Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss RAGHAV . VIKRAM RATHI .

of Comptr Department, Semester V th with

Roll No. 2113208 has completed a course of the necessary

experiments in the subject Data warehousing and mining under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In- Charge 19/10/23

Head of the Department

Date 19/10/2023

Principal

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.	select a dataset and perform exploratory data analysis using python (data preprocessing, transformation, discretization and visualization.		25/7/23	
2.	One case study on building Data warehouse DataMart. Write Detailed Problem Statement and design dimensional and modeling (Schema of Start and Snoflex)		1/8/23	
3.	Implementation of all dimension table and fact table based on Experiment 1 case study.		15/8/23	
4.	Implementation of OLAP operation: Slice Dice Rollup, Drilldown & Pivot on experiment 2		22/8/23	
5.	Implementation of Bayesian Algorithm		29/8/23	
6.	Perform data: Preprocessing task and Demonstrate performing classification Clustering, Association algorithm on data sets using data mining tool (WEKA/R tool)		5/9/23	
7.	Implementation of Clustering algorithm. (K-mean / K-median).		12/9/23	
8.	Implementation of any one Hierichical Clustring method.		26/9/23	
9.	Implementation of Association Rule Mining algorithm (Apriori		3/10/23	Sign 19/10/23
10.	Implementation of Page Rank / HITS Algorithm.		3/10/23	
	Assignment 1		25/9/23	
	Assignment 2		6/10/23	

Raghav V. Rathi

C34

2113208

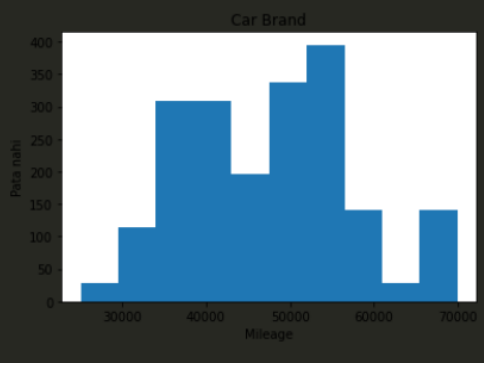
Practical 1

```
import pandas as pd
df = pd.read_csv("Car Data.csv") #I am working in Windows environment
#Reading the dataset in a dataframe using Pandas
print(df.head(3))
```

```
[3]
...
   Car ID   Brand  Model  Year   Color  Mileage  Price     Location
0       1  Toyota  Camry  2018   White    45000  18000  Los Angeles
1       2   Honda  Civic  2019    Blue    35000  16000     New York
2       3    Ford  Focus  2017  Silver    55000  14000      Chicago
```

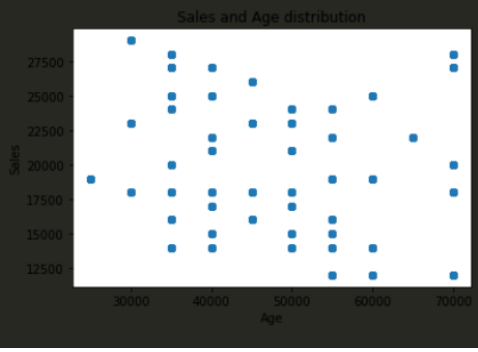
```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("Car Data.csv")
fig=plt.figure()
ax = fig.add_subplot(1,1,1)
ax.hist(df['Mileage'],bins = 10)
plt.title('Car Brand')
plt.xlabel('Mileage')
plt.ylabel('Pata nahi')
plt.show()
```

```
[12]
...
```



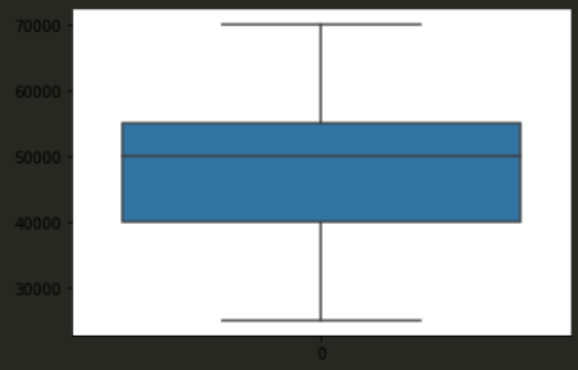
```
fig=plt.figure()

ax = fig.add_subplot(1,1,1)
ax.scatter(df['Mileage'],df['Price'])
plt.title('Mileage and Price distribution')
plt.xlabel('Mileage')
plt.ylabel('Price')
plt.show()
```

[13]




```
import seaborn as sns
sns.boxplot(df['Mileage'])
sns.despine()
```

[14]



```
df=pd.read_csv("Car Data.csv")
print(df)
test= df.groupby(['Brand','Price'])
test.size()
```

```
      Car ID      Brand      Model  Year   Color  Mileage  Price       Location
0          1     Toyota      Camry  2018   White    45000  18000    Los Angeles
1          2      Honda      Civic  2019    Blue    35000  16000       New York
2          3       Ford      Focus  2017  Silver    55000  14000        Chicago
3          4  Chevrolet      Cruze  2016     Red    60000  12000          Miami
4          5    Hyundai    Elantra  2018   Black    40000  15000  San Francisco
...      ...        ...        ...   ...     ...      ...    ...            ...
1995    1996    Hyundai   Palisade  2019  Silver    65000  22000  San Francisco
1996    1997     Toyota     Sienna  2018     Red    55000  16000         Dallas
1997    1998      Honda        Fit  2018    Gray    50000  14000        Atlanta
1998    1999       Ford     Fusion  2017   White    55000  19000        Phoenix
1999    2000  Chevrolet     Malibu  2016    Blue    30000  23000        Houston

[2000 rows x 8 columns]
```

```
Brand      Price
Chevrolet  12000    31
           14000    34
           15000    28
           16000    22
           17000    24
                    ..
Toyota     25000    23
           26000    10
           27000    24
           28000    12
           29000    13
Length: 85, dtype: int64
```

```
test= df.groupby(['Price'])
test.describe()
```

	Car ID								Year					Mileage						
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	max
Price																					
12000	169.0	1003.550296	579.066787	4.0	501.00	998.0	1495.00	1992.0	169.0	2017.562130	...	2019.0	2020.0	169.0	58343.195266	5529.194526	55000.0	55000.0	55000.0	60000.0	70000.0
14000	170.0	1001.158824	582.460343	3.0	501.75	1000.5	1499.25	1998.0	170.0	2017.582353	...	2019.0	2020.0	170.0	50029.411765	9553.602419	35000.0	40000.0	52500.0	60000.0	60000.0
15000	141.0	999.397163	580.030638	5.0	502.00	999.0	1496.00	1993.0	141.0	2017.510638	...	2019.0	2020.0	141.0	48936.170213	4957.264173	40000.0	50000.0	50000.0	50000.0	55000.0
16000	114.0	1000.728070	586.638063	2.0	500.75	999.5	1498.25	1997.0	114.0	2017.359649	...	2019.0	2020.0	114.0	45000.000000	7164.316690	35000.0	37500.0	45000.0	52500.0	55000.0
17000	140.0	1001.500000	575.758536	29.0	518.25	1007.5	1496.75	1986.0	140.0	2017.557143	...	2019.0	2020.0	140.0	44000.000000	4916.570133	40000.0	40000.0	40000.0	50000.0	50000.0
18000	198.0	996.939394	581.168337	1.0	499.50	998.0	1496.50	1995.0	198.0	2017.530303	...	2019.0	2020.0	198.0	44924.242424	11969.475483	30000.0	35000.0	45000.0	50000.0	70000.0
19000	142.0	1003.683099	584.055768	6.0	504.25	1002.5	1500.75	1999.0	142.0	2017.042254	...	2018.0	2020.0	142.0	49859.154930	12784.959886	25000.0	55000.0	55000.0	55000.0	60000.0
20000	84.0	998.500000	577.242312	15.0	504.25	993.5	1482.75	1972.0	84.0	2017.619048	...	2019.0	2020.0	84.0	46666.666667	16598.253162	35000.0	35000.0	35000.0	70000.0	70000.0
21000	112.0	996.500000	576.469747	13.0	504.75	996.5	1488.25	1980.0	112.0	2017.214286	...	2019.0	2020.0	112.0	45000.000000	5022.472023	40000.0	40000.0	45000.0	50000.0	50000.0
22000	141.0	1003.787234	580.020416	8.0	505.00	1002.0	1499.00	1996.0	141.0	2017.354610	...	2019.0	2020.0	141.0	54078.014184	8053.456154	40000.0	55000.0	55000.0	55000.0	65000.0
23000	113.0	1002.407080	581.634959	12.0	509.00	1006.0	1503.00	2000.0	113.0	2017.601770	...	2019.0	2020.0	113.0	43628.318584	8298.229900	30000.0	30000.0	45000.0	50000.0	50000.0
24000	84.0	1001.166667	577.254779	17.0	506.25	995.5	1484.75	1974.0	84.0	2017.630952	...	2019.0	2020.0	84.0	46666.666667	8549.407551	35000.0	35000.0	50000.0	55000.0	55000.0
25000	112.0	993.500000	576.380474	16.0	506.25	996.5	1486.75	1977.0	112.0	2017.758929	...	2019.0	2020.0	112.0	42500.000000	10354.091327	35000.0	35000.0	37500.0	45000.0	60000.0
26000	56.0	1002.500000	579.062157	24.0	513.25	1002.5	1491.75	1981.0	56.0	2017.250000	...	2018.0	2020.0	56.0	45000.000000	0.000000	45000.0	45000.0	45000.0	45000.0	45000.0
27000	112.0	1001.500000	576.438678	21.0	511.50	1002.0	1492.50	1983.0	112.0	2017.232143	...	2019.0	2020.0	112.0	45000.000000	14642.896381	35000.0	35000.0	37500.0	47500.0	70000.0
28000	56.0	998.500000	579.098202	19.0	508.75	998.5	1488.25	1978.0	56.0	2017.178571	...	2019.0	2020.0	56.0	52500.000000	17658.374269	35000.0	35000.0	52500.0	70000.0	70000.0
29000	56.0	1003.500000	579.062157	25.0	514.25	1003.5	1492.75	1982.0	56.0	2017.553571	...	2019.0	2020.0	56.0	30000.000000	0.000000	30000.0	30000.0	30000.0	30000.0	30000.0

17 rows × 24 columns

Experiment 2: One case study on building Data warehouse/Data Mart

Aim: Write Detailed Problem statement and design dimensional modelling (creation of star and snowflake schema)

Theory:

Case study:

The case study revolves around the application and significance of data warehousing and mining techniques within the context of a library management system. In an era where libraries are adapting to digital landscapes, the utilization of data-driven strategies has become imperative. This case study offers an in-depth exploration of the amalgamation of data warehousing and mining to extract valuable insights from the vast reservoir of data encompassing user activities, resource utilization, and operational records within a library system.

The case study delves into the essence of data mining, underscoring its pivotal role in unearthing concealed patterns, trends, and interconnections within the realm of library data.

Star Schema:

A star schema is a type of data modeling technique used in data warehousing to represent data in a structured and intuitive way. In a star schema, data is organized into a central fact table that contains the measures of interest, surrounded by dimension tables that describe the attributes of the measures.

The fact table in a star schema contains the measures or metrics that are of interest to the user or organization. For example, in a sales data warehouse, the fact table might contain sales revenue, units sold, and profit margins. Each record in the fact table represents a specific event or transaction, such as a sale or order.

The dimension tables in a star schema contain the descriptive attributes of the measures in the fact table. These attributes are used to slice and dice the data in the fact table, allowing users to analyze the data from different perspectives. For example, in a sales data warehouse, the dimension tables might include product, customer, time, and location.

In a star schema, each dimension table is joined to the fact table through a foreign key relationship. This allows users to query the data in the fact table using attributes from the dimension tables.

Some features of star schema:

- Central Fact Table
- Dimension Tables
- Denormalized structure
- Simple queries
- Aggregated data
- Fast performance

- Easy to understand

Advantages of Star Schema :

- Simpler Queries –

  Join logic of star schema is quite cinch in comparison to other join logic which are needed to fetch data from a transactional schema that is highly normalized.

- Simplified Business Reporting Logic –

  In comparison to a transactional schema that is highly normalized, the star schema makes simpler common business reporting logic, such as of reporting and period-over-period.

- Feeding Cubes –

  Star schema is widely used by all OLAP systems to design OLAP cubes efficiently. In fact, major OLAP systems deliver a ROLAP mode of operation which can use a star schema as a source without designing a cube structure.

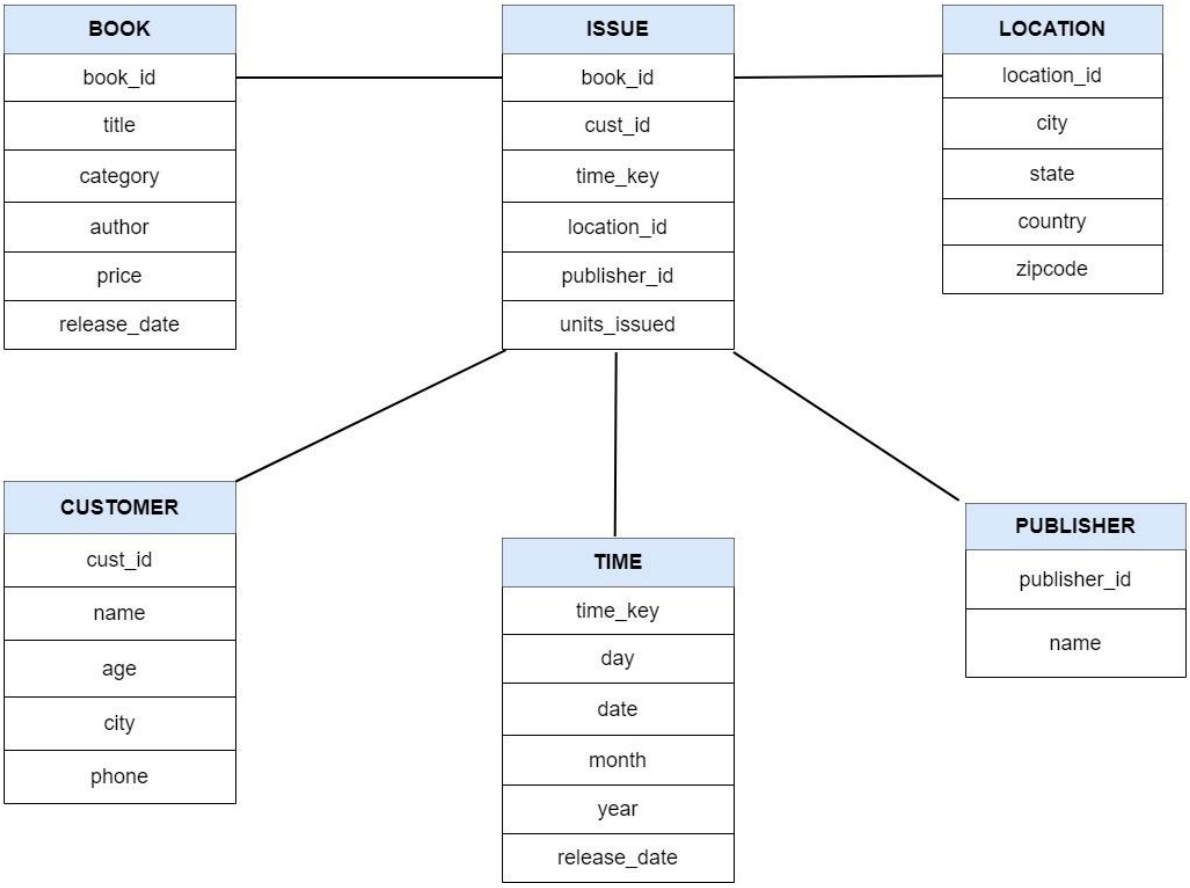


Fig 2.1 Star Schema of Library Management

Snowflake Schema:

The snowflake schema is a variant of the star schema. Here, the centralized fact table is connected to multiple dimensions. In the snowflake schema, dimensions are present in a normalized form in multiple related tables. The snowflake structure materialized when the dimensions of a star schema are detailed and highly structured, having several levels of

relationship, and the child tables have multiple parent tables. The snowflake effect affects only the dimension tables and does not affect the fact tables.

A snowflake schema is a type of data modeling technique used in data warehousing to represent data in a structured way that is optimized for querying large amounts of data efficiently. In a snowflake schema, the dimension tables are normalized into multiple related tables, creating a hierarchical or "snowflake" structure.

In a snowflake schema, the fact table is still located at the center of the schema, surrounded by the dimension tables. However, each dimension table is further broken down into multiple related tables, creating a hierarchical structure that resembles a snowflake.

For Example, in a sales data warehouse, the product dimension table might be normalized into multiple related tables, such as product category, product subcategory, and product details. Each of these tables would be related to the product dimension table through a foreign key relationship.

The snowflake design is the result of further expansion and normalization of the dimension table. In other words, a dimension table is said to be snowflaked if the low-cardinality attribute of the dimensions has been divided into separate normalized tables.

Features of Snowflake Schema:

- Normalization
- Hierarchical
- Multiple levels
- Joins
- Scalability

Advantages of Snowflake Schema

- It provides structured data which reduces the problem of data integrity.
- It uses small disk space because data are highly structured.

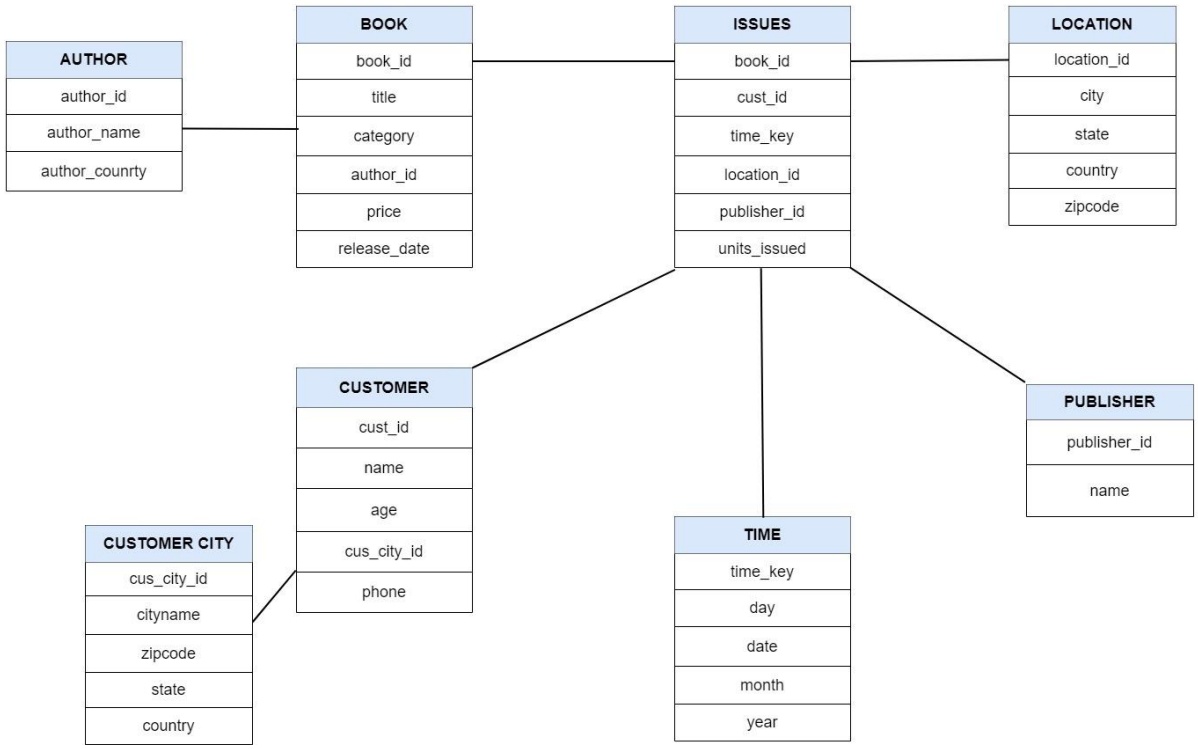


Fig 2.2 Snowflake schema of Library Management

Conclusion:

In conclusion, star and snowflake schema both provide an aggregated view of the data sets and both of them have their own differences, advantages and disadvantages. While star schema provides a more aggregated straight forward perspective of the data, while snowflake provides a normalised structure which helps in minimising redundancy. he advantage here is that such tables (normalized) are easy to maintain and save storage space. However, it also means that more joins will be needed to execute the query. This will adversely impact system performance.

However, the snowflake schema can also be more complex to query than a star schema because it requires more table joins.

The decision to use a snowflake schema versus a star schema in a data warehousing project will depend on the specific requirements of the project and the trade-offs between query performance, schema complexity, and data integrity.

Experiment 3: One case study on building Data warehouse/Data Mart

Aim: Implementation of all dimension table and fact table based on experiment 1 case study

Theory:

Case study:

All Libraries may create a data warehouse that keeps the record of the magazines, books issued by them to the library members with respect to dimensions that include time, publisher, book details, author details, member details, etc. These dimensions help keep track of measures around the month or the year and find out values like 'books issued this month' or 'books issued in the last two years' and other details about those measures.

The case study delves into the essence of data mining, underscoring its pivotal role in unearthing concealed patterns, trends, and interconnections within these measures and dimensions

Dimensional Modeling:

Dimensional Data Modeling is one of the data modeling techniques used in data warehouse design. The concept of Dimensional Modeling was developed by Ralph Kimball which is comprised of facts and dimension tables. Since the main goal of this modeling is to improve the data retrieval so it is optimized for SELECT OPERATION. The advantage of using this model is that we can store data in such a way that it is easier to store and retrieve the data once stored in a data warehouse. The dimensional model is the data model used by many OLAP systems.

Elements of Dimensional Data Model

Facts:

Facts are the measurable data elements that represent the business metrics of interest. For example, in a sales data warehouse, the facts might include sales revenue, units sold, and profit margins. Each fact is associated with one or more dimensions, creating a relationship between the fact and the descriptive data.

Dimension:

Dimensions are the descriptive data elements that are used to categorize or classify the data. For example, in a sales data warehouse, the dimensions might include product, customer, time, and location. Each dimension is made up of a set of attributes that describe the dimension. For example, the product dimension might include attributes such as product name, product category, and product price.

Attributes:

Characteristics of dimension in data modeling are known as characteristics. These are used to filter, search facts, etc. For a dimension of location, attributes can be State, Country, Zipcode, etc.

Fact Table:

In a dimensional data model, the fact table is the central table that contains the measures or metrics of interest, surrounded by the dimension tables that describe the attributes of the measures. The dimension tables are related to the fact table through foreign key relationships

Dimension Table:

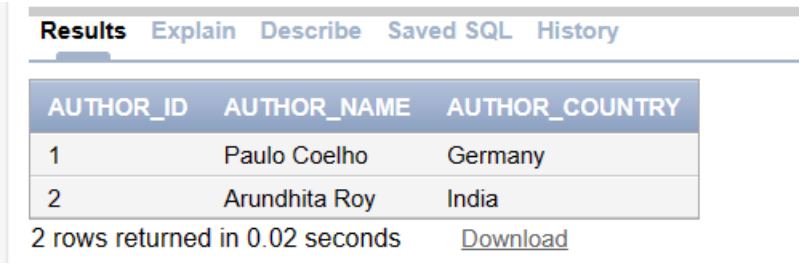
Dimensions of a fact are mentioned by the dimension table and they are basically joined by a foreign key. Dimension tables are simply de-normalized tables. The dimensions can be having one or more relationships.

Table creation outputs:

create table author(author_id int primary key, author_name varchar(50), author_country varchar(30))

insert into author values(1,'Paulo Coelho','Germany')

insert into author values(2,'Arundhita Roy','India')



create table book(book_id int primary key, title varchar(100), category varchar(50), author_id int, price varchar(30), release_date date)

ALTER TABLE book

ADD CONSTRAINT fk_authors

FOREIGN KEY (author_id)

REFERENCES author(author_id)

INSERT INTO BOOK (BOOK_ID, TITLE, CATEGORY, AUTHOR_ID, PRICE, RELEASE_DATE)

VALUES (1, 'Eleven', 'fiction', 1, 200, TO_DATE('2020-08-22', 'YYYY-MM-DD'));

INSERT INTO BOOK (BOOK_ID, TITLE, CATEGORY, AUTHOR_ID, PRICE, RELEASE_DATE)

VALUES (2, 'Cosmic Odyssey', 'science fiction', 2, 150, TO_DATE('2021-05-10', 'YYYY-MM-DD'));

INSERT INTO BOOK (BOOK_ID, TITLE, CATEGORY, AUTHOR_ID, PRICE, RELEASE_DATE)

VALUES (3, 'The Enigmatic Puzzle', 'mystery', 1, 180, TO_DATE('2019-11-15', 'YYYY-MM-DD'));

INSERT INTO BOOK (BOOK_ID, TITLE, CATEGORY, AUTHOR_ID, PRICE, RELEASE_DATE)

VALUES (4, 'Love in Bloom', 'romance', 2, 220, TO_DATE('2022-02-14', 'YYYY-MM-DD'));

INSERT INTO BOOK (BOOK_ID, TITLE, CATEGORY, AUTHOR_ID, PRICE, RELEASE_DATE)

VALUES (5, 'Realm of Dreams', 'fantasy', 1, 250, TO_DATE('2023-07-01', 'YYYY-MM-DD'));

BOOK_ID	TITLE	CATEGORY	AUTHOR_ID	PRICE	RELEASE_DATE
1	Eleven	fiction	1	200	08/22/2020
2	Cosmic Odyssey	science fiction	2	150	05/10/2021
3	The Enigmatic Puzzle	mystery	1	180	11/15/2019
4	Love in Bloom	romance	2	220	02/14/2022
5	Realm of Dreams	fantasy	1	250	07/01/2023

create table location(location_id int primary key, city varchar(30), state varchar(30), country varchar(30), zipcode varchar(10));

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (1, 'Mumbai', 'Maharashtra', 'India', '400001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (2, 'Delhi', 'Delhi', 'India', '110001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (3, 'Bangalore', 'Karnataka', 'India', '560001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (4, 'Chennai', 'Tamil Nadu', 'India', '600001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (5, 'Kolkata', 'West Bengal', 'India', '700001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (6, 'Hyderabad', 'Telangana', 'India', '500001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (7, 'Pune', 'Maharashtra', 'India', '411001');

INSERT INTO location (location_id, city, state, country, zipcode)

VALUES (8, 'Ahmedabad', 'Gujarat', 'India', '380001');

LOCATION_ID	CITY	STATE	COUNTRY	ZIPCODE
8	Ahmedabad	Gujarat	India	380001
1	Mumbai	Maharashtra	India	400001
2	Delhi	Delhi	India	110001
3	Bangalore	Karnataka	India	560001
4	Chennai	Tamil Nadu	India	600001
5	Kolkata	West Bengal	India	700001
6	Hyderabad	Telangana	India	500001
7	Pune	Maharashtra	India	411001

8 rows returned in 0.00 seconds Download

create table publisher(publisher_id int primary key, name varchar(100));

insert into publisher values(1,'Hiten publications')

insert into publisher values(2,'Sawan publications')

insert into publisher values(3,'vrajesh publications')

insert into publisher values(4,'Samarth publications')

insert into publisher values(5,'Raghyav publications')

PUBLISHER_ID	NAME
5	Raghyav publications
1	Hiten publications
3	vrajesh publications
4	Samarth publications

create table time(time_key int primary key, date_ date, day varchar(15), month varchar(20), year varchar(10))

INSERT INTO time (time_key, date_, day, month, year)

VALUES (1, TO_DATE('2023-08-01', 'YYYY-MM-DD'), 'Monday', 'August', '2023');

INSERT INTO time (time_key, date_, day, month, year)

VALUES (2, TO_DATE('2023-08-02', 'YYYY-MM-DD'), 'Tuesday', 'August', '2023');

INSERT INTO time (time_key, date_, day, month, year)

VALUES (3, TO_DATE('2023-08-03', 'YYYY-MM-DD'), 'Wednesday', 'August', '2023');

INSERT INTO time (time_key, date_, day, month, year)

VALUES (4, TO_DATE('2023-08-04', 'YYYY-MM-DD'), 'Thursday', 'August', '2023');

INSERT INTO time (time_key, date_, day, month, year)

VALUES (5, TO_DATE('2023-08-05', 'YYYY-MM-DD'), 'Friday', 'August', '2023');

TIME_KEY	DATE_	DAY	MONTH	YEAR
1	08/01/2023	Monday	August	2023
2	08/02/2023	Tuesday	August	2023
3	08/03/2023	Wednesday	August	2023
4	08/04/2023	Thursday	August	2023
5	08/05/2023	Friday	August	2023

create table customer_city(cus_city_id int primary key, cityname varchar(30), zipcode varchar(10), state varchar(30), country varchar(30))

insert into customer_city values(1,'ulhasnagar','421002','Maharashtra','India')

insert into customer_city values(2,'ambernath','421506','Maharashtra','India')

insert into customer_city values(3,'badlapur','421509','Maharashtra','India')

CUS_CITY_ID	CITYNAME	ZIPCODE	STATE	COUNTRY
1	ulhasnagar	421002	Maharashtra	India
2	ambernath	421506	Maharashtra	India
3	badlapur	421509	Maharashtra	India

3 rows returned in 0.06 seconds Download

create table customer(customer_id int primary key, name varchar(50), age int, cus_city_id int, phone varchar(20))

ALTER TABLE customer

ADD CONSTRAINT fk_custcity

FOREIGN KEY (cus_city_id)

REFERENCES customer_city(cus_city_id);

insert into customer values(1,'Hiten',19,1,'9898989898')

insert into customer values(2,'Sawan',19,2,'9898989898')

insert into customer values(3,'VRajesh',19,3,'9898989898')

insert into customer values(4,'Raghav',19,3,'9898989898')

insert into customer values(5,'Hiten',19,1,'9898989898')

insert into customer values(6,'Samrht',19,1,'9898989898')

insert into customer values(7,'Akash',19,1,'9898989898')

CUSTOMER_ID	NAME	AGE	CUS_CITY_ID	PHONE
1	Hiten	19	1	9898989898
2	Sawan	19	2	9898989898
3	VRajesh	19	3	9898989898
4	Raghav	19	3	9898989898
5	Hiten	19	1	9898989898
6	Samrht	19	1	9898989898
7	Akash	19	1	9898989898

create table issues(issue_id int primary key,book_id int, cust_id int, time_key int, location_id int, publisher_id int, units_issued int)

ALTER TABLE issues

ADD CONSTRAINT fk_book

FOREIGN KEY (book_id)

REFERENCES book(book_id);

ALTER TABLE issues

ADD CONSTRAINT fk_cust

FOREIGN KEY (cust_id)

REFERENCES customer(customer_id);

ALTER TABLE issues

ADD CONSTRAINT fk_tie

FOREIGN KEY (time_key)

REFERENCES time(time_key);

ALTER TABLE issues

ADD CONSTRAINT fk_location

FOREIGN KEY (location_id)

REFERENCES location(location_id);

ALTER TABLE issues

ADD CONSTRAINT fk_publisher

FOREIGN KEY (publisher_id)

REFERENCES publisher(publisher_id);

insert into issues values(1,1,1,1,1,1,100)

insert into issues values(3,3,3,3,3,3,230)

insert into issues values(4,4,4,4,4,4,220)

insert into issues values(5,4,2,4,4,3,220)

insert into issues values(6,4,2,4,4,3,220)

insert into issues values(7,1,2,4,4,3,120)

insert into issues values(8,1,2,3,4,3,120)

ISSUE_ID	BOOK_ID	CUST_ID	TIME_KEY	LOCATION_ID	PUBLISHER_ID	UNITS_ISSUED
1	1	1	1	1	1	100
3	3	3	3	3	3	230
4	4	4	4	4	4	230
5	4	4	4	4	3	220
6	4	2	4	4	3	220
7	1	2	4	4	3	120
8	1	2	3	4	3	120
2	2	2	2	1	1	230

Experiment 4

Aim: Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot based on experiment 1

Theory:

Case study:

All Libraries may create a data warehouse that keeps the record of the magazines, books issued by them to the library members with respect to dimensions that include time, publisher, book details, author details, member details, etc. These dimensions help keep track of measures around the month or the year and find out values like 'books issued this month' or 'books issued in the last two years' and other details about those measures.

The case study delves into the essence of data mining, underscoring its pivotal role in unearthing concealed patterns, trends, and interconnections within these measures and dimensions

OLAP:

OLAP stands for On-Line Analytical Processing. OLAP is a classification of software technology which authorizes analysts, managers, and executives to gain insight into information through fast, consistent, interactive access in a wide variety of possible views of data that has been transformed from raw information to reflect the real dimensionality of the enterprise as understood by the clients.

OLAP implement the multidimensional analysis of business information and support the capability for complex estimations, trend analysis, and sophisticated data modeling. It is rapidly enhancing the essential foundation for Intelligent Solutions containing Business Performance Management, Planning, Budgeting, Forecasting, Financial Documenting, Analysis, Simulation-Models, Knowledge Discovery, and Data Warehouses Reporting. OLAP enables end-clients to perform ad hoc analysis of record in multiple dimensions, providing the insight and understanding they require for better decision making.

OLAP Operations:

OLAP operations:

There are five basic analytical operations that can be performed on an OLAP cube:

1. Slice: It selects a single dimension from the OLAP cube which results in a new sub-cube creation. In the cube given in the overview section, Slice is performed on the dimension Time = “Q1”.

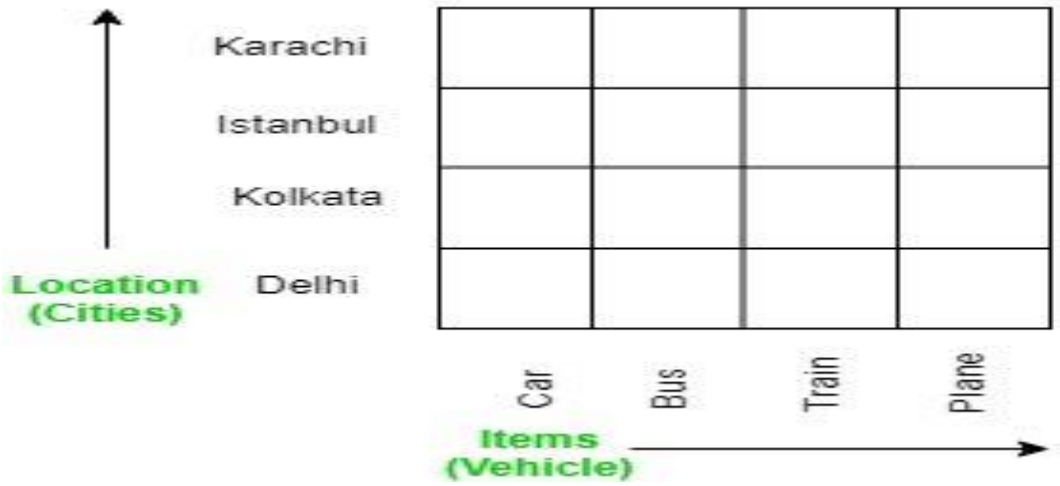


Fig 4.1 Slice

2. Dice: It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting following dimensions with criteria:

Location = “Delhi” or “Kolkata”

Time = “Q1” or “Q2”

Item = “Car” or “Bus”

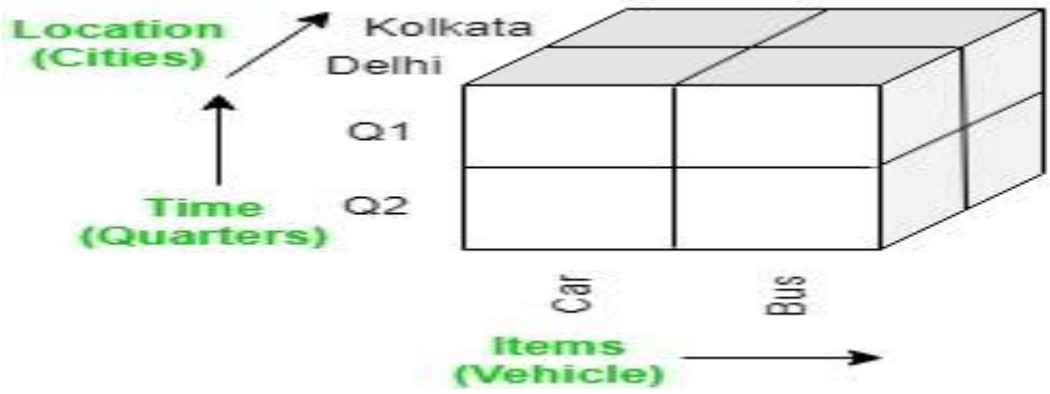


Fig 4.2 Dice

3. Roll up: It is just opposite of the drill-down operation. It performs aggregation on the OLAP cube. It can be done by:

Climbing up in the concept hierarchy

Reducing the dimensions

In the cube given in the overview section, the roll-up operation is performed by climbing up in the concept hierarchy of Location dimension (City -> Country).

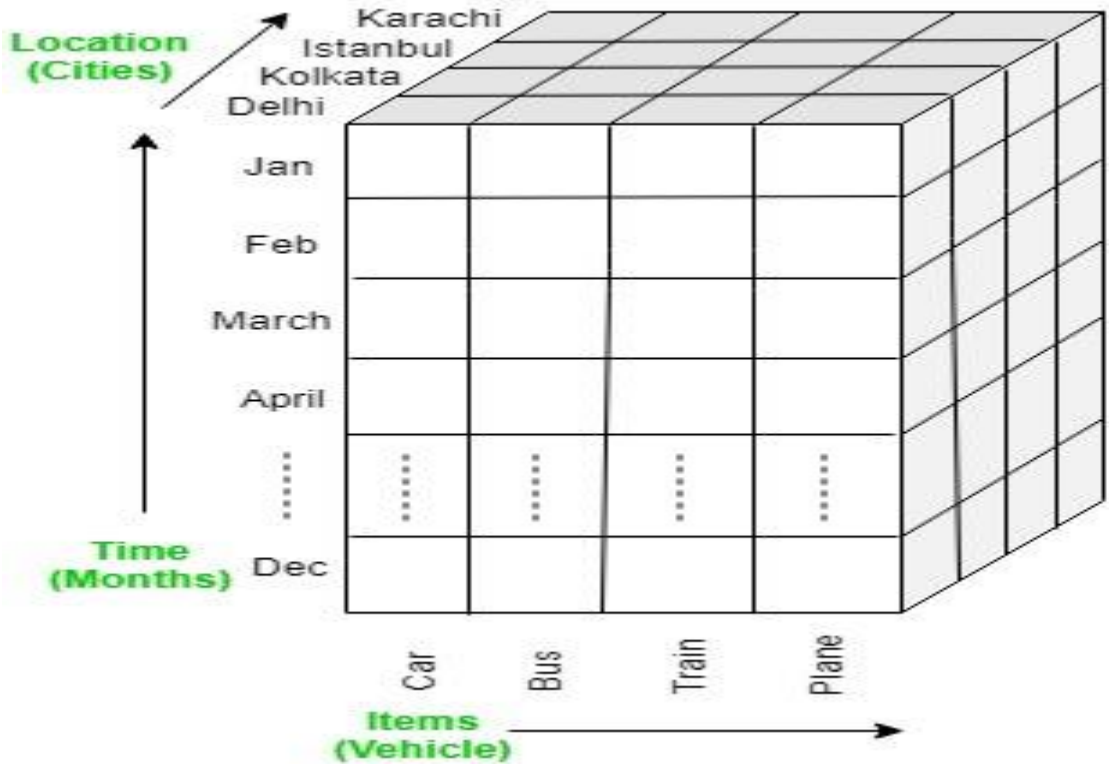


Fig 4.3 Roll up

4. Drill down: In drill-down operation, the less detailed data is converted into highly detailed data. It can be done by:

Moving down in the concept hierarchy

Adding a new dimension

In the cube given in overview section, the drill down operation is performed by moving down in the concept hierarchy of Time dimension (Quarter -> Month).

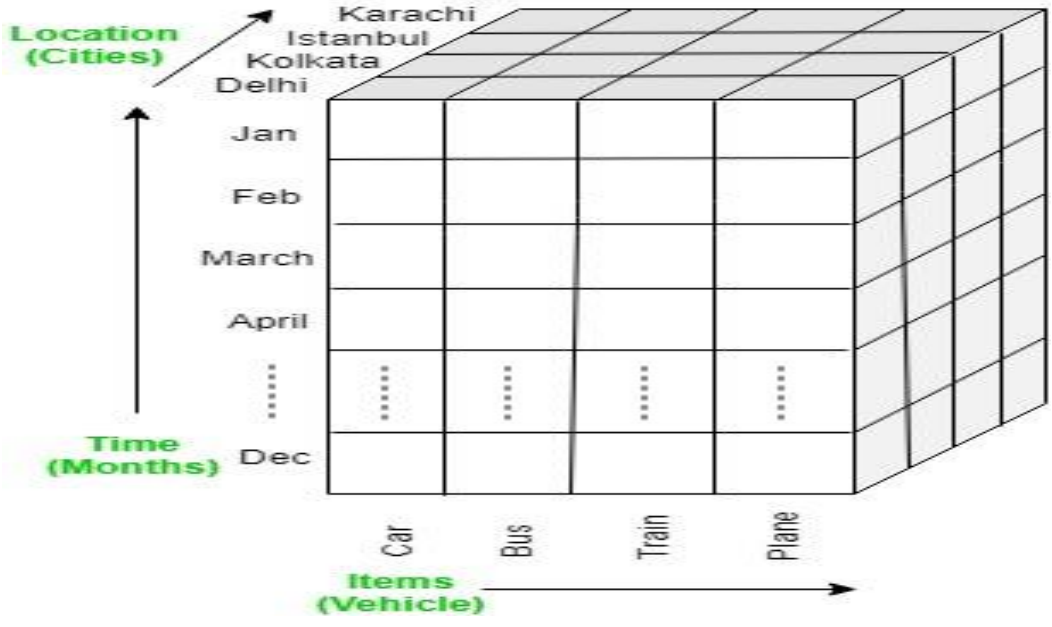


Fig 4.4 Drill-Down

5. Pivot: It is also known as rotation operation as it rotates the current view to get a new view of the representation. In the sub-cube obtained after the slice operation, performing pivot operation gives a new view of it.

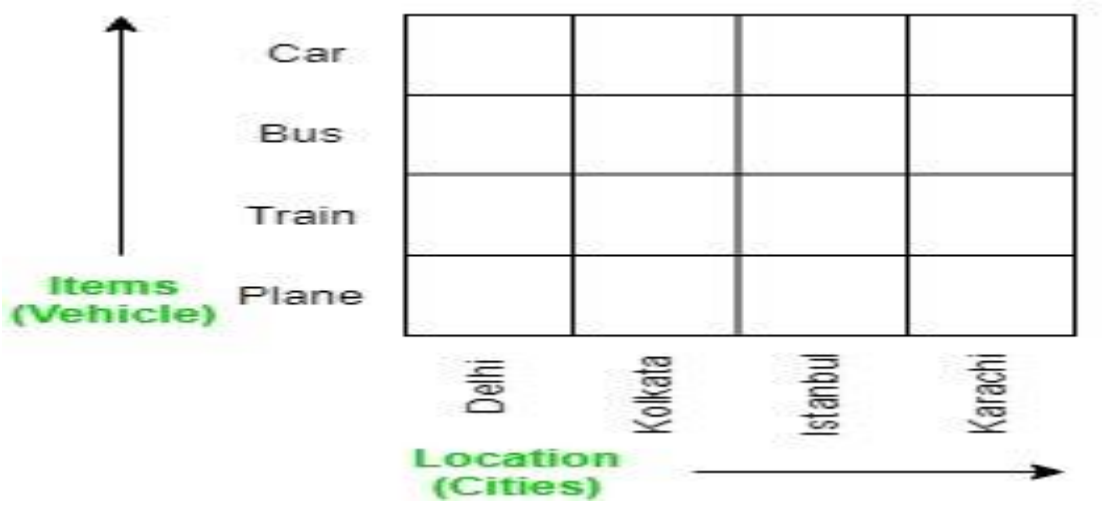


Fig 4.5 Pivot

Output:

Materialized View:

CREATE MATERIALIZED VIEW mv_issues_summary

REFRESH FAST

AS

SELECT location_id, time_key, SUM(units_issued) AS total_units_issued FROM issues

GROUP BY location_id, time_key;

LOCATION_ID	TIME_KEY	TOTAL_UNITS_ISSUED
4	11	10
1	6	10
1	9	5
3	11	7
5	8	20
4	4	790
1	2	230
1	1	100
3	3	230
4	7	12
More than 10 rows available. Increase rows selector to view more rows.		

Slice:

SELCT i.location_id,l.city,sum(i.units_issued) FROM issues i INNER JOIN location l on i.location_id = l.location_id GROUP BY i.location_id,l.city ORDER BY i.location_id

Results Explain Describe Saved SQL History

LOCATION_ID	CITY	SUM(I.UNITS_ISSUED)
1	Mumbai	345
2	Delhi	8
3	Bangalore	237
4	Chennai	932
5	Kolkata	20

5 rows returned in 0.01 seconds Download

Dice:

SELECT location_id, time_key, SUM(units_issued) AS total_units_issued

FROM issues

WHERE location_id = 1 AND time_key<=10 GROUP BY location_id, time_key;

LOCATION_ID	TIME_KEY	TOTAL_UNITS_ISSUED
1	6	10
1	9	5
1	2	230
1	1	100

4 rows returned in 0.00 seconds Download

Rollup:

SELECT year, SUM(units_issued) AS total_units_issued FROM issues JOIN time ON issues.time_key = time.time_key GROUP BY year;

YEAR	TOTAL_UNITS_ISSUED
2023	1512
2021	13
2022	17

3 rows returned in 0.00 seconds Download

Drilldown:

SELECT date_, SUM(units_issued) AS total_units_issued FROM issues JOIN time ON issues.time_key = time.time_key GROUP BY date_;

DATE_	TOTAL_UNITS_ISSUED
08/01/2023	100
08/04/2023	790
01/01/2021	13
08/03/2023	350
08/02/2023	230
10/01/2023	12
11/01/2023	20
09/01/2023	10
01/01/2022	17

9 rows returned in 0.01 seconds Download

Pivot:

```
SELECT location_id,
    SUM(CASE WHEN time_key = 1 THEN units_issued ELSE 0 END) AS day1_units,
    SUM(CASE WHEN time_key = 2 THEN units_issued ELSE 0 END) AS day2_units,
    SUM(CASE WHEN time_key = 3 THEN units_issued ELSE 0 END) AS day3_units,
    SUM(CASE WHEN time_key = 4 THEN units_issued ELSE 0 END) AS day4_units,
    SUM(CASE WHEN time_key = 5 THEN units_issued ELSE 0 END) AS day5_units
FROM issues
GROUP BY location_id;
```

LOCATION_ID	DAY1_UNITS	DAY2_UNITS	DAY3_UNITS	DAY4_UNITS	DAY5_UNITS
1	100	230	0	0	0
2	0	0	0	0	0
4	0	0	120	790	0
5	0	0	0	0	0
3	0	0	230	0	0

5 rows returned in 0.00 seconds Download

Experiment 5

Aim: Implementation of Bayesian algorithm

Theory:

Naïve Bayes Classifier Algorithm

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

It is mainly used in text classification that includes a high-dimensional training dataset.

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

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.

Bayes' Theorem:

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

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

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Algorithm:

- Import the CSV file of the dataset.
- For each dependent feature of the dataset, create a frequency table and
- count the occurrence of each value, concerning the positive and negative
- outcome of the class variable. For instance, count how many candidates have
- been selected who have STRONG/WEAK/AVG knowledge of DSA.
- Post the counting, we find the probability for all the values of the features.
- Then we take an arbitrary case input from the user for which prediction is to
- be made.
- We calculate the 2 probabilities using dependent features and target variable
- such that, in one case the target variable has a positive outcome and, in
- another case, it has a negative outcome.
- To make the prediction we divide each probability in Step 5 by the sum of
- both probabilities to perform normalization and convert the outcome into a
- valid probability value.
- If the value of probability is higher for the positive outcome of the target
- variable then the user input case is likely to occur, else the outcome is
- negative.

Flowchart:

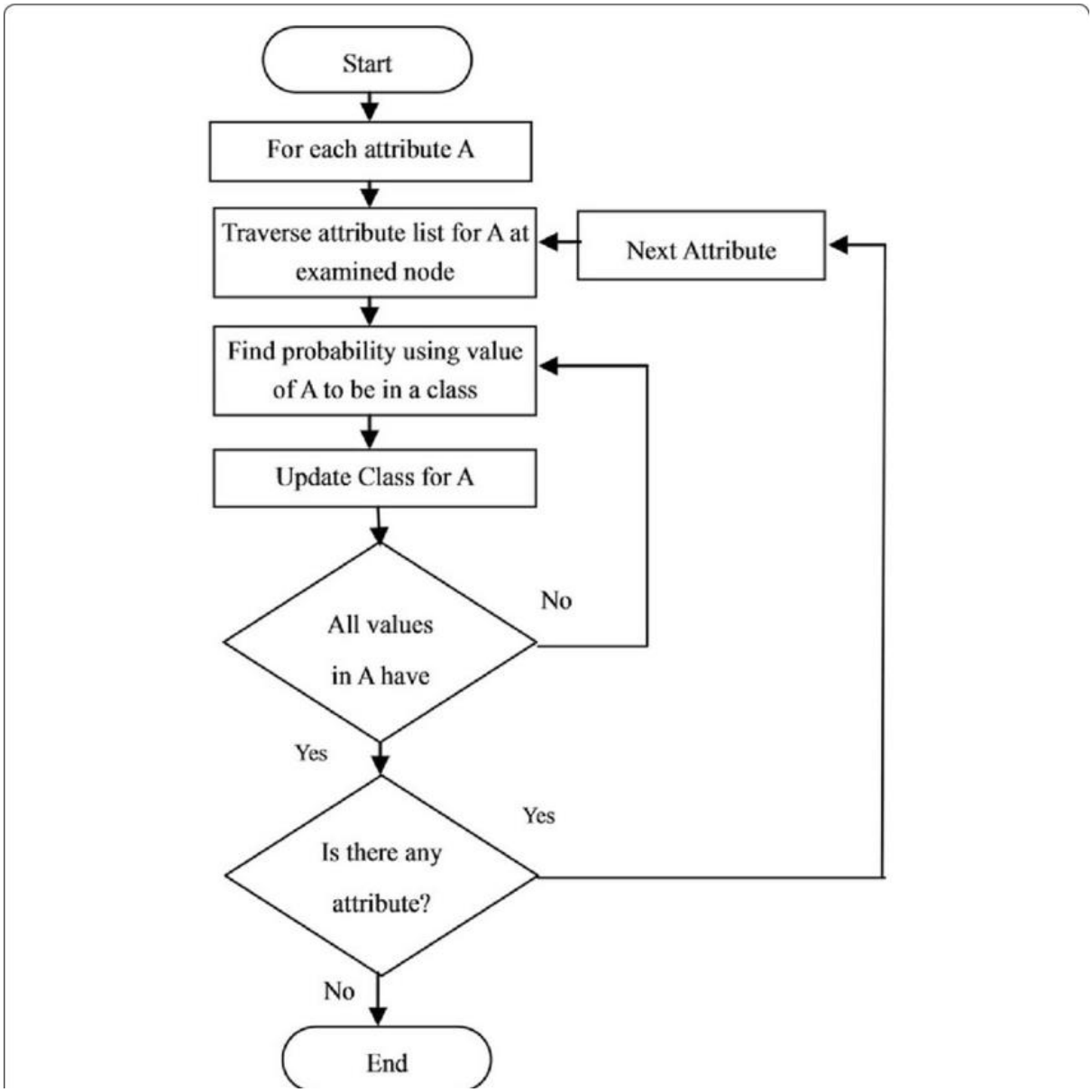


Fig 3.1 Flowchart of Naïve Bayes

Dataset used:

id	age	income	student	credit_rating	Buy_Computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no

7	middle_age	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Code:

```
import csv


def predict(test, PYES, PNO):
    pinyes = AGE_Y[test[0]] * INCOME_Y[test[1]] * STUDENT_Y[test[2]] * CREDIT_RATING_Y[test[3]] *
PYES
    pinno = AGE_N[test[0]] * INCOME_N[test[1]] * STUDENT_N[test[2]] * CREDIT_RATING_N[test[3]] *
PNO
    sumprob = pinyes + pinno
    pinyes = pinyes / sumprob
    pinno = pinno / sumprob
    print("Probability of Buying a Computer (Yes):", pinyes)
    print("Probability of Not Buying a Computer (No):", pinno)
    if pinyes > pinno:
        print("Prediction: Buy a Computer")
    else:
        print("Prediction: Do Not Buy a Computer")


def testInput():
    string = input("Enter comma-separated test tuple (age,income,student,credit_rating):")
    test = string.split(',')
    return test


def probabilityComputation(dictionary):
```

```
    value = sum(dictionary.values())
    for key in dictionary:
        dictionary[key] /= value


def preprocessing():
    PYES = PNO = 0
    with open('./Buy_Computer.csv', mode='r') as csv_file:
        csv_reader = csv.DictReader(csv_file)
        nyes = nno = 0
        for row in csv_reader:
            if row['Buy_Computer'] == 'yes':
                AGE_Y[row['age']] += 1
                INCOME_Y[row['income']] += 1
                STUDENT_Y[row['student']] += 1
                CREDIT_RATING_Y[row['credit_rating']] += 1
                nyes += 1
            else:
                AGE_N[row['age']] += 1
                INCOME_N[row['income']] += 1
                STUDENT_N[row['student']] += 1
                CREDIT_RATING_N[row['credit_rating']] += 1
                nno += 1
        PYES = nyes / (nyes + nno)
        PNO = nno / (nyes + nno)
        probabilityComputation(AGE_Y)
        probabilityComputation(AGE_N)
        probabilityComputation(INCOME_Y)
        probabilityComputation(INCOME_N)
        probabilityComputation(STUDENT_Y)
        probabilityComputation(STUDENT_N)
        probabilityComputation(CREDIT_RATING_Y)
        probabilityComputation(CREDIT_RATING_N)
        return PYES, PNO
```

```
AGE_Y = {
    'youth': 0,
    'middle_age': 0,
    'senior': 0
}

AGE_N = {
    'youth': 0,
    'middle_age': 0,
    'senior': 0
}

INCOME_Y = {
    'low': 0,
    'medium': 0,
    'high': 0
}

INCOME_N = {
    'low': 0,
    'medium': 0,
    'high': 0
}

STUDENT_Y = {
    'no': 0,
    'yes': 0
}

STUDENT_N = {
    'no': 0,
    'yes': 0
}
```

```
CREDIT_RATING_Y = {
    'fair': 0,
    'excellent': 0
}

CREDIT_RATING_N = {
    'fair': 0,
    'excellent': 0
}

PYES, PNO = preprocessing()
test = testInput()
predict(test, PYES, PNO)
```

Output:

```
Enter comma-separated test tuple (age,income,student,credit_rating):youth,low,yes,fair
Probability of Buying a Computer (Yes): 0.8605851979345954
Probability of Not Buying a Computer (No): 0.1394148020654045
Prediction: Buy a Computer
```

```
Enter comma-separated test tuple (age,income,student,credit_rating):middle_age,high,no,excellent
Probability of Buying a Computer (Yes): 1.0
Probability of Not Buying a Computer (No): 0.0
Prediction: Buy a Computer
```

Experiment 6

Aim: Perform data Pre-processing task and Demonstrate performing Classification, Clustering, Association algorithm on datasets using Weka Tool

Theory:

Weka is a collection of machine learning algorithms and data preprocessing tools that is widely used for data mining and predictive modeling. It is an open-source software tool developed by the University of Waikato in New Zealand. Weka stands for "Waikato Environment for Knowledge Analysis."

Key features of Weka include:

- Diverse Machine Learning Algorithms: Weka provides a comprehensive set of machine learning algorithms for tasks such as classification, regression, clustering, association rule mining, and more. It includes both traditional statistical algorithms and newer techniques from the field of artificial intelligence.
- User-Friendly Interface: Weka offers a graphical user interface (GUI) that makes it accessible to users with varying levels of technical expertise. This GUI allows users to build, evaluate, and visualize machine learning models without the need for extensive programming knowledge.
- Data Preprocessing: Weka includes a wide range of data preprocessing tools for cleaning, transforming, and preparing data for analysis. These tools help users handle missing values, normalize data, and perform feature selection, among other tasks.
- Experimentation and Evaluation: Weka provides tools for conducting experiments and evaluating the performance of machine learning models. Users can easily compare different algorithms and techniques to determine which ones are most suitable for their specific datasets.
- Integration and Extensibility: Weka can be integrated into other software applications and workflows through its Java API. It also supports the development of custom machine learning algorithms and extensions.
- Education and Research: Weka is often used in educational settings to teach machine learning concepts and techniques. It is also a popular choice for research projects due to its flexibility and extensibility.
- Open Source: Weka is distributed under an open-source license, which means that it is freely available for anyone to use, modify, and distribute.
- Weka has been widely adopted in both academia and industry for tasks such as data analysis, predictive modeling, and knowledge discovery from data. Its user-friendly interface, extensive library of algorithms, and active community of users and developers make it a valuable tool for a wide range of data mining and machine learning applications.

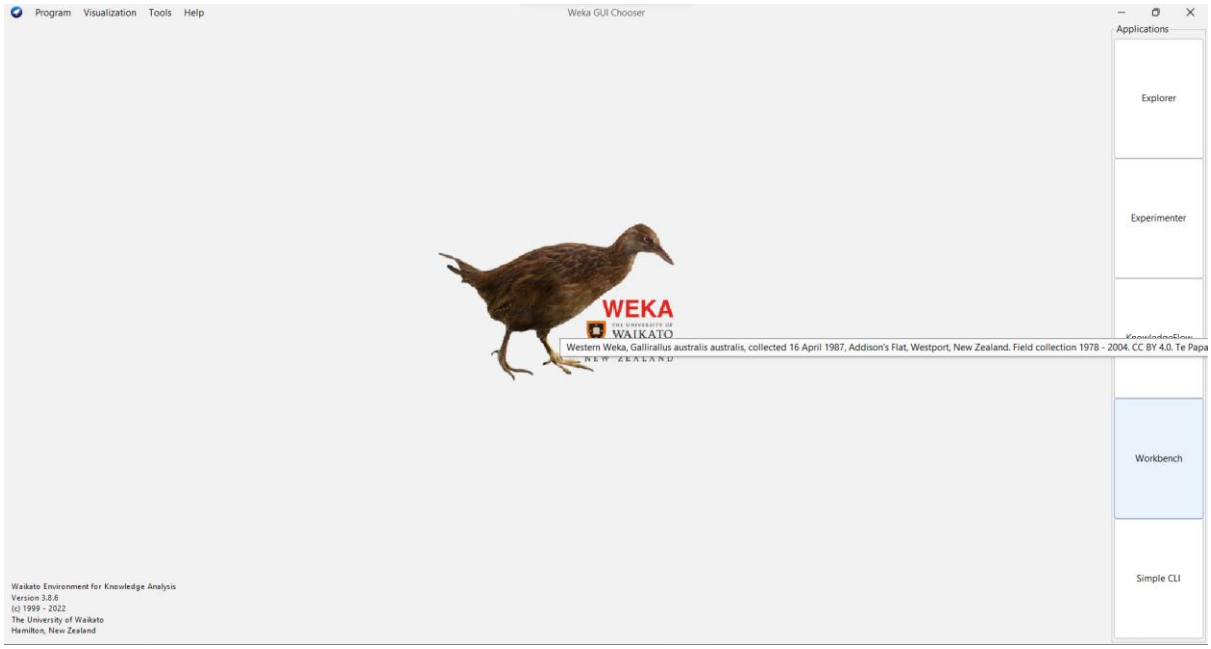


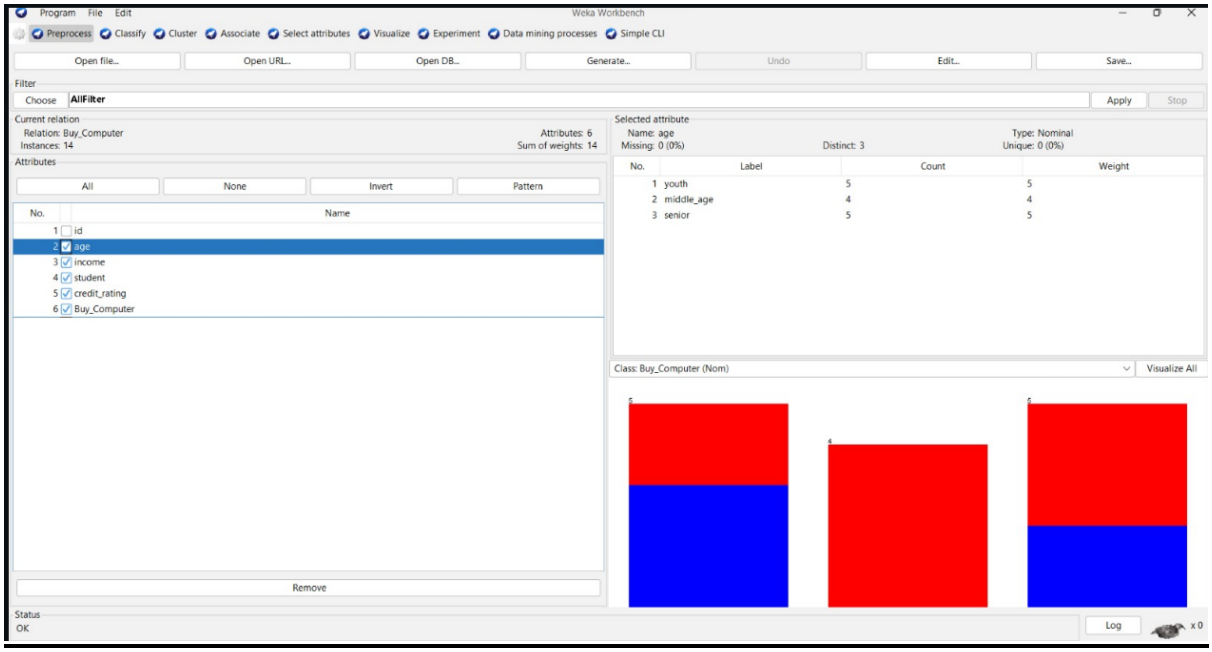
Fig 6.1 Weka Launching Page

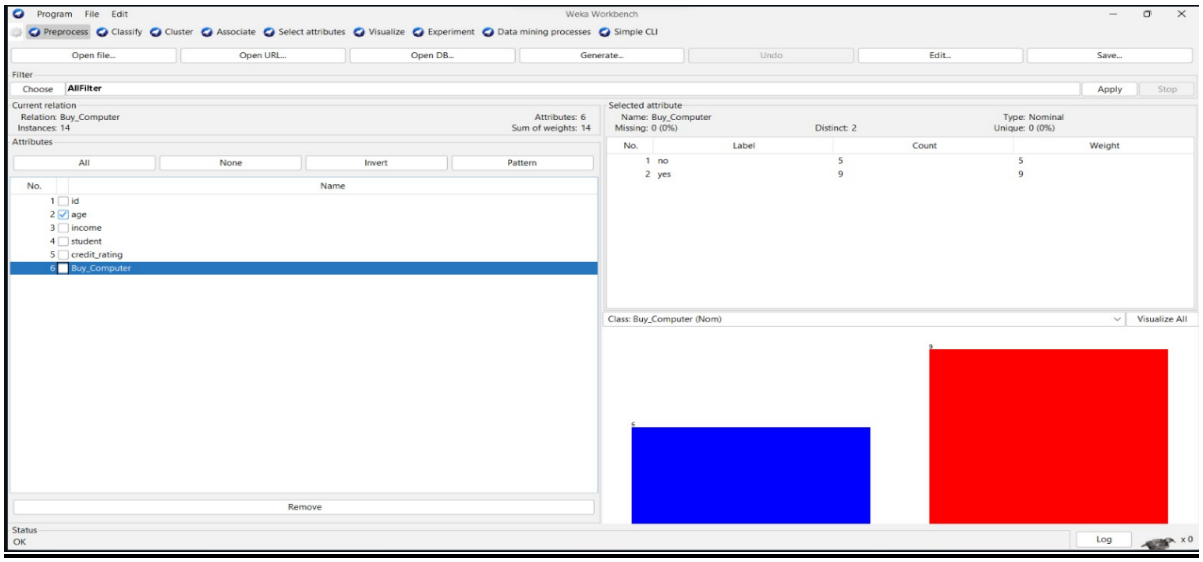
Dataset:

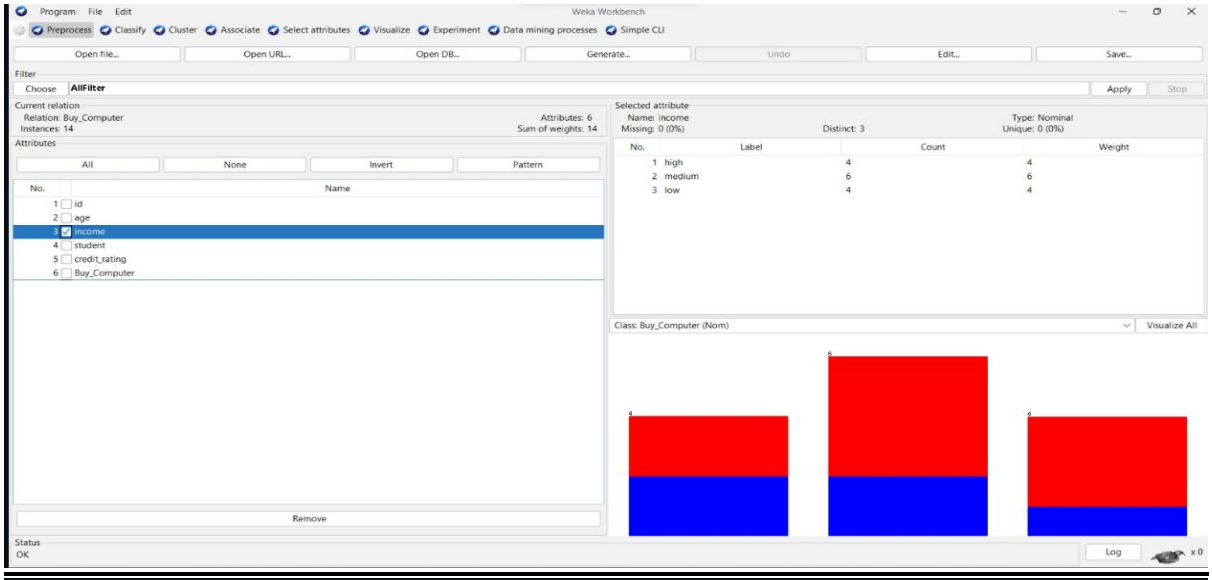
id	age	income	student	credit_rating	Buy_Computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

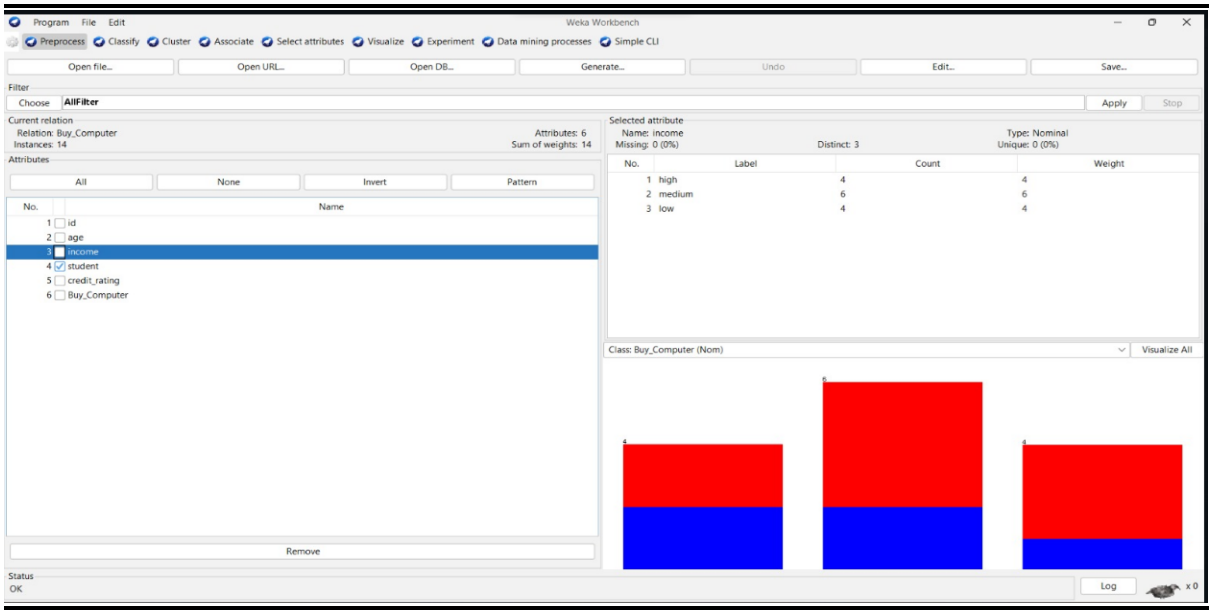
Output:

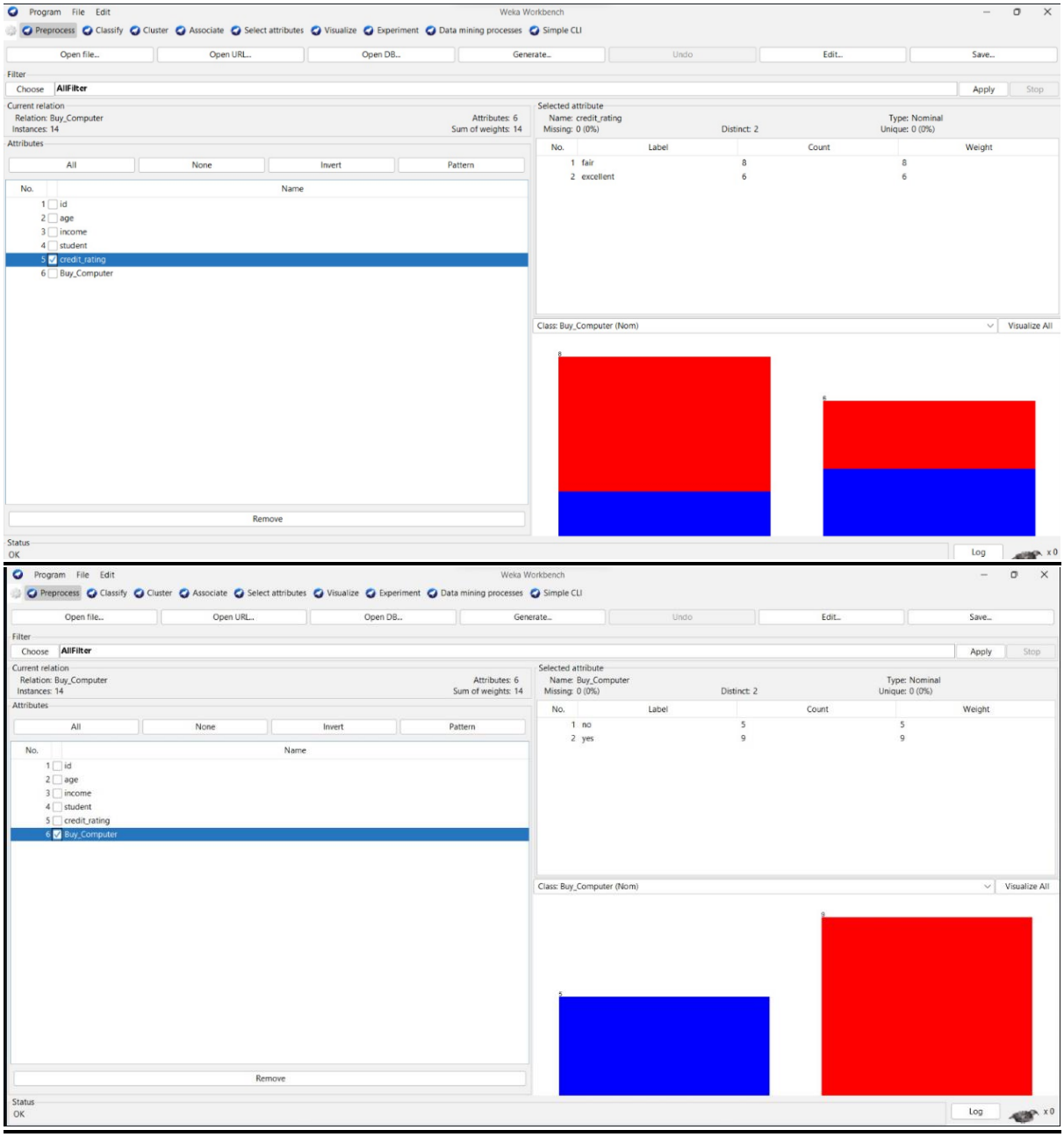
1. Preprocessing





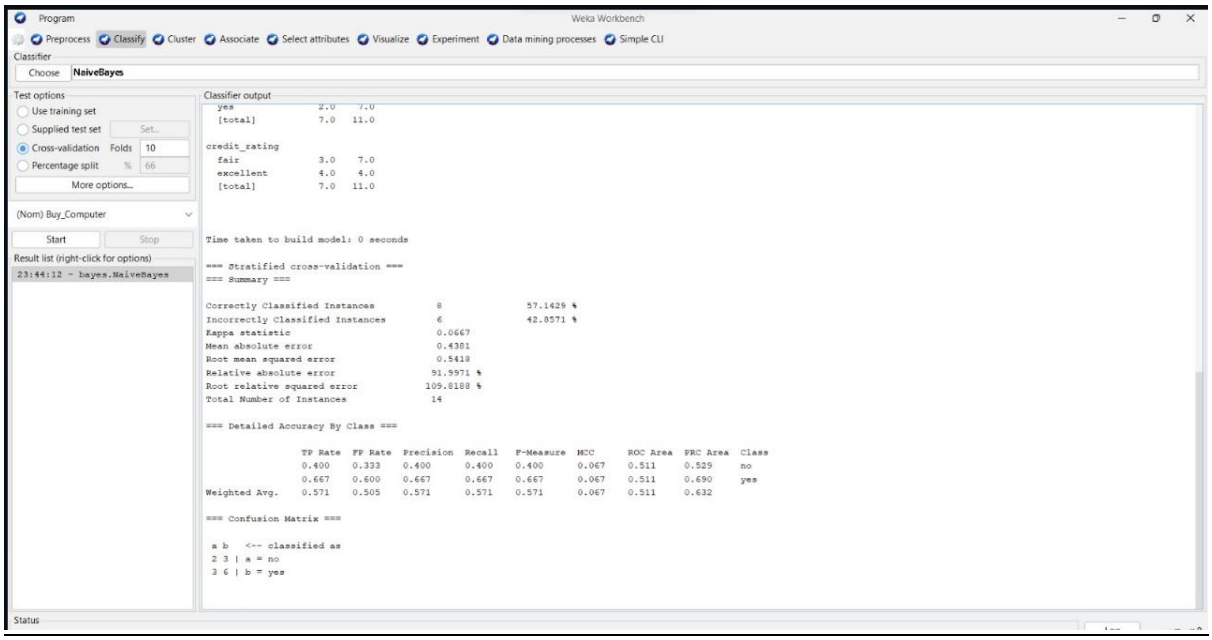

Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
Attributes: 6
Sum of weights: 14
Attributes
All None Invert Pattern
No. Name
1 id
2 age
3 income
4 student
5 credit_rating
6 Buy_Computer
Remove
Selected attribute
Name: income
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 high 4 4
2 medium 6 6
3 low 4 4
Class: Buy_Computer (Nom)
Visualize All
Status
OK
Log
x 0


Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
Attributes: 6
Sum of weights: 14
Attributes
All None Invert Pattern
No. Name
1 id
2 age
3 income
4 student
5 credit_rating
6 Buy_Computer
Remove
Selected attribute
Name: income
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 high 4 4
2 medium 6 6
3 low 4 4
Class: Buy_Computer (Nom)
Visualize All
Status
OK
Log
x 0

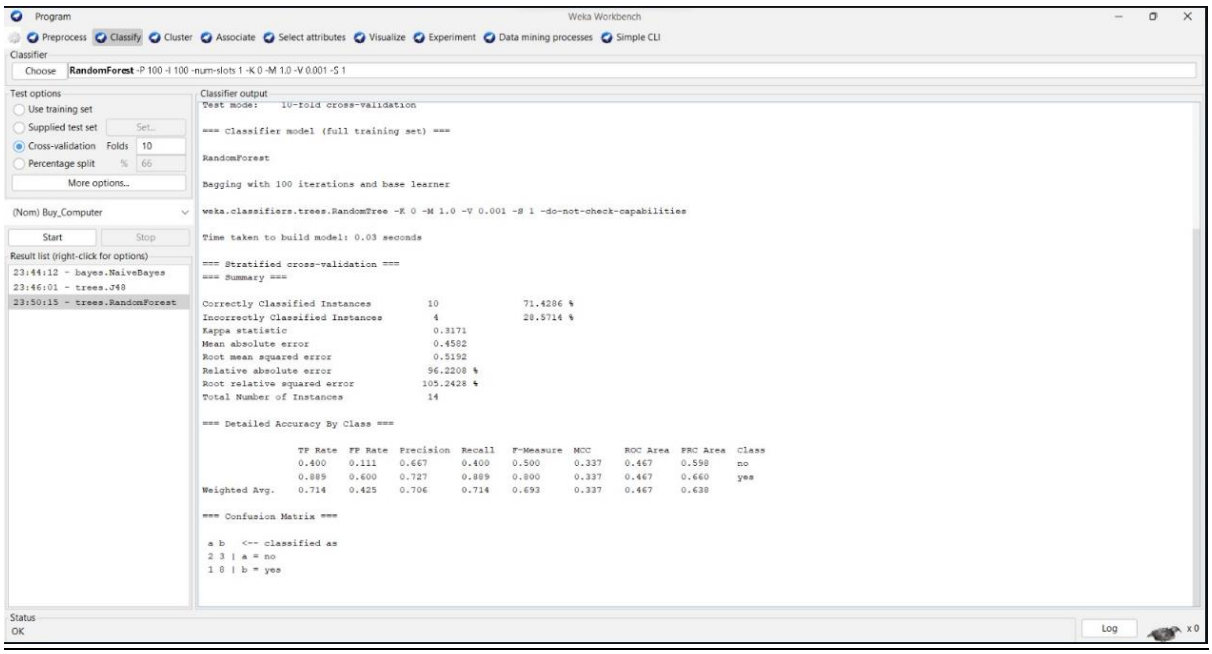

Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
Attributes: 6
Sum of weights: 14
Attributes
All None Invert Pattern
No. Name
1 id
2 age
3 income
4 student
5 credit_rating
6 Buy_Computer
Remove
Selected attribute
Name: credit_rating
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 fair 8 8
2 excellent 6 6
Class: Buy_Computer (Nom)
Visualize All
Status
OK
Log
x 0
Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
Attributes: 6
Sum of weights: 14
Attributes
All None Invert Pattern
No. Name
1 id
2 age
3 income
4 student
5 credit_rating
6 Buy_Computer
Remove
Selected attribute
Name: Buy_Computer
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 no 5 5
2 yes 9 9
Class: Buy_Computer (Nom)
Visualize All
Status
OK
Log
x 0

2. Classification

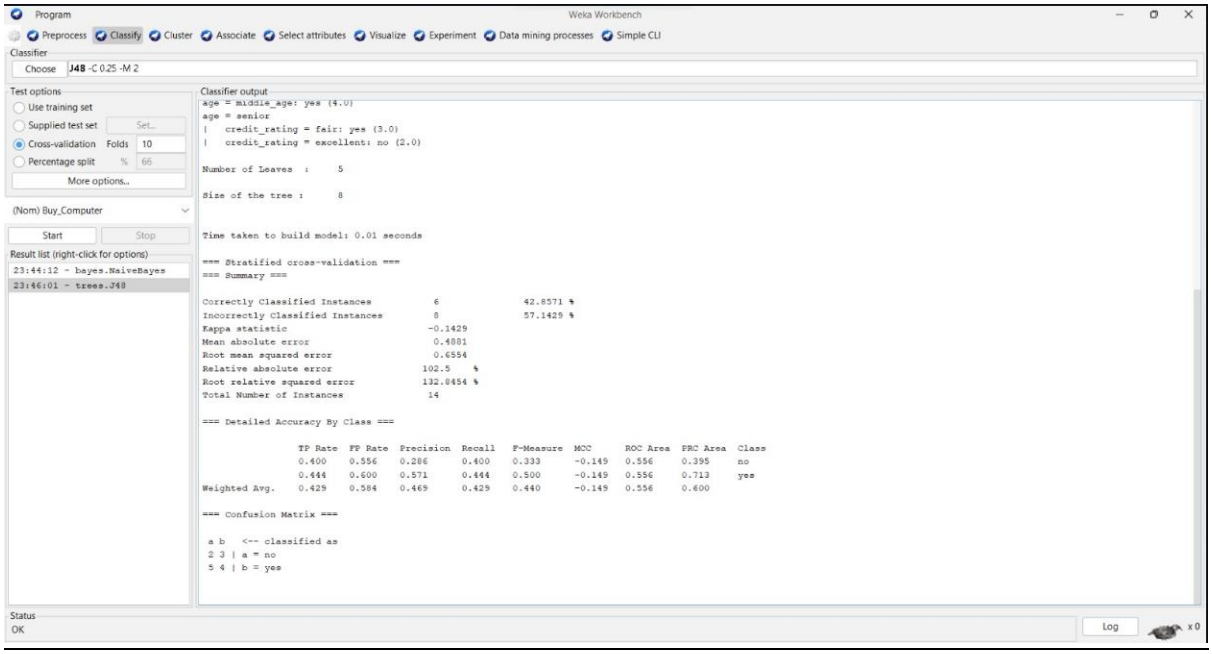
a. Naïve Bayes:

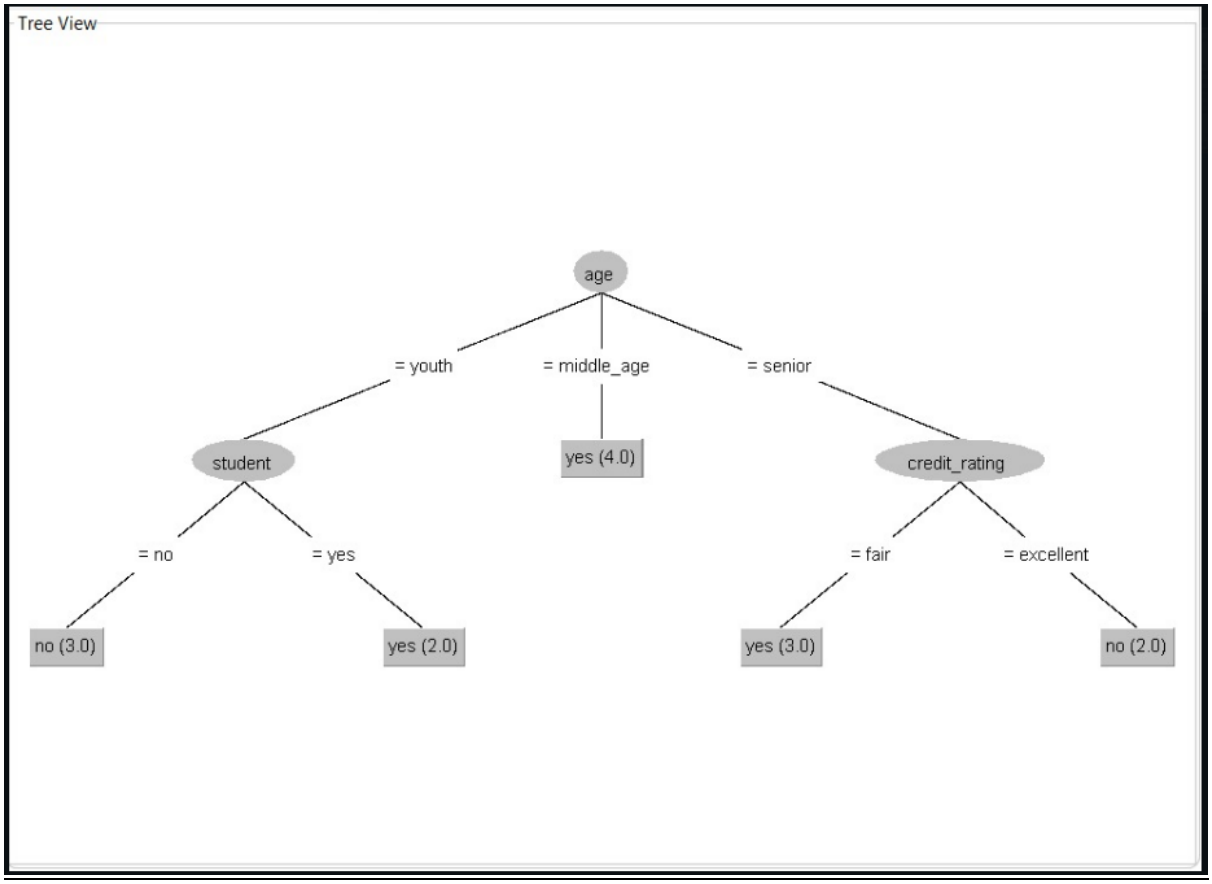


b. Random Forest:



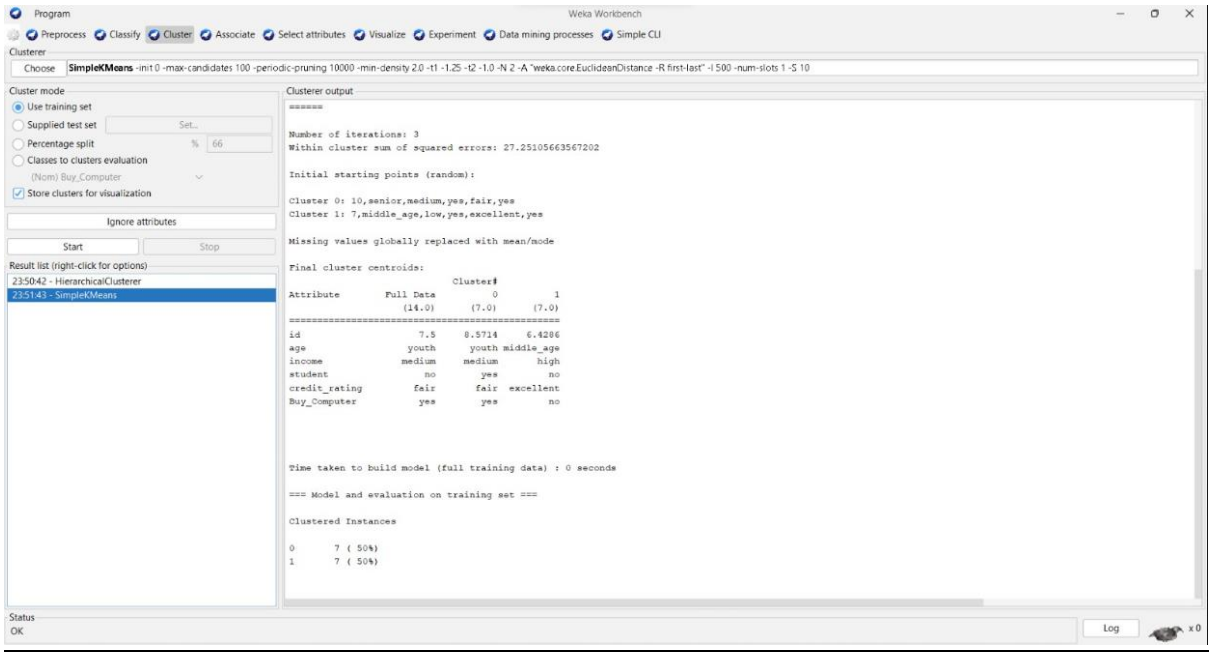
c. J48


Program
Weka Workbench
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Experiment
Data mining processes
Simple CLI
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) Buy_Computer
Start
Stop
Result list (right-click for options)
23:44:12 - bayes.NaiveBayes
23:46:01 - trees.J48
Classifier output
age = middle_age: yes (4.0)
age = senior
| credit_rating = fair: yes (3.0)
| credit_rating = excellent: no (2.0)
Number of Leaves : 5
Size of the tree : 8
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 6 42.8571 %
Incorrectly Classified Instances 8 57.1429 %
Kappa statistic -0.1429
Mean absolute error 0.4081
Root mean squared error 0.6554
Relative absolute error 102.5 %
Root relative squared error 132.0454 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.400 0.556 0.286 0.400 0.333 -0.149 0.556 0.395 no
0.444 0.600 0.571 0.444 0.500 -0.149 0.556 0.713 yes
Weighted Avg. 0.429 0.584 0.469 0.429 0.440 -0.149 0.556 0.600
=== Confusion Matrix ===
a b <-- classified as
2 3 | a = no
5 4 | b = yes
Status
OK
Log
x 0

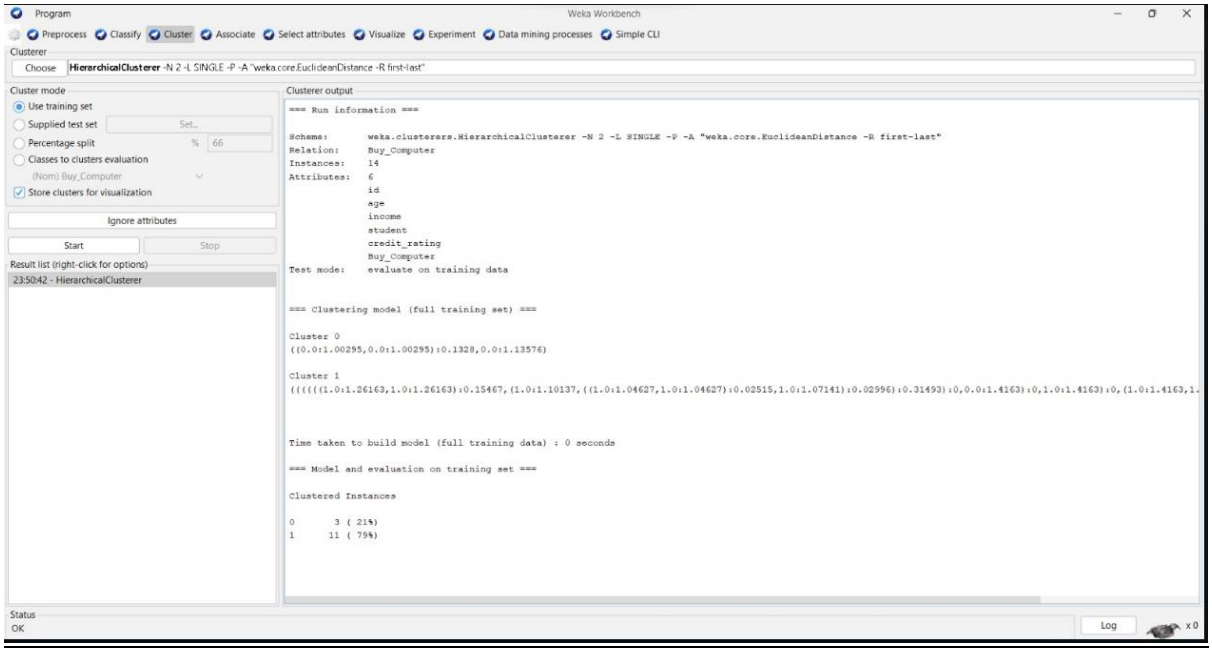

Tree View
age
= youth
= middle_age
= senior
student
yes (4.0)
credit_rating
= no
= yes
= fair
= excellent
no (3.0)
yes (2.0)
yes (3.0)
no (2.0)

3. Clustering

a. Kmeans

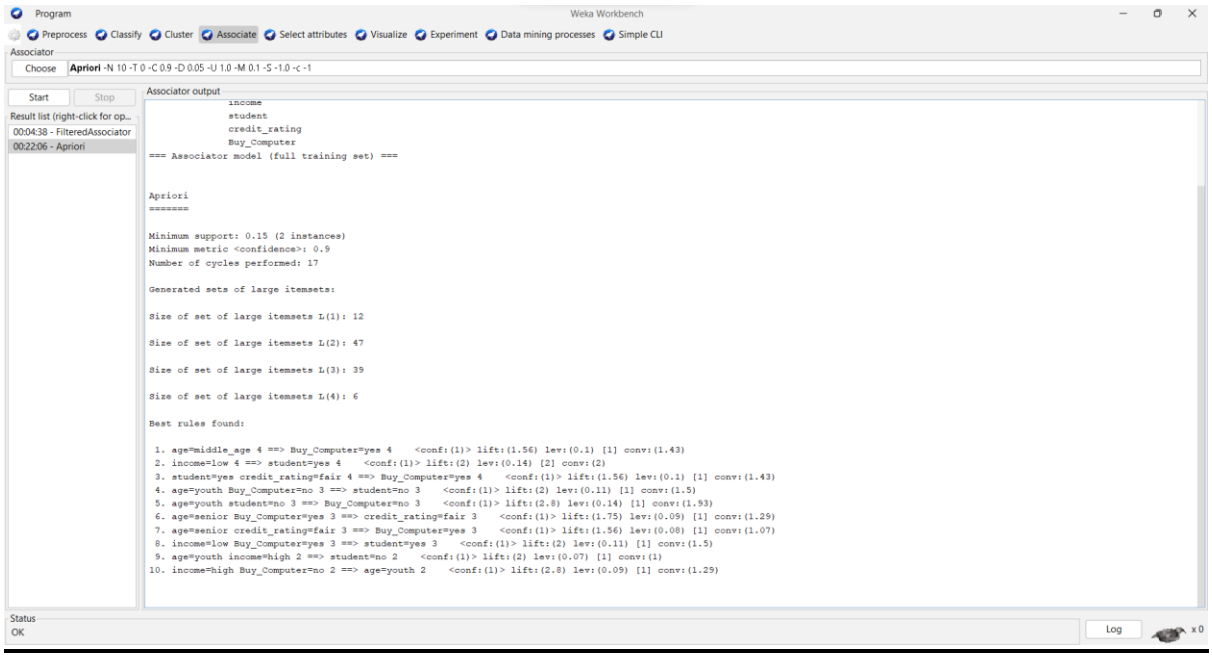


b. Hierarchical Clustering



4. Association

a. Apriori



Experiment 7

Aim: Implementation of clustering algorithms (K-means)

Theory:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

Determines the best value for K center points or centroids by an iterative process.

Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

Algorithm:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Flowchart:

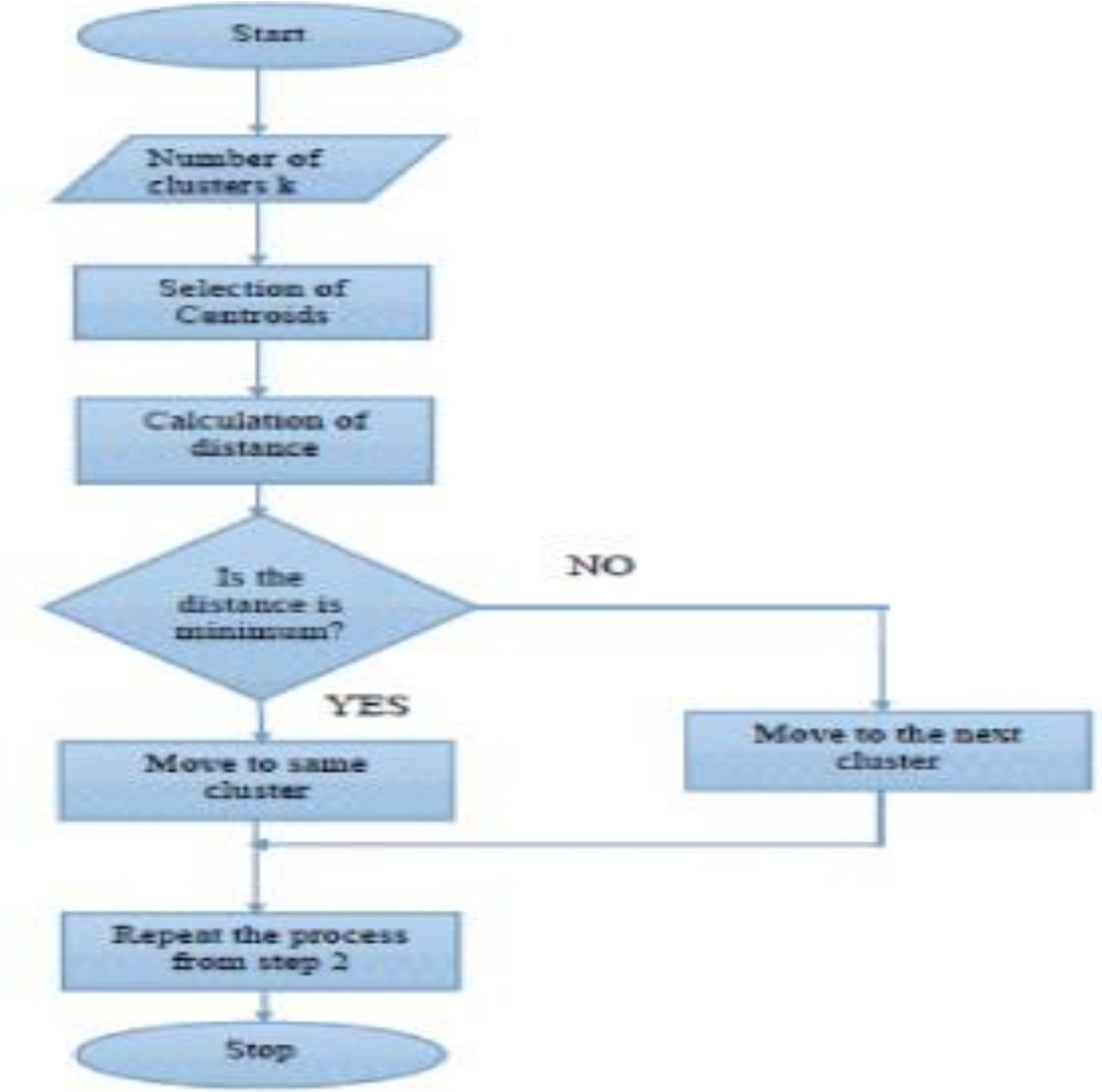


Fig 7.1 Flowchart of K-Means

Example:

Cluster the following eight points (with (x, y) representing locations) into three clusters:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).

The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-

P(a, b) = |x2 – x1| + |y2 – y1|

Use K-Means Algorithm to find the three cluster centers after the second iteration.

Iteration 1:

P(A1, C1) = |x2 – x1| + |y2 – y1

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

Iteration 2:

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1

A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

Therefore, new clusters:

Cluster-01:

A1(2, 10)

A8(4, 9)

Cluster-02:

A3(8, 4)

A4(5, 8)

A5(7, 5)

A6(6, 4)

Cluster-03:

A2(2, 5)

A7(1, 2)

Center of Cluster-01

= ((2 + 4)/2, (10 + 9)/2) = (3, 9.5)

Center of Cluster-02

= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4) = (6.5, 5.25)

Center of Cluster-03

= ((2 + 1)/2, (5 + 2)/2) = (1.5, 3.5)

Ans:

- C1(3, 9.5)
- C2(6.5, 5.25)
- C3(1.5, 3.5)

Code:

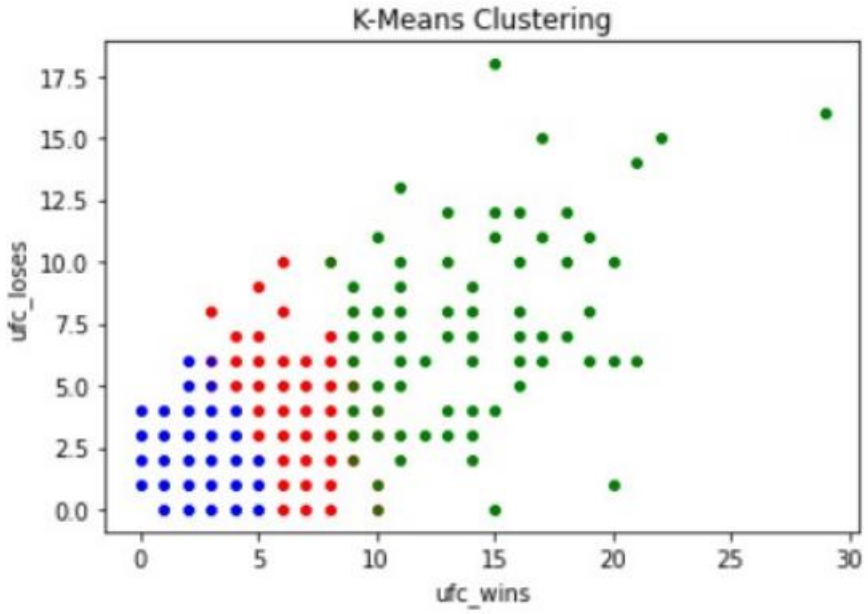
```
import csv
import matplotlib.pyplot as plt


def initializeCentroids(k, data):
    centroids = []
    for i in range(k):
        centroids.append(data[i])
    return centroids


def plot(clusters):
    colors = ['red', 'green', 'blue', 'yellow', 'black', 'purple', 'pink', 'cyan']
    i = 0
    for cluster in clusters:
        x = [point[0] for point in cluster]
        y = [point[1] for point in cluster]
        plt.scatter(x, y, s=10, c=colors[i])
```

```
        i += 1

def distance(xc, yc, px, py):
    x = (px - xc) ** 2
    y = (py - yc) ** 2
    distance = (x + y) ** 0.5
    return round(distance, 2)

def mean(cluster):
    sumx = 0
    sumy = 0
    for coord in cluster:
        sumx += coord[0]
        sumy += coord[1]
    return (round(sumx / len(cluster), 2), round(sumy / len(cluster), 2))

data = []
k = int(input('Enter the number of clusters: '))

with open('ufc_master_data.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader:
        data.append((float(row['ufc_wins']), float(row['ufc_loses'])))

centroids = initializeCentroids(k, data)
cluster_coords = [[] for _ in range(k)]  # Initialize cluster_coords with empty lists for each cluster

while True:
    for coords in data:
        point_distance = [distance(centroid[0], centroid[1], coords[0], coords[1]) for centroid in centroids]
        cluster_coords[point_distance.index(min(point_distance))].append(coords)

    prev_cents = centroids
    centroids = [mean(cluster) for cluster in cluster_coords]
```

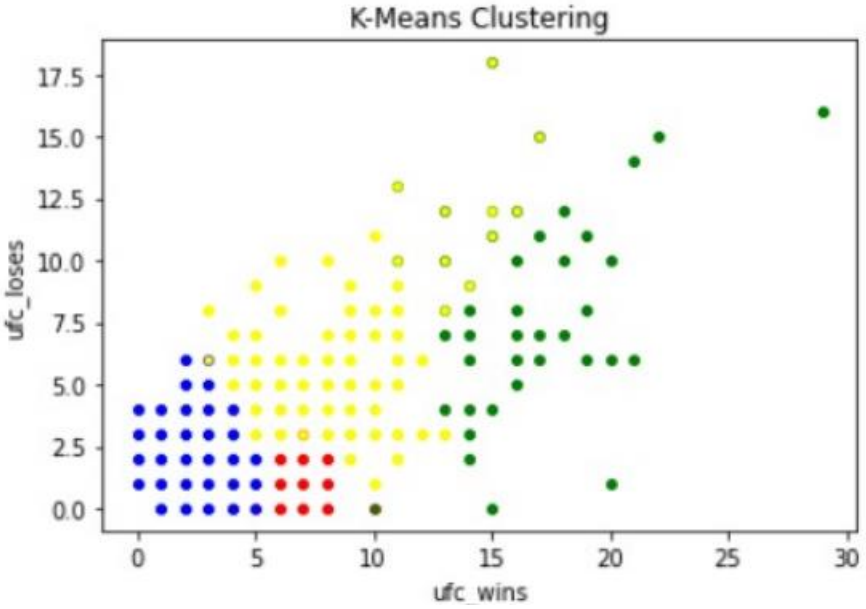
```
    if prev_cents == centroids:
        break

i = 1
for centroid in centroids:
    print('Centroid ', i, ': ', centroid)
    i += 1

plot(cluster_coords)
plt.title('K-Means Clustering')
plt.xlabel('ufc_wins')
plt.ylabel('ufc_loses')
plt.show()
```

Output:

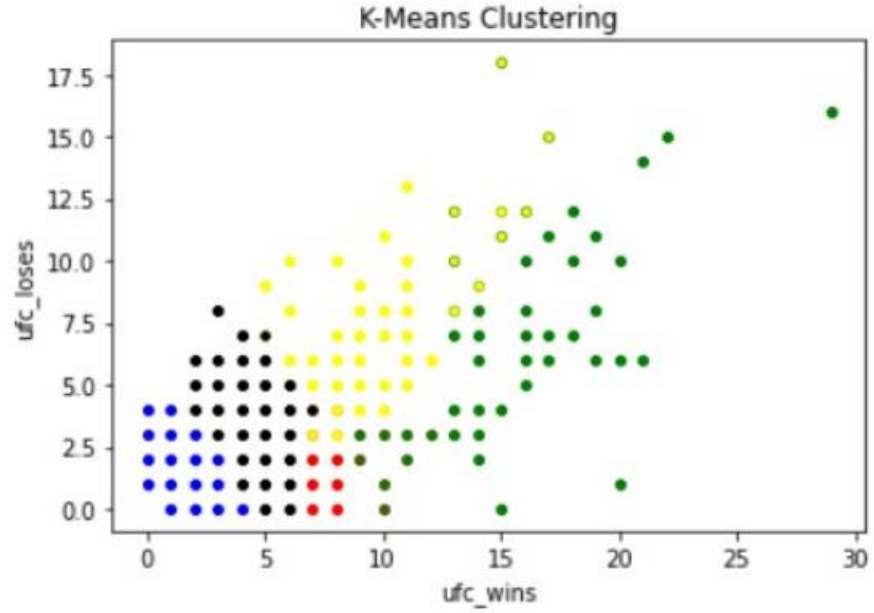
```
Enter the number of clusters: 3
Centroid  1 :  (6.41, 3.7)
Centroid  2 :  (14.56, 7.7)
Centroid  3 :  (1.74, 1.49)
```



```
Enter the number of clusters: 4
Centroid  1 :  (5.16, 2.36)
Centroid  2 :  (15.83, 7.92)
Centroid  3 :  (1.49, 1.53)
Centroid  4 :  (8.27, 5.43)
```




```
Enter the number of clusters: 5
Centroid  1 :  (8.9, 2.52)
Centroid  2 :  (16.39, 8.49)
Centroid  3 :  (1.41, 1.3)
Centroid  4 :  (8.6, 6.68)
Centroid  5 :  (4.43, 3.06)
```



Experiment 8

Aim: Implementation of any one hierarchical clustering method

Theory:

A Hierarchical clustering method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data point as a separate cluster. Then, it repeatedly executes the subsequent steps:

Identify the 2 clusters which can be closest together, and

Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called Dendrogram (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are broken up (top-down view).

Hierarchical clustering is a method of cluster analysis in data mining that creates a hierarchical representation of the clusters in a dataset. The method starts by treating each data point as a separate cluster and then iteratively combines the closest clusters until a stopping criterion is reached. The result of hierarchical clustering is a tree-like structure, called a dendrogram, which illustrates the hierarchical relationships among the clusters.

Agglomerative: Initially consider every data point as an individual Cluster and at every step, merge the nearest pairs of the cluster. (It is a bottom-up method). At first, every dataset is considered an individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed.

The algorithm for Agglomerative Hierarchical Clustering is:

Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)

Consider every data point as an individual cluster

Merge the clusters which are highly similar or close to each other.

Recalculate the proximity matrix for each cluster

Repeat Steps 3 and 4 until only a single cluster remains.

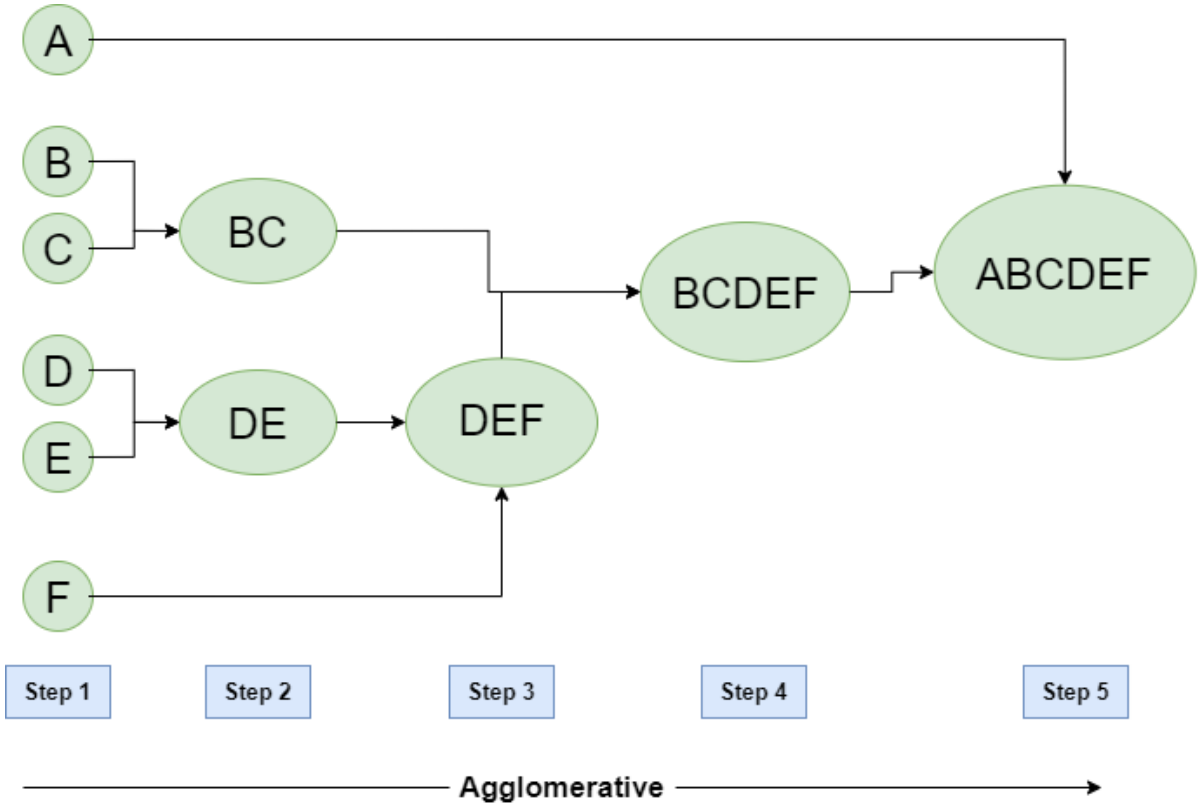
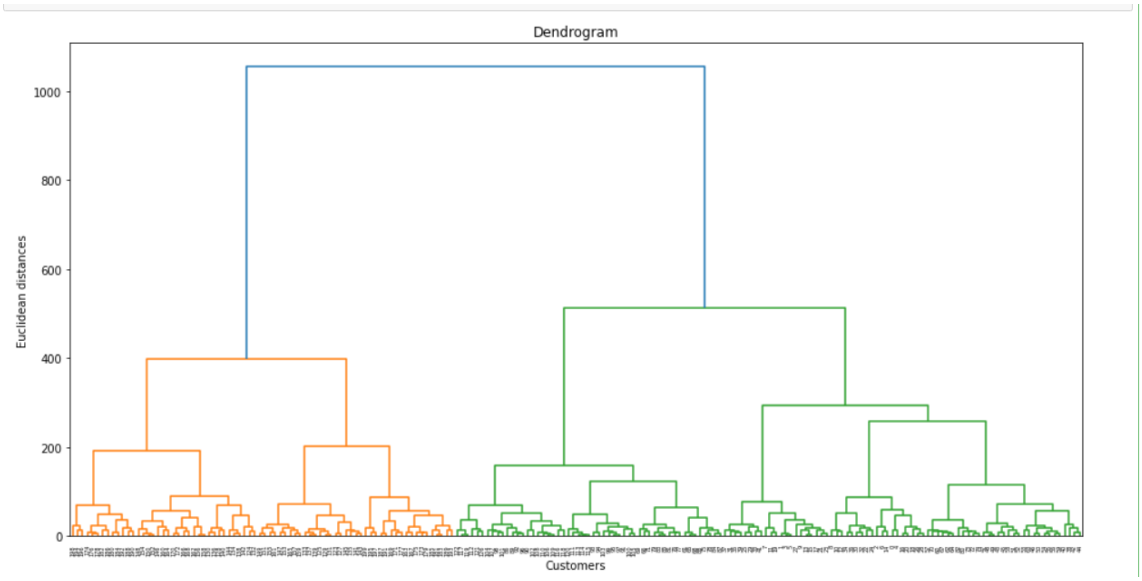


Fig 9.1 Agglomerative Hierarchical Clustering

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn import preprocessing
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

df = pd.read_csv('Mall_Customers.csv')

label_encoder = preprocessing.LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
```

```
linkage_matrix = sch.linkage(df, method="ward")


agglomerative = AgglomerativeClustering(n_clusters=None, affinity='euclidean',
linkage='ward', distance_threshold=0)
agglomerative.fit(df)


cluster_labels = agglomerative.labels_


plt.figure(1, figsize=(16, 8))
dendrogram = sch.dendrogram(linkage_matrix)
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()


distance_matrix = sch.distance.pdist(df)
distance_matrix = sch.distance.squareform(distance_matrix)
distance_matrix
```



In [4]:
```
print(distance_matrix)
```

```
[[  0.          42.05948169  33.10589071 ... 229.18115106 233.87603554
  237.78561773]
 [ 42.05948169   0.          75.02666193 ... 225.6257964  240.38094766
  232.75093985]
 [ 33.10589071  75.02666193   0.         ... 234.29468624 230.96753019
  243.8852189 ]
 ...
 [229.18115106 225.6257964  234.29468624 ...   0.          57.07889277
   14.49137675]
 [233.87603554 240.38094766 230.96753019 ...  57.07889277   0.
   65.03845017]
 [237.78561773 232.75093985 243.8852189  ...  14.49137675  65.03845017
    0.        ]]
```

Experiment 9

Aim: Implementation of Association Rule Miningalgorithm (Apriori)

Theory:

Apriori is a widely used algorithm in data mining and machine learning for finding frequent itemsets in transactional databases. It is a fundamental algorithm for association rule mining, which aims to discover relationships or patterns in data. Apriori is specifically used to identify frequent itemsets and generate association rules based on these itemsets.

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

- Support: Support measures how frequently an itemset appears in the dataset. It is defined as the ratio of the number of transactions containing the itemset to the total number of transactions. High support indicates that an itemset is common in the dataset.
- Confidence: Confidence measures the strength of an association rule in terms of the likelihood that the consequent (the item on the right side of the rule) will be purchased when the antecedent (the item(s) on the left side of the rule) is/are purchased.
- Lift: Lift measures how much more likely the consequent is to be purchased when the antecedent is purchased compared to when the consequent is purchased without the antecedent.

Program:

```
import csv
from itertools import combinations


def find_frequency(string, freqs):
    for dictionary in freqs:
        if (string in list(dictionary.keys())):
            return dictionary[string]


def power_set(string):
    power_set = set({})
    for i in range(0, len(string) + 1):
        for element in combinations(string, i):
```

```
            power_set.add(' '.join(element))
    power_set -= set({''})
    power_set.remove(' '.join(string))
    power_set = list(power_set)
    power_set.sort()
    return power_set

def association(keys, confidence, all_freqs):
    final_rules = []
    for key in keys:
        lhs = power_set(key.split(' '))
        rhs = []
        elements = set(key) - {' '}
        for elem in lhs:
            to_join = list(elements - set(elem) - {' '})
            to_join.sort()
            rhs.append(' '.join(to_join))
        for l, r in zip(lhs, rhs):
            rule = l + ' -> ' + r
            string = l + " " + r
            temp_lst = string.split(' ')
            temp_lst.sort()
            string = ' '.join(temp_lst)
            conf = (find_frequency(string, all_freqs) / find_frequency(l, all_freqs)) * 100
            if (conf >= confidence):
                print(rule + " with a confidence of " + str(conf) + "%")

def combine(keys, key_len):
    final_keys = []
    for lst1 in keys:
```

```
        for lst2 in keys:
            temp_lst = lst1[:]
            temp_lst.extend(lst2)
            temp_lst = list(set(temp_lst))
            temp_lst.sort()
            if (len(temp_lst) == key_len + 1):
                if (temp_lst not in final_keys):
                    final_keys.append(temp_lst)
    return final_keys

def filter_comb(keys, data, support):
    final_keys = []
    frequency = {}
    for key in keys:
        present = True
        freq = 0
        for value in list(data.values()):
            for item in key:
                if (item not in value):
                    present = present and False
            if present == True:
                freq = freq + 1
            present = True
        frequency[' '.join(key)] = freq
    frequency = dict(filter(lambda elem: elem[1] >= support, frequency.items()))
    return frequency

items = []
item_set = []
data = {}
```

```
previous = {}
support = int(input("Enter the minimum support (in percent): "))
confidence = int(input("Enter the minimum confidence (in percent): "))
frequency = {}
key_len = 1
all_freqs = []

# Provided dataset
transactions = [
    [1, 2, 3, 4],
    [1, 4, 2, 5, 4],
    [3, 4, 5],
    [1, 3, 4],
    [3, 4, 6],
    [1, 3, 4, 6],
    [1, 6],
    [1, 3, 4],
    [3, 4, 2, 6],
    [1, 4, 3],
    [1, 3],
    [1, 3, 6, 5]
]

# Convert transaction data to a dictionary
for i, transaction in enumerate(transactions):
    data[i + 1] = list(map(str, transaction))

# Combine items into a single list
items = [item for sublist in data.values() for item in sublist]
```

```
# Get unique items
item_set = list(set(items))
item_set.sort()

# Calculate support threshold
support = (support / 100) * len(data)

# Calculate item frequency
for item in item_set:
    frequency[item] = items.count(item)

# Filter items based on support
frequency = dict(filter(lambda elem: elem[1] >= support, frequency.items()))
key_list = []

# Convert frequent items to a list of lists
for key in list(frequency.keys()):
    key_list.append([key])

while (True):
    key_list = combine(key_list, key_len)
    all_freqs.append(frequency)
    previous = frequency.copy()
    frequency = filter_comb(key_list, data, support)
    key_list = []
    for key in list(frequency.keys()):
        key_l = key.split(' ')
        key_list.append(key_l)
    key_len += 1
    if (len(key_list) <= 1):
```

```
    break

print("The strong association rules are as follows:")
if (not frequency):
    association(previous, confidence, all_freqs)
else:
    all_freqs.append(frequency)
    association(frequency, confidence, all_freqs)
```

Output:

```
Enter the minimum support (in percent): 30
Enter the minimum confidence (in percent): 20
The strong association rules are as follows:
1 -> 3 4 with a confidence of 55.55555555555556%
1 3 -> 4 with a confidence of 71.42857142857143%
1 4 -> 3 with a confidence of 83.33333333333334%
3 -> 1 4 with a confidence of 50.0%
3 4 -> 1 with a confidence of 62.5%
4 -> 1 3 with a confidence of 50.0%
```

```
Enter the minimum support (in percent): 50
Enter the minimum confidence (in percent): 40
The strong association rules are as follows:
1 -> 3 with a confidence of 77.77777777777779%
3 -> 1 with a confidence of 70.0%
1 -> 4 with a confidence of 66.66666666666666%
4 -> 1 with a confidence of 60.0%
3 -> 4 with a confidence of 80.0%
4 -> 3 with a confidence of 80.0%
```

```
Enter the minimum support (in percent): 30
Enter the minimum confidence (in percent): 40
The strong association rules are as follows:
1 -> 3 4 with a confidence of 55.55555555555556%
1 3 -> 4 with a confidence of 71.42857142857143%
1 4 -> 3 with a confidence of 83.33333333333334%
3 -> 1 4 with a confidence of 50.0%
3 4 -> 1 with a confidence of 62.5%
4 -> 1 3 with a confidence of 50.0%
```

Experiment 10

Aim: Implementation of Page Rank Algorithm

Theory:

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.

The above centrality measure is not implemented for multi-graphs.

Working:

Step 1: Web Graph Representation

Represent the web as a directed graph, where web pages are nodes, and hyperlinks are directed edges.

Step 2: Initialize PageRank

Assign an initial PageRank value to each page, typically a uniform value or equally distributing PageRank across all pages.

Step 3: Damping Factor

Introduce a damping factor (usually around 0.85) to account for random user behavior and ensure convergence.

Step 4: Iterative Calculation

Iteratively update PageRank values for each page. During each iteration:

Step 5: PageRank Calculation

Calculate the new PageRank value for each page using the formula:

PR(A) = (1 - d) / N + d * (PR(B)/L(B) + PR(C)/L(C) + ... + PR(N)/L(N))

PR(A) is the new PageRank value for page A.

d is the damping factor.

N is the total number of pages.

PR(B), PR(C), ..., PR(N) are the current PageRank values of pages linking to page A.

L(B), L(C), ..., L(N) are the number of outbound links on pages B, C, ..., N.

Step 6: Iterate Until Convergence

Repeat the PageRank calculation iteratively until PageRank values stabilize (converge). This typically requires several iterations.

Step 7: Ranking

Rank pages based on their final PageRank scores. Higher PageRank scores indicate greater importance.

Step 8: Handling Sink Nodes

Address "sink nodes" (pages with no outbound links) to redistribute PageRank and prevent loss.

Step 9: Personalized PageRank

Calculate personalized PageRank to provide customized rankings based on user preferences or interests.

Step 10: Handling Dead-Ends

Consider how to handle "dead-end" pages (pages with no inbound links) to ensure they receive some PageRank value.

These steps summarize the key concepts of the PageRank algorithm, which plays a crucial role in web search and information retrieval systems.

Dataset:

	A	B	C	D
A	0	1	0	1
B	1	0	1	0
C	0	1	0	0
D	0	0	1	0

Program:

```
import numpy as np


# Load the adjacency matrix from pagerank.csv
adjacency_matrix = np.genfromtxt('pagerank.csv', delimiter=',', skip_header=1, usecols=range(1, 5))
damping_factor = 0.85
num_iterations = 100
num_nodes = len(adjacency_matrix)
pagerank = np.ones(num_nodes) / num_nodes
for _ in range(num_iterations):
```

```
    new_pagerank = np.zeros(num_nodes)

    for i in range(num_nodes):
        for j in range(num_nodes):
            if adjacency_matrix[j, i] == 1:
                new_pagerank[i] += pagerank[j] / np.sum(adjacency_matrix[j])

    pagerank = (1 - damping_factor) / num_nodes + damping_factor * new_pagerank
# Create a list of nodes and their corresponding PageRank scores
nodes = ['A', 'B', 'C', 'D']
pagerank_scores = list(pagerank)

# Sort the nodes based on their PageRank scores in descending order
sorted_nodes = [node for _, node in sorted(zip(pagerank_scores, nodes), reverse=True)]

# Print the ranked nodes
for i, node in enumerate(sorted_nodes):
    print(f'{i+1}. Node {node}: {pagerank_scores[nodes.index(node)]}')
```

Output:

```
1. Node B: 0.37824245632609843
2. Node C: 0.30174695606140081
3. Node A: 0.19825304393859183
4. Node D: 0.12175754367390153
```

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

DWM 11.1

:- Assignments :- (01) :-

Q. What is Metadata? Explain Data warehouse Metadata with Example.

⇒ Def^n :- Data warehouse metadata are pieces of information stored in one or more special purpose metabata repositories that include.

(a) Info on the contents of the data warehouse, ther loc & their structure
(b) Info on the process that take place in DW back-stage concering the refreshment of the warehouse with clean up-to-date, semantically, and structurally reconciled data.
(c) Information on the implicit semantics of data along with any other kind of data that aids the end-user explore the information of the warehouse.
(d) Information on the infrastructure and physical characteristics of compnets and the sources of the data warehouse. and.
(e) info including security, authorizations and usage statics that aids the administrator runs the operation of the data warhouse as appropriate.

⇒ DW Metadata :- It has specific Metadata that desc. table metadata requirements typical are -

→ Physical name
→ logical name
→ Type : fact, Dimension, Bridge.
→ Role : legacy, OLTP, stage.
→ DBMS : DB2, informix
→ location
→ Defination
→ Notes

⇒ Metadata describes columns within Table

Physical Name	logical Name
Order in Table length Nullable / Required Edit Rule Notes	Data type Decimal Points Default value Definition

Example of Metadata for Customer Sales data Warehouse

+ Entity Name : Customer
→ Alias Names : Account, client
→ Definition : A person or an organization that purchases goods or services from the company.
→ Remarks : Customer entity includes regular, casual, and past customers.
→ Source Systems : Finished Goods Order, Maintenance Contracts, Online Sales
→ Create Date : June 15, 2007
→ Last Update Date : June 21, 2009
→ Update Cycle : Weekly
→ Last Full Refresh Date : December 29, 2008
→ Full Refresh Cycle : Every six months
→ Data Quality Reviewed : July 25, 2009
→ Last Deduplication : June 05, 2009
→ Planned Archival : Every six months
→ Responsible User : Rallani H

Pain B+

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

DWM 12.1

-: Assignments :- (02)

Q1 Write short notes on

a) Multilevel Association rules and Multidimensional association rules
b) Web Usage Mining.

⇒ a) i) Multilevel Association Rule:

Defn: It is a type of DM technique that extends traditional association rule mining by considering multiple levels of abstraction within the data.

Hierarchy: In multilevel association rule mining data is organized in a hierarchical structure. This hierarchy can be based on attrib., categ., or any other form of hierarchy present in the data.

Ex:- Consider a retail dataset. Multilevel association rules can be applied to find associations and patterns not only at the level of individual products but also at higher levels, such as product categ. departments or even across the entire store.

UseCase: Multilevel association rules are useful in various domains like retail, where you want to uncover associations.

and patterns across different level of product hierarchy to make informed decisions about product placement, promotion and inventory management.

Multidimensional Association Rules:-

Defn: Multidimensional association rule focus on analyzing data with multiple dimensions or attributes. These rules are often associated with data cubes where each dimension represents a diff. attribute or characteristic.

Data Cubes: In Multidimensional association rule mining data is typically organized in a multidimensional cube, allowing for the analysis of relationship between various dimensions.

Eg: Consider analysis sales data with multiple dimension like product, time, loc., & customer segment. Multidimensional association rules can reveal interesting insights about how these dimensions are correlated.

Use cases: MDs find applications in areas such as business intelligence, where organization want to gain a deeper understanding of their data by exploring.

relationship betn diff. attri. and dimensions. This help in decision making, trend analysis and resources allocation.

In Summary, both multilevel & multidimensional association rule mining techniques are valuable in extracting meaningful patterns and relationship from complex datasets but they focus on different aspects of data hierarchy and dimensionality.

⇒ **Web usage Mining.**

Def": Web usage Mining is a subset of data mining that involves the discovery and analysis of valuable patterns and knowledge from web data, particularly the interactions and behaviors of users on websites.

Types:

- Clickstream Analysis: Examines user clicks, navigation paths and interactions with web pages.
- Pageview Analysis: Shows which web pages are most frequently accessed by users.

Session Analysis: Studies which web pages are most frequently accessed by users.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

⇒ Applications:

→ Personalization: Web Usage Mining is crucial for creating personalized user experiences by offering tailored content product recommendations or advertisements based on a user's past behaviour.

→ Content Improvement: It helps website owners optimize content and layout by identifying which pages are popular which are frequently visited and how users navigate the sites.

→ Marketing Optimization: Marketers use web usage data to target specific user segments with relevant advertising and promotional campaigns.

⇒ Challenges:

→ Privacy Concerns: The collection of user data raises privacy issues and organizations must adhere to regulations like GDPR to protect user information.

→ Data Volume: Websites generate massive amounts of data meaning efficient data storage retrieval and processing.

Methods:

Data Missing: web data can be noisy and incomplete due to factors like bot traffic incomplete user sessions and server errors.

Web usage Mining is a valuable tool for web-based businesses and organizations looking to enhance user experience increase engagement and make data-driven decisions to improve their online presence.

Rajvir A

Naive Bayes Classifier.

Car Theft example Attributes are color type original & the subject, stolen be either yes or no.

Data set:

Car No	Color	type	orignal	stolen.
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic.	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes.

We want to classify a Red, domestic, SUV unseen sample. Note there is no example of a Domestic SUV. SUVs is our data set.

$P(\text{yes}) = 5/10$

$P(\text{NO}) = 5/10.$

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Color:
P(Red/Yes) = 3/5 P(Red/No) = 2/5
P(Yellow/Yes) = 2/5 P(Yellow/No) = 3/5

Type
P(SUV/Yes) = 1/5 P(SUV/No) = 3/5
P(Sports/Yes) = 4/5 P(Sports/No) = 2/5

Origin
P(Domestic/Yes) = 2/5 P(Domestic/No) = 3/5
P(Imported/Yes) = 3/5 P(Imported/No) = 2/5

An unseen sample x = <Red, Domestic, SUV>
P(x/Yes) . P(Yes)
= P(Red, Yes) . P(Domestic . Yes) . P(SUV/Yes) . P(Yes)
$= \frac{3}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{5}{10}$
= 0.024

P(x/No) P(No)
= P(Red/No) . P(Domestic/No) . P(SUV/No) . P(No)
$= \frac{2}{5} \times \frac{3}{5} \times \frac{3}{5} \times \frac{5}{10}$
= 0.072.

Since 0.072 > 0.024, our example gets classified as 'No'.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Apriori example:

T/D	items
T1	I1 I2 I5
T2	I2 I4
T3	I2 I3
T4	I1 I2 I4
T5	I1 I3
T6	I2 I3
T7	I1 I3
T8	I1 I2 I3 I5
T9	I1 I2 I3
~~I9~~	

minimum confidence = 50%.
minimum support count is 2.

i) k = 1

I1	6
I2	7
I3	6
I4	2
I5	2

ii)

Itemset	sup_count
I_1, I_2	4
I_1, I_3	4
I_1, I_4	1
I_1, I_5	2
I_2, I_3	4
I_2, I_4	2
I_2, I_5	2
I_3, I_4	0
I_3, I_5	1
I_4, I_5	0

∴ remove

I_3, I_4	$0 < 2$
I_3, I_5	$1 < 2$
I_4, I_5	$0 < 2$.

iii)

I_2, I_2, I_3	2
I_1, I_2, I_5	2

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Now,

confidence $(A \rightarrow B)$ = support count $(A \cup B)$ / support count (A).

$(I1, I2) \rightarrow I3$	2/4	50%.
$(I1, I3) \Rightarrow I2$	2/4	50%.
$(I2, I3) \rightarrow I1$	2/4	50%.
$I1 \rightarrow (I2, I3)$	2/6	33%.
$I2 \rightarrow (I1, I3)$	2/7	28%.
$I3 \rightarrow (I1, I2)$	2/6	33%.

∴ min confidence is 50%.

∴ first three $(I1, I2) \rightarrow I3$
$(I1, I3) \rightarrow I2$
$(I2, I3) \rightarrow I1$

can be consider as strong association rules.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

9). Perform Agglomerative Algorithm on the following data & plot a dendogram using single link Approach. the given data indicates the distance b/w elements.

Item	E	A	C	B	D.
E	0	1	2	2	3
A	(1)	0	2	5	3
C	2	2	0	1	6
B	2	5	1	0	3
D	3	3	6	3	0

Pair E, A.

	(E, A)	C	B	D.
(E, A)	0			
C	2	0		
B	2	(1)	0	
D	3	6	3	0

Pair (B, C).

	(E, A)	(B, C)	D.
(E, A)	0		
(B, C)	(2)	0	
D	3	3	0

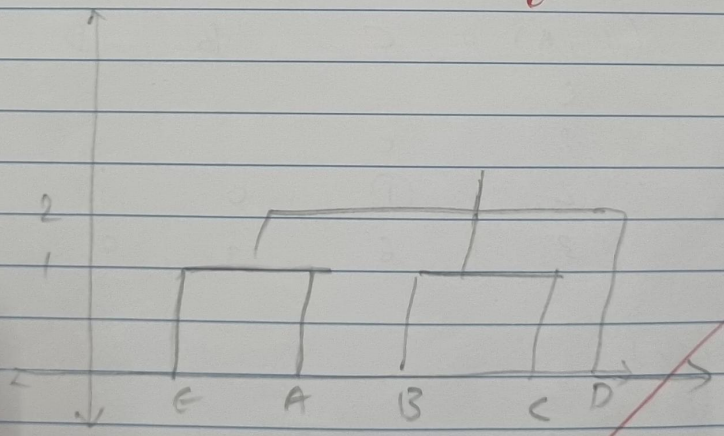
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Pair (E,A) & (B,C).

	((E,A), (B,C))	D
(E,A), (B,C)	0	
D	2	0

Pair (EA, JB, C), D.

	(E, A,B,C,D)
(E,A,B,C,D)	0



clarify