Experiment 1: Exploratory data analysis using Python

Aim: To understand the data through Exploratory data analysis

- Data cleaning- Missing Values,remove outliers
- Data transformation- Min-max normalization, Z-score normalization, Decimal Scaling
- Data Discretization- Binning
- Data analysis and Visualization

Theory:

Data Cleaning:

Data cleaning, also known as data cleansing or data scrubbing, is a crucial step in the data preparation process of any data analysis or machine learning project. It involves identifying and correcting errors, inconsistencies, and inaccuracies in datasets to ensure that the data is accurate, reliable, and ready for analysis.

Some common problems found in datasets are:

- Missing Values
- Outliers
- Inconsistent Formats, etc.

Some functions used for data cleaning:

- dropna()
- fillna()
- drop_duplicates()

Data transformation:

Data transformation refers to the process of converting or altering the raw data in a way that makes it more suitable for analysis, modeling, or visualization. It involves applying various mathematical, statistical, or logical operations to the data to achieve specific objectives, such as improving data quality, normalizing scales, handling outliers, or making the data conform to assumptions required by certain analytical methods.

Some of the techniques involved in data transformation are:

- Normalization
- Standardization
- Min-Max normalization
- Decimal Scaling

Data discretization:

Data discretization, also known as binning or discretization, is the process of converting continuous or numeric data into discrete bins or intervals. In other words, it involves dividing a continuous variable's range into smaller, non-overlapping intervals and assigning data points to the corresponding interval. This transformation is particularly useful when working with data analysis, visualization, or certain types of machine learning algorithms that benefit

from reduced data granularity or when data is naturally presented in grouped or categorized form.

Some functions that help us perform data discretization:

- pandas.qcut
- pandas.cut

Data visualisation:

Data visualization is the graphical representation of information and data. It involves using visual elements like charts, graphs, and maps to present complex data in a more accessible and understandable format. Data visualization is a powerful tool for exploring, analyzing, and communicating insights from data, making it an essential part of data analysis

The data can be presented in various forms via data visualisation for a better understanding of the data and some of those forms are:

- Pie-chart
- Histogram
- Scatterplot, etc.

Data analysis:

Data analysis is the process of inspecting, cleaning, transforming, and interpreting data to extract meaningful insights, discover patterns, and make informed decisions. It involves using various techniques, tools, and methodologies to understand the underlying structure of data, identify trends, relationships, and anomalies, and derive actionable information from the data.

Some functions used for data analysis are:

- .head()
- .info()
- .describe()

Steps:

1) **Load the libraries Download the data set from kaggle/ other sources**


```
In [2]: import pandas as pd
```

2) **Read the file –select appropriate file read function according to data type of file**

```
In [2]: import pandas as pd

        df = pd.read_csv("ufc_master_data.csv")
        df.head()
```

Out[2]:

	ranking	name	age	weight	gender	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests
0	C	Brandon Moreno	27	125	Male	67	8	2	2.0	NaN	19.0	5.0	2.0	NaN
1	1	Deiveson Figueiredo	33	125	Male	65	9	2	1.0	NaN	20.0	2.0	1.0	NaN
2	2	Askar Askarov	29	125	Male	65	3	0	1.0	NaN	14.0	0.0	1.0	NaN
3	3	Alexandre Pantoja	31	125	Male	65	8	3	NaN	NaN	24.0	5.0	NaN	NaN
4	4	Alex Perez	29	125	Male	64	6	2	NaN	NaN	24.0	6.0	NaN	NaN

3) Describe the attributes name, count no of values, and find min, max, data type, range, quartile, percentile, box plot and outliers.

Attribute names and data-type:

```
In [64]: attributes_info = df.dtypes

In [65]: attributes_info

Out[65]: ranking            object
         name               object
         age                 int64
         weight              int64
         gender             object
         height              int64
         ufc_wins            int64
         ufc_loses           int64
         ufc_draws         float64
         ufc_no_contests   float64
         mma_wins          float64
         mma_loses         float64
         mma_draws         float64
         mma_no_contests   float64
         dtype: object
```

No. of values:

```
In [67]: count_values

Out[67]: ranking           177
         name              617
         age               617
         weight            617
         gender            617
         height            617
         ufc_wins          617
         ufc_loses         617
         ufc_draws          44
         ufc_no_contests    43
         mma_wins          615
         mma_loses         615
         mma_draws          82
         mma_no_contests    68
         dtype: int64
```

Min and Max:

```
In [70]: max_val = numeric_col.max()

In [71]: max_val

Out[71]: age                 44.0
         weight             265.0
         height              79.0
         ufc_wins            29.0
         ufc_loses           18.0
         ufc_draws            2.0
         ufc_no_contests      2.0
         mma_wins            59.0
         mma_loses           21.0
         mma_draws            2.0
         mma_no_contests      2.0
         dtype: float64

In [72]: min_val = numeric_col.min()

In [73]: min_val

Out[73]: age                 21.0
         weight             115.0
         height              59.0
         ufc_wins             0.0
         ufc_loses            0.0
         ufc_draws            0.0
         ufc_no_contests      1.0
         mma_wins             1.0
         mma_loses            0.0
         mma_draws            0.0
         mma_no_contests      1.0
         dtype: float64
```

Range:

```
In [74]: range = max_val - min_val

In [75]: range

Out[75]: age                 23.0
         weight             150.0
         height              20.0
         ufc_wins            29.0
         ufc_loses           18.0
         ufc_draws            2.0
         ufc_no_contests      1.0
         mma_wins            58.0
         mma_loses           21.0
         mma_draws            2.0
         mma_no_contests      1.0
         dtype: float64
```

Quartile:

```
In [76]: quartiles = numeric_col.quantile([0.25, 0.50, 0.75,1.0])
```

```
In [77]: quartiles
```

Out[77]:

	age	weight	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests
0.25	28.0	135.0	67.0	1.0	1.0	1.0	1.0	10.0	2.0	1.0	1.0
0.50	31.0	145.0	69.0	3.0	2.0	1.0	1.0	14.0	4.0	1.0	1.0
0.75	34.0	170.0	72.0	6.0	4.0	1.0	1.0	18.0	6.0	1.0	1.0
1.00	44.0	265.0	79.0	29.0	18.0	2.0	2.0	59.0	21.0	2.0	2.0

Percentile:

```
In [79]: percentiles = numeric_col.quantile([0.1,0.9])
         percentiles
```

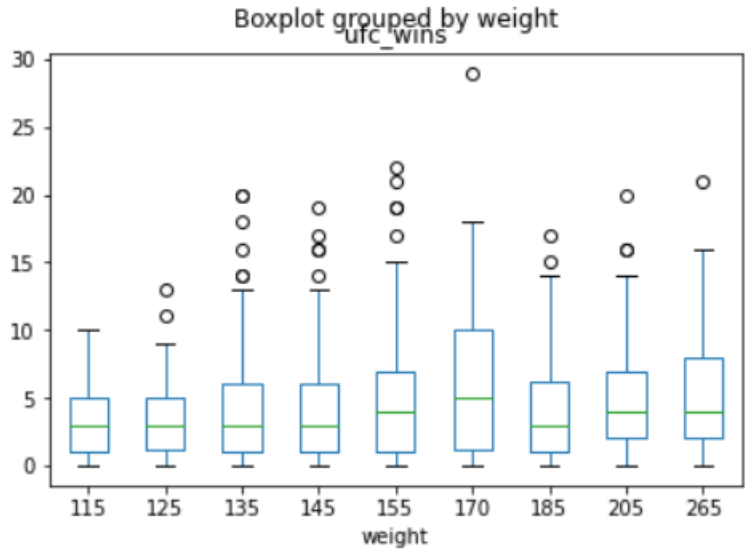
Out[79]:

	age	weight	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests
0.1	26.0	125.0	65.0	0.0	0.0	1.0	1.0	7.0	1.0	1.0	1.0
0.9	37.0	205.0	75.0	11.0	6.0	1.0	1.0	23.0	9.0	2.0	1.0

Boxplot and outliers:

```
In [80]: df.boxplot(by = 'weight', column = ['ufc_wins'],grid = False)
```

```
Out[80]: <AxesSubplot:title={'center':'ufc_wins'}, xlabel='weight'>
```



4) Perform cleaning,transformation , discretization and analysis

In [81]:
```
df.drop_duplicates(inplace = True)
df.fillna(df.mean(),inplace = True)
```

```
C:\Users\Komal\AppData\Local\Temp/ipykernel_7208/1661239573.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.fillna(df.mean(),inplace = True)
```

In [82]:
```
df.head()
```

Out[82]:

	ranking	name	age	weight	gender	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests
0	C	Brandon Moreno	27	125	Male	67	8	2	2.0	1.046512	19.0	5.0	2.000000	1.088235
1	1	Deiveson Figueiredo	33	125	Male	65	9	2	1.0	1.046512	20.0	2.0	1.000000	1.088235
2	2	Askar Askarov	29	125	Male	65	3	0	1.0	1.046512	14.0	0.0	1.000000	1.088235
3	3	Alexandre Pantoja	31	125	Male	65	8	3	1.0	1.046512	24.0	5.0	1.097561	1.088235
4	4	Alex Perez	29	125	Male	64	6	2	1.0	1.046512	24.0	6.0	1.097561	1.088235

In [83]:
```
df['gender_encoded'] = df['gender'].map({'Male':0,'Female':1})
df.drop('gender', axis=1, inplace=True)
df.head()
```

Out[83]:

king	name	age	weight	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests	gender_encoded
C	Brandon Moreno	27	125	67	8	2	2.0	1.046512	19.0	5.0	2.000000	1.088235	0
1	Deiveson Figueiredo	33	125	65	9	2	1.0	1.046512	20.0	2.0	1.000000	1.088235	0
2	Askar Askarov	29	125	65	3	0	1.0	1.046512	14.0	0.0	1.000000	1.088235	0
3	Alexandre Pantoja	31	125	65	8	3	1.0	1.046512	24.0	5.0	1.097561	1.088235	0
4	Alex Perez	29	125	64	6	2	1.0	1.046512	24.0	6.0	1.097561	1.088235	0

In [88]:
```
age_bins = [15,21,31,36,46,float('inf')]
age_labels = ['15-20','21-30','31-35','36-45','46+']
df['age_group'] = pd.cut(df['age'],bins=age_bins,labels=age_labels)
```

In [97]:
```
df.drop('age',axis=1,inplace=True)
df.head()
```

Out[97]:

ame	weight	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests	gender_encoded	age_group
don reno	-0.92852	-0.693834	0.730117	-0.331949	10.140677	0.0	0.661012	0.106242	6.165279	2.358119e-15	-0.486285	21-30
son redo	-0.92852	-1.242323	0.950402	-0.331949	0.000000	0.0	0.812136	-0.757267	-0.666517	2.358119e-15	-0.486285	31-35
skar arov	-0.92852	-1.242323	-0.371306	-1.068684	0.000000	0.0	-0.094606	-1.332940	-0.666517	2.358119e-15	-0.486285	21-30
ndre ntoja	-0.92852	-1.242323	0.730117	0.036419	0.000000	0.0	1.416630	0.106242	0.000000	2.358119e-15	-0.486285	21-30
Alex erez	-0.92852	-1.516567	0.289548	-0.331949	0.000000	0.0	1.416630	0.394078	0.000000	2.358119e-15	-0.486285	21-30

```
In [100]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 617 entries, 0 to 616
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ranking          177 non-null    object
 1   name             617 non-null    object
 2   weight           617 non-null    float64
 3   height           617 non-null    float64
 4   ufc_wins         617 non-null    float64
 5   ufc_loses        617 non-null    float64
 6   ufc_draws        617 non-null    float64
 7   ufc_no_contests  617 non-null    float64
 8   mma_wins         617 non-null    float64
 9   mma_loses        617 non-null    float64
 10  mma_draws        617 non-null    float64
 11  mma_no_contests  617 non-null    float64
 12  gender_encoded   617 non-null    float64
 13  age_group        617 non-null    category
dtypes: category(1), float64(11), object(2)
memory usage: 68.3+ KB
```

```
In [101]: df.describe()
```

Out[101]:

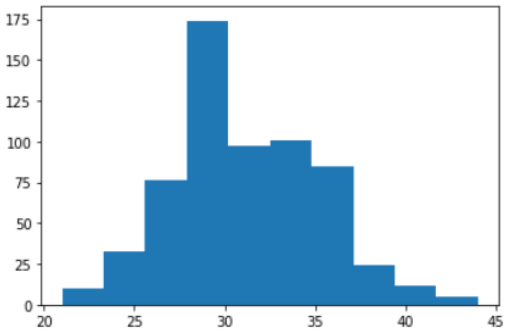
	weight	height	ufc_wins	ufc_loses	ufc_draws	ufc_no_contests	mma_wins	mma_loses	mma_draws	mma_no_contests	gen
count	6.170000e+02	6.170000e+02	617.000000	6.170000e+02	617.000000	6.170000e+02	6.170000e+02	617.000000	6.170000e+02	6.170000e+02	
mean	-4.606436e-17	4.606436e-17	0.000000	-1.151609e-17	0.000000	4.318534e-18	-2.303218e-17	0.000000	-1.151609e-17	4.030631e-17	
std	1.000811e+00	1.000811e+00	1.000811	1.000811e+00	1.000811	1.000811e+00	1.000811e+00	1.000811	1.000811e+00	1.000811e+00	
min	-1.195729e+00	-2.887789e+00	-1.032161	-1.068684e+00	-10.140677	-8.366261e-01	-2.059212e+00	-1.332940	-7.498312e+00	-9.370611e-01	
25%	-6.613105e-01	-6.938340e-01	-0.811876	-7.003166e-01	0.000000	1.036448e-16	-6.991001e-01	-0.757267	9.212872e-17	-7.753356e-17	
50%	-3.941012e-01	-1.453451e-01	-0.371306	-3.319489e-01	0.000000	1.036448e-16	-9.460581e-02	-0.181594	9.212872e-17	-7.753356e-17	
75%	2.739220e-01	6.773882e-01	0.289548	4.047866e-01	0.000000	1.036448e-16	5.098884e-01	0.394078	9.212872e-17	-7.753356e-17	
max	2.812410e+00	2.597099e+00	5.356096	5.561935e+00	10.140677	1.715084e+01	6.705955e+00	4.711624	6.165279e+00	9.682965e+00	

5) Give visualization of statistical description of data – in form of histogram, scatter plot, pie chart,Give correlation matrix

Histogram:

```
In [3]: plt.hist(df['age'])
```

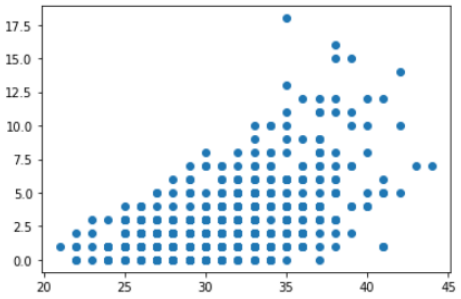
```
Out[3]: (array([ 10.,  33.,  76., 174.,  97., 101.,  85.,  24.,  12.,   5.]),
         array([21. , 23.3, 25.6, 27.9, 30.2, 32.5, 34.8, 37.1, 39.4, 41.7, 44. ]),
         <BarContainer object of 10 artists>)
```



Scatter plot:

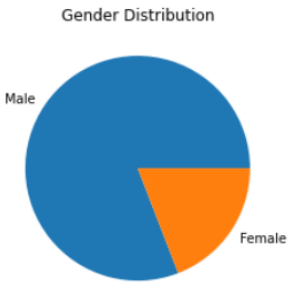
```
In [5]: plt.scatter(df['age'],df['ufc_loses'])
```

```
Out[5]: <matplotlib.collections.PathCollection at 0x1d0bab3b280>
```



Piechart:

```
In [7]: gender_counts = df['gender'].value_counts()

        plt.pie(gender_counts, labels=['Male','Female'])
        plt.title('Gender Distribution')
        plt.show()
```



Correlation matrix:

```
In [8]: two_df = df[['gender','age','weight']]
        correlation_matrix = two_df.corr()
        print(correlation_matrix)
```

```
                age    weight
age        1.000000  0.190549
weight     0.190549  1.000000
```

Frequency table:

In [8]:
```
test= df.groupby(['gender','weight'])
test.size()
```

Out[8]:
```
gender  weight
Female  115       43
        125       43
        135       26
        145        6
Male    125       31
        135       82
        145       78
        155       82
        170       74
        185       68
        205       42
        265       42
dtype: int64
```

EXPERIMENT 2

AIM: To perform a case study on building Data warehouse/Data Mart and establishing a problem statement for the same and designing a dimensional model (Star + Snowflake Schema).

PROBLEM STATEMENT:

In the context of an **airline reservation system**, the challenge lies in efficiently managing and analyzing vast amounts of data to optimize operational and strategic decision-making. The objective is to design and implement a data warehouse that can consolidate data from multiple sources into a central repository and deliver insights to improve passenger experience, revenue generation, and operational efficiency.

THEORY:

The Star Schema and Snowflake Schema are two common approaches to dimensional modeling in data warehousing, each with its unique characteristics and use cases.

The Star Schema is a simplified and denormalized structure that represents a central fact table connected to multiple dimension tables. In this schema, the fact table typically contains quantitative data (facts) that represent business metrics, such as sales revenue, quantities sold, or profit, and foreign keys that link to dimension tables. Dimension tables store descriptive attributes related to the facts, such as customer names, product categories, or time periods. This structure simplifies and accelerates query performance because it reduces the need for joins and allows for quick data retrieval. It's particularly useful for scenarios where fast and straightforward access to aggregated data is essential, making it well-suited for business intelligence and reporting applications.

The Snowflake Schema is a more normalized extension of the Star Schema. In this schema, dimension tables are further broken down into sub-dimensions, resulting in a structure resembling the branching of a snowflake. This approach reduces data redundancy by removing some attributes from dimension tables and storing them in separate related tables. While it maintains data integrity and conserves storage space, the Snowflake Schema can be more complex and may require additional joins when querying data, potentially impacting query performance. This schema is often chosen when there is a need for data consistency, data governance, or when dealing with data sources that evolve over time.

In summary, the Star Schema is a simpler and more denormalized structure, ideal for scenarios where query performance and simplicity are paramount. On the other hand, the Snowflake Schema is more normalized, which enhances data integrity and minimizes data redundancy, but may introduce complexity due to the increased number of joins. The choice between these two schemas depends on the specific needs of your data warehousing project and the trade-offs between query performance and data integrity.

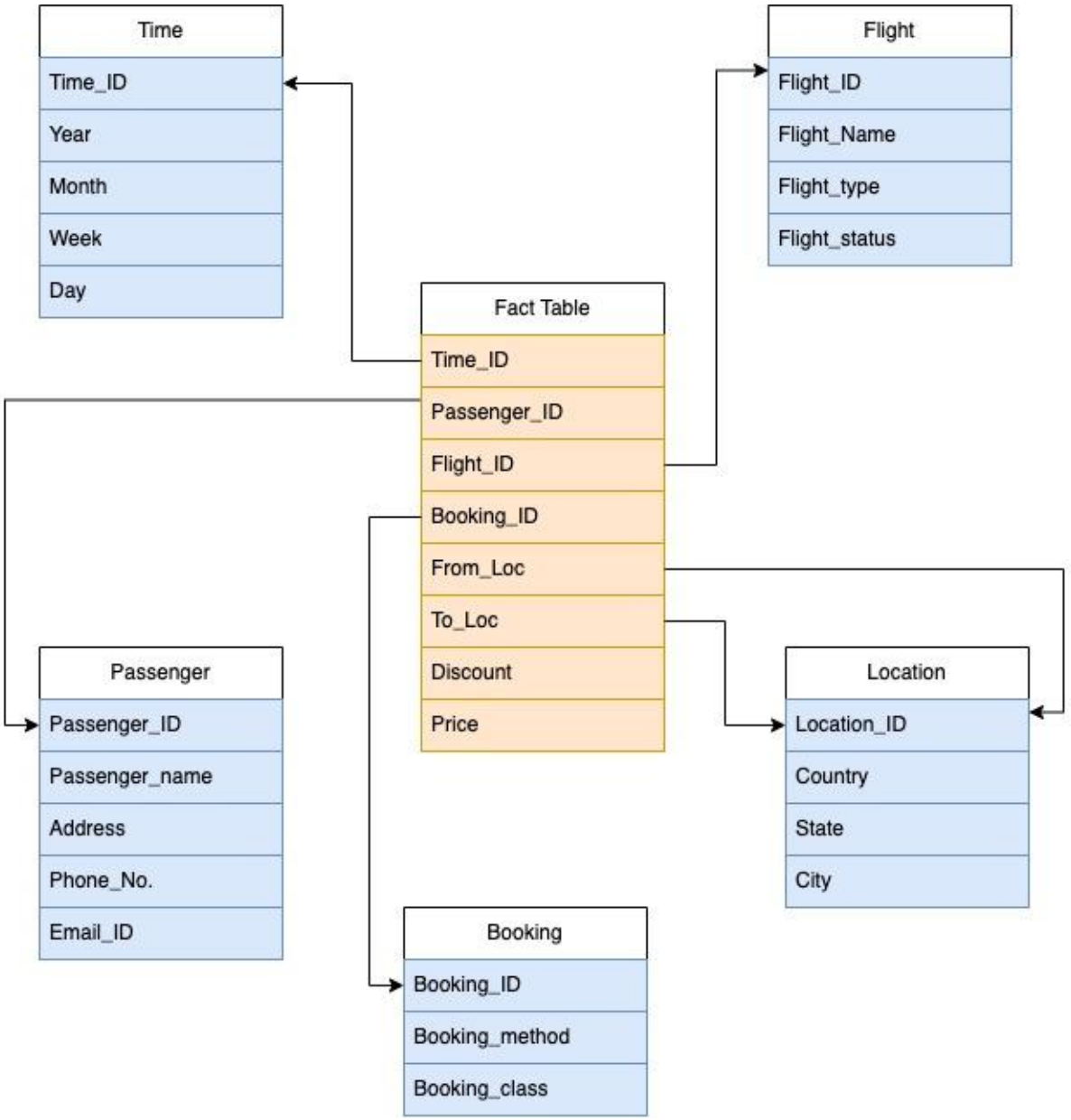
A Data Warehouse and a Data Mart are both integral components of an organization's data management and business intelligence strategy.

A Data Warehouse is a centralized repository that stores and manages a vast amount of historical and current data from various sources within an organization. Its primary function is to support complex data analysis, reporting, and decision-making processes. Data Warehouses are typically designed to be highly structured, with data organized into well-defined schemas, such as Star or Snowflake schemas. They are optimized for query performance and are used to generate reports, conduct in-depth data analysis, and provide a comprehensive view of an organization's data. Data Warehouses are particularly valuable for large enterprises that require a comprehensive and integrated view of their data to support strategic planning and business intelligence.

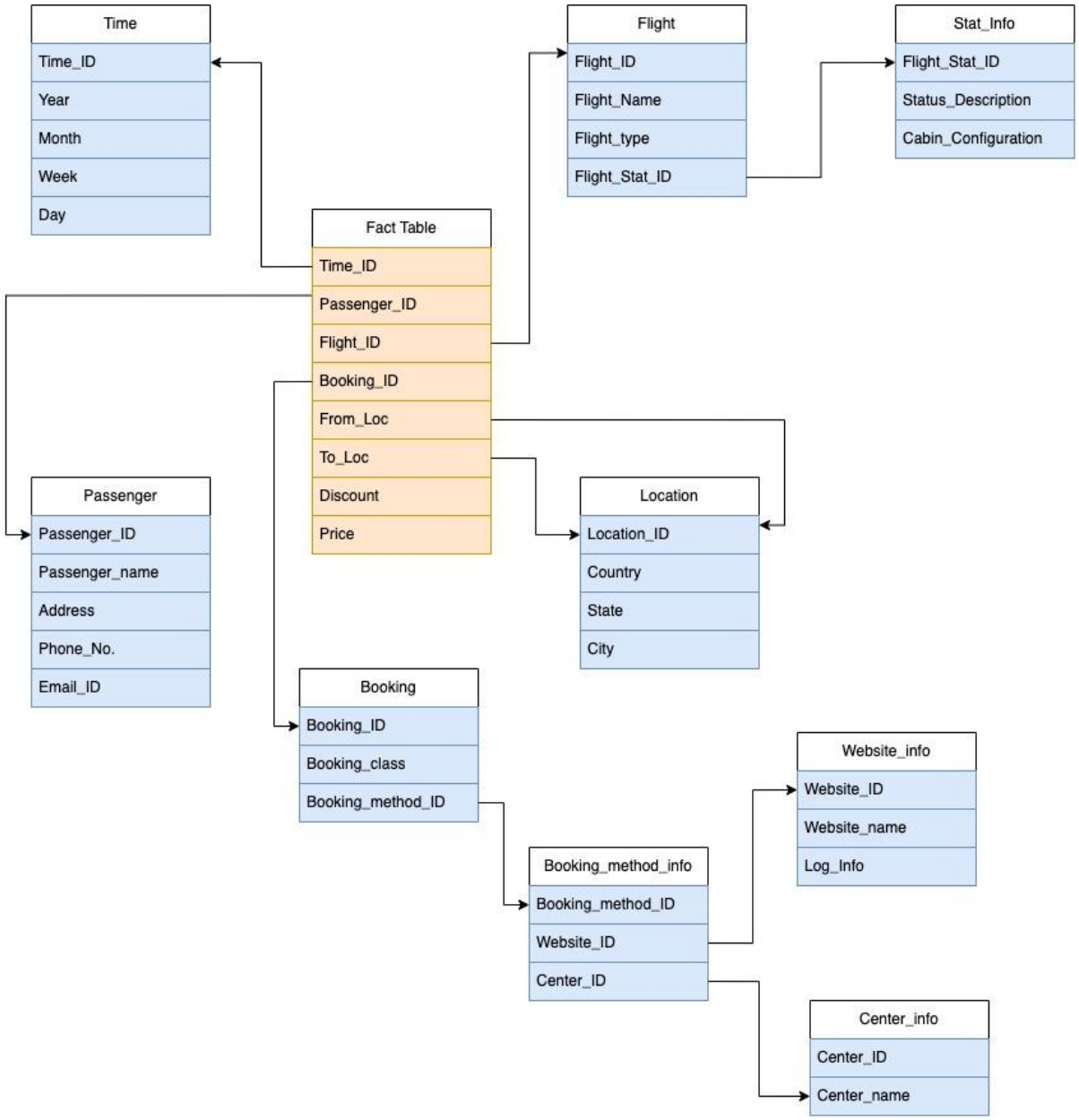
On the other hand, a Data Mart is a subset or specialized segment of a Data Warehouse. Data Marts are designed to meet the specific needs of a particular business unit or group within an organization. They contain a tailored subset of data from the Data Warehouse, focusing on a specific subject area, such as sales, marketing, finance, or human resources. Data Marts are often smaller, more agile, and designed for the unique requirements of individual departments, making it easier for those teams to access and analyze data specific to their functions. Data Marts are particularly useful for decentralizing data access and analysis, allowing business units to independently manage and utilize their data while still benefiting from the organization's overarching Data Warehouse.

In summary, a Data Warehouse is a comprehensive, centralized repository that stores a broad range of data for organization-wide reporting and analysis, while a Data Mart is a smaller, focused subset of the Data Warehouse, customized to meet the specific needs of individual business units or departments. Both play crucial roles in supporting data-driven decision-making and business intelligence within an organization.

STAR SCHEMA


Time
Time_ID
Year
Month
Week
Day
Flight
Flight_ID
Flight_Name
Flight_type
Flight_status
Fact Table
Time_ID
Passenger_ID
Flight_ID
Booking_ID
From_Loc
To_Loc
Discount
Price
Passenger
Passenger_ID
Passenger_name
Address
Phone_No.
Email_ID
Location
Location_ID
Country
State
City
Booking
Booking_ID
Booking_method
Booking_class

SNOWFLAKE SCHEMA:



CONCLUSION:

The developed data warehousing system for the airline reservation system offers a robust and flexible platform for consolidating and analyzing vast amounts of data. It employs a hybrid schema, combining elements of both the Star and Snowflake Schemas, ensuring efficient query performance and data integrity. The system comprises fact tables capturing critical booking and flight details, while dimension tables house descriptive attributes. These dimensions include information about flights, passengers, time, locations, booking classes, booking methods, and more, catering to a comprehensive set of data analysis and reporting needs. By offering a well-structured data environment, the system empowers the airline to enhance passenger experience, optimize revenue generation, and improve operational efficiency through data-driven decision-making.

EXPERIMENT 3

AIM: Implementation of all dimension table and fact table based on the experiment 2 case study.

THEORY:

In a data warehousing and business intelligence context, fact and dimension tables are fundamental components of the star schema or snowflake schema, two common data modelling techniques used for organising and structuring data.

A **fact table** represents the core data in a data warehouse, typically containing quantitative data known as measures. These measures are numerical values that provide insights into the business, such as sales revenue, profit, quantity sold, or any other metrics that organisations want to analyse. Fact tables are often large and contain historical data, making them the central focus of analytical queries. Each row in the fact table represents a specific event or transaction, and these events are associated with keys from dimension tables.

Dimension tables, on the other hand, provide context and descriptive information for the data stored in the fact table. They contain attributes or characteristics that help in categorising, filtering, and grouping data in the fact table. Common examples of dimension tables include customer, product, time, location, and various other entities. These tables are typically smaller than fact tables and serve as the entry points for drilling down into the detailed data.

Fact and dimension tables work in conjunction within a star schema or snowflake schema. The fact table references the dimension tables using keys to form relationships. These relationships facilitate querying and aggregating data for analytical purposes. By separating the quantitative data (in the fact table) from the descriptive attributes (in dimension tables), organisations can efficiently analyse and report on their data. This separation also simplifies the maintenance and scalability of the data warehouse, as dimension tables can often be reused across multiple fact tables, and new measures can be added without altering the dimension structure. In summary, fact and dimension tables are essential components of data warehousing and play a critical role in enabling organisations to make informed, data-driven decisions by providing a structured and efficient way to store and analyse data.

CODE:

```
CREATE TABLE PASSENGER(
p_id int PRIMARY KEY,
p_name varchar(50) not null,
address varchar(50),
phone_no int not null,
email_id varchar(50) not null);

DESC Passenger;

CREATE TABLE BOOKING(
    booking_id int primary key,
    booking_method varchar(50) not null,
    booking_class varchar(50) not null
);

DESC BOOKING;

CREATE TABLE LOCATION(
location_key varchar(50) primary key,
country varchar(50) not null,
state varchar(50) not null,
city varchar(50) not null
);

DESC LOCATION;

CREATE TABLE FLIGHT(
flight_id int PRIMARY KEY,
flight_name varchar(50) not null,
flight_type varchar(50) not null,
flight_status varchar(50) not null
);

DESC passenger;

CREATE TABLE TIME(
time_id int PRIMARY KEY,
day date not null,
month varchar(20) not null,
week int not null,
year varchar(50) not null);
```

```
CREATE TABLE FACT( time_id int references TIME(time_id) NOT NULL,

flight_id int references FLIGHT(flight_id)NOT NULL,

p_id int references PASSENGER(p_id)NOT NULL,

booking_id int references BOOKING(booking_id)NOT NULL,

from_loc varchar(50) references LOCATION(location_key)NOT NULL,
to_loc varchar(50) references LOCATION(location_key) NOT NULL,

price int not null,

discount int not null

);

desc fact;

INSERT INTO PASSENGER VALUES(121121,'Chaitanya
Kakade','Atlanta
Height,Prabhadevi',9619032253,'kakadechaitanya77@gmail.com'); select
* from passenger;

    INSERT INTO PASSENGER VALUES(128361,'Aman Karlupia','Link
    Road,Bandra',8899455763,'AmanKarlupia11@gmail.com');

    INSERT INTO PASSENGER VALUES(898546,'Rishi Kewalya','Mira
    Road',9137171684,'rishikewalya2004@gmail.com');

        INSERT INTO PASSENGER VALUES(459875,'Ritesh
        Katare','Solapur',8180991488,'rkatare247@gmail.com');

        INSERT INTO PASSENGER VALUES(857896,'Raj
        Upasane','Akola',9874563212,'krka@gmail.com');

INSERT INTO LOCATION VALUES('A114','India','Maharashtra','Mumbai');
INSERT INTO LOCATION VALUES('A115','India','West Bengal','Kolkata');
INSERT INTO LOCATION VALUES('A116','India','Maharashtra','Pune');
INSERT INTO LOCATION VALUES('A117','India','Karnataka','Bangalore');
INSERT INTO LOCATION VALUES('A118','India','Tamil Nadu','Chennai');
        select * from location;

INSERT INTO BOOKING VALUES(5699875,'MakeMyTrip','Economy');
INSERT INTO BOOKING VALUES(3654535,'Ixigo','Economy-Premium');
INSERT INTO BOOKING VALUES(5658745,'Goibibo','Business');
INSERT INTO BOOKING VALUES(4566328,'AirIndia','Economy');
INSERT INTO BOOKING VALUES(5699873,'MakeMyTrip','Economy');
        select * from booking;

INSERT INTO flight VALUES(123,'Air India','Domestic','On time'); INSERT
        INTO flight VALUES(133,'Go Airs','Domestic','Delayed');
```

INSERT INTO flight VALUES(223,'Indigo','Domestic','On time');INSERT INTO flight VALUES(423,'Spice Jet','Domestic','On time'); INSERT INTO flight VALUES(23665,'Emirates','International','On time');
select * from flight; desc time;

INSERT INTO time VALUES(1,'01-JAN-2023','January',1,2023);
INSERT INTO time VALUES(2,'02-JAN-2023','January',1,2023);
INSERT INTO time VALUES(3,'03-JAN-2023','January',1,2023);
INSERT INTO time VALUES(4,'10-MAR-2023','March',2,2023);
INSERT INTO time VALUES(5,'05-JAN-2023','January',1,2023);
select * from time; select * from fact;

desc fact;

INSERT INTO fact VALUES(1, 123, 121121, 5699875, 'A114', 'A115', 5000, 200);
INSERT INTO fact VALUES(2, 133, 128361, 3654535, 'A115', 'A116', 6000, 250);
INSERT INTO fact VALUES(3, 223, 898546, 5658745, 'A116', 'A117', 8000, 300);
INSERT INTO fact VALUES(4, 423, 459875, 4566328, 'A114', 'A117', 7000, 280);
INSERT INTO fact VALUES(5, 23665, 857896, 5699873, 'A117', 'A118', 10000, 400);

select * from time; select * from fact;

select * from location;

select * from passenger;

select * from booking;

desc time; desc fact;

desc location;

desc passenger;

desc booking;

SCREEN SHOTS(Table Description)

TABLE TIME

Column	Null?	Type
TIME_ID	NOT NULL	NUMBER
DAY	NOT NULL	DATE
MONTH	NOT NULL	VARCHAR2(20)
WEEK	NOT NULL	NUMBER
YEAR	NOT NULL	VARCHAR2(50)

TABLE FACT

Column	Null?	Type
TIME_ID	NOT NULL	NUMBER
FLIGHT_ID	NOT NULL	NUMBER
P_ID	NOT NULL	NUMBER
BOOKING_ID	NOT NULL	NUMBER
FROM_LOC	NOT NULL	VARCHAR2(50)
TO_LOC	NOT NULL	VARCHAR2(50)
PRICE	NOT NULL	NUMBER
DISCOUNT	NOT NULL	NUMBER

TABLE BOOKING

Column	Null?	Type
BOOKING_ID	NOT NULL	NUMBER
BOOKING_METHOD	NOT NULL	VARCHAR2(50)
BOOKING_CLASS	NOT NULL	VARCHAR2(50)

TABLE LOCATION

Column	Null?	Type
LOCATION_KEY	NOT NULL	VARCHAR2(50)
COUNTRY	NOT NULL	VARCHAR2(50)
STATE	NOT NULL	VARCHAR2(50)
CITY	NOT NULL	VARCHAR2(50)

TABLE PASSENGER

Column	Null?	Type
P_ID	NOT NULL	NUMBER
P_NAME	NOT NULL	VARCHAR2(50)
ADDRESS	-	VARCHAR2(50)
PHONE_NO	NOT NULL	NUMBER
EMAIL_ID	NOT NULL	VARCHAR2(50)

SCREEN SHOTS(table values)

TIME:

TIME_ID	DAY	MONTH	WEEK	YEAR
1	01-JAN-23	January	1	2023
3	03-JAN-23	January	1	2023
4	10-MAR-23	March	2	2023
5	05-JAN-23	January	1	2023
2	02-JAN-23	January	1	2023

Passenger:

P_ID	P_NAME	ADDRESS	PHONE_NO	EMAIL_ID
121121	Chaitanya Kakade	Atlanta Height,Prabhadevi	9619032253	kakadechaitanya77@gmail.com
128361	Aman Karlupia	Link Road,Bandra	8899455763	AmanKarlupia11@gmail.com
898546	Rishi Kewalya	Mira Road	9137171684	rishikewalya2004@gmail.com
459875	Ritesh Katare	Solapur	8180991488	rkatare247@gmail.com
857896	Raj Upasane	Akola	9874563212	krka@gmail.com

Location:

LOCATION_KEY	COUNTRY	STATE	CITY
A114	India	Maharashtra	Mumbai
A115	India	West Bengal	Kolkata
A116	India	Maharashtra	Pune
A117	India	Karnataka	Bangalore
A118	India	Tamil Nadu	Chennai

Booking:

BOOKING_ID	BOOKING_METHOD	BOOKING_CLASS
5699875	MakeMyTrip	Economy
3654535	Ixigo	Economy-Premium
5658745	Goibibo	Business
5699873	MakeMyTrip	Economy
4566328	AirIndia	Economy

Fact Table:

TIME_ID	FLIGHT_ID	P_ID	BOOKING_ID	FROM_LOC	TO_LOC	PRICE	DISCOUNT
2	133	128361	3654535	A115	A116	6000	250
3	223	898546	5658745	A116	A117	8000	300
1	123	121121	5699875	A114	A115	5000	200
4	423	459875	4566328	A114	A117	7000	280
5	23665	857896	5699873	A117	A118	10000	400

CONCLUSION:

I was able to successfully create the Dimensional Model for the Airline Reservation System.

EXPERIMENT NO. 4

AIM: Implementation of OLAP operations: SLICE,DICE,ROLLUP,DRILLDOWN and PIVOT TABLE operations based on the case study.

THEORY:

1. **Roll-up (Drill-Up):** Roll-up is an operation that involves aggregating data from a lower level of detail to a higher one within a dimensional hierarchy. It's particularly useful for summarizing data for higher-level analysis. For example, in a time dimension, you might roll up daily sales data to monthly or yearly sales. This operation simplifies data exploration, aids in identifying long-term trends, and facilitates high-level decision-making.

2. **Drill-down (Roll-down):** Drill-down is the reverse of roll-up, allowing you to explore data in greater detail. You move from a higher level of aggregation to a lower level within a dimension. This operation is valuable for investigating specific aspects of the data. For instance, drilling down from yearly sales to quarterly or monthly sales provides insights into shorter-term patterns and helps pinpoint specific issues or opportunities.

3. **Slice:** Slicing involves selecting a specific "slice" or subset of the multidimensional data cube. You fix one or more dimensions to view data from a specific perspective. For example, you can slice data to focus solely on a particular region, product category, or time period. Slicing simplifies data exploration within specific contexts.

4. **Dice:** Dicing is a more advanced operation where you create a subcube by selecting specific values across multiple dimensions. This operation allows you to analyze a well-defined subset of the data. For instance, dicing might involve examining sales data for particular products, regions, and time periods, providing a very detailed view of a specific scenario.

5. **Pivot (Rotate):** Pivoting data involves reorienting the data cube to view it from different dimensions or axes. It's akin to changing the layout of data, effectively switching rows and columns. This operation is helpful when you

want to explore data in different contexts or when you need to compare dimensions in new ways.

6. **Ranking:** Ranking is the process of assigning a rank or order to dimension members based on a specific measure. For example, you might rank products by sales revenue. Ranking is used to identify top performers, underperformers, or to prioritize items based on a specific criterion.

7. **Top-N Analysis:** Top-N analysis helps in selecting the top or bottom "N" items from a dimension based on a measure. This operation is commonly used to identify, for instance, the top 10 customers with the highest purchase amounts or the lowest-performing products.

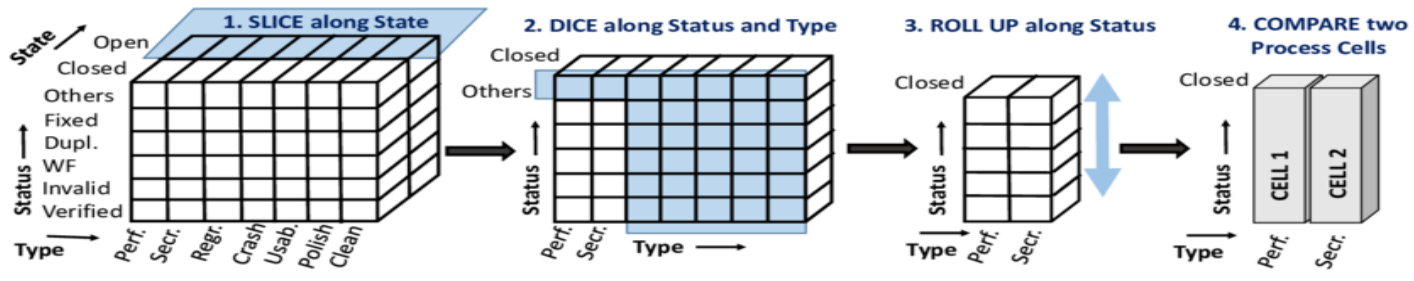
8. **Cross-tabulation (Pivot Table):** Cross-tabulation, also known as pivot tables, is used to create summary tables that display the relationship between two dimensions. It's particularly valuable for comparing data along multiple dimensions, such as examining sales by product category and region simultaneously.

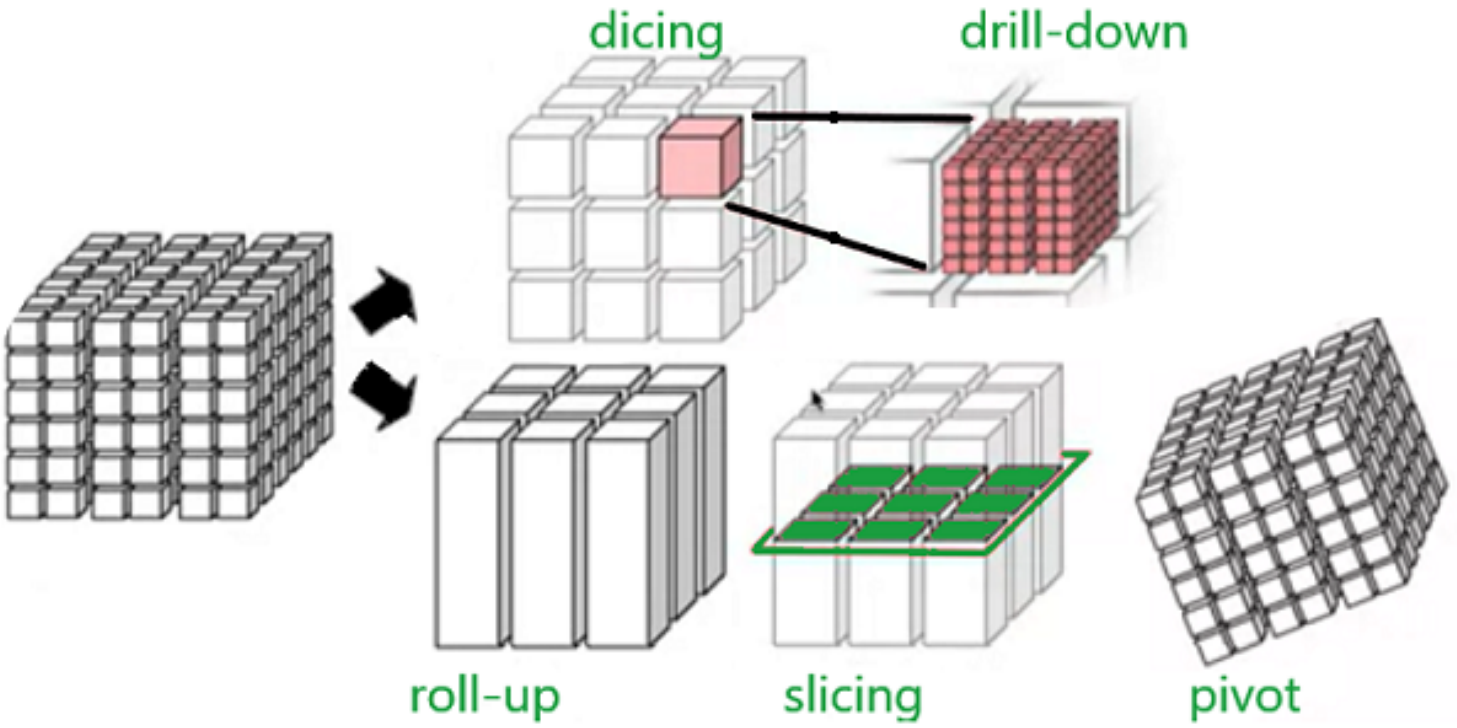
9. **Time Series Analysis:** Time series analysis is a specialized OLAP operation for examining data trends over time. It's used to understand seasonal variations, cycles, and patterns in data. This operation helps in forecasting and decision-making based on historical data.

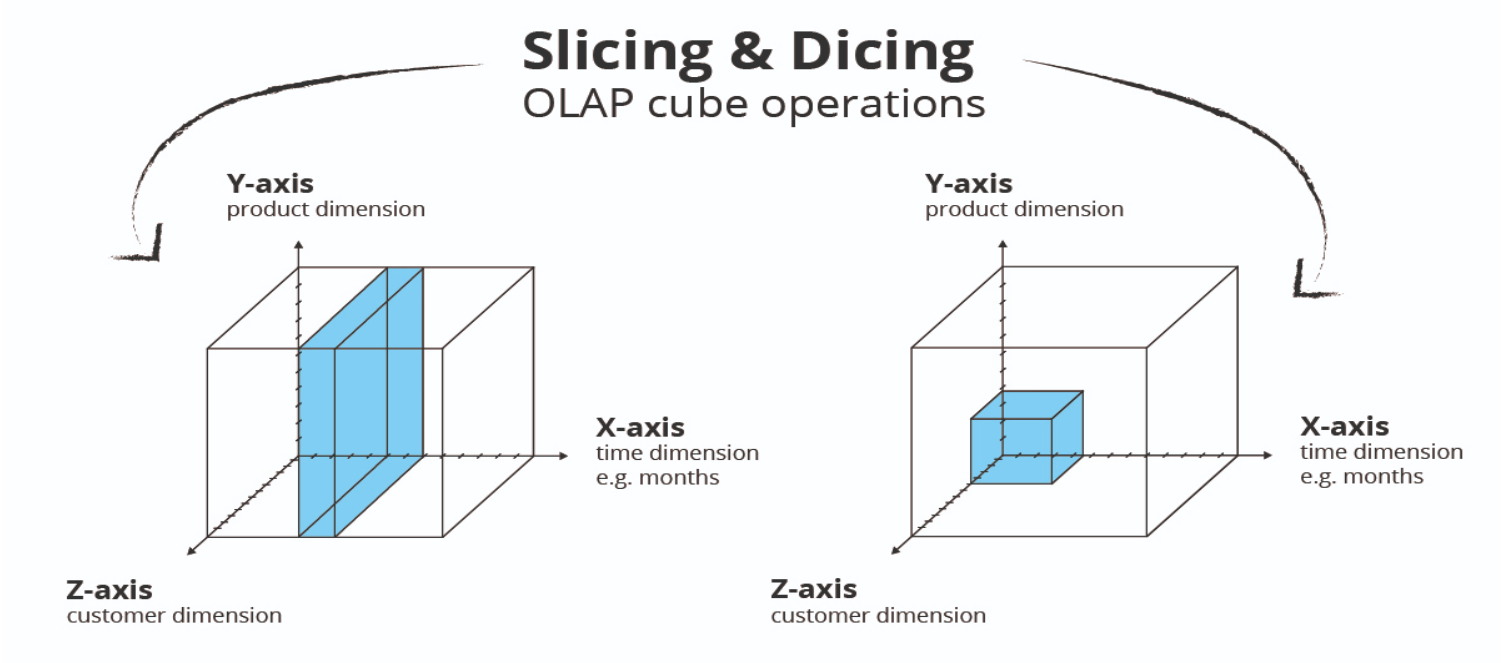
10. **Aggregate (Consolidation):** Aggregation is the process of summarizing data to a higher level of detail. It reduces the data's granularity, making it more manageable for analysis or reporting. For instance, you might aggregate daily sales data to get total monthly sales.

These OLAP operations are critical for exploring, analyzing, and gaining insights from multidimensional data. They provide powerful tools for decision-makers and analysts to navigate complex data structures efficiently and extract valuable information. OLAP operations are widely used in data warehousing, business intelligence, and data analysis to support data-driven decision-making.

vner	Person who manages issue	D	Clustering	Long Term, Short Term Contributors: Duration and productivity based [12]
	Event Attributes : Different for each event of a case			
tivity	Task performed	ND	–	Activities captured for issue resolution progression [3]
mestamp	Time when activity is performed	ND	–	Datetime
tor	Person who performs activity	ND	–	People participating in resolution process




dicing
drill-down
roll-up
slicing
pivot


Slicing & Dicing
OLAP cube operations
Y-axis
product dimension
X-axis
time dimension
e.g. months
Z-axis
customer dimension
Y-axis
product dimension
X-axis
time dimension
e.g. months
Z-axis
customer dimension

CODE:

```
select * from Fact;

select * from Location; select * from passenger;

select * from fact;

select state from Location ;

desc location;

select price from Location l,Fact f where f.from_loc = l.location_key;

insert into location values('B114','USA','Texas','LA'); insert into location
values('B123','Japan','Null','Tokyo');
  select l.country,l.state,l.city,sum(f.price) from fact f natural join location l
  group by rollup(l.country,l.state,l.city);

select flight_id,P_id,sum(price) from fact group by rollup(flight_id,P_id);

-- Example of a simple pivot SELECT

MAX(CASE WHEN year = 2023 THEN price END) AS year_2023_sales,
MAX(CASE WHEN year = 2022 THEN price END) AS year_2022_sales

FROM FACT JOIN TIME ON FACT.time_id = TIME.time_id;

-- Drill Down SELECT year, month, SUM(price) as quarterly_sales

FROM FACT

JOIN TIME ON FACT.time_id = TIME.time_id

GROUP BY year, month;

-- slice SELECT *

FROM FACT

WHERE from_loc = 'A114';

-- Dice

SELECT *
```

FROM FACT WHERE from_loc = 'A114' AND to_loc = 'A115' AND flight_id = 123;

-- consolidation SELECT year, SUM(price) as yearly_sales

FROM FACT

JOIN TIME ON FACT.time_id = TIME.time_id

GROUP BY year;

OUTPUT:

COUNTRY	STATE	CITY	SUM(F.PRICE)
USA	Texas	LA	36000
USA	Texas	-	36000
USA	-	-	36000
India	Karnataka	Bangalore	36000
India	Karnataka	-	36000
India	Tamil Nadu	Chennai	36000
India	Tamil Nadu	-	36000
India	Maharashtra	Pune	36000
India	Maharashtra	Mumbai	36000
India	Maharashtra	-	72000
India	West Bengal	Kolkata	36000
India	Maharashtra	-	72000
India	West Bengal	Kolkata	36000
India	West Bengal	-	36000
India	-	-	180000
Japan	Null	Tokyo	36000
Japan	Null	-	36000
Japan	-	-	36000
-	-	-	252000

FLIGHT_ID	P_ID	SUM(PRICE)
123	121121	5000
123	–	5000
133	128361	6000
133	–	6000
223	898546	8000
223	–	8000
423	459875	7000
423	–	7000
23665	857896	10000
23665	–	10000

YEAR_2023_SALES	YEAR_2022_SALES
10000	–

Download CSV

YEAR	MONTH	QUARTERLY_SALES
2023	March	7000
2023	January	29000

Download CSV

TIME_ID	FLIGHT_ID	P_ID	BOOKING_ID	FROM_LOC	TO_LOC	PRICE	DISCOUNT
1	123	121121	5699875	A114	A115	5000	200
4	423	459875	4566328	A114	A117	7000	280

Download CSV

2 rows selected.

TIME_ID	FLIGHT_ID	P_ID	BOOKING_ID	FROM_LOC	TO_LOC	PRICE	DISCOUNT
1	123	121121	5699875	A114	A115	5000	200

Download CSV

YEAR	YEARLY_SALES
2023	36000

Download CSV

To analyse the data using Excel Pivot Table.

Theory:

A PivotTable is an interactive way to quickly summarize large amounts of data. You can use a PivotTable to analyze numerical data in detail, and answer unanticipated questions about your data.
A pivot table is a powerful data analysis tool that allows you to summarize and analyze data from a dataset in a structured way. In this case, i have created a pivot table using the given dataset, with the "to_loc" column as the columns in the pivot table and the "from_loc" column as the rows. The values displayed in the pivot table represent the "Price" for each combination of "from_loc" and "to_loc."

Analysis:

Flight Prices Comparison: The pivot table allows you to compare flight prices from different "from_loc" to different "to_loc" easily. You can quickly identify which routes have higher or lower prices and potentially investigate the reasons behind these variations.

Popular Routes: The pivot table might reveal the most popular routes based on the number of flights booked and their respective prices. This information can be used to focus marketing efforts and improve services on those routes.

Seasonal Trends: If you have data spanning over a specific period, the pivot table could help identify seasonal price variations. You may notice price fluctuations based on factors like holidays, events, or peak travel times.

Revenue Analysis: By summing up the prices for each "from_loc" and "to_loc" combination, you can determine the revenue generated from different routes. This information can be valuable for business decision-making and resource allocation.

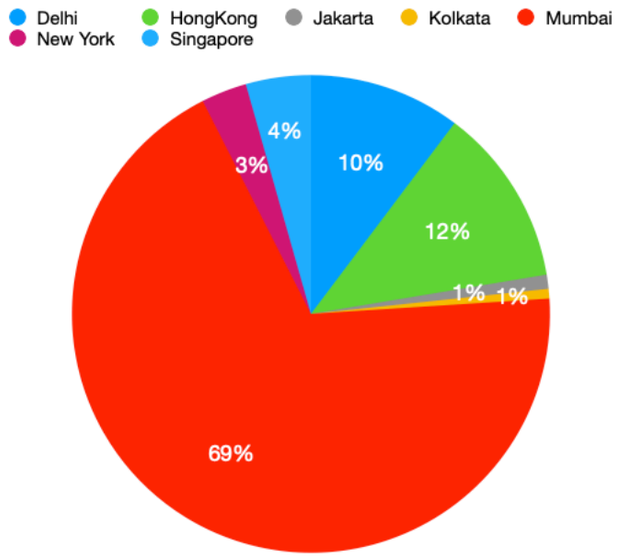
Price Outliers: The pivot table can also highlight any unusual or extreme prices for specific routes. These outliers may indicate potential data issues or special circumstances that need further investigation.

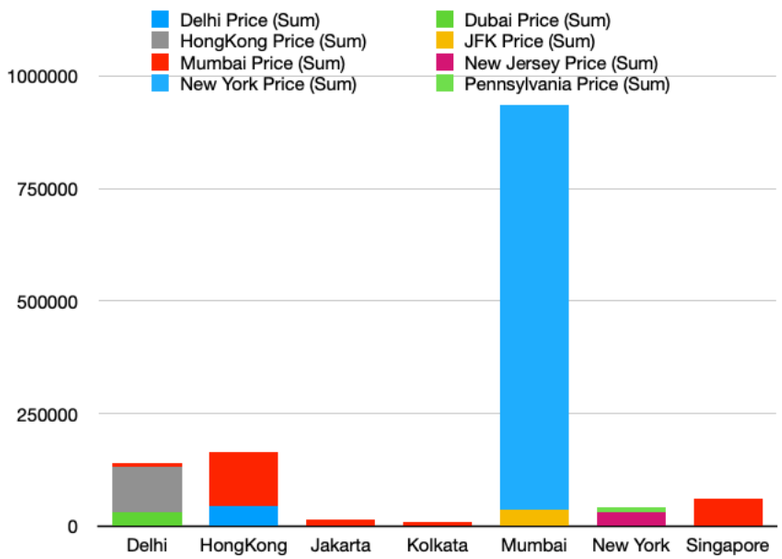
Fact Table

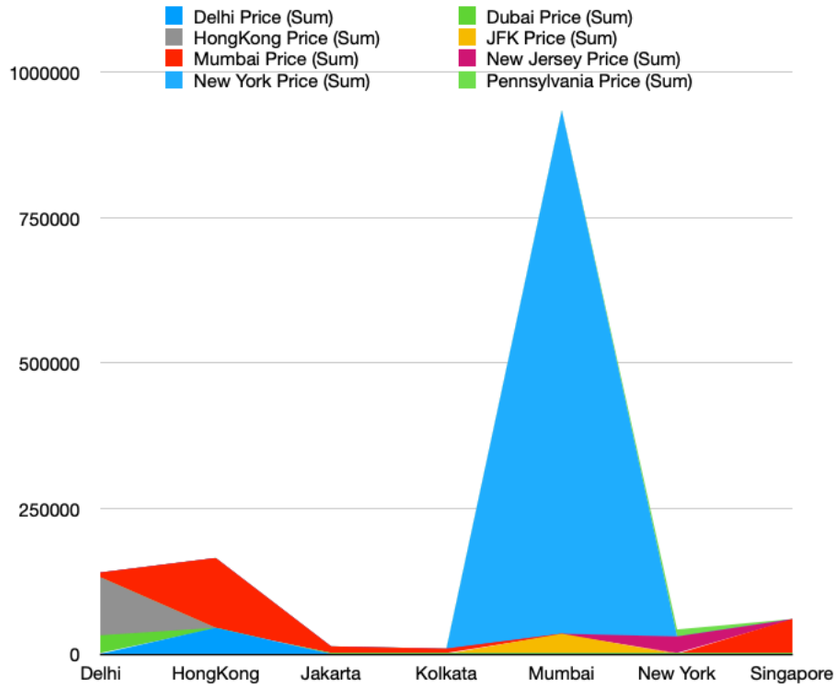
	Time_ID	Flight_ID	Passenger_ID	Booking_ID	Price	from_loc	to_loc
	1	F100	A112	121190	32000	Delhi	Dubai
	2	G101	B113	454422	12000	New York	Pennsylvania
	3	H102	C114	676721	8000	Delhi	Mumbai
	4	I103	D115	128973	59999	Singapore	Mumbai
	5	J104	E116	883134	900000	Mumbai	New York
	6	K105	F117	398673	100000	Delhi	HongKong
	7	L106	G118	908778	120000	HongKong	Mumbai
	8	M107	H119	454667	30000	New York	New Jersey
	9	N108	I120	325643	45000	HongKong	Delhi
	10	O109	J121	347765	34787	Mumbai	JFK
	11	P110	K122	112233	12999	Jakarta	Mumbai
	12	Q111	L123	332244	8999	Kolkata	Mumbai

Table 1 Pivot

to_loc	Delhi	Dubai	HongKong	JFK	Mumbai	New Jersey	New York	Pennsylvania	Grand Total
from_loc	**Price (Sum)**								
Delhi		32000	100000		8000				**140000**
HongKong	45000				120000				**165000**
Jakarta					12999				**12999**
Kolkata					8999				**8999**
Mumbai				34787			900000		**934787**
New York						30000		12000	**42000**
Singapore					59999				**59999**
Grand Total	**45000**	**32000**	**100000**	**34787**	**209997**	**30000**	**900000**	**12000**	**1363784**


Delhi
HongKong
Jakarta
Kolkata
Mumbai
New York
Singapore
4%
10%
3%
12%
1%
1%
69%


Delhi Price (Sum)
Dubai Price (Sum)
HongKong Price (Sum)
JFK Price (Sum)
Mumbai Price (Sum)
New Jersey Price (Sum)
New York Price (Sum)
Pennsylvania Price (Sum)
1000000
750000
500000
250000
0
Delhi
HongKong
Jakarta
Kolkata
Mumbai
New York
Singapore


Delhi Price (Sum)
Dubai Price (Sum)
HongKong Price (Sum)
JFK Price (Sum)
Mumbai Price (Sum)
New Jersey Price (Sum)
New York Price (Sum)
Pennsylvania Price (Sum)
1000000
750000
500000
250000
0
Delhi
HongKong
Jakarta
Kolkata
Mumbai
New York
Singapore

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 simplest 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:

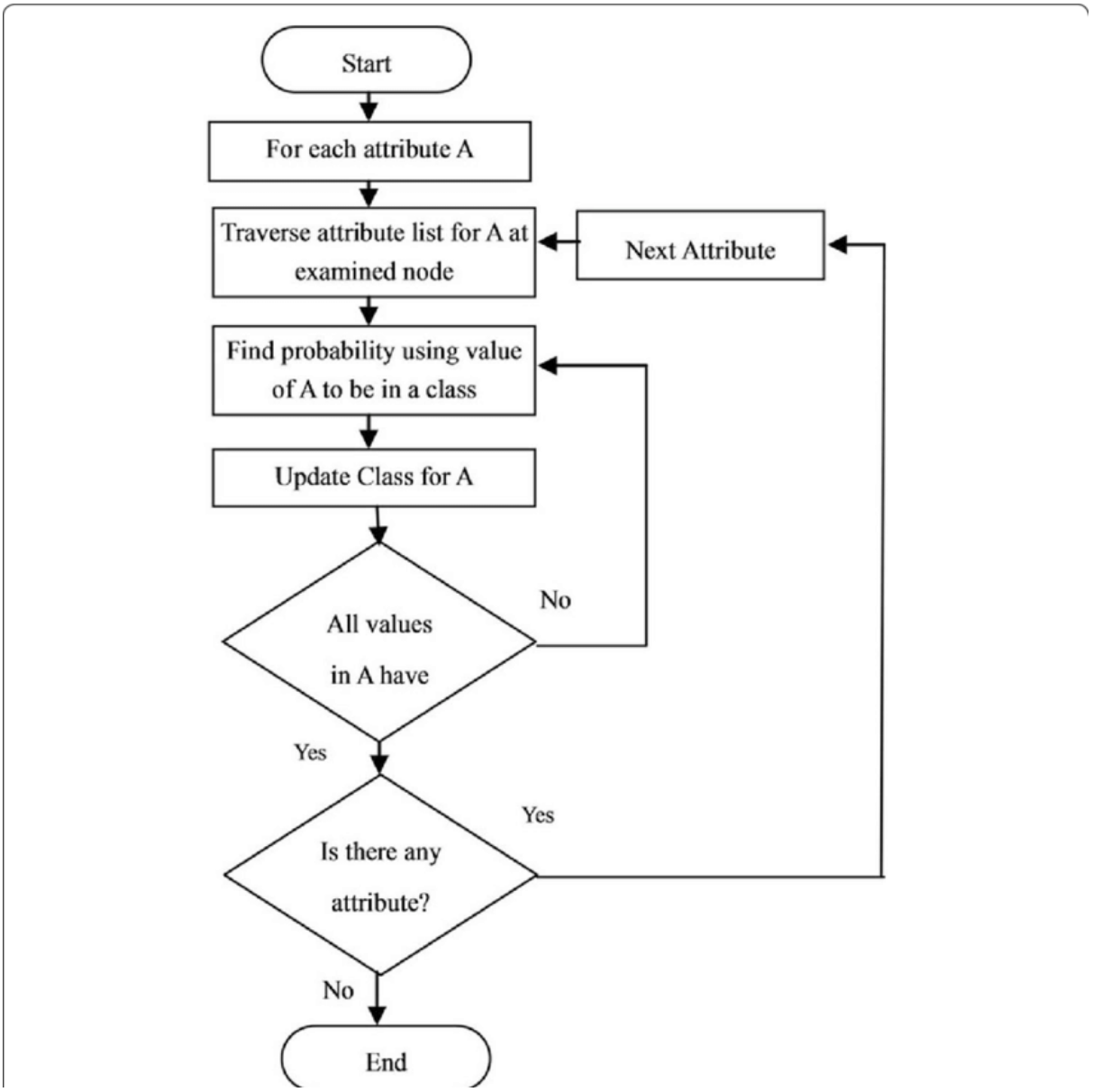


Fig 3.1 Flowchart of Naïve Bayes

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
```

Type

P(SUV | Yes) = 1/5
P(SPORTS | Yes) = 4/5

P(SUV | No) = 3/5
P(SPORTS | No) = 2/5

Origin

P(Domestic | Yes)
P(Imported | Yes)

P(Domestic | No) = 3/5
P(Imported | No) = 2/5

An unseen example X = <Red, Domestic, SUV>

$$P(X|Yes) \cdot P(Yes) = P(Red|Yes) \cdot P(Domestic|Yes) \cdot P(SUV|Yes) \cdot P(Yes)$$
$$= 3/5 \times 2/5 \times 1/5 \times 5/10$$
$$= 0.024$$

$$P(X|No) \cdot P(No) = P(Red|No) \cdot P(Domestic|No) \cdot P(SUV|No) \cdot P(No)$$
$$= 2/5 \times 3/5 \times 5/10 \times 3/5$$
$$= 0.072$$

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

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.

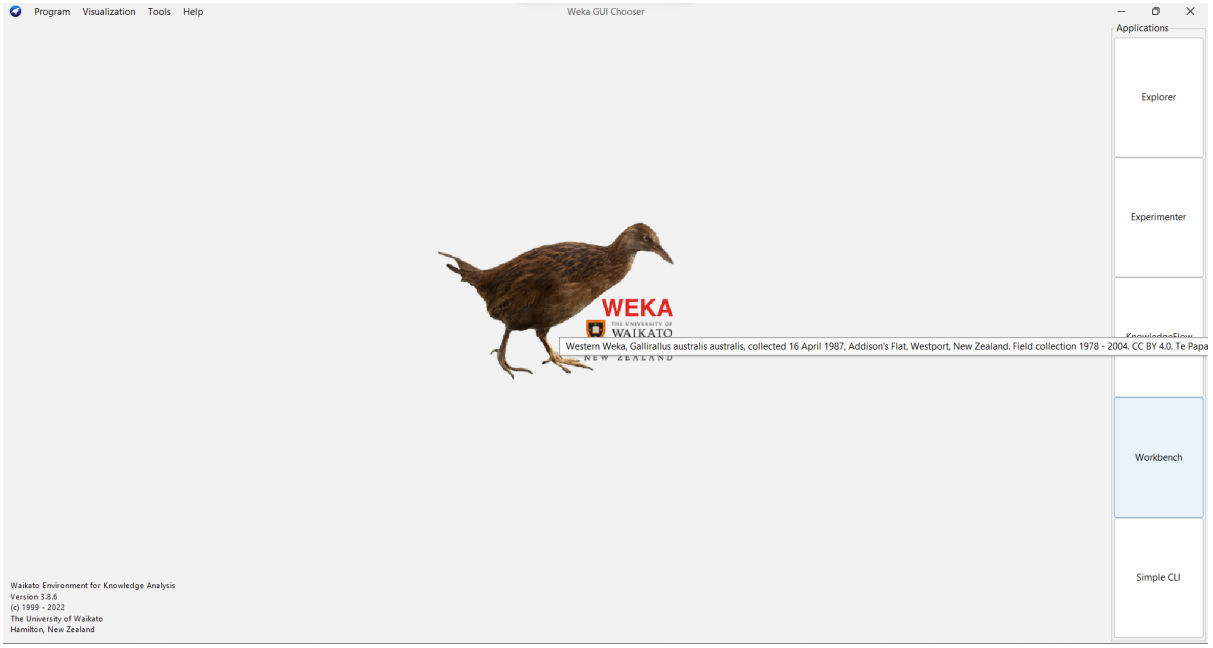


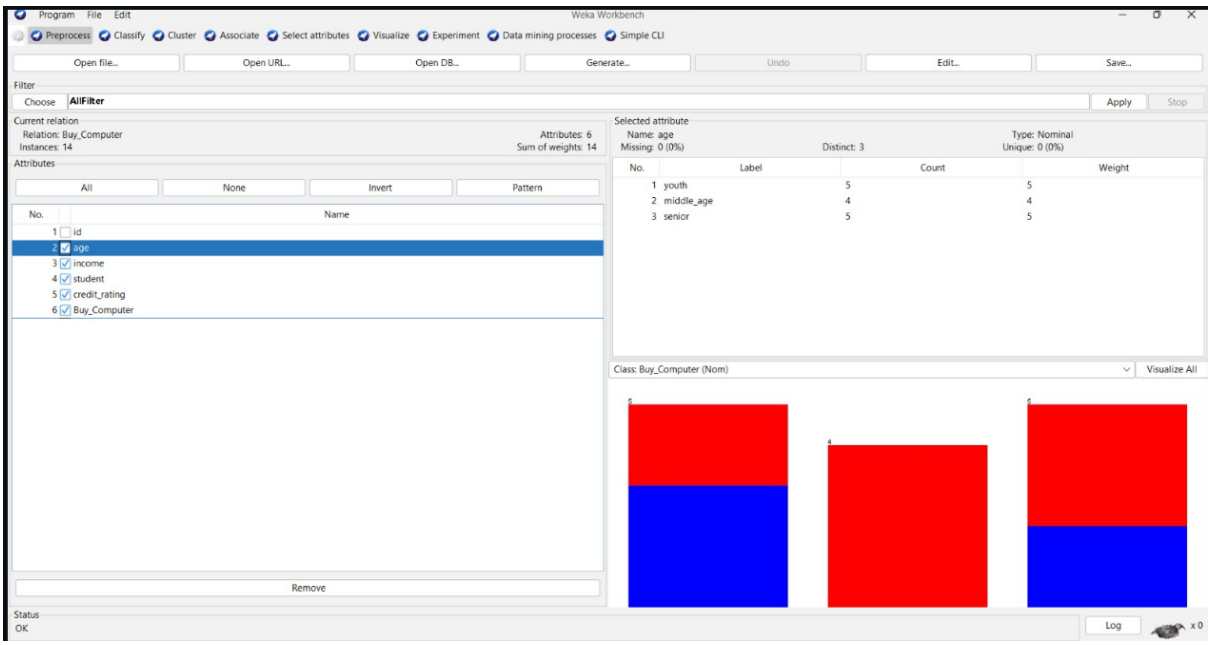
Fig 6.1 Weka Launching Page

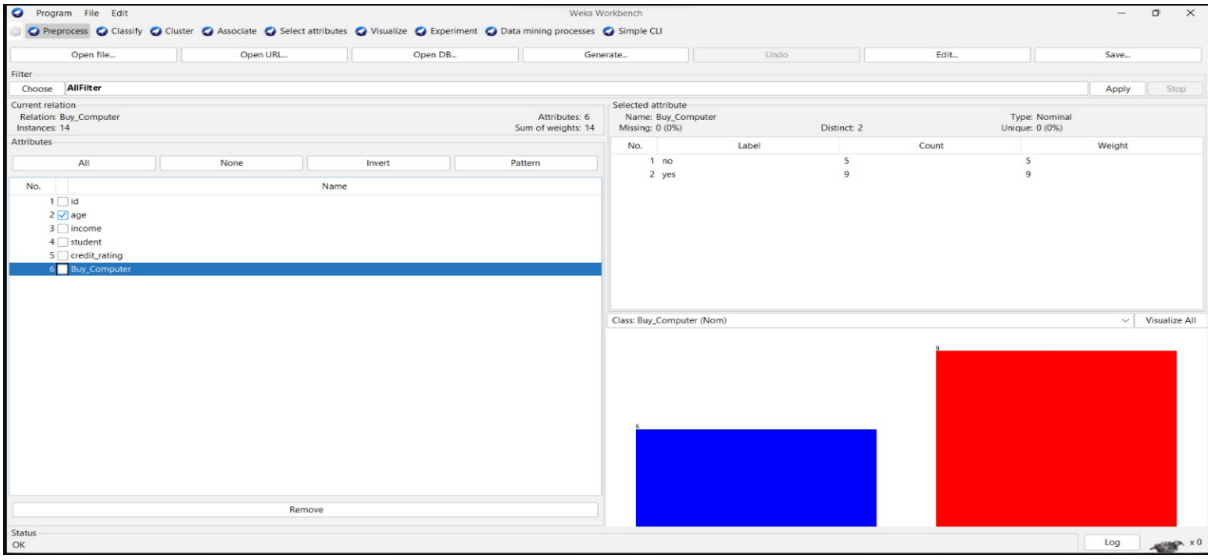
Dataset:

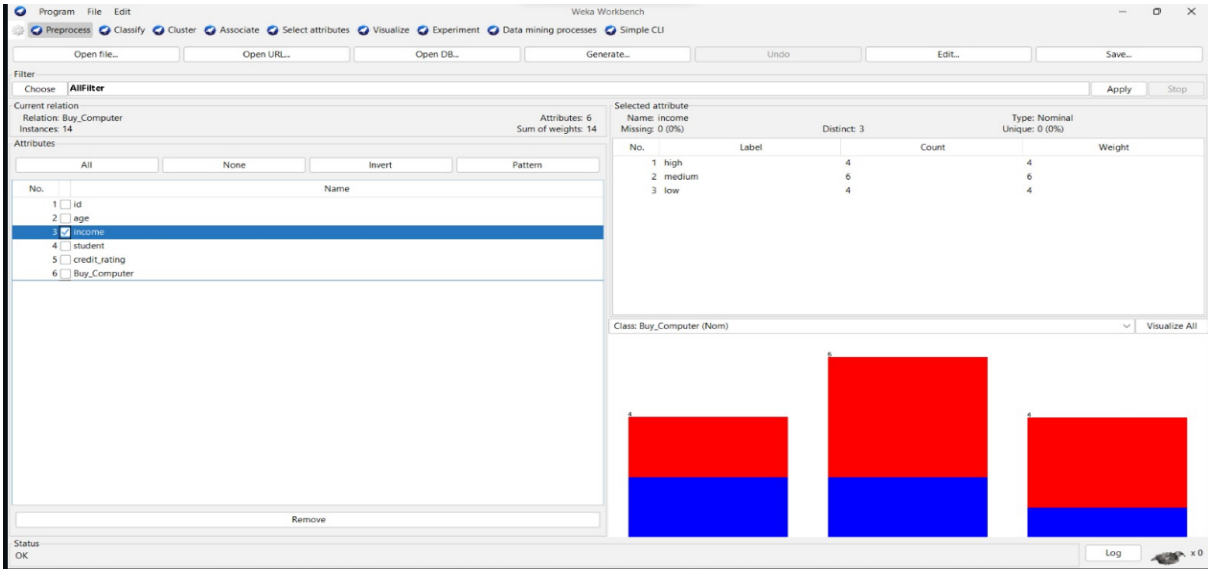
id	age	income	student	credit_rating	Buy_Computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes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

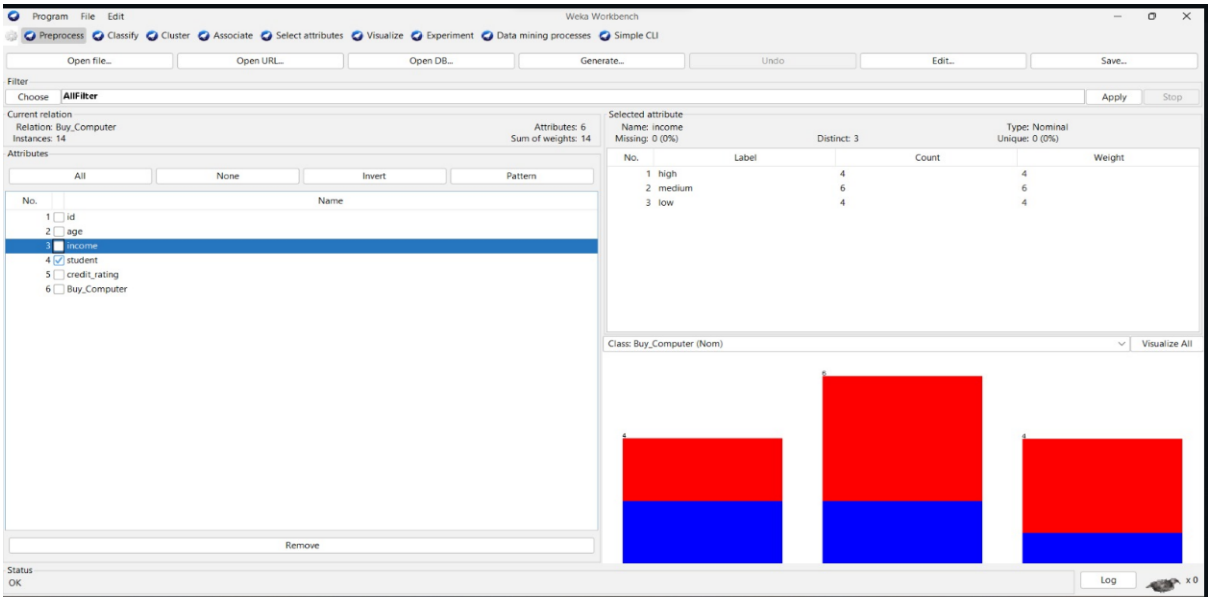
Output:

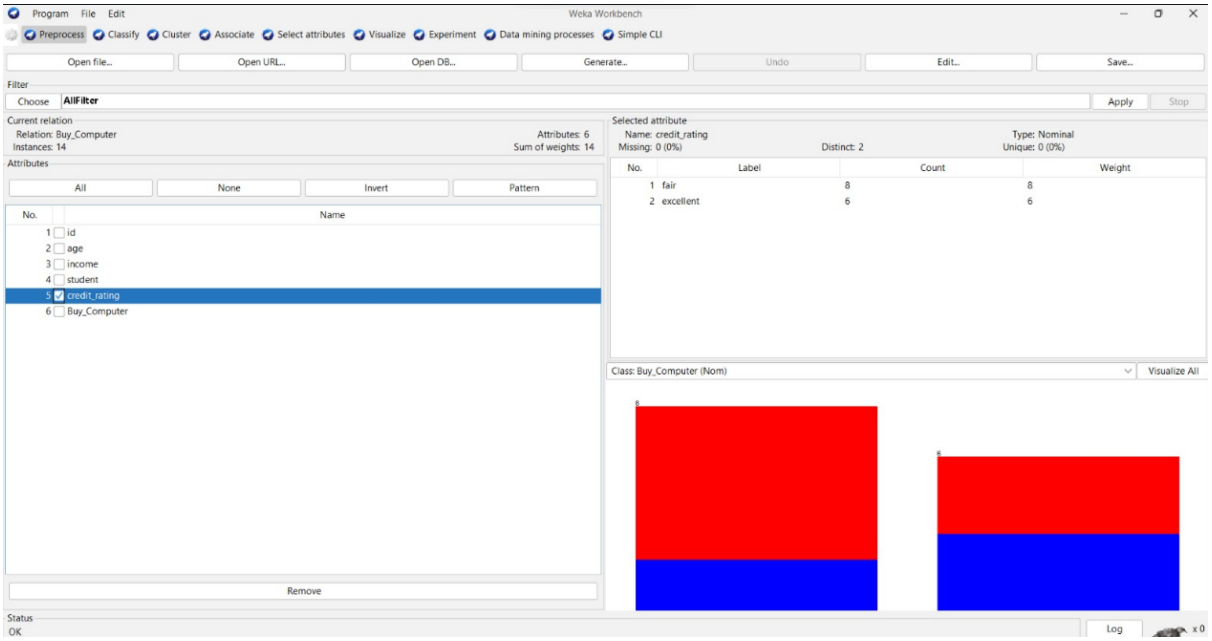
1. Preprocessing

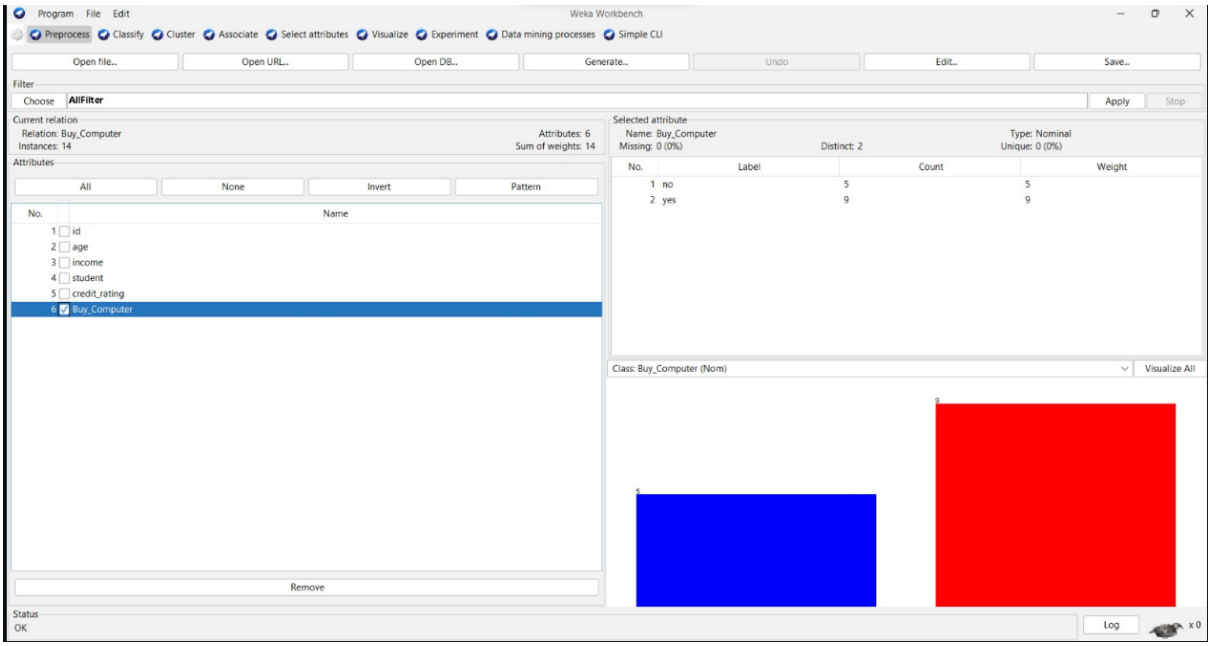





Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
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

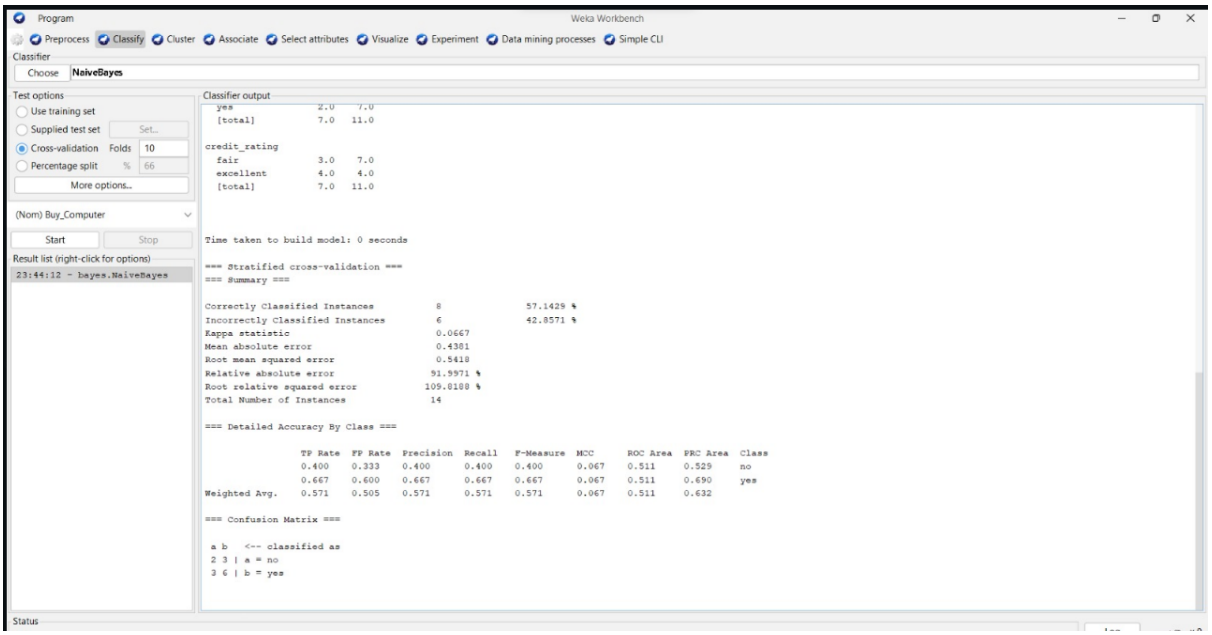

Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
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

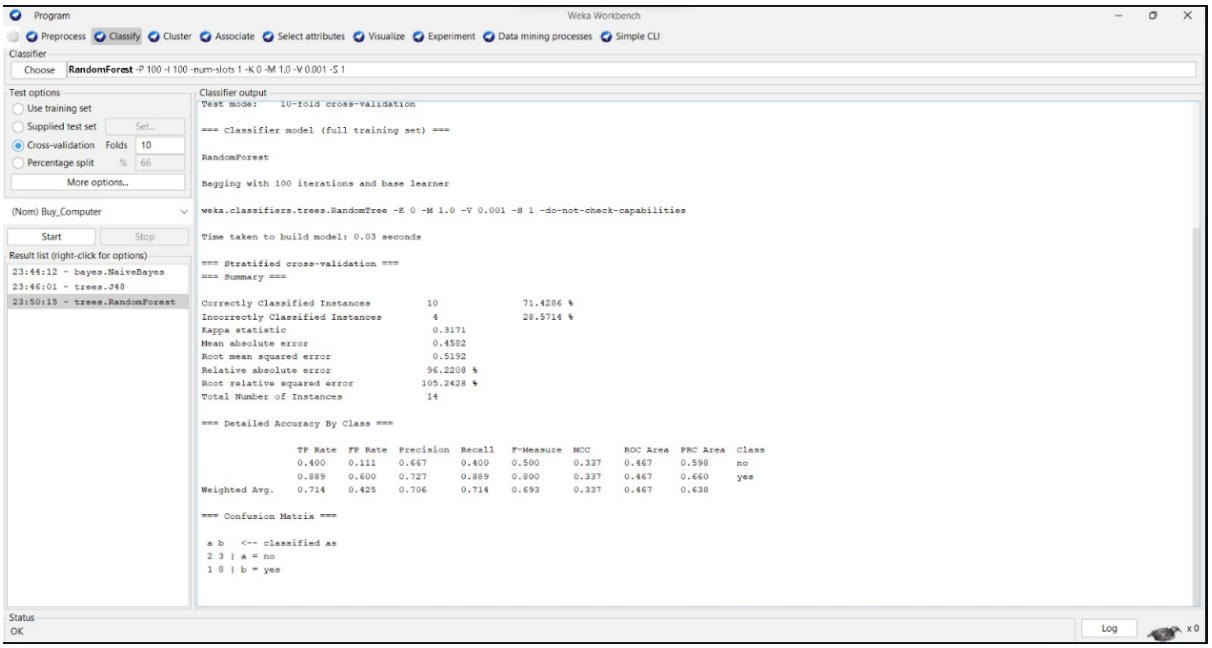

Program File Edit
Weka Workbench
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose AllFilter
Apply Stop
Current relation
Relation: Buy_Computer
Instances: 14
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

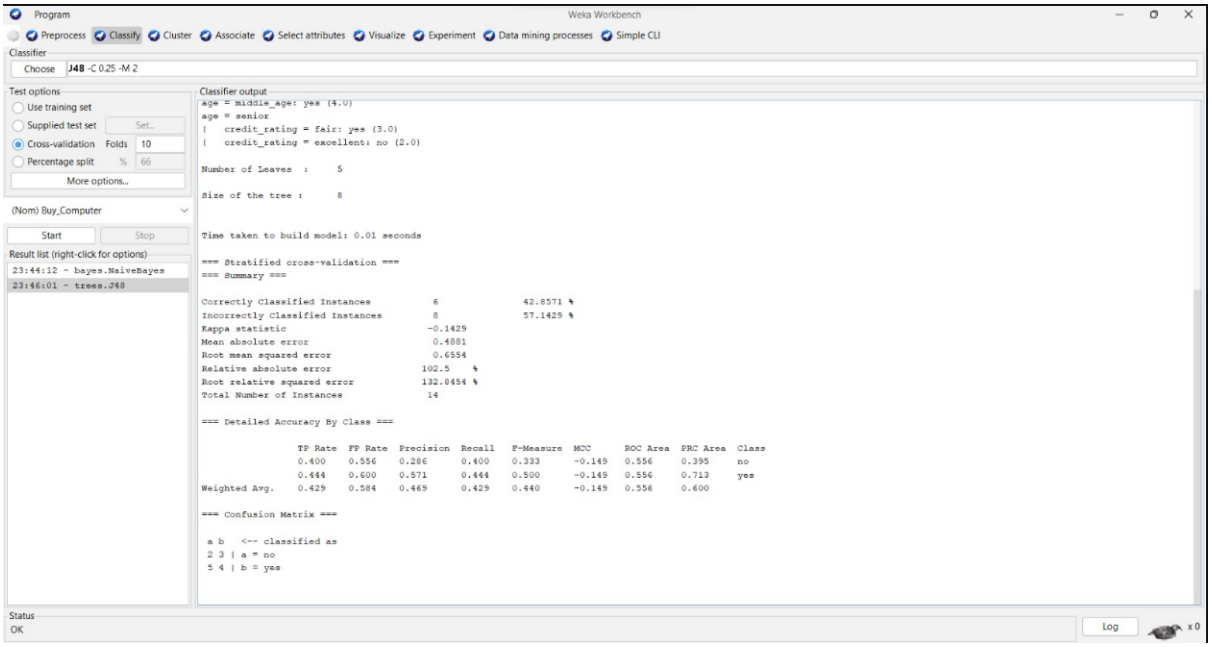
a. Naïve Bayes:

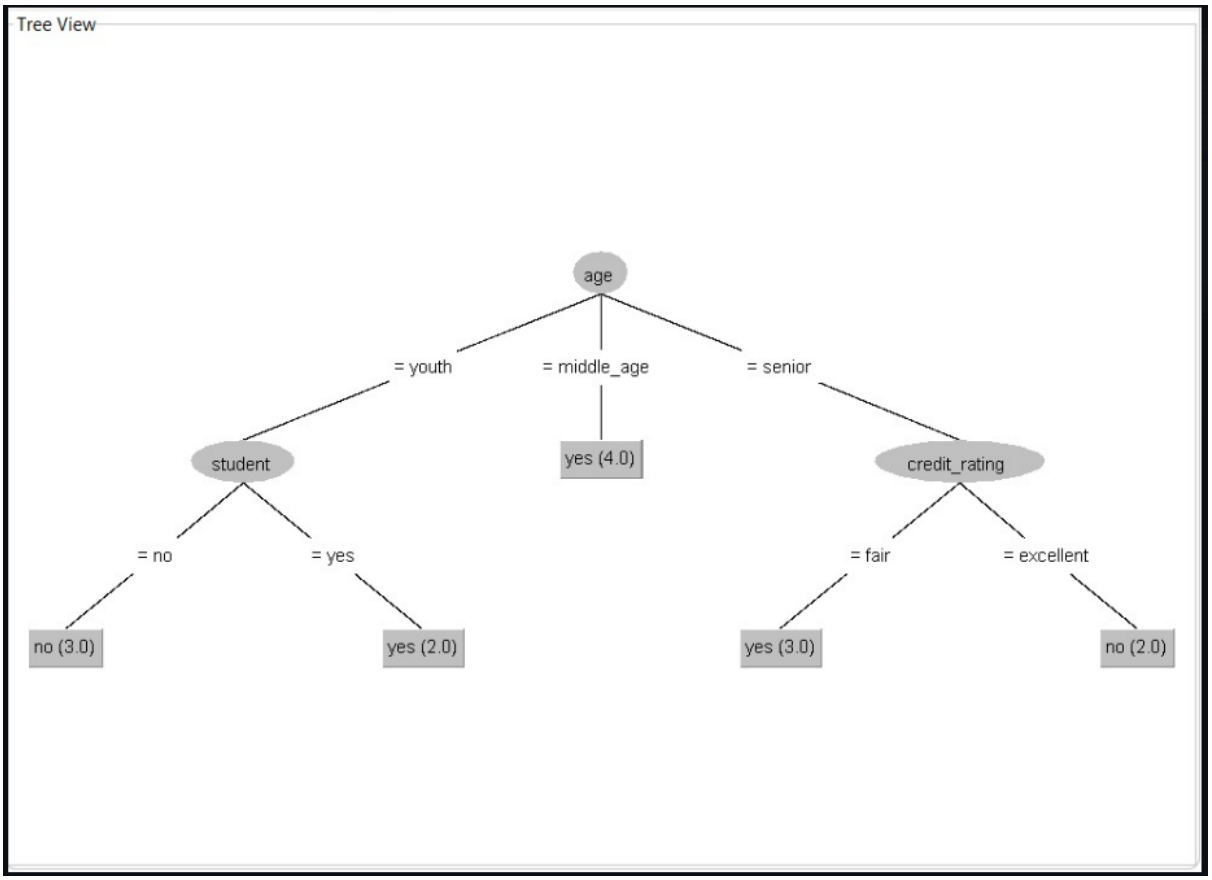


b. Random Forest:



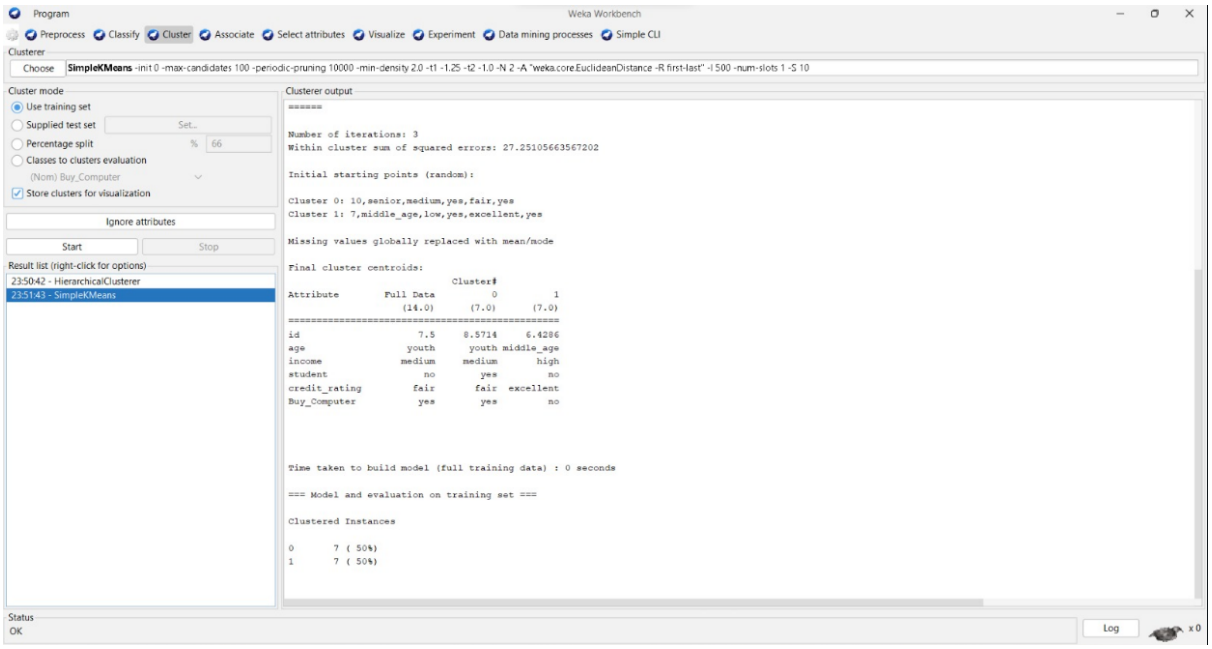
c. J48


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 0.4881
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

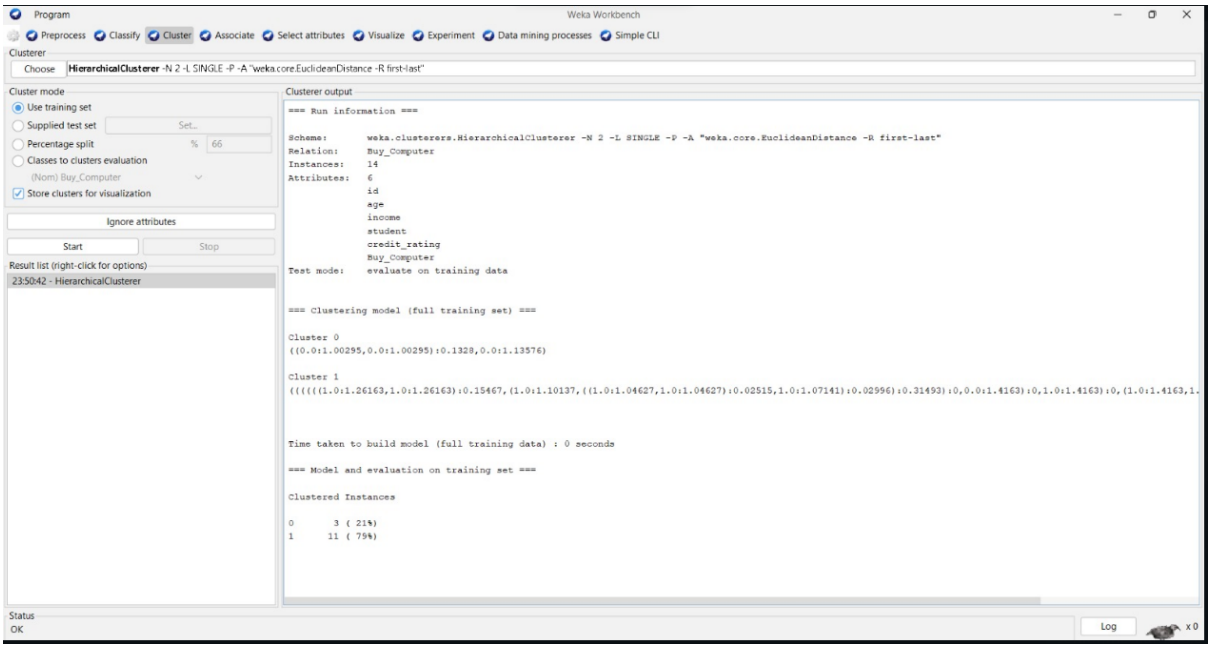

Tree View
age
= youth
= middle_age
= senior
student
yes (4.0)
credit_rating
= no
= yes
= fair
= excellent
no (3.0)
yes (2.0)
yes (3.0)
no (2.0)

3. Clustering

a. Kmeans

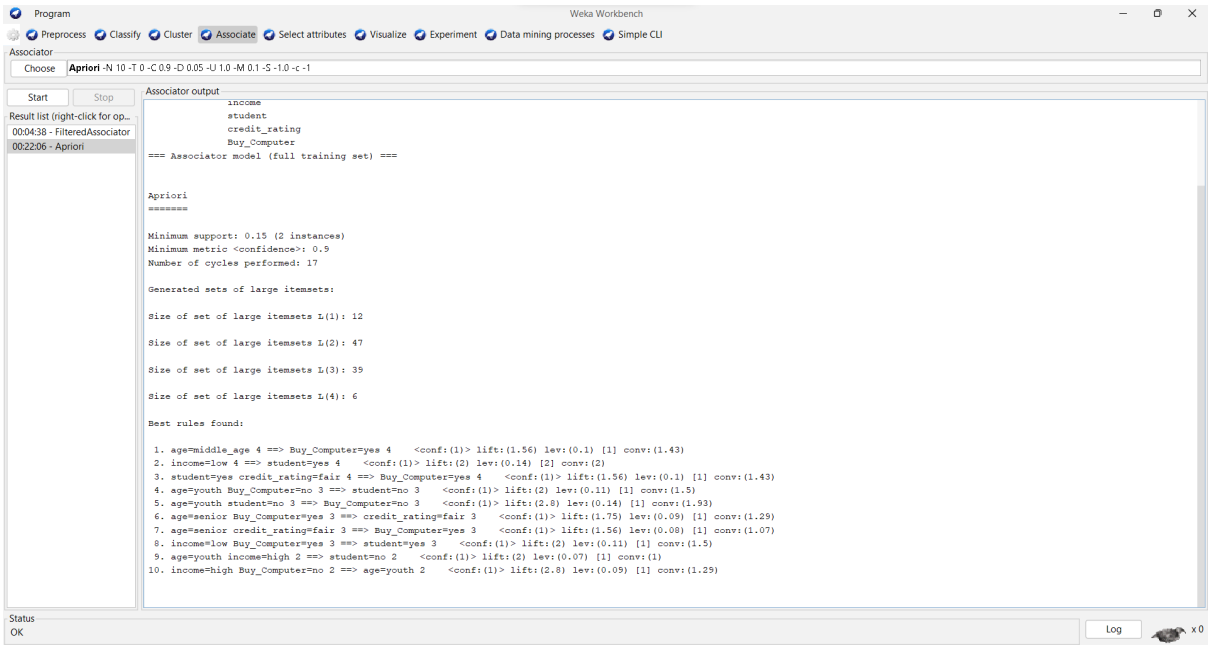


b. Hierarchical Clustering



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:

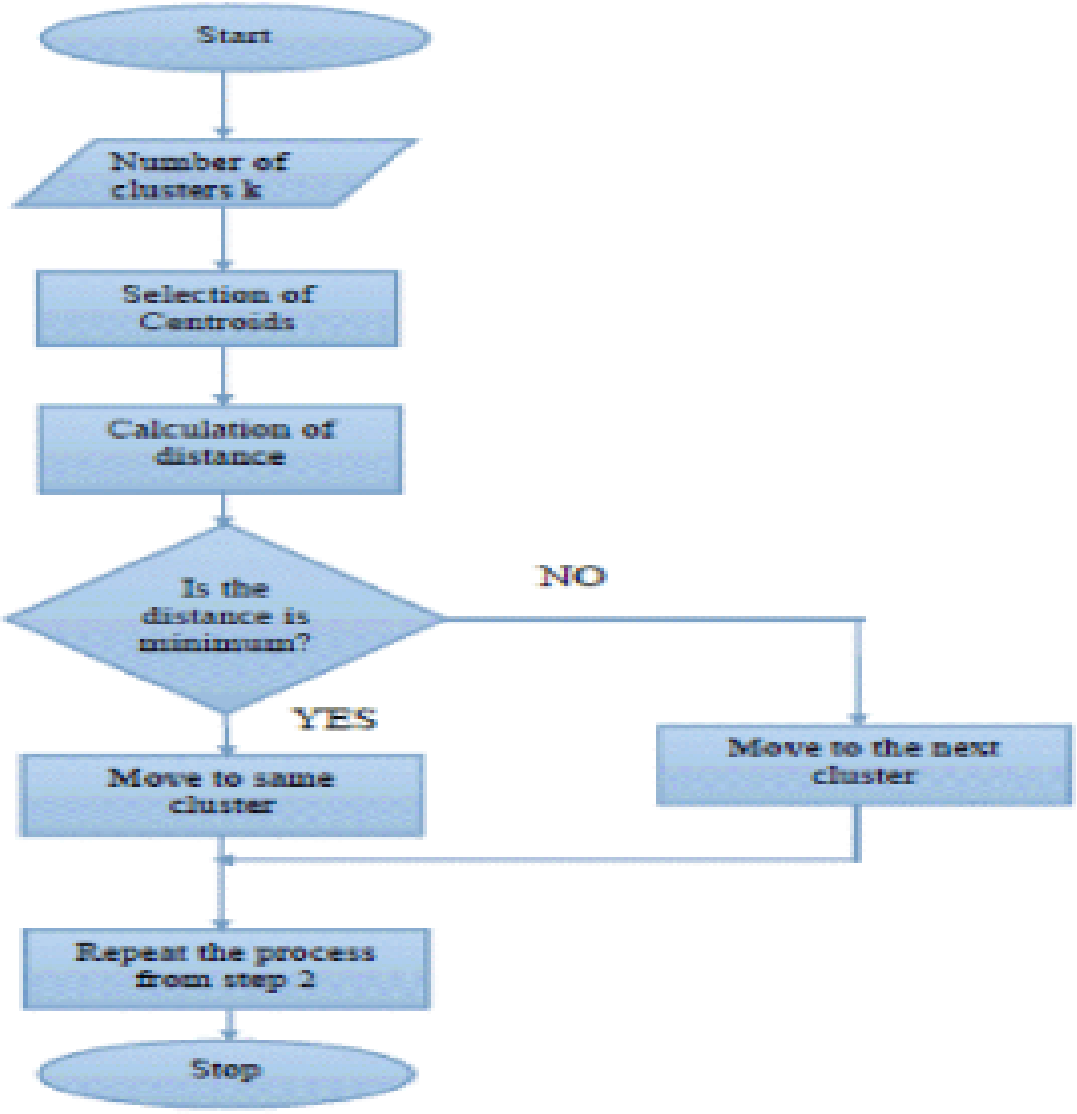


Fig 7.1 Flowchart of K-Means

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:

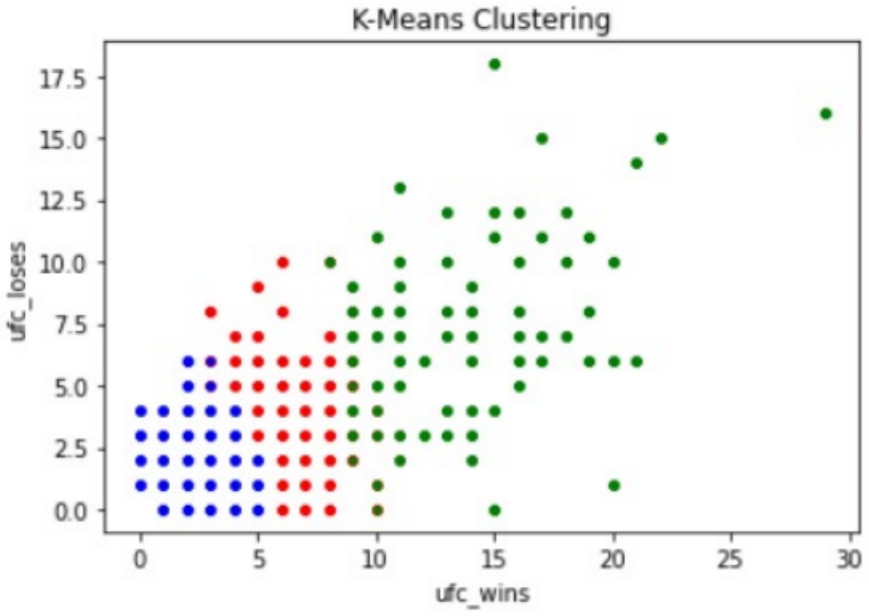
```
import csv
import matplotlib.pyplot as plt


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]
```

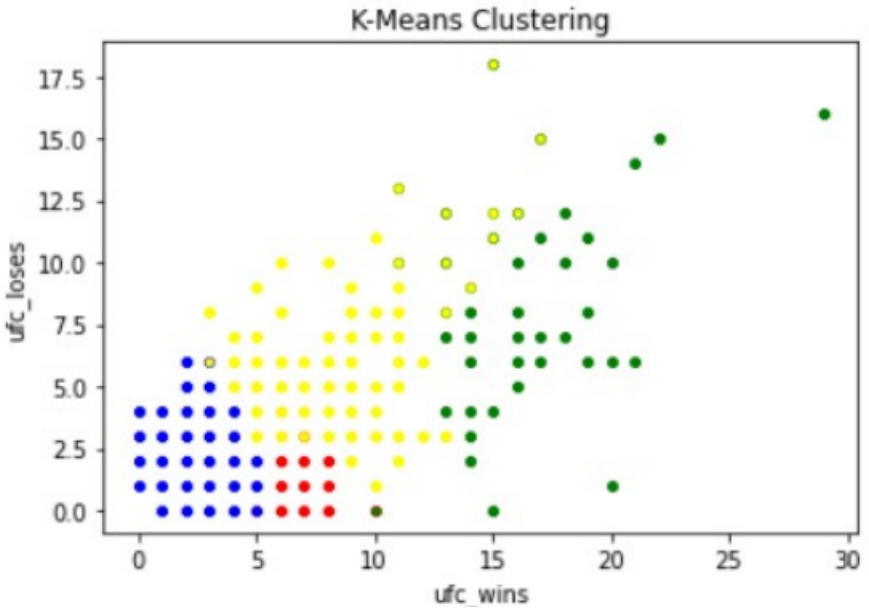
```
    if prev_cents == centroids:
        break

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:

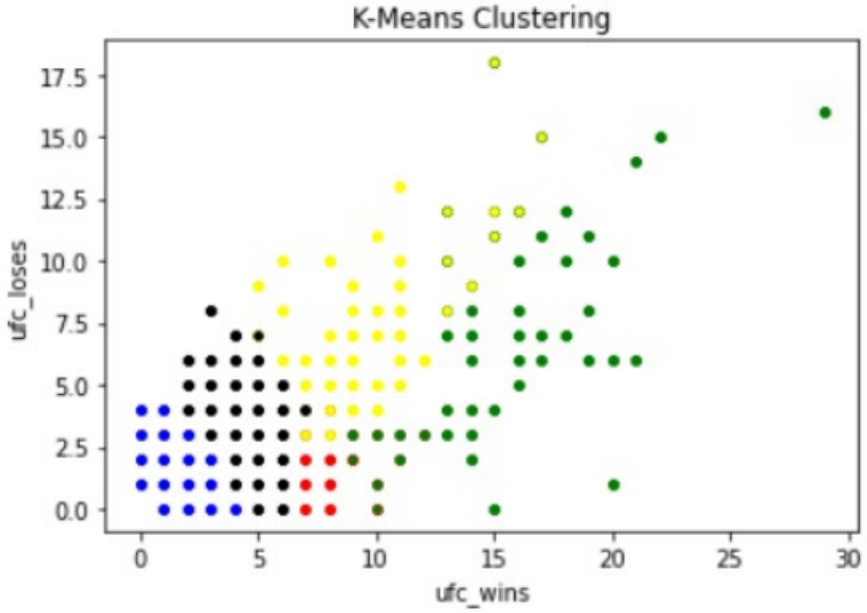
```
Enter the number of clusters: 3
Centroid  1 :  (6.41, 3.7)
Centroid  2 :  (14.56, 7.7)
Centroid  3 :  (1.74, 1.49)
```



```
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.

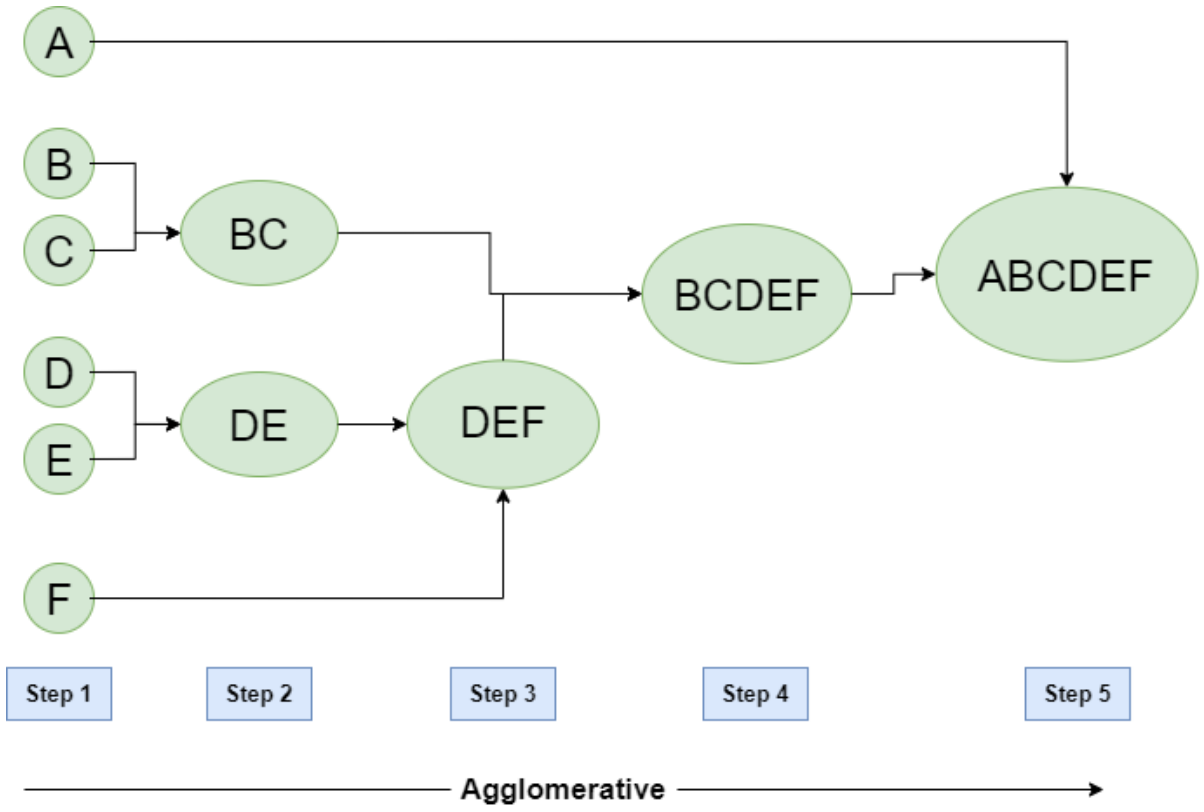
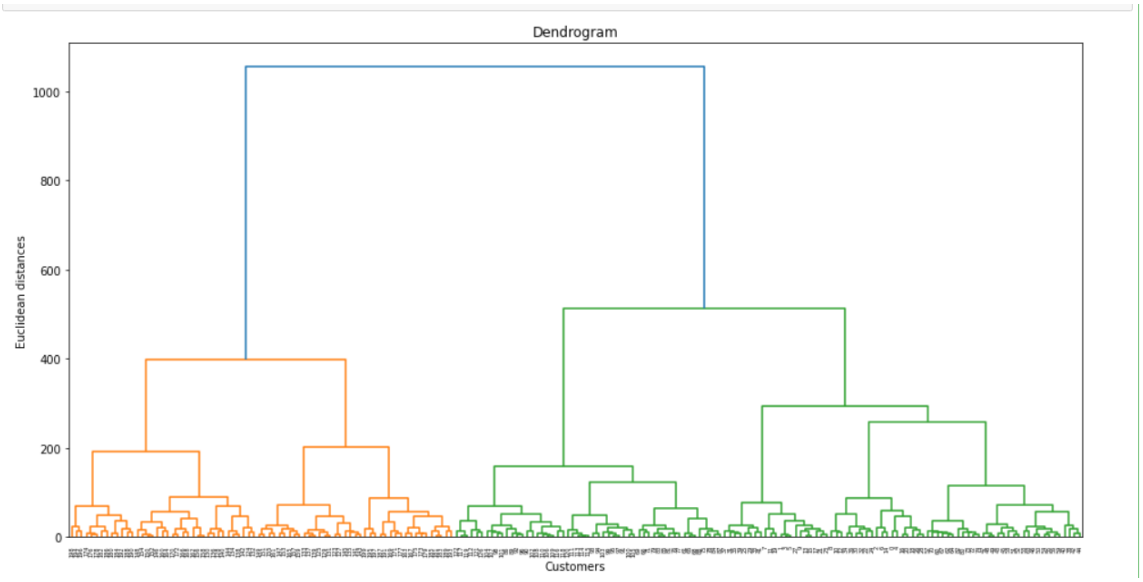


Fig 9.1 Agglomerative Hierarchical Clustering

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.        ]]
```

Example:

Q: Perform Agglomerative Algorithm on the following data and plot a dendogram using single link Approach. The given data indicates the distance between 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)

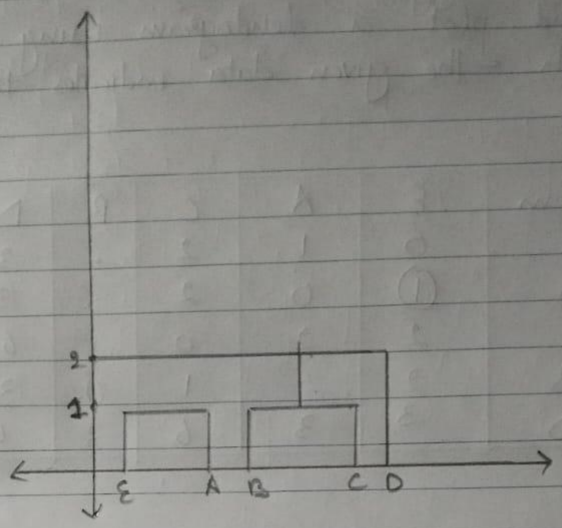
	(E, A)	C	B	D
(E, A)	0			
C	2	0		
B	2	(1)	0	
D	3	6	3	0

Pair (B, C)

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

Pair (E, A) and (B, C)

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



Experiment 9

Aim: Implementation of Association Rule Miningalgorithm (Apriori)

Theory:

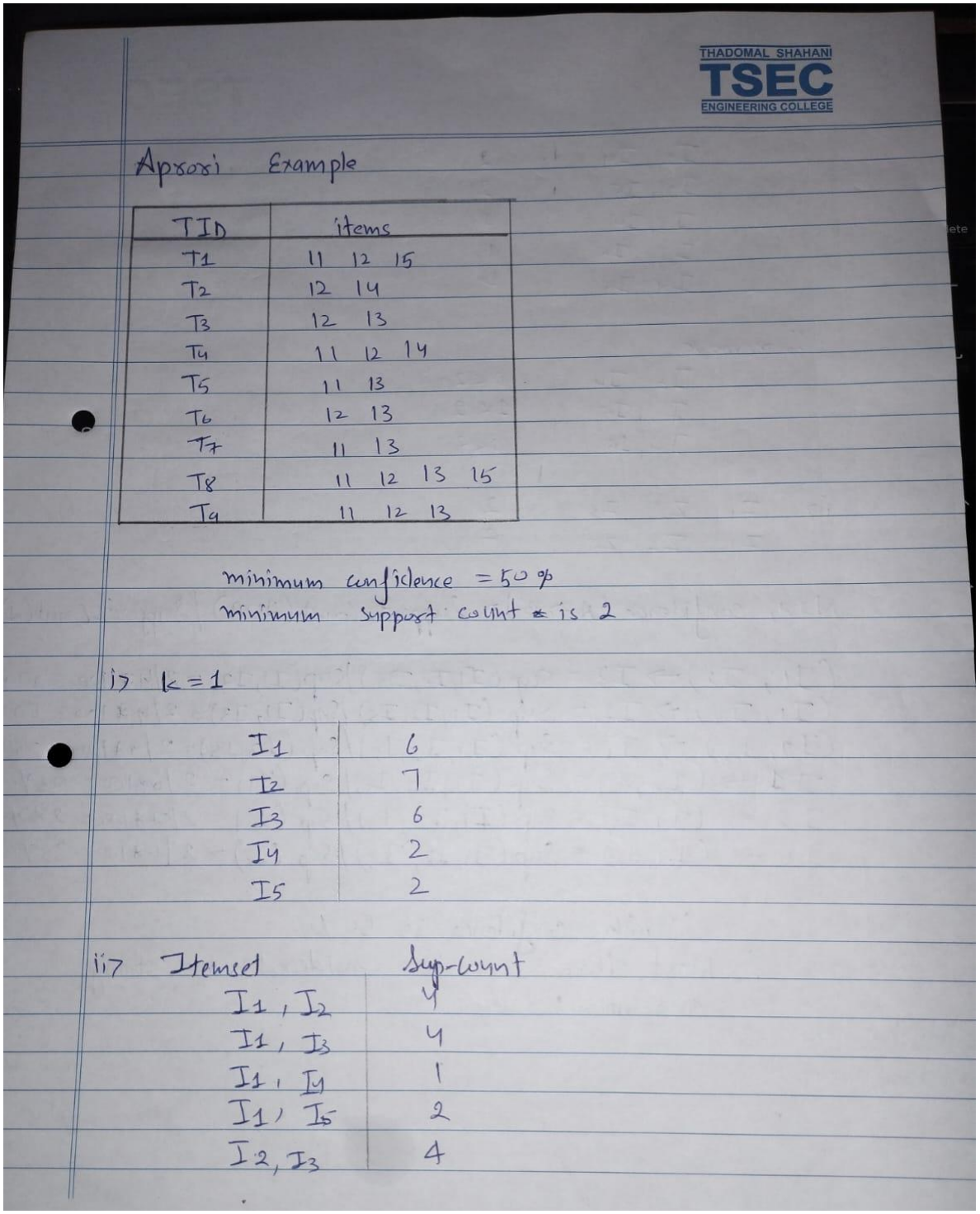
Apriori is a widely used algorithm in data mining and machine learning for finding frequent itemsets in transactional databases. It is a fundamental algorithm for association rule mining, which aims to discover relationships or patterns in data. Apriori is specifically used to identify frequent itemsets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

Sum:



THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Aprsori Example

TID	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

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

I1, I2	4
I1, I3	4
I1, I4	1
I1, I5	2
I2, I3	4

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

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_1	I_2	I_3	2
I_1	I_2	I_5	2

Now, confidence $(A \rightarrow B)$ = Support_count $(A \cup B)$ / Support_count (A)

$(I_1, I_2) \rightarrow I_3$ = Sup (I_1, I_2, I_3) / Sup (I_1, I_2) = $2/4 * 100 = 50\%$
$(I_1, I_3) \rightarrow I_2$ = Sup (I_1, I_2, I_3) / Sup (I_1, I_3) = $2/4 * 100 = 50\%$
$(I_2, I_3) \rightarrow I_1$ = Sup (I_1, I_2, I_3) / Sup (I_2, I_3) = $2/4 * 100 = 50\%$
$I_1 \rightarrow (I_2, I_3)$ = Sup (I_1, I_2, I_3) / Sup (I_1) = $2/6 * 100 = 33\%$
$I_2 \rightarrow (I_1, I_3)$ = Sup (I_1, I_2, I_3) / Sup (I_2) = $2/7 * 100 = 28\%$
$I_3 \rightarrow (I_1, I_2)$ = Sup (I_1, I_2, I_3) / Sup (I_3) = $2/6 * 100 = 33\%$

∵ min confidence is 50%
∴ First three can be considered as strong association rules.

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
```