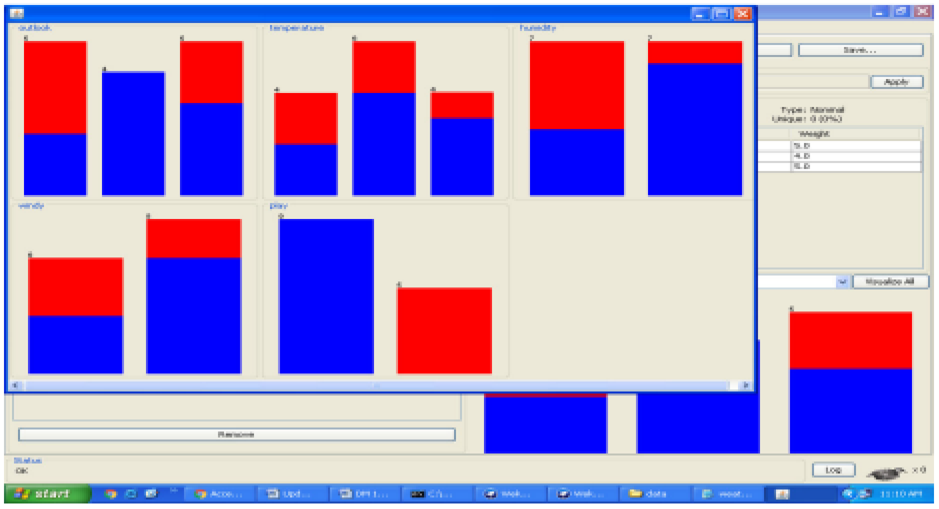
sunny,mild,normal,TRUE,yes

overcast,mild,high,TRUE,yes

overcast,hot,normal,FALSE,yes

rainy,mild,high,TRUE,no

(6) Visualize the data in various dimensions



Experiment-3

AIM: Perform following data preprocessing tasks using Python

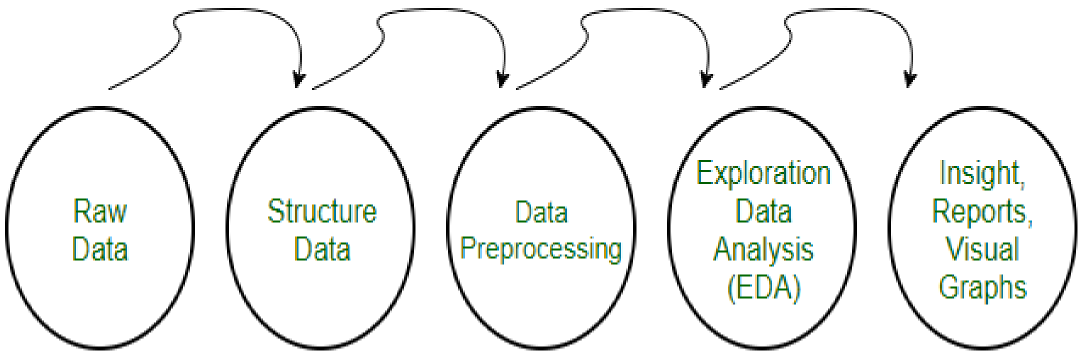
i) Rescale Data

ii) Binarize Data

iii)Standardize Data

Description:

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.


Raw Data
Structure Data
Data Preprocessing
Exploration Data Analysis (EDA)
Insight, Reports, Visual Graphs

Need of Data Preprocessing

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

Another aspect is that the data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithm are executed in one data set, and best out of them is chosen.

This article contains 3 different data preprocessing techniques for machine learning.

i). Rescale Data :

When our data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale.This is useful for optimization algorithms used in the core of machine learning algorithms like gradient descent.It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like K-Nearest Neighbors.We can rescale your data using scikit-learn using the MinMaxScaler class.

Algorithm:

Step1: Begin by identifying the minimum and maximum values for the data set.

Step2: Calculate the difference between the min and max values.

Step3: For each data point, subtract the minimum value from it and calculate the percentage of the difference between the min and max values that it represents.

Step4: Multiply the percentage by the new range desired for the data set.

Step5: Add the newly calculated value to the minimum value of the new range to rescale the data point.

Step6: Repeat for each data point.

ii). Binarize Data (Make Binary) :

We can transform our data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0.This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.We can create new binary attributes in Python using scikit-learn with the Binarizer class.

Binarization is a preprocessing technique which is used when we need to convert the data into binary numbers i.e., when we need to binarize the data. The scikit-learn function named Sklearn.preprocessing.binarize() is used to binarize the data.This binarize function is having threshold parameter, the feature values below or equal this threshold value is replaced by 0 and value above it is replaced by 1.

Algorithm:

Step 1: Begin by defining the threshold of what is considered a “binary” value.

Step 2: Iterate through the set of data, testing each value to see if it meets or surpasses the thresholds defined.

Step 3: If a value surpasses the threshold, it is considered a “1” or true value.

Step 4: If a value does not meet the threshold, it is considered a “0” or false value.

Step 5: Replace all data that has been tested with a 0 or a 1, according to the value of the data.

Step 6: End the process when all data in the set has been tested, and all 0s and 1s have been allocated.

iii) Standardize Data :

Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.We can standardize data using scikit-learn with the StandardScaler class.

Standardization is used on the data values that are normally distributed. Further, by applying standardization, we tend to make the mean of the dataset as 0 and the standard deviation equivalent to 1.

That is, by standardizing the values, we get the following statistics of the data distribution

mean = 0

standard deviation = 1

$$z = \frac{x - \mu}{\sigma}$$

Thus, by this the data set becomes self explanatory and easy to analyze as the mean turns down to 0 and it happens to have an unit variance.

Algorithm:

Step 1: Define the data set by creating a list of numerical values

Step 2: Calculate the mean (μ) and standard deviation (σ) of the data

Step 3:. For each x-value of the dataset, subtract the mean from the x-value and divide by the standard deviation

Step 4: Store the calculated z-scores in a new list

Step 5: Output the new list containing the standardized values

Experiment-4

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

Description:

Chi-Square test is a statistical test which is used to find out the difference between the observed and the expected data we can also use this test to find the correlation between categorical variables in our data. The purpose of this test is to determine if the difference between 2 categorical variables is due to chance, or if it is due to a relationship between them.

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

It is important to note that the variables to be compared should have only 2 categories i.e 1 and 0 the chi-square test fails to determine the correlation between variables with more than 2 categories.

While conducting the chi-square test we have to initially consider 2 hypothesis i.e the Null Hypothesis and the Alternate Hypothesis.

- H0 (Null Hypothesis) = The 2 variables to be compared are independent.
- H1 (Alternate Hypothesis) = The 2 variables are dependent.

if the p-value obtained after conducting the test is less than 0.05 we reject the Null hypothesis and accept the Alternate hypothesis and if the p-value is greater that 0.05 we accept the Null hypothesis and reject the Alternate hypothesis. Now, let’s move onto the implementation. If you want to know more about the theory behind the Chi-square test you can check this out.

Algorithm:

Step1: Define the null (H0) and alternative (H1) hypothesis.

Step2: Determine the value of alpha (α) for according to the domain you are working. Ideally α=0.05 that means you are willing to take 0.5% of risk/ margin of error.

Step3: Check the data for Nans or other kind of errors.

Step4: Check the assumptions for the test.

Step5: At last perform the test and draw your conclusion whether to reject or support null hypothesis (H0) .

Experiment-5

AIM: AIM: Demonstrate performing classification on data sets

Load each dataset into Weka and run 1d3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.

Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix.

Load each dataset into Weka and perform Naïve-bayes classification and k-Nearest Neighbour classification.

Interpret the results obtained.

Plot RoC Curves

Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce

which classifier is performing best and poor for each dataset and justify.

Description:

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 by clicking in the Test options box. There are four test modes:

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

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

the file to test on.

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

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

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. The classifier's predictions are remembered so that they can be visualized. This option is selected by default.

6. **Output predictions.** The predictions on the evaluation data are output. Note that in the case of a cross-validation the instance numbers do not correspond to the location in the data!

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

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

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

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

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

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.

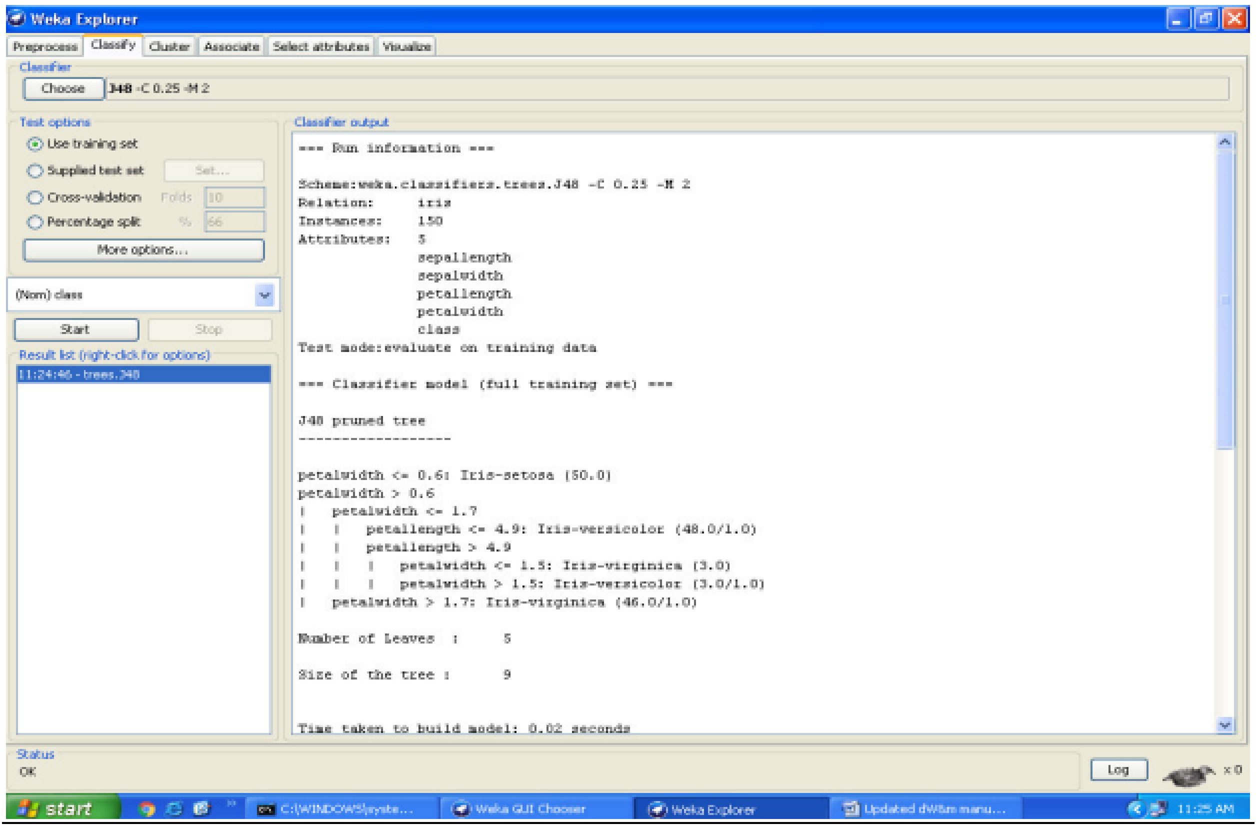
By default, the class is taken to be the last attribute in the data. If you want
to train a classifier to predict a different attribute, click on the box below the Test options box to bring up a drop-down list of attributes to choose from.

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.

(i) Load each dataset into Weka and run 1d3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.

<u>Description:</u>



Classifier: J48 -C 0.25 -M 2

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

(Nom) class

Result list (right-click for options): 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
```

Status: OK

The Classifier Output Text

The text in the Classifier output area has scroll bars allowing you to browse

the results. Clicking with the left mouse button into the text area, while holding Alt

and Shift, brings up a dialog that enables you to save the displayed output

in a variety of formats (currently, BMP, EPS, JPEG and PNG). Of course, you

can also resize the Explorer window to get a larger display area.

The output is

Split into several sections:

1. Run information. A list of information giving the learning scheme options, relation name,

instances, attributes and test mode that were involved in the process.

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

produced on the full training data.

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

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

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

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

7. Source code (optional). This section lists the Java source code if one chose "Output source code" in the "More options" dialog.

Algorithm:

Step1: Open WEKA Tool.

Step2: Click on WEKA Explorer.

Step3: Click on Preprocessing tab button.

Step4: Click on open file button.

Step5: Choose WEKA folder in C drive.

Step6: Select and Click on data option button.

Step7: Choose iris data set and open file.

Step8: Click on classify tab and Choose J48 algorithm and select use training set test option.

Step9: Click on start button.

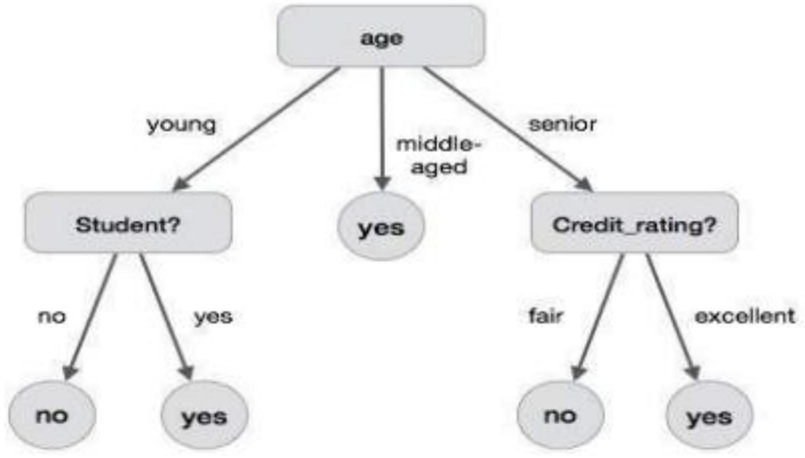
Step10: Click on classify tab and Choose ID3 algorithm and select use training set test option.

Step11: Click on start button.

(ii) Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix.

Description:

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 orprecondition.

• 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.

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.

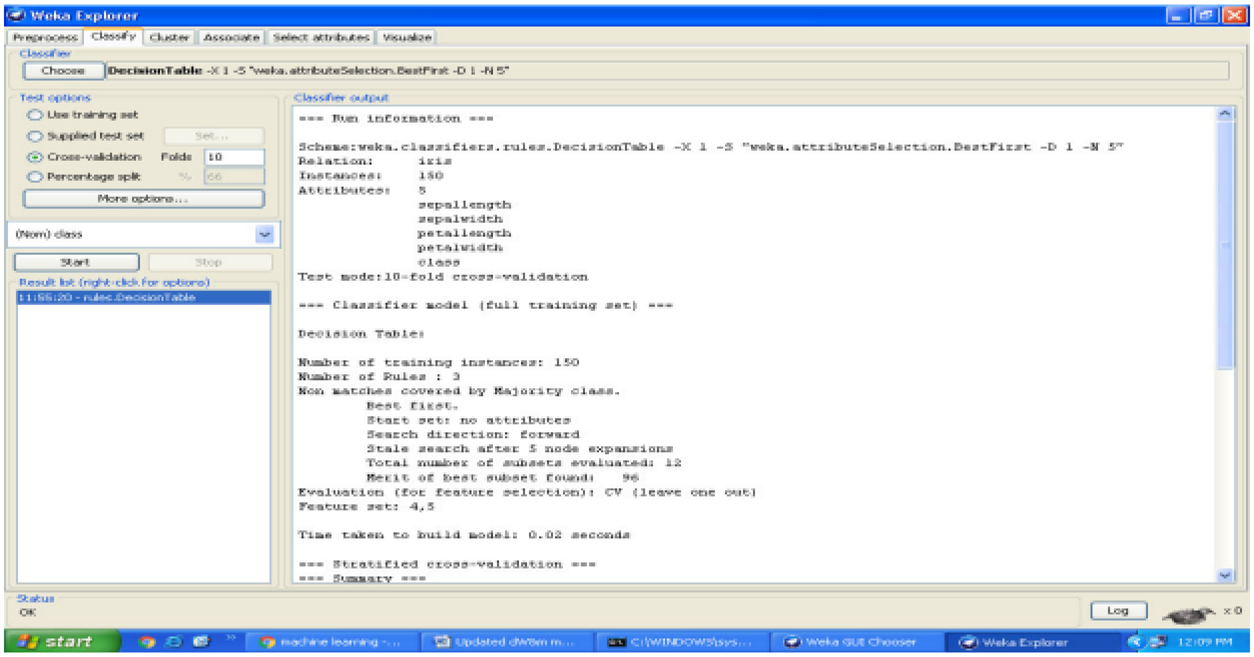
Rule Induction Using Sequential Covering Algorithm

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 to each leaf in a decision tree corresponds to a rule.

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

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



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

Cross-validation Folds 10

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 ===
```

OK

Algorithm:

Step1: Open WEKA Tool.

Step2: Click on WEKA Explorer.

Step3: Click on Preprocessing tab button.

Step4: Click on open file button.

Step5: Choose WEKA folder in C drive.

Step6: Select and Click on data option button.

Step7: Choose iris data set and open file.

Step8: Click on classify tab and Choose decision table algorithm and select cross-validation folds value-10 test option.

Step9: Click on start button.

(iii) Load each dataset into Weka and perform Naïve-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained.

<u>Description:</u>

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

KNN (K — Nearest Neighbors) is one of many (supervised learning) algorithms used in data mining and machine learning, it's a classifier algorithm where the learning is based "how similar" is a data (a vector) from other .

<u>Algorithm:</u>

Step1: Open WEKA Tool.

Step2: Click on WEKA Explorer.

Step3: Click on Preprocessing tab button.

Step4: Click on open file button.

Step5: Choose WEKA folder in C drive.

Step6: Select and Click on data option button.

Step7: Choose iris data set and open file.

Step8: Click on classify tab and Choose Naïve-bayes algorithm and select use training set test option.

Step9: Click on start button.

Step10: Click on classify tab and Choose k-nearest neighbor and select use training set test option.

Step11: Click on start button.

(iv) Plot RoC Curves Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

Description:

While working on a classification model, we feel a need of a metric which can show us how our model is performing. A metric which can also give a graphical representation of the performance will be very helpful.

ROC curve can efficiently give us the score that how our model is performing in classifing the labels. We can also plot graph between False Positive Rate and True Positive Rate with this ROC (Receiving Operating Characteristic) curve. The area under the ROC curve give is also a metric.

Algorithm to identify plot RoC Curves.

Step1: Open WEKA Tool.

Step2: Click on WEKA Explorer.

Step3: Click on Visualize button.

Step4: Click on right click button.

Step5: Select and Click on polyline option button

Algorithm for run ID3 and J48 Classification in WEKA

Step1: Open WEKA Tool.

Step2: Click on WEKA Explorer.

Step3: Click on Preprocessing tab button.

Step4: Click on open file button.

Step5: Choose WEKA folder in C drive.

Step6: Select and Click on data option button.

Step7: Choose iris data set and open file.

Step8: Click on classify tab and Choose J48 algorithm and select use training set test option

Step9: Click on start button.

Step10: Click on classify tab and Choose ID3 algorithm and select use training set test option.

Step11: Click on start button.

Step12: Click on classify tab and Choose Naïve-bayes algorithm and select use training set test option.

Step13: Click on start button.

Step14: Click on classify tab and Choose k-nearest neighbor and select use training set test option.

Step15: Click on start button.

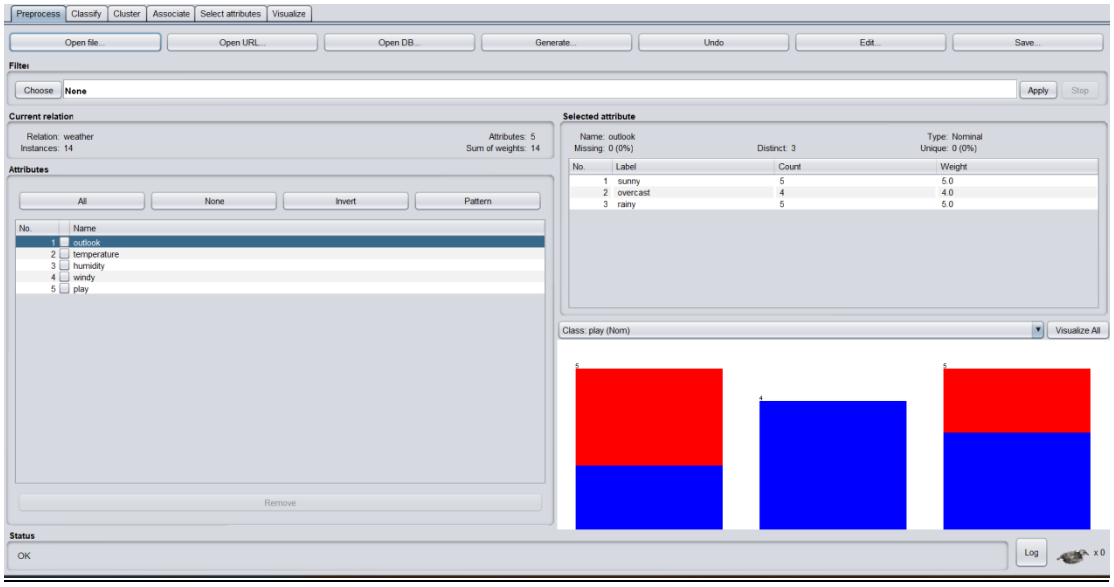
Experiment-6

AIM: Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observations (using WEKA).

Description:

Setting Test Data

Load the weather dataset into the weka.



Filter: None

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

Attributes

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

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

We will use the preprocessed weather data file from the previous lesson. Open the saved fileby using the Open file ... option under the Preprocess tab, click on the Classify tab, and youwould see the following screen –

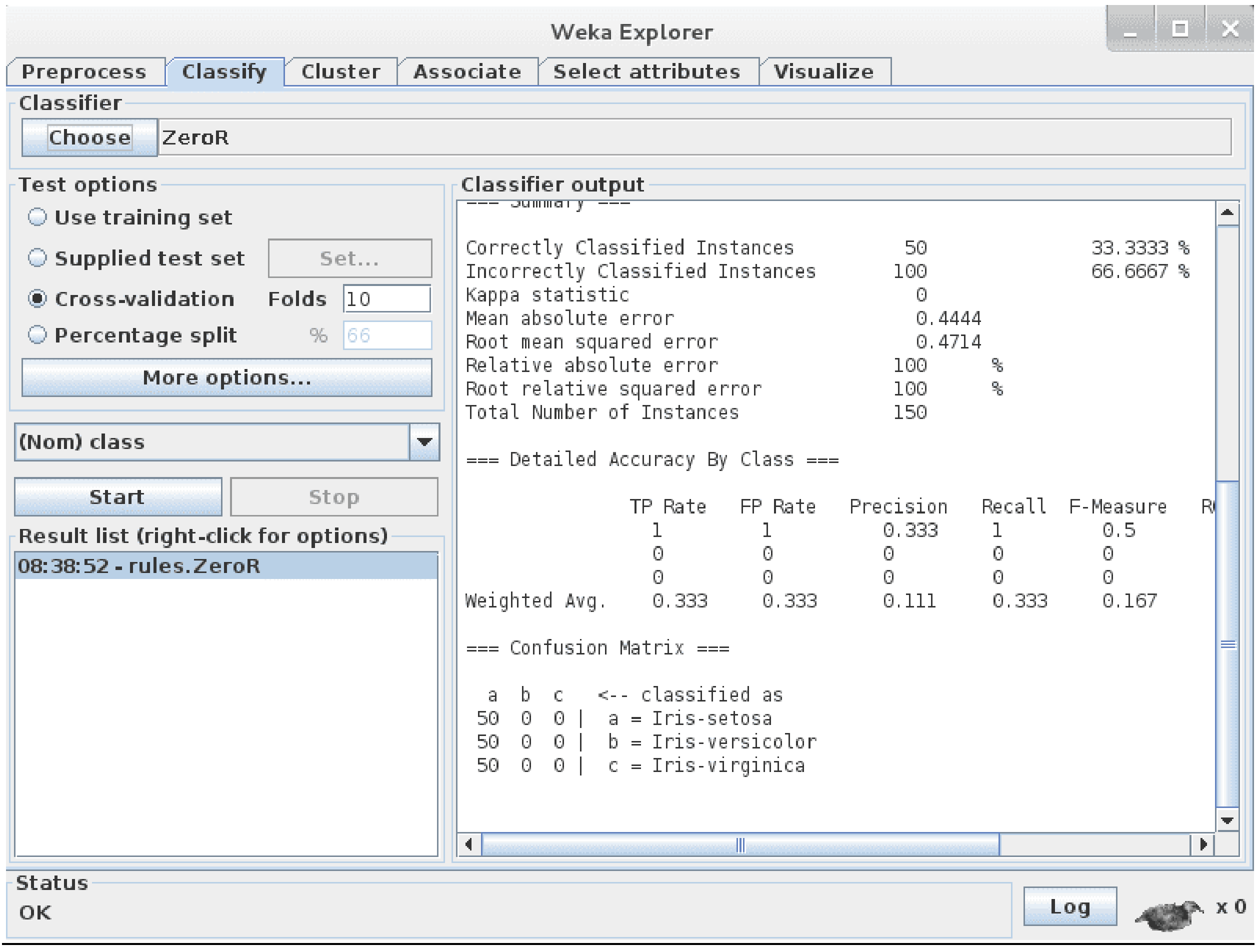
Before you learn about the available classifiers, let us examine the Test options. You will notice four testing options as listed below −

- Training set
- Supplied test set
- Cross-validation
- Percentage split

Unless you have your own training set or a client supplied test set, you would use cross-validation or percentage split options. Under cross-validation, you can set the number of folds in which entire data would be split and used during each iteration of training. In the percentage split, you will split the data between training and testing using the set split percentage.Now, keep the default play option for the output class –

Next, you will select the classifier.
Selecting Classifier
Click on the Choose button and select the following classifier −
weka→classifiers>trees>J48
This is shown in the screenshot below-

Click on the Start button to start the classification process. After a while, the classification results would be presented on your screen as shown here –
Let us examine the output shown on the right hand side of the screen.
It says the size of the tree is 6. You will very shortly see the visual representation of the tree.
In the Summary, it says that the correctly classified instances as 2 and the incorrectly classified instances as 3, It also says that the Relative absolute error is 110%. It also shows the Confusion Matrix. Going into the analysis of these results is beyond the scope of this tutorial. However, you can easily make out from these results that the classification is not acceptable and you will need more data for analysis, to refine your features selection, rebuild the model and so on until you are satisfied with the model's accuracy. Anyway, that's what WEKA is all about. It allows you to test your ideas quickly.
The Iris Flower dataset is a famous dataset from statistics and is heavily borrowed by researchers in machine learning. It contains 150 instances (rows) and 4 attributes (columns) and a class attribute for the species of iris flower (one of setosa, versicolor, and virginica).
Visualize Results
To see the visual representation of the results, right click on the result in the Result list box. Several options would pop up on the screen as shown here –



ZeroR

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

(Nom) class

08:38:52 - rules.ZeroR

```
=== Summary ===

Correctly Classified Instances          50               33.3333 %
Incorrectly Classified Instances       100               66.6667 %
Kappa statistic                          0
Mean absolute error                      0.4444
Root mean squared error                  0.4714
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall   F-Measure   R
                 1         1         0.333      1         0.5
                 0         0         0          0         0
                 0         0         0          0         0
Weighted Avg.    0.333     0.333     0.111      0.333     0.167

=== Confusion Matrix ===

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

Status: OK

Algorithm:

Step1: Load the weather dataset into the weka.

Step2: Click the “Open file…” button to open a data set and double click on the “data” directory.

Step3: Select the “iris.arff” file to load the Iris dataset.

Step4: choose a machine learning algorithm to model the problem and make predictions.

Step5: Click the “Classify” tab. This is the area for running algorithms against a loaded dataset in Weka.

Step6: The “ZeroR” algorithm is selected by default.

Step7: Click the “Start” button to run this algorithm.

Experiment-7

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

Description:

Naive Bayes is among one of the most simple and powerful algorithms for classification based on Bayes' Theorem with an assumption of independence among predictors. Naive Bayes model is easy to build and particularly useful for very large data sets. There are two parts to this algorithm:

- Naive
- Bayes

The Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that a particular fruit is an apple or an orange or a banana and that is why it is known as "Naive".

Algorithm:

Step 1: Calculate the prior probability for given class labels

Step 2: Find Likelihood probability with each attribute for each class

Step 3: Put these value in Bayes Formula and calculate posterior probability.

Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

Step 5: In the case of multiple features calculate prior probability for given class labels

Step6: Calculate conditional pprobability with each attribute for each class.

Step7: Multiply same class conditional probability.

Step8: Multiply prior probability with multiply same class conditional probability.

Step9: See which class has higher probability, higher probability class belongs to given input set step.

Experiment-8

AIM: Demonstrate performing clustering of 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. Derive interesting insights and explain.

Description:

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.

<u>Algorithm for run K-mean Clustering algorithms in WEKA</u>

Step1: Open WEKA Tool.

Step2: Click on WEKA Explorer.

Step3: Click on Preprocessing tab button.

Step4: Click on open file button.

Step5: Choose WEKA folder in C drive.

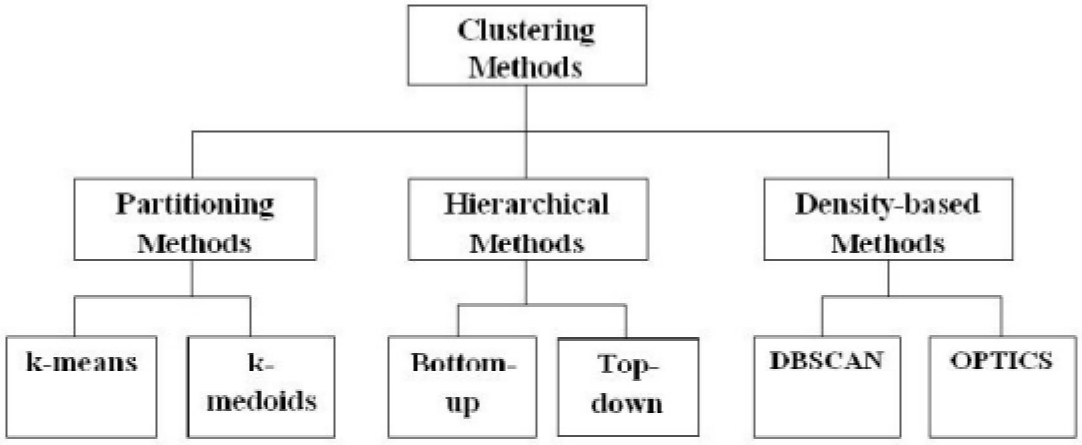
Step6: Select and Click on data option button.

Step7: Choose iris data set and open file.

Step8: Click on cluster tab and Choose k-mean and select use training set test option.

Step9: Click on start button.

Clustering Algorithms And Techniques in WEKA, They are

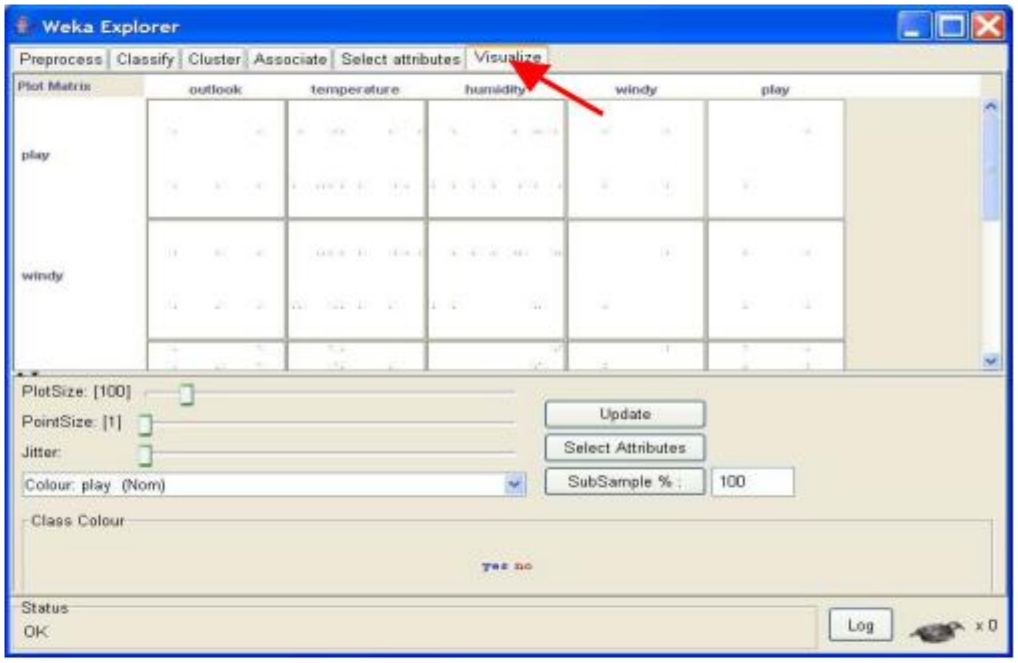


Visualize Features

WEKA’s visualization allows you to visualize a 2-D plot of the current working relation. Visualization is very useful in practice, it helps to determine difficulty of the learning problem. WEKA can visualize single attributes (1-d) and pairs of attributes (2-d), rotate 3-d visualizations (Xgobi-style). WEKA has “Jitter” option to deal with nominal attributes and to detect “hidden” data points.

Access To Visualization From The Classifier, Cluster And Attribute Selection Panel Is Available From A Popup Menu. Click The Right Mouse Button Over An Entry In The Result List To Bring Up The Menu. You Will Be Presented With Options For Viewing Or Saving The Text Output And --- Depending On The Scheme --- Further Options For Visualizing Errors, Clusters, Trees Etc.

To open Visualization screen, click ‘Visualize’ tab

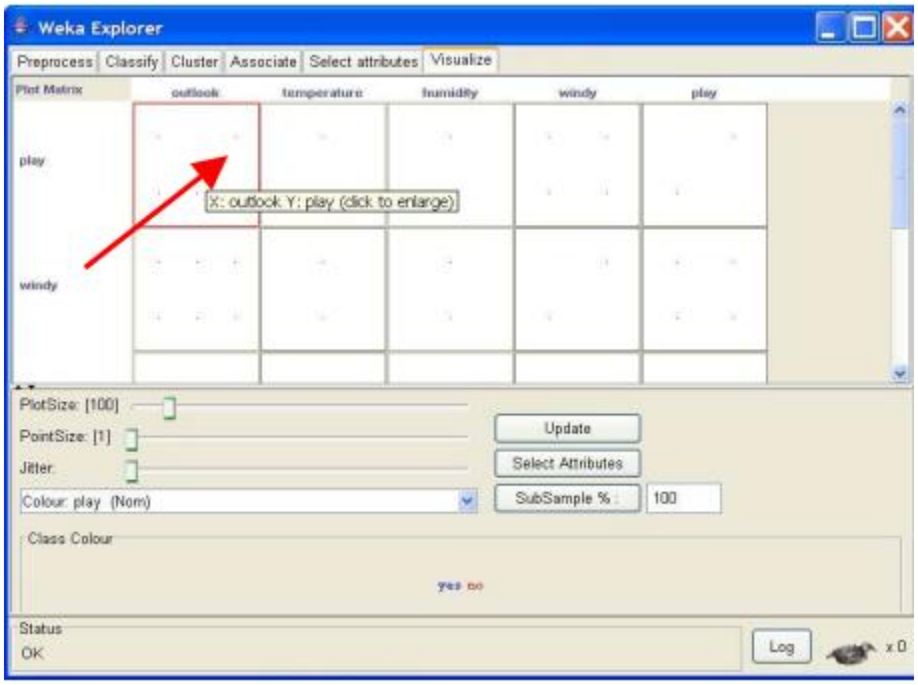


Select a square that corresponds to the attributes you would like to visualize. For example, let’s choose ‘outlook’ for X – axis and ‘play’ for Y – axis. Click anywhere inside the square that corresponds to ‘play o

Changing the View:

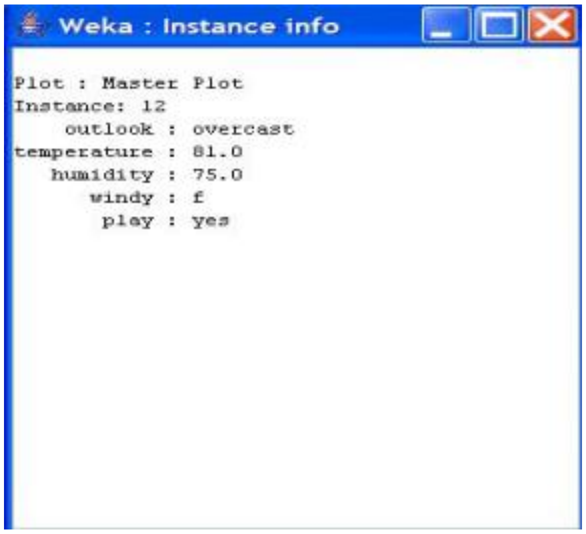
In the visualization window, beneath the X-axis selector there is a drop-down list, ‘Colour’, for choosing the color scheme. This allows you to choose the color of points based on the attribute selected. Below the plot area, there is a legend that describes what values the colors correspond to. In your example, red represents ‘no’, while blue represents ‘yes’. For better visibility you should change the color of label ‘yes’. Left-click on ‘yes’ in the ‘Class colour’ box and select lighter color from the color palette.

n the left and 'outlook' at the top.



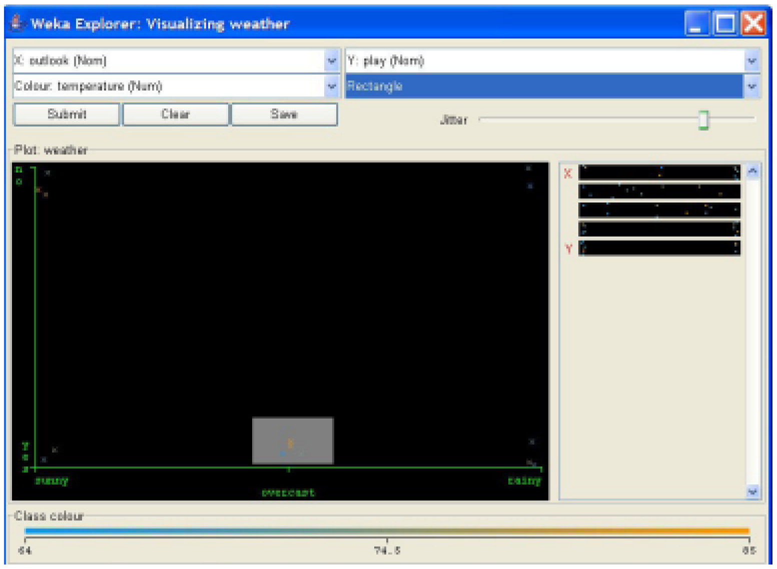
Algorithm:

Step1: Select Instance. Click on an individual data point. It brings up a window listing attributes of the point. If more than one point will appear at the same location, more than one set of attributes will be shown.

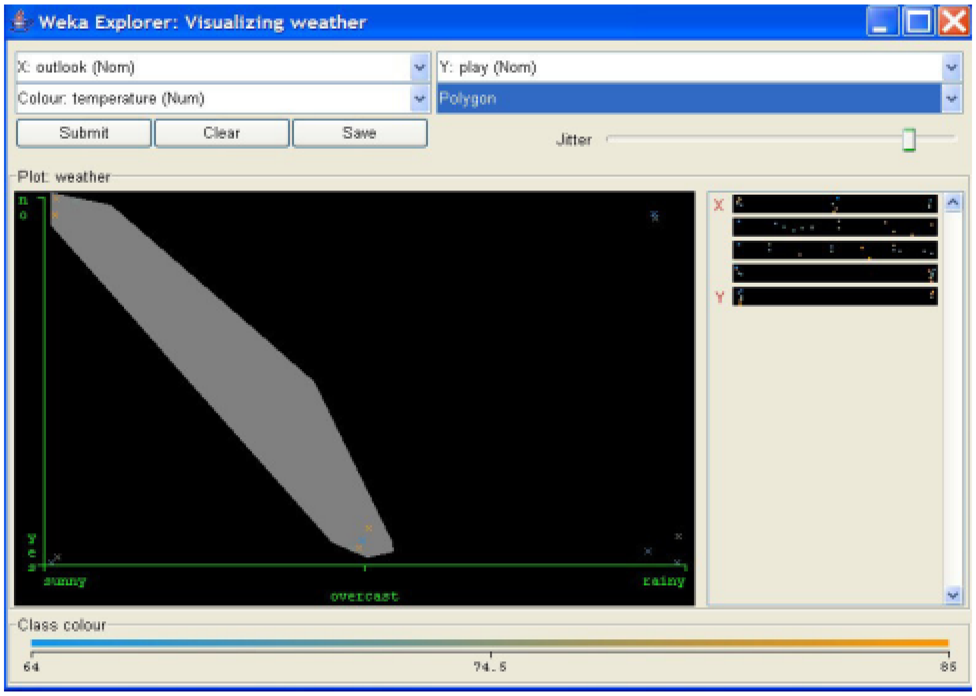


```
Plot : Master Plot
Instance: 12
    outlook : overcast
temperature : 81.0
   humidity : 75.0
      windy : f
       play : yes
```

Step2: Rectangle. You can create a rectangle by dragging it around the point.



Step3: Polygon. You can select several points by building a free-form polygon. Left-click on the graph to add vertices to the polygon and right-click to complete it.



Step4: Polyline. To distinguish the points on one side from the once on another, you can build a polyline. Left-click on the graph to add vertices to the polyline and right-click to finish.

Experiment-9

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

Description:

K-means is an unsupervised learning method for clustering data points. The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster.

First, each data point is randomly assigned to one of the K clusters. Then, we compute the centroid (functionally the center) of each cluster, and reassign each data point to the cluster with the closest centroid. We repeat this process until the cluster assignments for each data point are no longer changing.

K-means clustering requires us to select K, the number of clusters we want to group the data into. The elbow method lets us graph the inertia (a distance-based metric) and visualize the point at which it starts decreasing linearly. This point is referred to as the "eblow" and is a good estimate for the best value for K based on our data.

Algorithm:

Step 1: Select the value of K to decide the number of clusters (n_clusters) to be formed.

Step 2: Select random K points that will act as cluster centroids (cluster_centers).

Step 3: Assign each data point, based on their distance from the randomly selected points (Centroid), to the nearest/closest centroid, which will form the predefined clusters.

Step 4: Place a new centroid of each cluster.

Step 5: Repeat step no.3, which reassigns each datapoint to the new closest centroid of each cluster.

Step 6: If any reassignment occurs, then go to step 4; else, go to step 7.

Step 7: Finish

Experiment-10

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

Description:

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.

Algorithm:

Step 1: Importing the required libraries

Step 2: Loading and exploring the data

Step 3: Cleaning the Data

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

Step 5: Hot encoding the Data

Step 6: Building the models and analyzing the results

Experiment-11

AIM: Write a Python program to generate frequent item sets / association rules using FP-growth Tree algorithm.

Description:

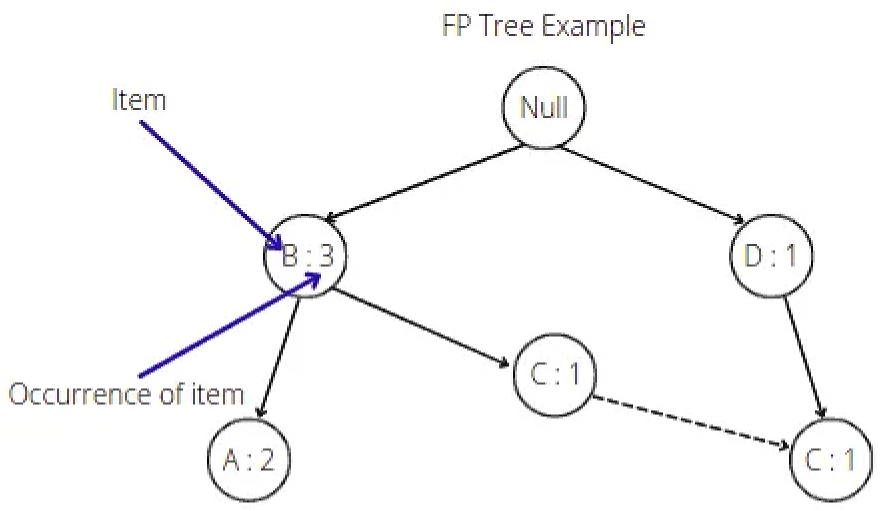
FP-growth algorithm is a tree-based algorithm for frequent itemset mining or frequent-pattern mining used for market basket analysis. The algorithm represents the data in a tree structure known as FP-tree, responsible for maintaining the association information between the frequent items.

The algorithm compresses frequent items into an FP-tree from the database while retaining association rules. Then it splits the database data into a set of conditional databases (a special kind of projected database), each of which is associated with one frequent data item.

FP-Tree

FP-tree is the core concept of the FP-growth algorithm. The FP-tree is a compressed representation of the database itemset, storing the DB itemset in memory and keeping track of the association between items.

The tree is constructed by taking each itemset and adding it as a subtree. The FP-tree's whole idea is that items that occur more frequently will be more likely to be shared.



The root node in the FP-tree is null. Each node of the subtree stores at least the item name and the support (or item occurrence) number. Additionally, the node may contain a link to the node with the same name from another subtree (represents another itemset from the database).

Building FP-Tree

The FP-growth algorithm uses the following steps to build FP-tree from the database.

- Scan itemsets from the database for the first time
- Find frequent items (single item patterns) and order them into a list `L` in frequency descending order. For example, `L = {A:5, C:3, D;2, B:1}`
- For each transaction order its frequent items according to the order in `L`
- Scan the database the second time and construct FP-tree by putting each frequency ordered transaction onto it.

Algorithm:

Step1: Take all transactions from a database and create a list of all items which appear inside the database.

Step2: Sort this list by the frequency of the item in all transactions

Step3: Then begin a recursive tree, starting with the item with the highest frequency and use the list to create nodes extending down from the top node.

Step4: For each node, find all transactions which contain the item.

Step5: Add a node to each path down the tree.These nodes represent each item in the transaction.

Step6: Recursively repeat this process and as each item appears, traverse down the tree and add it to the path.

Step7: Each path of nodes leading to a leaf node represents a set of items which often occur together.

Experiment-12

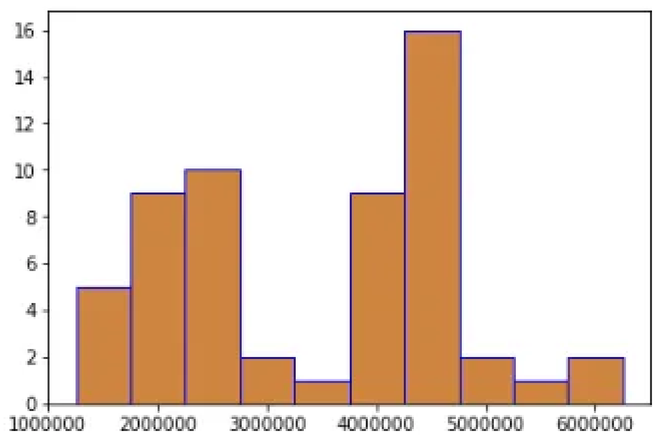
AIM: Visualize the datasets using matplotlib in python.(Histogram, Box plot, Bar chart, Pie chart etc.,).

Description:

Matplotlib is a powerful Python library used for creating visualizations of data, such as histograms, box plots, bar charts, pie charts, and more. Each of these diagrams provides a different way to view and interpret the same data set, allowing users to identify patterns, compare distributions, and draw conclusions. Histograms and bar charts provide insights into the frequency of values in the data set, while box plots and pie charts provide a visual representation of summary statistics. Regardless of the type of visualization used, matplotlib is an essential tool for examining the data set more closely and deepening the user's understanding.

Histogram

A histogram takes in a series of data and divides the data into a number of bins. It then plots the frequency data points in each bin (i.e. the interval of points). It is useful in understanding the count of data ranges.



Algorithm:

Step1: Prompt the user to input a list of values

Step2: Initialize a dictionary to store the values

Step3: Iterate through the list of values

Step4: For each value, check if it already exists in the dictionary. If it does, increment its value in the dictionary

Step5: If it does not exist, add the value to the dictionary with a value of 1

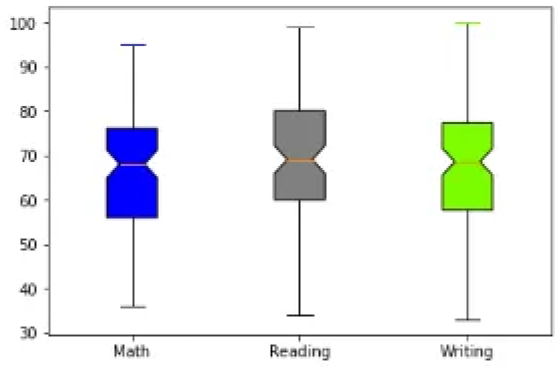
Step6: After iterating through the list, create a loop to print out the histogram

Step7: Within the loop, access the keys in the dictionary and print the associated values, multiplied by a * character for each occurrence of the value

Step8: After printing, exit the loop

<u>Box plot:</u>

A boxplot is a graphical tool for representing data sets, designed to show the distribution of data in five different categories, including the median, quartiles, minimum and maximum values. A boxplot can be used to compare multiple data sets, identify potential outliers, and interpret differences between groups. In Data Mining, boxplots can be used to compare variables to see how they are distributed, to show if there are significant differences between different groups, or to analyze overall trends in the data.



<u>Algorithm:</u>

Step1: Create a list of data points

Step2: Remove outliers

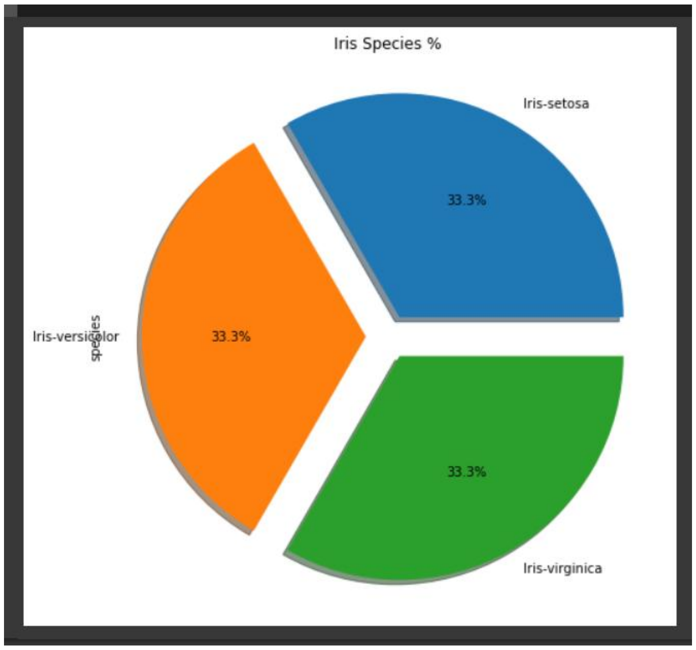
Step3: Calculate median, first quartile (Q1), third quartile (Q3) and interquartile range (IQR)

Step4: Calculate lower cut-off (Q1 − 1.5*IQR) and upper cut-off (Q3 + 1.5*IQR) value

Step5: Plot data points as a box using lower cut-off, Q1, median, Q3 and upper cut-off as the boundary of the box. Points outside the boundary are outliers.

Pie chart

A Pie Chart is a useful tool for displaying data visually in data mining. It allows a viewer to easily understand the relative proportions between different data points in a dataset. It usually uses a circle divided into slices, each slice representing a percentage of the total dataset. The size and color of each slice can vary, helping to communicate the data's hierarchy. Pie charts are also very space-efficient, easily representing complex data in a simple format.

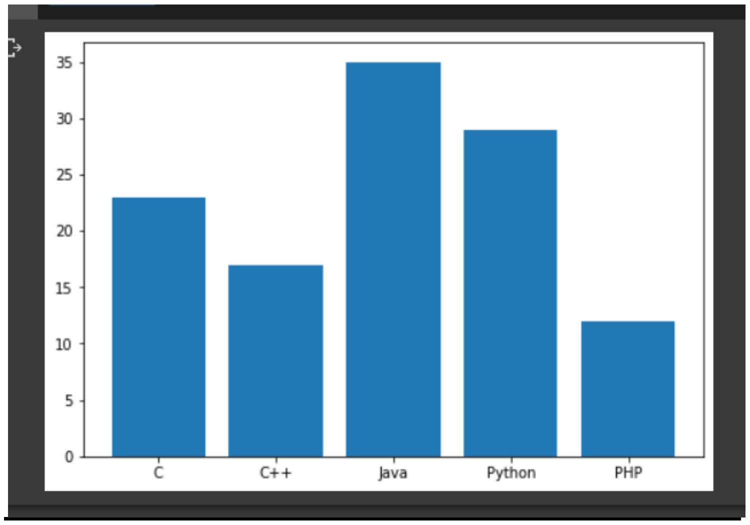


Algorithm:

Input: List of values and labels of a Pie chart
Step 1: Import the "matplotlib" library and use the pyplot module.
Step 2: Create a list of values and labels for the Pie chart from the user-defined variables.
Step 3: Create a figure to store the results of the Pie chart.
Step 4: Use the "pie()" function of the pyplot module to create a Pie chart within the figure.
Step 5: Assign the values and labels from the list declared earlier to the Pie chart.
Step 6: Generate the Pie chart by calling the "show()" function of the pyplot module.
Step 7: Close the figure window by calling the "close()" function of the pyplot module.

Bar chart:

A bar chart in data mining is a type of graph that is used to compare the values of different data sets. It is a graphical representation of data that is organized by categories and is used to analyze relationships between variables. The bars represent the different data points, with the length of the bar being proportional to the value of the data point. They are used to compare distributions and reveal patterns in the data. Bar charts can be used to compare values across categories, analyze changes over time, and compare two or more variables.



<u>Algorithm:</u>

Step1: Gather the necessary data for the graph.
Step2: Organize the data into categories, if applicable.
Step3: Determine the type of graph, bar chart, and the appropriate axes- labels, range, scale etc.
Step4: Draw the bar chart with the appropriate labels, scales.
Step5: Attribute the appropriate data points to the axes and the chart.
Step6: Add a scale for each axis (if necessary).
Step7: Analyze the information in the chart to gain insight.

Augmented Experiments

Experiment -13

AIM: Write a Python program to prepare a simulated data set with unique instances.

Description:

A simulated data set with unique instances in Python can be created using a variety of techniques, such as generating random numbers, creating objects randomly from a predefined library of Python classes, or directly modifying existing data structures. Python's random module can be used to generate random numbers and produce pseudo-random datasets. Specialized Python libraries, such as NumPy, SciPy, and others, provide routines for creating and manipulating random and pseudo-random data structures. Finally, many existing Python libraries offer methods and functions to generate simulated data sets with unique instances**.**To prepare a simulated data set with unique instances in Python, you can use the sample() function from the random module. This function enables you to generate a specified number of unique elements from a provided list of sequence or set. Also, you can utilize the unique_everseen() method from itertools module to get unique instances from an existing comparative data set. Of course, you should also remember to set your random state so that each run will always generate the same simulated data set with unique instances.

We will use the sklearn library that provides various generators for simulating classification data.

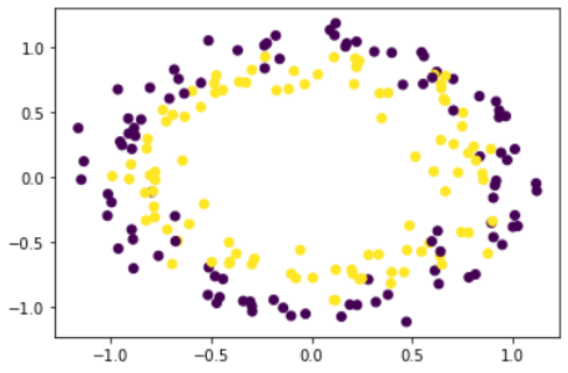
- **Single Label Classification:**

Single Label Classification is a supervised machine learning technique commonly used for categorizing data into a single class. It is an iterative process, starting with a dataset composed of labeled data and using algorithms to find patterns in the data and produce classifications. It is often used to identify objects in images or individuals in audio data, and can also be used to determine the sentiment of or relevance of texts.

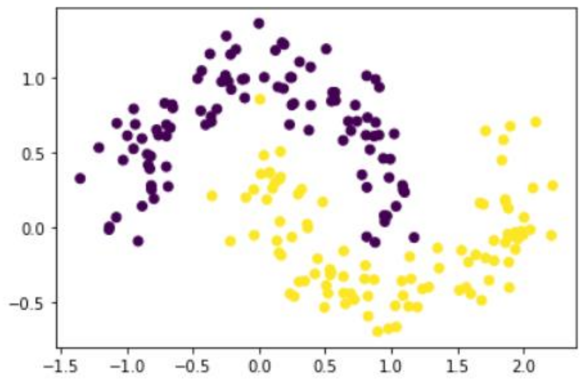
Here we are going to see single-label classification, for this we will use some visualization techniques.

Example 1: Using make_circles()

make_circles generates 2d binary classification data with a spherical decision boundary



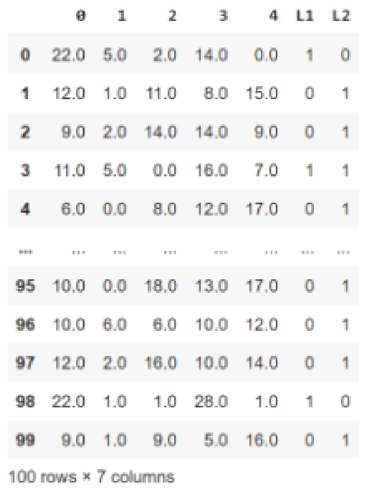
Example 2: Using make_moons()



- **Multi-Label Classification:**

Multi-label classification is a supervised learning task in which multiple labels are assigned to each instance of data. It is a type of problem in machine learning, where multiple labels need to be applied to each instance of data. It is often used in natural language processing and computer vision tasks where multiple tags are associated to each data point. Multi-label classification is a more challenging task than traditional classification tasks as it deals with multiple labels associated with each data instance make_multilabel_classification() generates a random multi-label classification problem.

Example:



	0	1	2	3	4	L1	L2
0	22.0	5.0	2.0	14.0	0.0	1	0
1	12.0	1.0	11.0	8.0	15.0	0	1
2	9.0	2.0	14.0	14.0	9.0	0	1
3	11.0	5.0	0.0	16.0	7.0	1	1
4	6.0	0.0	8.0	12.0	17.0	0	1
...	...	...	...	...	...	...	...
95	10.0	0.0	18.0	13.0	17.0	0	1
96	10.0	6.0	6.0	10.0	12.0	0	1
97	12.0	2.0	16.0	10.0	14.0	0	1
98	22.0	1.0	1.0	28.0	1.0	1	0
99	9.0	1.0	9.0	5.0	16.0	0	1

100 rows × 7 columns

Algorithm:

Step 1 : Import the library - GridSearchCv

Step 2 : Generating the data

Here we are using make_classification to generate a classification data. We have stored features and targets.

- **n_samples:** It signifies the number of samples(row) we want in our dataset. By default it is set to 100
- **n_features:** It signifies the number of features(columns) we want in our dataset. By default it is set to 20
- **n_informative:** It is used to set the number of informative class. By default it is set to 2
- **n_redundant :** It is used to set number of redundant features. The features which can be generated as random linear combinations of the informative features. By default it is set to 2
- **n_classes :** This signifies the number of classes in target dataset.

Step 3 : Viewing the dataset

Experiment -14

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

Description:

Dissimilarity Matrix:
The dissimilarity matrix (also called distance matrix) describes pairwise distinction between M objects. It is a square symmetrical MxM matrix with the (ij)th element equal to the value of a chosen measure of distinction between the (i)th and the (j)th object. The diagonal elements are either not considered or are usually equal to zero - i.e. the distinction between an object and itself is postulated as zero.

A closely related and the opposite concept is the similarity matrix . Both types of description are often used for the same data.Any reasonable measure of dissimilarity may be used, including subjective scores of dissimilarity. The only requirement is that the greter distinction between two objects, the greater the value the measure of dissimilarity.As far as the dissimilarity matrix is specified, a corresponding similarity matrix can be calculated, e.g by transformations like

Sij = C - Dij,

Sij = 1/(Dij+e),

where Sij are pairwise values of a similarity measure, Dij - of a dissimilarity measure.
In many branches of multivariate statistical analysis the initial data are represented as the dissimilarity (or distance) matrices, and/or as similarity matrices, e.g. in multidimensional scaling , correspondence analysis , cluster analysis .

Algorithm:

Step 1: Input the dataset in a form of a table with the columns representing the attributes and the rows representing the instances.

Step 2: Initialize an empty table for the dissimilarity matrix.

Step 3: Consider the first two rows from the dataset and calculate the Euclidean distance between the instances.

Step 4: Enter the calculated distance in the dissimilarity matrix as the element for the first row and the first column.

Step 5: Now take the same instance from the first row and another instance from the third row and repeat the same procedure.

Step 6: Enter the calculated result in the dissimilarity matrix as the element for the first row and the second column.

Step 7: Repeat the same procedure for all combinations of instances.

Step 8: Finally, display the dissimilarity matrix.

Experiment -15

AIM: Write a program of cluster analysis using DB SCAN algorithm Python programming language

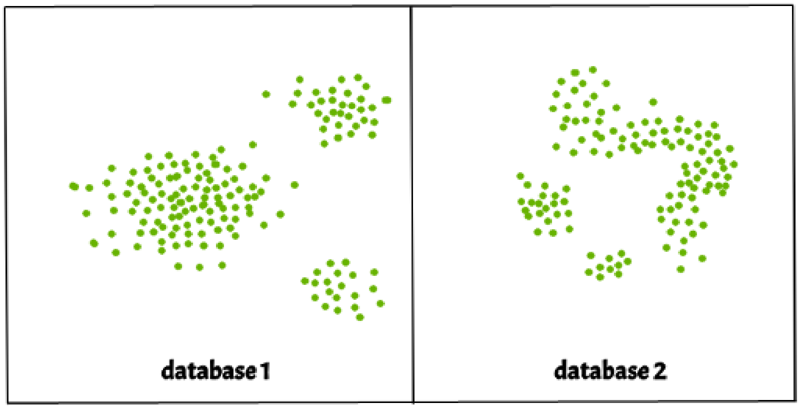
Description:

Clustering analysis or simply Clustering is basically an Unsupervised learning method that divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense. It comprises many different methods based on differential evolution.

E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), Spectral clustering (graph distance) etc.

Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches. Here we will focus on Density-based spatial clustering of applications with noise (DBSCAN) clustering method.

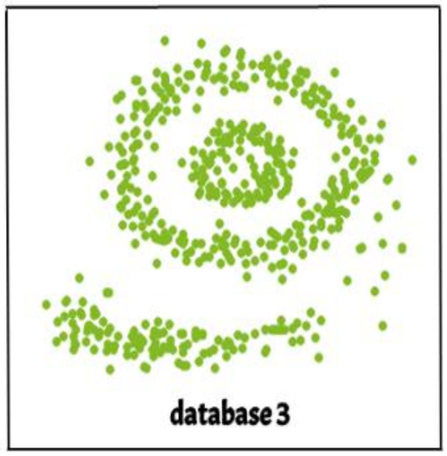
Clusters are dense regions in the data space, separated by regions of the lower density of points. The DBSCAN algorithm is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.



Why DBSCAN?

Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.Real life data may contain irregularities, like:

- Clusters can be of arbitrary shape such as those shown in the figure below.
- Data may contain noise.



The figure below shows a data set containing nonconvex clusters and outliers/noises. Given such data, k-means algorithm has difficulties in identifying these clusters with arbitrary shapes.

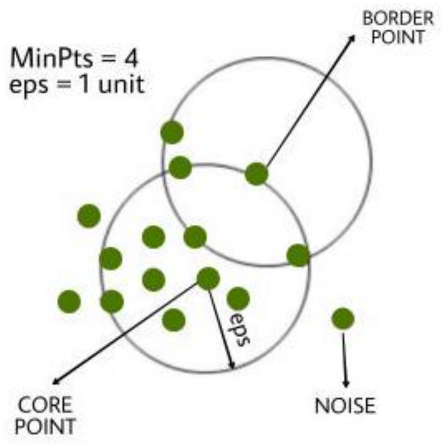
DBSCAN algorithm requires two parameters:

- **eps :** It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters. One way to find the eps value is based on the k-distance graph.
- **MinPts:** Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

In this algorithm, we have 3 types of data points.

1. **Core Point:** A point is a core point if it has more than MinPts points within eps.

2. **Border Point:** A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.
3. **Noise or outlier:** A point which is not a core point or border point.



Algorithm:

Step 1: Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.

Step 2: For each core point if it is not already assigned to a cluster, create a new cluster.

Step 3: Find recursively all its density connected points and assign them to the same cluster as the core point.

Step 4: A point a and b are said to be density connected if there exist a point c which has a sufficient number of points in its neighbors and both the points a and b are within the eps distance. This is a chaining process. So, if b is neighbor of c, c is neighbor of d, d is neighbor of e, which in turn is neighbor of a implies that b is neighbor of a.

Step 5: Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise**.**

Experiment -16

AIM: Demonstrate Web/Text Mining using WEKA Tool

Description:

Weka – Waikato Environment for Knowledge Analysis (Weka) :

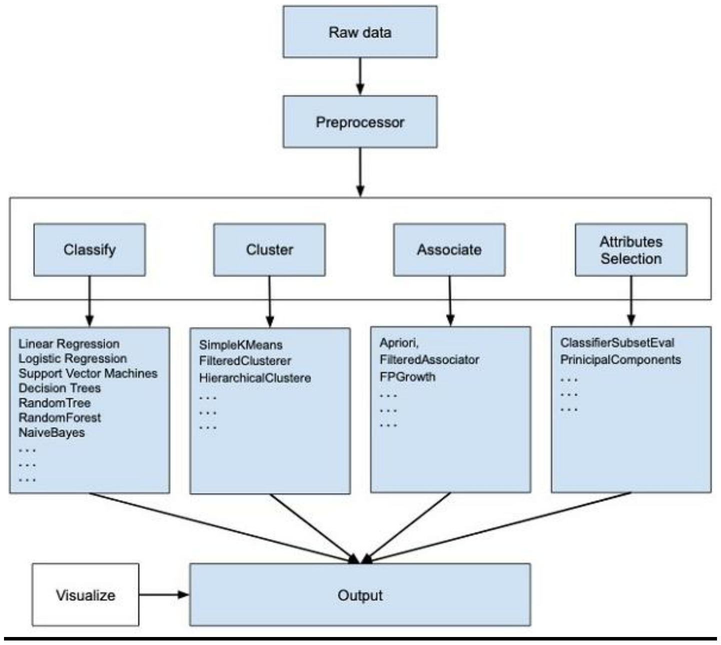
It is a suite of machine learning software developed at the University of Waikato, New Zealand. The program is written in Java. It contains a Collection of visualization tools and algorithms for data analysis and predictive modeling coupled with graphical user interface. Weka supports several standard data mining tasks, more specifically, data pre-processing, clustering, classification, regressing, visualization and feature selection.

Prerequisites

- Know about how to install weka tool
- Know about tools of the weka tool
- Know about upload the dataset
- Know about filetypes of the dataset
- Know about machine learning algorithm

WEKA is a data mining tool which can be used to do web or text mining. It can be used to pre-process, classify, cluster, visualize and analyze data sets. It includes algorithms like Naive Bayes, Linear regression and Logistic regression for text classification, pattern recognition and clustering. It also has tools to recognize and extract text from HTML, XML, PDF and other web formats. Additionally, the software allows for association rule mining and predictive analytics.

Using WEKA tool, Web/Text mining includes various activities such as gathering, extracting, structuring, cleaning, and mining data from the web or text. It helps to mainly understand the user's behavior in retail, e-commerce, sentiment analysis, and document classification. It also helps in analyzing the relationships between different words in documents. By applying WEKA's data mining functions, professionals can quickly and easily find patterns and correlations in web or text data.

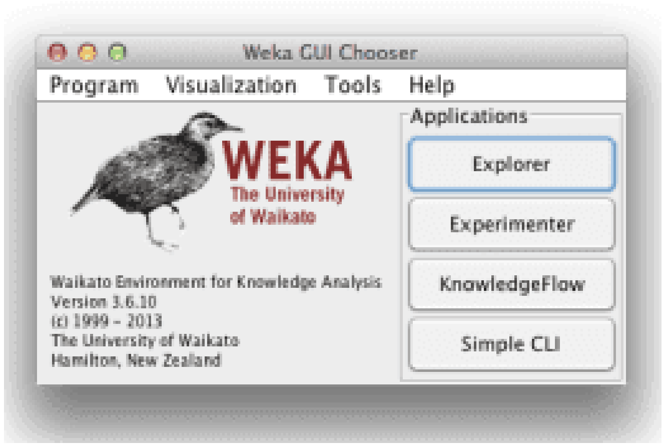


1. How to download Weka and Install

We can download the weka tool by clicking the Weka Download page and find a version of weka depends on your computer OS. (Windows, Mac, or Linux). Weka needs java. If we already install java don't care about but if not use this link to install Java.

2. Let's Start Weka

If you installed a successful weka tool you get an image looks like on your desktop or by double clicking on the weka.jar file. This is a GUI window that means we don't know about any idea just click and use , yes like that this window displays four types of options , Explorer, Experimenter, KnowledgeExplorer and Simple CLI Command line interface.



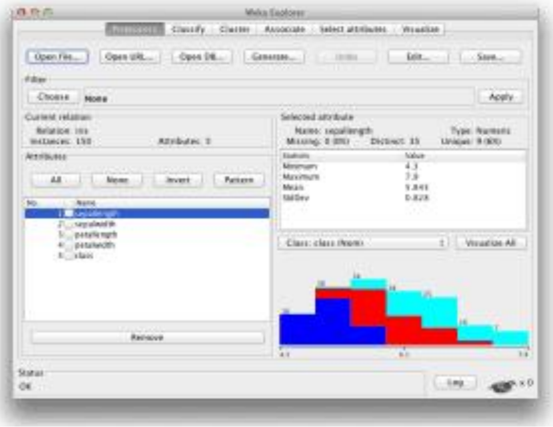
Weka GUI Chooser

Click the “Explorer” button to launch the Weka Explorer.

This GUI lets you load datasets and run classification algorithms. It also provides other features, like data filtering, clustering, association rule extraction, and visualization, but we won’t be using these features right now.

3. How to Open the data/iris.arff Dataset

First you go to the “Open file” button to open the data set and double click on the data directory. Weka tools provide some common machine learning datasets. Otherwise we can create our own dataset and load it to future use. Now we are going to upload the Iris dataset. In machine learning before we move on a particular dataset we must know about that data clearly then only we can find better patterns . Here is an iris flower image.



Filter: None

Current relation
Relation: iris
Instances: 150   Attributes: 5

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

Weka Explorer Interface with the Iris dataset loaded

The Iris Flower dataset is a famous dataset from statistics and is heavily borrowed by researchers in machine learning. It contains 150 instances (rows) and 4 attributes (columns) and a class attribute for the species of iris flower (one of setosa, versicolor, and virginica).

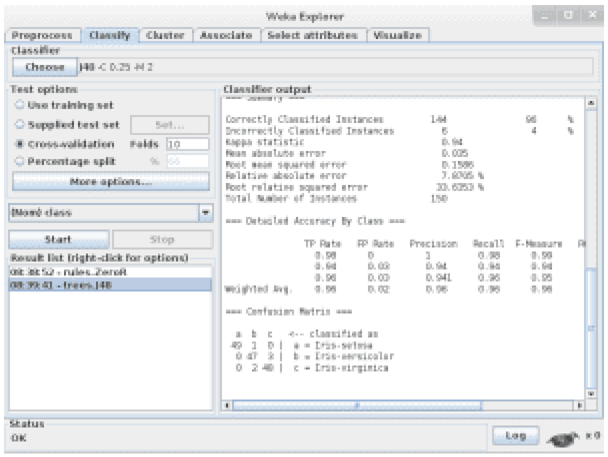
4.Now Select and Run an Algorithm

Here one point is noted , this data set is a previously cleaned data set so we don't need to do any data preprocessing, but in case our own dataset means we must concentrate on the EDA (Exploratory Data Analysis) and Data preprocessing part. Now we loaded a dataset, then we can choose a machine learning algorithm to model the problem and make predictions

Click the "Classify" tab. This is the area for running algorithms against a loaded dataset in Weka.

You will note that the "ZeroR" algorithm is selected by default.

Click the "Start" button to run this algorithm.



Classifier: J48 -C 0.25 -M 2

Cross-validation Folds 10

(Nom) class

08:38:52 - rules.ZeroR
08:39:41 - trees.J48

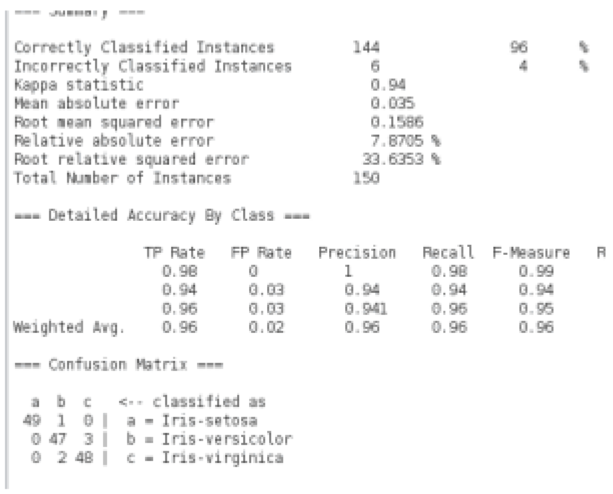
```
Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure
                 0.98      0          1          0.98     0.99
                 0.94      0.03       0.94       0.94     0.94
                 0.96      0.03       0.941      0.96     0.95
Weighted Avg.    0.96      0.02       0.96       0.96     0.96

=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Status: OK

Weka J48 algorithm results on the Iris flower dataset

5. Compare the Results

Here running the J48 algorithm, we can note the results in the "Classifier output" section.The algorithm was run with 10-fold cross-validation: this means it was given an opportunity to make a prediction for each instance of the dataset (with different training folds) and the presented result is a summary of those predictions.

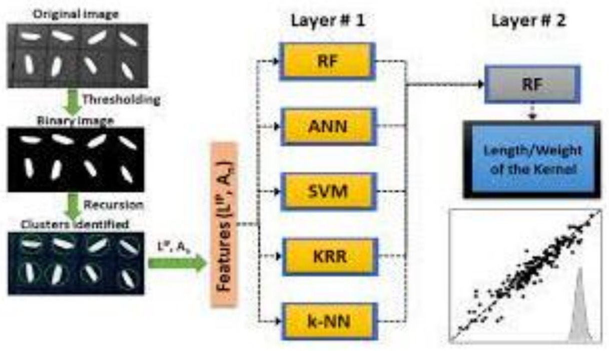


```
=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   R
                 0.98      0         1         0.98      0.99
                 0.94      0.03      0.94      0.94      0.94
                 0.96      0.03      0.941     0.96      0.95
Weighted Avg.    0.96      0.02      0.96      0.96      0.96

=== Confusion Matrix ===

  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 47  3 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

ust the results of the J48 algorithm on the Iris flower dataset in Weka

Firstly, note the Classification Accuracy. You can see that the model achieved a result of 144/150 correct or 96%, which seems a lot better than the baseline of 33%.

Secondly, look at the Confusion Matrix. You can see a table of actual classes compared to predicted classes and you can see that there was 1 error where an Iris-setosa was classified as an Iris-versicolor, 2 cases where Iris-virginica was classified as an Iris-versicolor, and 3 cases where an Iris-versicolor was classified as an Iris-setosa (a total of 6 errors). This table can help to explain the accuracy achieved by the algorithm.



<u>Algorithm:</u>

Step 1: Install WEKA and load the required dataset.

Step 2: Use mining algorithms such as Apriori or Naïve Bayes for Web mining or K Means for text mining

Step 3: Preprocess the data if needed (e.g. convert HTML tags to plain text).

Step 4: Train the model by adjusting parameters of the algorithms and set criteria for the evaluation

Step 5: Visualize the results using tools provided by WEKA to make sense of the data.

Step 6:. Evaluate the results and generate insights from the mined data.