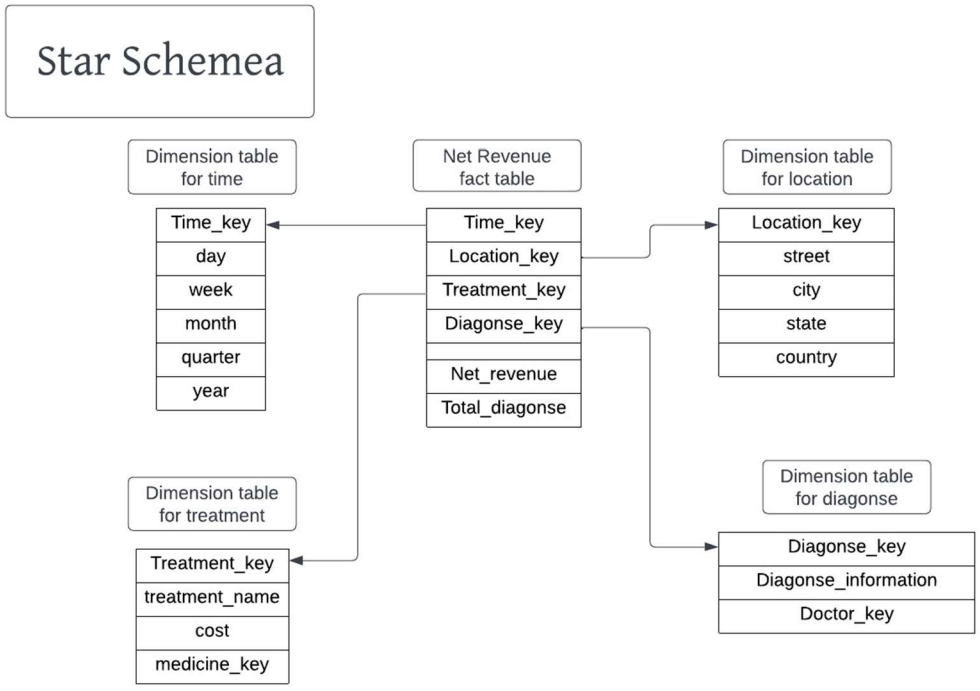
Topic: Dental Clinical Data System

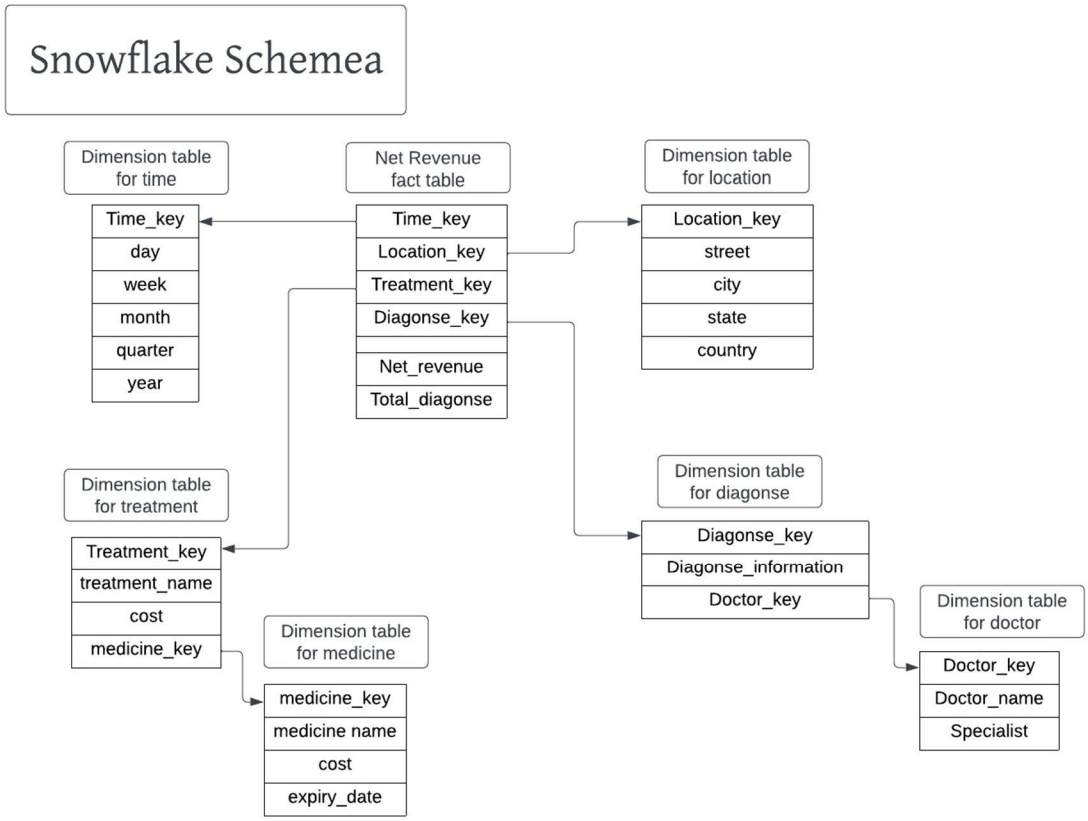
Goal: Our goal is to analyze the clinical data through revenue generation of each branch through Treatment analysis of each patient in every branch

We are creating dental clinical system to keep a proper track of patients under services. We also need them for the follow up sessions of every patient. This system will increase the patients under our services at different location thus increasing the profit. And we can also get to know the profit of different branches. Thus, to achieve the goal of better decision-making using dimensions like Time, Location, Treatment, Diagnosis and facts included are: Revenue generated, Profit, Expense, No. of treatments.

Information Diagram:

Business Process: Treatment Data Analysis				
Time	Location	Treatment	Disease Diagnosed	
Day	Street	Treatment key	Disease name	
Week	City	Treatment name	Disease information	
Month	State	Treatment cost	Doctor key	
Quarter	Country	Medicines used		
years				
Facts: Revenue generated, Profit, Expense, No. of treatments.				





1. Justify whether slowly changing dimension modeling is required for the problem selected?

Ans:

Slowly Changing Dimension (SCD) Modeling: SCD modeling is typically used in data warehousing and business intelligence when you have data that changes over time, such as historical customer addresses or product prices. The choice of whether to implement SCD modeling for a particular problem depends on the nature of the problem. Here are some justifications for and against using SCD modeling:

Justification for SCD Modeling:

- **Historical Analysis**: If your problem requires historical analysis, SCD modeling is crucial. For instance, analyzing how customer demographics change over time or tracking product price fluctuations over the years.
- **Auditing and Compliance**: When you need to maintain an audit trail and ensure compliance with historical data changes, SCD modeling helps you keep a record of all changes.

Justification against SCD Modeling:

- **Static Data**: If the data in your problem domain doesn't change over time, there's no need for SCD modeling. For example, if you're dealing with reference data that remains constant, like a list of countries and their attributes, you don't need SCD modeling.

2. Justify why Fact-less fact table modeling is not required for the problem selected? Give theexamples of fact less fact tables with its type.

Ans:

Fact-less Fact Table Modeling: Fact-less fact tables are used to represent events or facts that don't contain any measures or numeric values. These tables are primarily used to establish relationships between dimensions. Whether fact-less fact tables are required depends on the problem you're dealing with. Here are some justifications for not using fact-less fact table modeling:

Justification against Fact-less Fact Table Modeling:

- **Lack of Relationships**: If your problem doesn't involve establishing relationships between dimensions, there's no need for fact-less fact tables. For instance, in a simple sales reporting system, you might not need fact-less fact tables if you're only interested in sales figures without intricate dimension relationships.

Examples of Fact-less Fact Tables:

- **Date Dimension Bridge**: A fact-less fact table could be used to establish relationships between date dimensions. For example, to track holidays, weekends, or special events for analytical purposes without containing any measures.
- **Student Enrollment**: In an educational context, a fact-less fact table could be used to track student enrollments, helping to identify enrollment trends and relationships between students and courses.

3.Justify your approach to deal with large dimension tables for the problem selected.

Ans:

Large Dimension Table Management: When dealing with large dimension tables, you may encounter performance and storage challenges. To justify your approach for managing large dimension tables in your problem domain, consider the following:

- **Data Partitioning**: Use data partitioning techniques to break down large dimension tables into smaller, more manageable pieces. For instance, you can partition a time dimension table by year or month.
- **Indexing**: Implement appropriate indexing strategies to optimize query performance for large dimensions. This could involve using clustered and non-clustered indexes.
- **Aggregations and Summarizations**: Consider pre-computing aggregations and summaries to reduce the complexity and size of your dimension tables, especially when dealing with historical data.
- **Caching**: Implement caching mechanisms to store frequently accessed parts of large dimension tables in memory for faster retrieval.

4.Justify the use of junk dimension and surrogate key for the problem selected.

Ans:

Junk Dimension and Surrogate Key Usage: Junk dimensions are used to combine several low-cardinality attributes into a single dimension to simplify the data warehouse structure. Surrogate keys are used as unique identifiers for dimension members. Their use depends on the specific

problem domain:

- **Junk Dimension Justification**: If your problem domain contains multiple low-cardinality attributes that don't need to be analyzed separately, but their combinations are meaningful, you can use a junk dimension. For example, in a retail environment, you could use a junk dimension to combine different discount types and payment methods.
- **Surrogate Key Justification**: Surrogate keys are valuable when you need to maintain a stable and unique identifier for dimension members, especially when dealing with slowly changing dimensions. These keys help in tracking and auditing changes to dimension attributes over time. For example, in a customer dimension, a surrogate key could be used to uniquely identify customers, even if their names or addresses change.

Name: Shivprasad CP

Roll No: 9696

Batch: A

Experiment 2

Implement SQL queries for OLAP operations:
Part 1

```
create table dim_time(time_key serial primary key, days date, weeks int, months int, quarter int,
years int);
select * from dim_time;

insert into dim_time(days, weeks, months, quarter, years)
values  ('2016-03-08', 33, 8, 3, 2016),
('2018-06-27', 27, 26, 6, 2018),
('2019-04-30', 30, 18, 4, 2019),
('2015-06-01', 1, 22, 6, 2015),
('2015-12-06', 6, 49, 12, 2015),
('2012-12-21', 21, 51, 12, 2012);

create table dim_location(location_key serial primary key, street varchar, city varchar, states varchar,
country varchar);
select * from dim_location;

insert into dim_location(street, city, states, country)
values ('gully chowl', 'Navi Mumbai', 'Maharashtra', 'India'),
('Film city road', 'Mumbai', 'Maharashtra', 'India'),
('Sector 24', 'Noida', 'U.P', 'India'),
('Aul market road', 'Patamundi', 'Orrisa', 'India'),
('Shivaji Chowk', 'Pune', 'Maharashtra', 'India');

insert into dim_location(street, city, states, country)
values ('New market', 'Noida', 'U.P', 'India');

create table dim_treatment(treatment_key serial primary key, treatment_name varchar, costs int,
medicine_info varchar);
select * from dim_treatment;

insert into dim_treatment(treatment_name, costs, medicine_info)
values('Root Canal', 2000, 'Yes'),
('Braces', 2500, 'No'),
('Teeth whitening', 3000, 'Yes'),
('Root Canal', 2500, 'Yes'),
('Wisdom extract', 4000, 'No'),
('Root Canal', 3000, 'Yes');
```

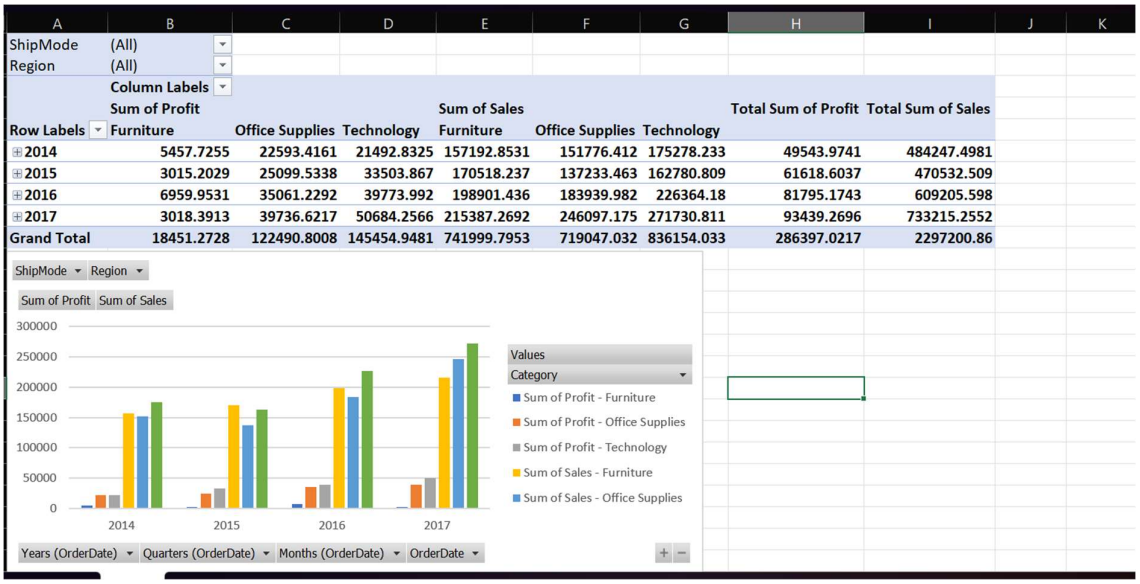
```
create table dim_diagonse(diagonse_key serial primary key, diagonse_info varchar, Doctor_key int);
select * from dim_diagonse;

insert into dim_diagonse(diagonse_info, Doctor_key)
values('Root Canal', 20),
('Braces', 21),
('Teeth whitening', 40),
('Root Canal', 72),
('Wisdom extract', 34),
('Root Canal', 40);

create table fact_revenue(time_key int references dim_time(time_key),
                location_key int references dim_location(location_key),
                treatment_key int references dim_treatment(treatment_key),
                diagonse_key int references dim_diagonse(diagonse_key),
                net_revenue decimal(19,4), total_diagonse int,
                primary key(time_key, location_key, treatment_key, diagonse_key)
               );

insert into fact_revenue values(1,1,1,1,20000.50, 10),
(2,2,2,2,10000.50, 5),
(3,3,3,3,15000.50, 8),
(4,4,4,4,22000.50, 12),
(5,5,5,5,24000.50, 15),
(6,6,6,6,18000.50, 9);

select * from fact_revenue;

drop table fact_revenue;

-- Roll Up Operations
select city, states, sum(net_revenue) from dim_location inner join
fact_revenue on dim_location.location_key = fact_revenue.location_key
group by rollup(states, city) order by states,city;

--Cube operations

select years, quarter, total_diagonse, sum(net_revenue) from fact_revenue natural inner join
dim_time
group by cube(years, quarter, total_diagonse);

--Slice Operations
select states, city, sum(net_revenue) from dim_location inner join fact_revenue on
fact_revenue.location_key = dim_location.location_key where states = 'Maharashtra' group by states,
city;

--Dice Operations
select states, city, sum(net_revenue) from dim_location inner join fact_revenue on
fact_revenue.location_key = dim_location.location_key where states = 'Maharashtra' and city =
'Mumbai'
group by states, city;
```

Part 2:

Perform data analysis with visualization for the following:

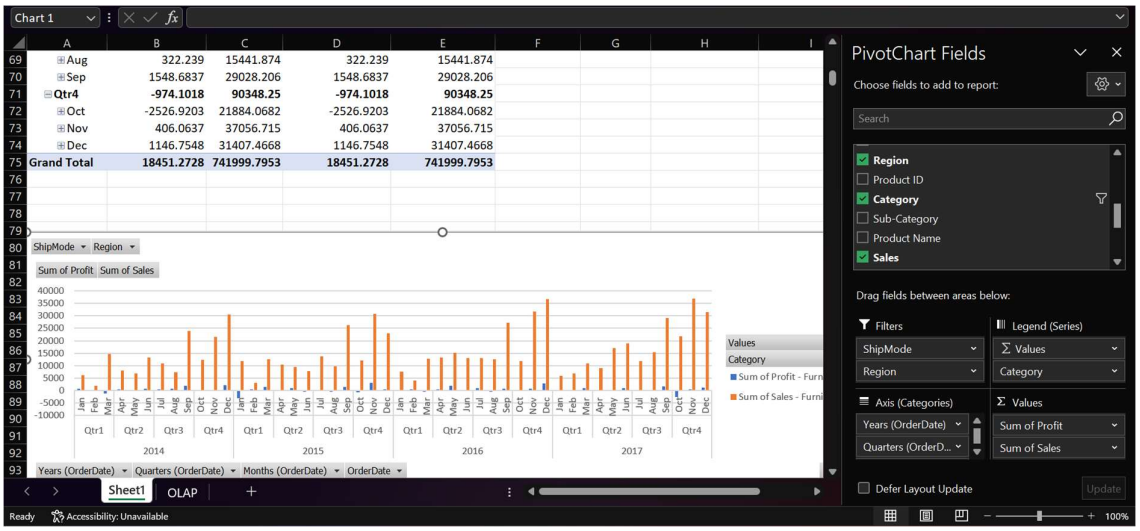
a)To view monthly, quarterly, yearly profit, sales of each category, region wise

b) Comparison of sales and profit on various years.



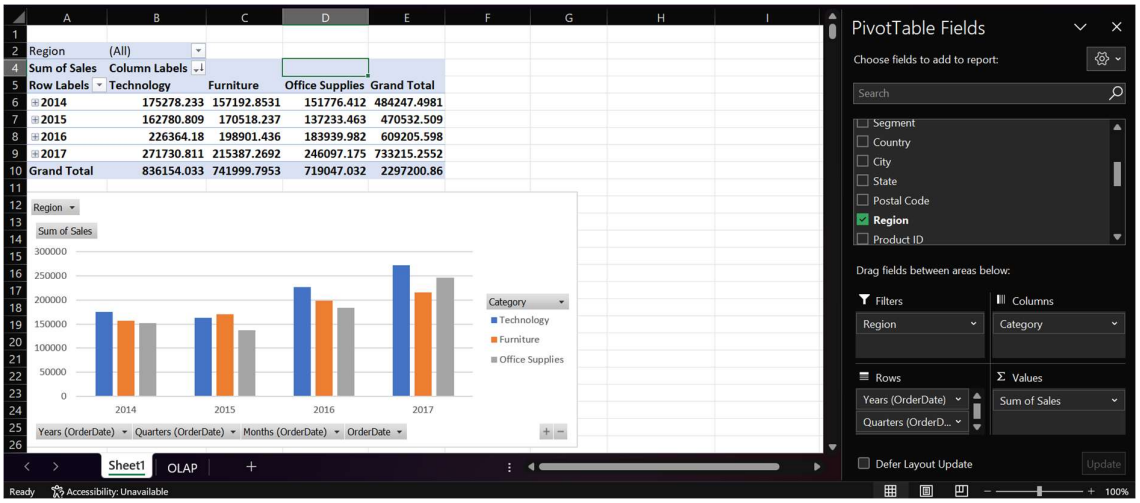
ShipMode	(All)								
Region	(All)								
	Column Labels								
	Sum of Profit			Sum of Sales			Total Sum of Profit	Total Sum of Sales	
Row Labels	Furniture	Office Supplies	Technology	Furniture	Office Supplies	Technology			
2014	5457.7255	22593.4161	21492.8325	157192.8531	151776.412	175278.233	49543.9741	484247.4981	
2015	3015.2029	25099.5338	33503.867	170518.237	137233.463	162780.809	61618.6037	470532.509	
2016	6959.9531	35061.2292	39773.992	198901.436	183939.982	226364.18	81795.1743	609205.598	
2017	3018.3913	39736.6217	50684.2566	215387.2692	246097.175	271730.811	93439.2696	733215.2552	
Grand Total	18451.2728	122490.8008	145454.9481	741999.7953	719047.032	836154.033	286397.0217	2297200.86	

c) Comparison of sales in various months for product category =furniture.

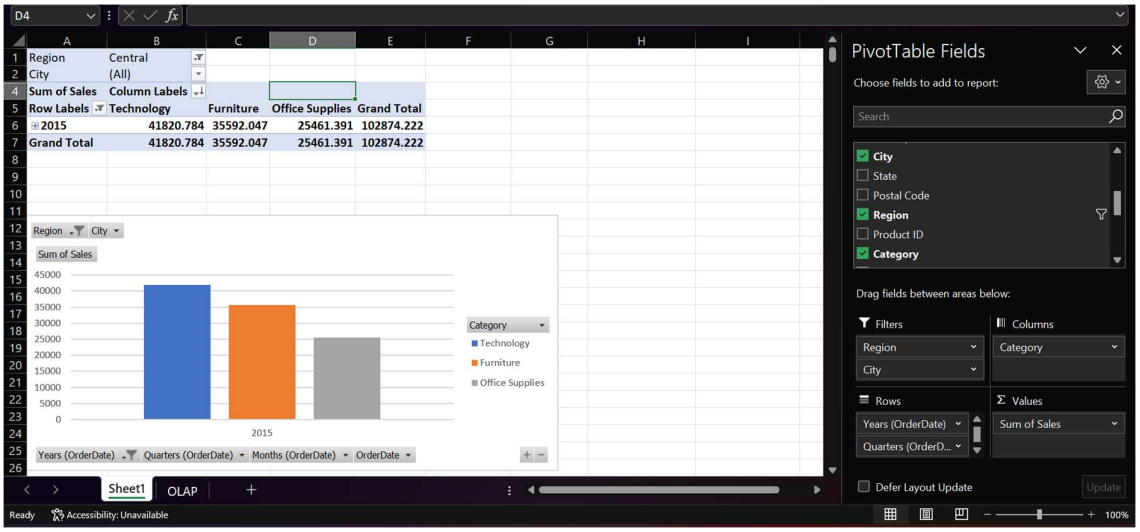


Aug	322.239	15441.874	322.239	15441.874
Sep	1548.6837	29028.206	1548.6837	29028.206
Qtr4	-974.1018	90348.25	-974.1018	90348.25
Oct	-2526.9203	21884.0682	-2526.9203	21884.0682
Nov	406.0637	37056.715	406.0637	37056.715
Dec	1146.7548	31407.4668	1146.7548	31407.4668
Grand Total	18451.2728	741999.7953	18451.2728	741999.7953

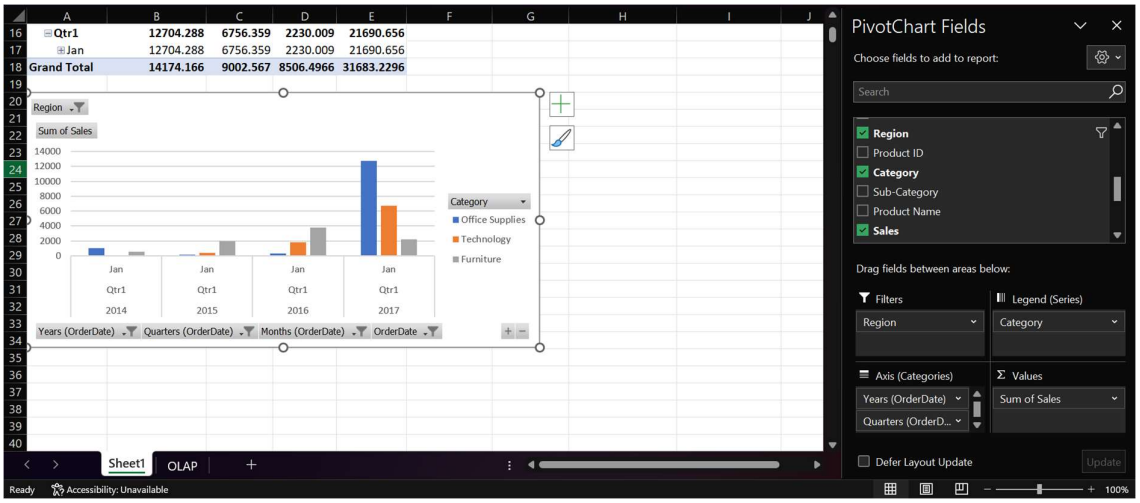
d) Need to know which product has more demand on which location?



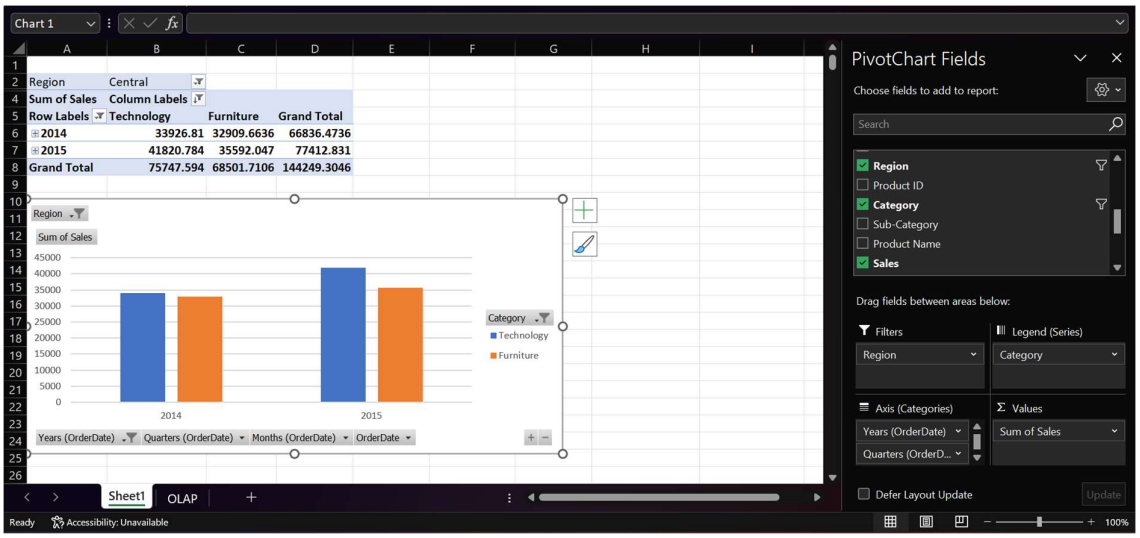
e) What is sale of product category wise, city wise for year=2015 ?



f) What is sales of product categories of Jan month of each year?



g) What is trend of sales on year 2014 and 2015 for product category furniture and technology?



Part 3:

List down the data Extraction Transformation and Loading processes applicable for your DW system.

Extract:

- Data Source Identification: Different sources of dental clinical data, which could include electronic health records, appointment systems, billing systems, patient information, etc.
- Data Extraction: Extract relevant data from the identified sources using appropriate methods, such as database queries, flat file exports, etc.
- Data Profiling: Analyze and profile the extracted data to understand its quality, structure, and potential issues.

Transform:

- Data Cleansing: Cleaning the extracted data by identifying missing values, inconsistencies, errors, and duplicates.

- Data Transformation: Convert data into a common format and standardize units, terminologies, and codes to ensure consistency.
- Data Integration: Integrate data from various sources into a unified format, considering data types, relationships, and hierarchies.
- Data Validation: Validate the transformed data to ensure that it meets the defined quality standards and business rules.

Loading:

- Staging: Store the cleaned and transformed data in a staging area, separate from the data warehouse, to facilitate further validation.
- Data Warehouse Loading: Load the validated and transformed data into the data warehouse.

Experiment No: 03	TE AI&DS
Date of Performance:	Roll No: 9696
Aim: Apply data Exploration techniques on given data to organize data (Tutorial)	
CO3: Apply data exploration and Data preprocessing techniques to organize and prepare data for data mining	

Rubrics for assessment of Experiment:

Sr. No	Parameters	Exceed Expectations(EE)	Meet Expectations (ME)	Below Expectations (BE)
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

NAME:- CHAVARATTIL SHIVPRASAD PREMARAJAN.

ROLLNO: 9696.

Tutorial 1. DWM Experiment 3.

1]

→ 18, 20, 21, 21, 24, 25, 25, 26, 27, 27, 29, 29, 29, 29, 35, 38, 38, 40, 40, 40, 40, 41, 45, 50, 51, 57, 75.

a) mode = ~~8~~ 33.15

median = 29.

b)

modes are 29 & 40.

∴ Data's modality = bimodal.

c) mid range = (18 + 75)//2.

mid range = 46.5

d)

→ Since the data is already sorted

Q_1 = (27+1)/4 = 7th value

Q_3 = 3* (27+1)/4 = 21st value.

Q_1 = 25

Q_3 = 40

c) minimum = 18 $Q_2 = \frac{n+50}{100} = \frac{270+1}{2} = 135.5 \approx 136^{th}$

$Q_1 = 25$ $Q_3 = 40$ $IQR = Q_3 - Q_1 = 40 - 25 = 15$

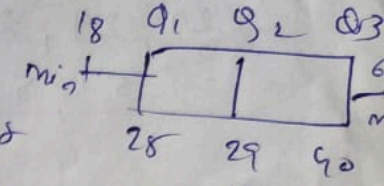
Median = 29

$Q_3 = 40$

Maximum = 75

Lower limit $= Q_1 - 1.5 \times IQR$
$= 25 - 1.5 \times 15 = 2.5$

Upper limit $= Q_3 + 1.5 \times IQR$
$= 40 + 1.5 \times 15 = 62.5$



d) A quantile-quantile (q-q) plot is used to compare the quantiles of data distribution with quantiles of a theoretical distribution like a normal distribution. It helps to determine if the data follows a specific distribution. A quatile plot on the other hand, typically refers to a plot of the quantiles of data itself.

2].

→

age	frequency	C.f.
1-5	300	300.
6-15	550	850
16-20	450	1300
21-50	1200	2500
51-80	800	3300
81-110.	65	3365.

$n = 110.$, $n/2 = 110/2 = 55.$

55 th item lies in the class of 51-80.

$\therefore L_1 = 51.$

$n = 110.$

$(\Sigma f_{req})_1 = 2500.$

$\Sigma (f_{req})_m = 800.$

$$\text{median} = L_1 + \left(\frac{n/2 - \Sigma(f_{req})_1}{\Sigma (f_{req})_m} \right) \times \text{width}$$

$$= 51 + \left(\frac{55 - 2500}{800} \right) \times 29$$

$$= 51 - 29.07 = 21.93 //.$$

3]

→ a)

Mean = (23 + 23 + 27 + 27 + 39 + 41 + 49 + 49 + 50 + 52 + 54 + 54 + 56 + 57 + 58 + 58 + 60 + 61) / 18

= 45.06.

Median = 50.

Standard deviation = $\sqrt{\frac{\Sigma(x - \bar{x})^2}{n}}$.

= 2(23 − 45.06)² + 2(27 − 45.06)² + (39 − 45.06)² + (41 − 45.06)² + (47 − 45.06)² + (49 − 45.06)² + (50 − 45.06)² + (52 − 45.06)² + 2(54 − 45.06)² + (56 − 45.06)² + (57 − 45.06)².

Standard deviation = $\sqrt{\frac{2970.44}{18}}$ = 12.84.

For Jad:

Mean = $\frac{516.1}{18}$ = 28.67.

median :

$N = 18 \Rightarrow \frac{N}{2} = \frac{18}{2} = 9$

Median :- ~~$N = 18$~~ $\frac{(9^{th} + 10^{th})}{2} = \frac{30.2 + 33.6}{2} = 31.9$.

fat	fat − 28.67	(fat − 28.67)²
7.5	−21.17	448.16
24.5	−4.17	17.38
5.8	−22.87	523.03
15.8	−12.87	165.67
33.4	4.73	22.37
28.9	−4.77	22.75
31.4	2.73	7.45
29.2	0.53	0.2809
30.2	1.53	2.3409
33.6	4.93	24.3049
44.5	15.83	250.58
28.8	0.13	0.0169
31.4	2.73	7.4529
33.2	4.53	20.52
32.1	3.43	11.76
34.9	6.23	38.81
40.2	11.53	132.94
35.7	7.1	50.41

$\Sigma \Rightarrow 1746.1827$

$\sigma = \sqrt{\frac{1746.1827}{18}}$

$= 9.84$.

b) box plot

for qz

$$Q_1 = \frac{N \times 25}{100} = \frac{18 \times 25}{100} = 4.5 = 5^{th} = 39$$

$$Q_2 = \frac{N \times 50}{100} = 9 = \frac{50+52}{2} = 51$$

$$Q_3 = \frac{N \times 75}{100} = 13.5 = 57$$

$$IQR = Q_3 - Q_1 = 57 - 39 = 18$$

Lower limit = $Q_1 - IQR \times 1.5$

$= 39 - 18 \times 1.5$

$= 12$

Upper limit = $Q_3 + IQR \times 1.5$

$= 57 + 18 \times 1.5$

$= 84$

No outliers

for fat:-

5·8, 7·5, 15·8, 23·9, 24·5, 28·8, 29·2, 30·2, 31·4, 31·4, 32·1, 33·2, 33·4336, 34·9, 35·7, 40·2, 49·5

$Q_1 = 4 \cdot 5 \simeq 5^{th} = 24 \cdot 5$

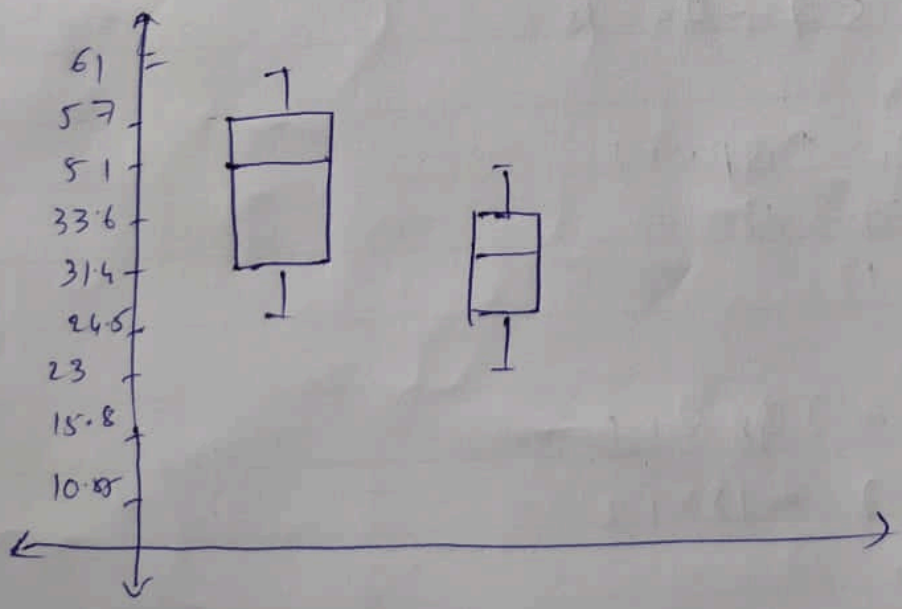
$Q_2 = \frac{9^{th} + 10^{th}}{2} = \frac{31 \cdot 4 + 31 \cdot 4}{2} = 31 \cdot 4$

$Q_3 = 14^{th} = 33 \cdot 6$

$IQR = 33 \cdot 6 - 24 \cdot 5$
$= 9 \cdot 1$

Lower Limit $= 24 \cdot 5 - (9 \cdot 1 \times 1 \cdot 5)$
$= 10 \cdot 85$

Upper limit $= 33 \cdot 6 + (9 \cdot 1 \times 1 \cdot 5)$
$= 47 \cdot 25$



4)

→

Age	Ward	Gender	Test	Health Progress	Fees
56	Ward A	F	P	Good	13 674
76	Ward B	M	N	Better	12343
23	Ward B	M	N	Bitter	6542
47	Ward C	F	N	Best	3459

a) Nominal attribute :- Ward.

$$d(P,j) = \frac{P-M}{P}$$

$$d(2,1) = \frac{1-0}{1} = 1$$

$$d(3,1) = \frac{1-0}{1} = 1$$

$$d(3,2) = \frac{1-1}{1} = 0$$

$$d(4,1) = \frac{1-1}{1} = 0$$

$$d(4,2) = \frac{1-0}{1} = 1$$

$$d(4,3) = \frac{1-0}{1} = 1$$

$$d(i,j) = \begin{bmatrix} 0 & & & \\ 1 & 0 & 0 & \\ 1 & 0 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

11) Asymmetric Test

1 P → 1
2 N → 0
3 N → 0
4 N → 0

P \ Obj	1	0
1	q	r
0	s	t

$$d(i,j) = \frac{r+s}{q+r+s}$$

$$d(2,1) = \frac{1+0}{0+1+0} = 1$$

$$d(3,1) = \frac{0+1}{0+0+1} = 1$$

$$d(3,2) = \frac{0}{0} = 0$$

$$d(4,1) = \frac{0+1}{0+0+1} = 1$$

$$d(4,2) = \frac{0}{0} = 0$$

$$d(4,3) = \frac{0}{0} = 0$$

$$d(i,j) = \begin{bmatrix} 0 & & \\ 1 & 0 & \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

→ more similarity.

Symmetric.

Obj i \ Obj j	1	0
1	q	r
0	s	t

F → 0
M → P.

gender

F → 0
M → 1
N → 1
P → 0.

$$d(2,1) = \frac{1+0}{0+1+0+0} = 1$$

$d(3,1) = 1$
$d(3,2) = 0$
$d(4,1) = 0$
$d(4,2) = 1$
$d(4,3) = 1$

$$d(i,j) = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 0 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Ordinal attribute

Health progress

Good → 3

Better → 2

Better → 2

Bust → 1

$z_{if} = \frac{r_{if} - 1}{r_{if} - 1}$

$Bust = \frac{1-1}{3-1} = 0$

$Better = \frac{2-1}{3-1} = \frac{1}{2} = 0.5$

$Good = \frac{3-1}{3-1} = 1$

$d(2,1) = 0.5$

$d(3,1) = 0.5$

$d(3,2) = 0$

$$d = \begin{bmatrix} 0 & & & \\ 0.5 & 0 & & \\ 0.5 & 0 & 0 & \\ 1 & 0.5 & 0.5 & 0 \end{bmatrix}$$

Numeric attributes

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}|$$

$$d(2,1) = |12343 - 13674| + |76 - 50|$$
$$= 1331 + 26$$
$$= 1357$$

$$d(3,1) = |6542 - 13674| + |23 - 566|$$
$$= 7132 + 33$$
$$= 7165$$

$$d(3,2) = |6542 - 12343| + |23 - 76|$$
$$= 7132 + 33$$
$$= 7165$$

$$d(3,2) = 5854$$
$$d(4,1) = 10224$$
$$d(4,2) = 8913$$
$$d(4,3) = 3107$$

$$d(i,j) = \begin{bmatrix} 0 & & \\ 1357 & 0 & \\ 7165 & 5854 & 0 \\ 10224 & 8913 & 3107 \end{bmatrix}$$

Engineer.

5]

→

a)

Euclidean Distance = $\sqrt{(32-20)^2 + (10-0)^2 + (40-31)^2 + (20-5)^2}$

$= 27.037$.

b) Manhattan Distance = $|32-20| + |10-0| + |40-31| + |20-5|$

$= 46$ //.

Experiment No: 04	TE AI&DS
Date of Performance:	Roll No: 9696
Aim: Apply data exploration and Data preprocessing techniques to organize data for data mining	
Related CO3: Apply data exploration and Data preprocessing techniques to organize and prepare data for data mining	

Rubrics for assessment of Experiment:

Sr. No	Parameters	Exceed Expectations(EE)	Meet Expectations (ME)	Below Expectations (BE)
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

Theory:

Data Exploration:

Measure of Central Tendency

A measure of central tendency (also referred to as measures of centre or central location) is a summary measure that attempts to describe a whole set of data with a single value that represents the middle or centre of its distribution.

There are three main measures of central tendency: the mean, the median and the mode. Each of these measures describes a different indication of the typical or central value in the distribution.

What is the mean?

The mean is the sum of the value of each observation in a dataset divided by the number of observations. This is also known as the arithmetic average.

Looking at the retirement age distribution again:

54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

The mean is calculated by adding together all the values (54+54+54+55+56+57+57+58+58+60+60 = 623) and dividing by the number of observations (11) which equals 56.6 years.

The population mean is indicated by the Greek symbol μ (pronounced 'mu'). When the mean is calculated on a distribution from a sample it is indicated by the symbol $\bar{x}$ (pronounced X-bar).

What is the median?

The median is the *middle value* in distribution when the values are arranged in ascending or descending order.

The median divides the distribution in half (there are 50% of observations on either side of the median value). In a distribution with an odd number of observations, the median value is the middle value.

Looking at the retirement age distribution (which has 11 observations), the median is the middle value, which is 57 years:

54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

When the distribution has an even number of observations, the median value is the mean of

the two middle values. In the following distribution, the two middle values are 56 and 57, therefore the median equals 56.5 years:

52, 54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

The median cannot be identified for categorical nominal data, as it cannot be logically ordered.

Standard Deviation (Dispersion)

The **standard deviation** is a statistic that measures the dispersion of a dataset relative to its mean and is calculated as the square root of the variance.

Formula

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

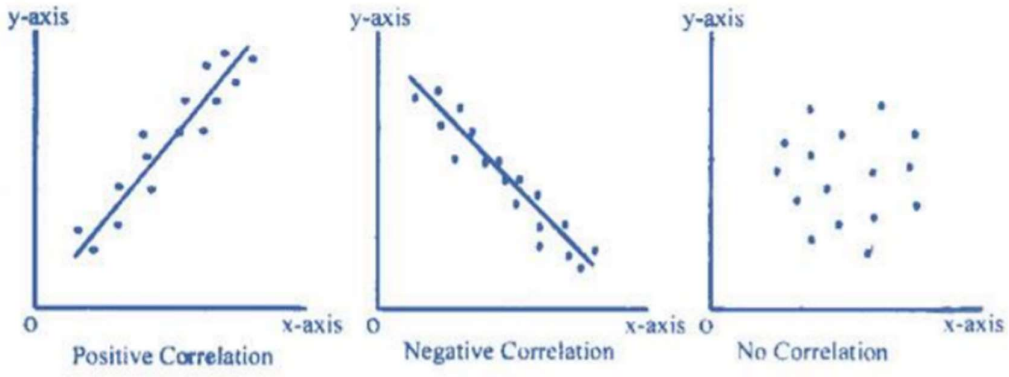
μ = the population mean

CORRELATION

Definition: The Correlation is a statistical tool used to measure the relationship between two or more variables, i.e. the degree to which the variables are associated with each other, such that the change in one is accompanied by the change in another.

Types of Correlation:

1. **Positive Correlation** A correlation in the same direction is called a positive correlation. If one variable increases the other also increases and when one variable decreases the other also decreases. For example, the length of an iron bar will increase as the temperature increases. • Price and Supply • Sales and Expenditure on Advertisement • Yield and Fertilizer Applied
2. **Negative Correlation** Correlation in the opposite direction is called a negative correlation. Here if one variable increases the other decreases and vice versa. For example, the volume of gas will decrease as the pressure increases, or the demand for a particular commodity increases as the price of such commodity decreases. Examples: • Price and Demand • Yield and Weed
3. **No Correlation or Zero Correlation** If there is no relationship between the two variables such that the value of one variable changes and the other variable remains constant, it is called no or zero correlation.



Methods of Determining Correlation:

- Pearson's Coefficient of Correlation.
- Spearman's Rank Correlation Coefficient;
- Kendall

Pearson *r* correlation

Pearson *r* correlation is the most widely used correlation statistic to measure the degree of the relationship between linearly related variables. For example, in the stock market, if we want to measure how two stocks are related to each other, Pearson *r* correlation is used to measure the degree of relationship between the two. The point-biserial correlation is conducted with the Pearson correlation formula except that one of the variables is dichotomous. The following formula is used to calculate the Pearson *r* correlation:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

r_{xy} = Pearson r correlation coefficient between x and y

n = number of observations

x_i = value of x (for ith observation)

y_i = value of y (for ith observation)

Data preprocessing

Handling Missing Values

One of the important stages of data mining is preprocessing, where we prepare the data for mining. Real-world data tends to be incomplete, noisy, and inconsistent and an important task when preprocessing the data is to fill in missing values, smooth out noise and correct inconsistencies.

If we specifically look at dealing with missing data, there are several techniques that can be used. Choosing the right technique is a choice that depends on the problem domain — the data's domain and our goal for the data mining process.

So, there are several techniques which can be used for handling missing values.

1. Ignore the data row
This is usually done when the class label is missing (assuming your data mining goal is classification), or many attributes are missing from the row (not just one). However, you'll obviously get poor performance if the percentage of such rows is high.

2. Use a global constant to fill in for missing values
Decide on a new global constant value, like "unknown", "N/A" or minus infinity, that will be used to fill all the missing values.
This technique is used because sometimes it just doesn't make sense to try and predict the missing value.

3. Use attribute mean
Replace missing values of an attribute with the mean (or median if its discrete) value for that attribute in the database.

4. Use a data mining algorithm to predict the most probable value
The value can be determined using regression, inference based tools using Bayesian formalism, decision trees, clustering algorithms (K-Mean\Median etc.).

Feature Scaling- Normalization and Standardization

Normalization
Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.
Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

Standardization (z-score normalization_

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Here's the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

μ is the mean of the feature values and σ is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.

Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.

Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

Binning

Data binning, bucketing is a data pre-processing method used to minimize the effects of small observation errors. The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin. This has a smoothing effect on the input data and may also reduce the chances of overfitting in the case of small datasets.

There are 2 methods of dividing data into bins:

Equal Frequency Binning: bins have an equal frequency.

```
Input:[5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]
Output:
[5, 10, 11, 13]
[15, 35, 50, 55]
[72, 92, 204, 215]
```

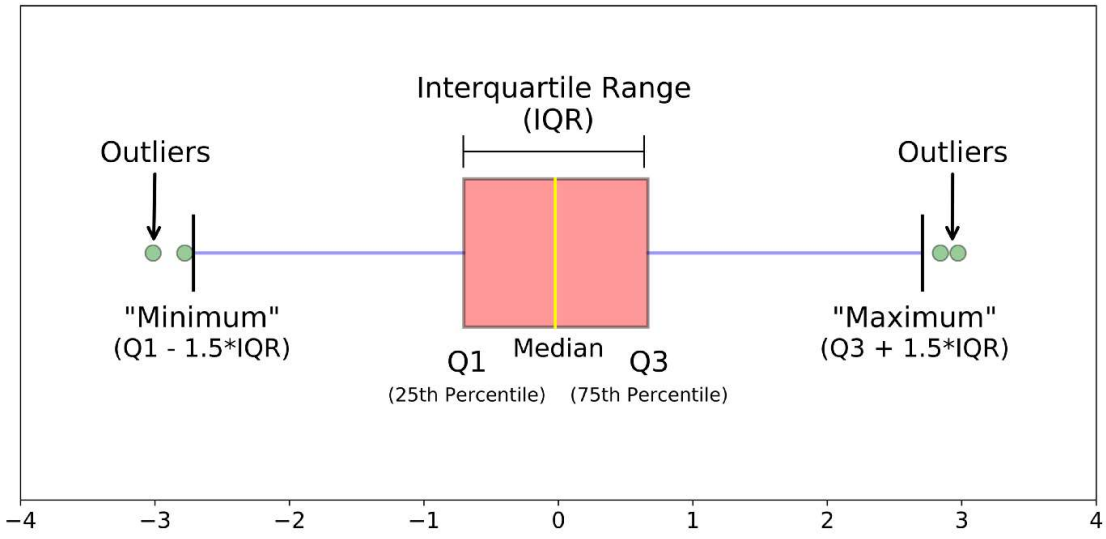
Equal Width Binning : bins have equal width with a range of each bin are defined as [min + w], [min + 2w] …. [min + nw] where w = (max – min) / (no of bins).

```
Input: [5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215]
[5, 10, 11, 13, 15, 35, 50, 55,72]

[92]

[204, 215]
```

Data visualization

BOX PLOT:
A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It

can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

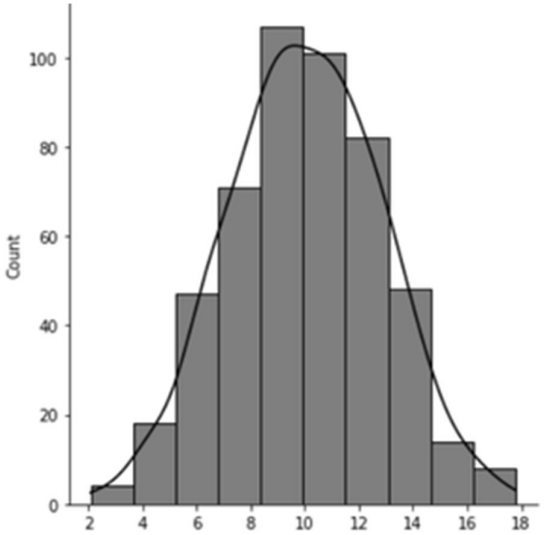


Histogram

A Histogram is a variation of a bar chart in which data values are grouped together and put into different classes. This grouping enables you to see how frequently data in each class occur in the dataset.

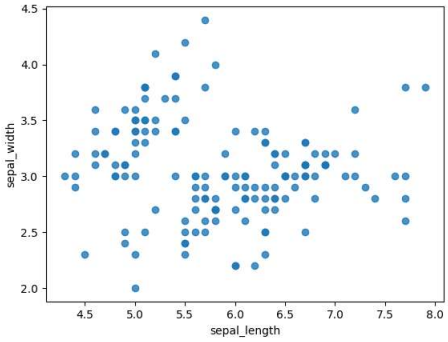
The histogram graphically shows the following:

- Frequency of different data points in the dataset.
- Location of the centre of data.
- The spread of dataset.



Scatter plot:

Scatter plots are **a commonly used data visualisation tool within data science**. They allow us to plot two numerical variables, as points, on a two-dimensional graph. From these plots, we can understand if there is a relationship between the two variables, and what the strength of that relationship is.



Implementation:

1) Load the libraries
2) Download the data set from classroom (realEstate_trans_full.csv)
3) Read the file –select appropriate file read function according to data type of file.
4) Display attributes in the data set-10 samples.
5) Describe the attributes name, count no of values, and find min, max, mean,data type, range, quartile, percentile.
6) Display correlation between any two attributes.
7) Download the dataset from classroom (realEstate_trans_less_dirty.csv)
8) Identify missing values fill them with mean.
9) Implement data normalization and standardization on real estate data.
10) Implement binning techniques on real estate data.

Code:

 # data normalization

```
import pandas as pd

csv_read = pd.read_csv('D:/DWM/Exp 4/realEstate_trans_imputed.csv')

def normalize(col):
    return (col - col.min()) / (col.max() - col.min())

def standardize(col):
    return (col - col.mean()) / col.std()

cols = ['price_mean','sq__ft']

for col in cols:
    csv_read['n_' + col] = normalize(csv_read[col])
    csv_read['s_' + col] = standardize(csv_read[col])
```

```
with open('D:/DWM/Exp 4/realEstate_trans_imputed2.csv', 'w') as write_csv:
    write_csv.write(csv_read.to_csv(sep=',', index=False))
```

 # data smoothing

```
import numpy as np
import pandas as pd

# read the data
csv_read = pd.read_csv('D:/DWM/Exp 4/realEstate_trans_imputed2.csv')

# create bins for the price that are based on the
# linearly spaced range of the price values

bins = np.linspace(
    csv_read['price_mean'].min(),
    csv_read['price_mean'].max(),
    6
)

# and apply the bins to the data
csv_read['b_price'] = np.digitize(
csv_read['price_mean'],
bins
)

# print out the counts for the bins
counts_b = csv_read['b_price'].value_counts()
print(counts_b.sort_index())

# and write to a file

with open('D:/DWM/Exp 4/realEstate_trans_binning.csv', 'w') as write_csv:
    write_csv.write(csv_read.to_csv(sep=',', index=False))

# OutPut :-
# 1    350
# 2    480
# 3    118
# 4     26
# 5      4
# 6      3
# Name: b_price, dtype: int64
```

 # handling missing value:

```
import pandas as pd
# read the data
```

```
csv_read =
pd.read_csv('/home/universe/Desktop/9689/realEstate_trans_less_dirty -
realEstate_trans_less_dirty.csv')

# impute mean in place of NaNs

csv_read['price_mean'] =
csv_read['price'].fillna(csv_read.groupby('zip')['price'].transform('mean'))

# impute median in place of NaNs
csv_read['price_median'] =
csv_read['price'].fillna(csv_read.groupby('zip')['price'].transform('median')
)

# and write to a file
with open('/home/universe/Desktop/9689/realEstate_trans_imputed.csv', 'w') as
write_csv:
    write_csv.write(csv_read.to_csv(sep=',' ,index=False))
```

 # mean, median and correlation

```
import pandas as pd
import numpy as np

dataframe =
pd.read_csv('/home/universe/Desktop/9689/realEstate_trans_less_dirty -
realEstate_trans_less_dirty.csv')
# print(dataframe.columns)
dtype = (dataframe.dtypes)
# print(dtype)

# print(dataframe.head(10))
# print(dataframe[['beds','baths', 'sq__ft']].describe)

#Mean
print(dataframe[['beds','baths', 'sq__ft']].mean())
#Mode
print(dataframe[['beds','baths', 'sq__ft']].mode())
#Median
print(dataframe[['beds','baths', 'sq__ft']].median())
#Standrad Deviation
print(dataframe[['beds','baths', 'sq__ft']].std())


# Coorelation
print(dataframe[['beds','baths', 'sq__ft']].corr(method = "pearson"))
```

```
print("\nKendall")
print(dataframe[['beds','baths', 'sq__ft']].corr(method = "kendall"))


print("\nSpearman")
print(dataframe[['beds','baths', 'sq__ft']].corr(method = "spearman"))


dataframe['price_mean'] =
dataframe['price'].fillna(dataframe.groupby('zip')['price'].transform('mean'))
dataframe['price_median'] =
dataframe['price'].fillna(dataframe.groupby('zip')['price'].transform('median'
))
```

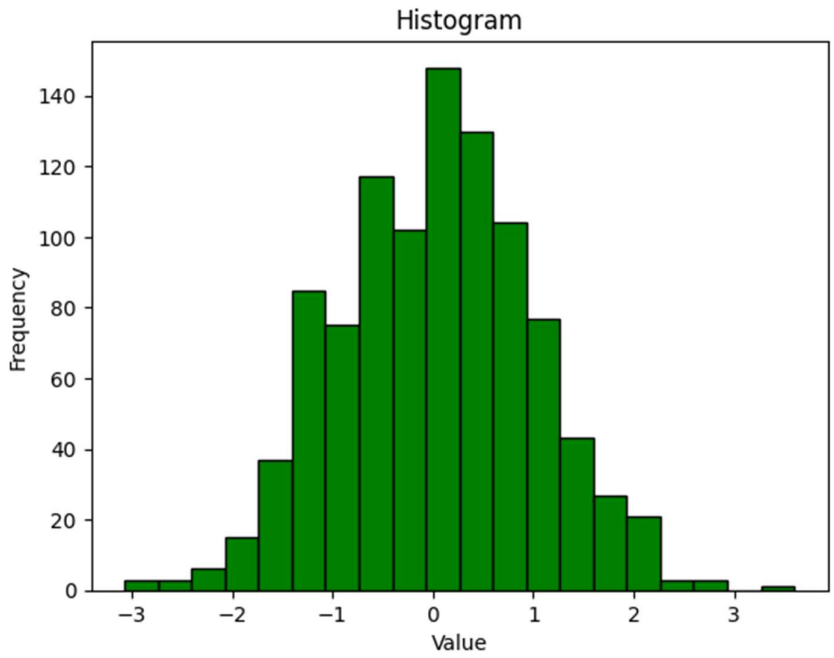
Post lab:

1. Apply data visualization for statistical description of data – in the form of histogram, scatter plot and box plot.

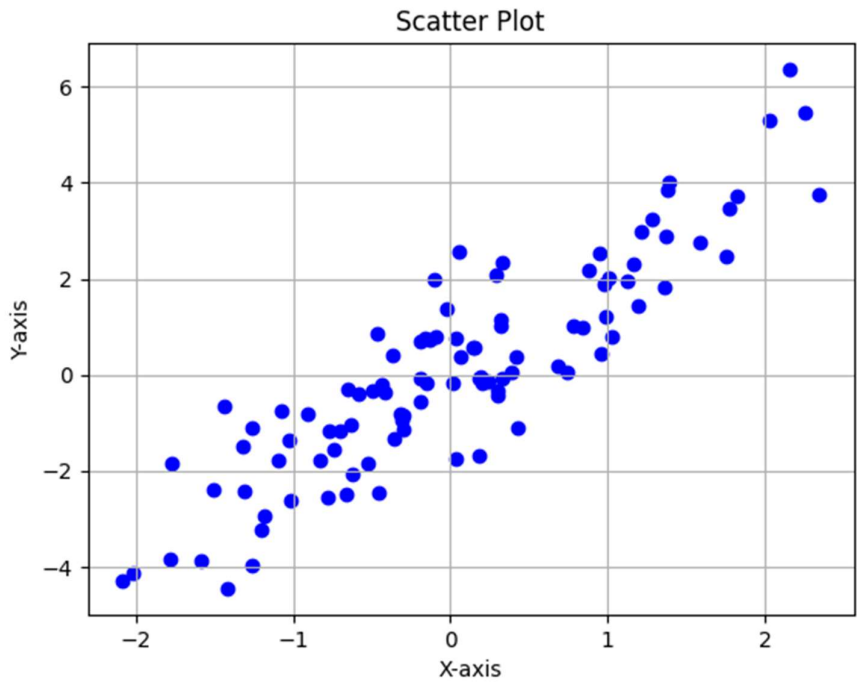
Code:

```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Generate some example data
data = np.random.randn(1000)  # Replace with your own dataset

# Create a histogram
plt.hist(data, bins=20, color='green', edgecolor='black')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram')
plt.show()
```




```
# Scatter Plot
# Generate some example data
x = np.random.randn(100)
y = 2 * x + np.random.randn(100)  # Replace with your own dataset

# Create a scatter plot
plt.scatter(x, y, c='blue', marker='o')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.grid(True)
plt.show()
```



Conclusion:
I am able to understand the different data preprocessing technique and able to implement the code.

Experiment No: 05	TE AI&DS
Date of Performance:	Roll No: 9696
Aim: To Implement Naïve Bayesian classification algorithm to build classification model and predict class for unseen data.	
Related CO5: Implement Classification, Clustering and Association mining techniques to extract knowledge	
Objective: To learn how classification is used for Prediction.	

Rubrics for assessment of Experiment:

Sr. No	Parameters	Exceed Expectations(EE)	Meet Expectations (ME)	Below Expectations (BE)
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

Theory:

Naive Bayes classifiers are a family of "probabilistic classifiers" based on Bayes' theorem with strong independence between the features. They are among the simplest Bayesian network models and are capable of achieving high accuracy levels.
Bayes theorem states mathematically as:
P(A|B) = (P(B|A) * P(A))/ P(B)
where A and B are events and P(B) != 0.
P(A|B) is a conditional probability: the probability of event A occurring given that B is true.
P(B|A) is also a conditional probability: the probability of event B occurring given that A is true.
P(A) and P(B) are the probabilities of observing A and B respectively without any given conditions.
A and B must be different events.

Bayes Theorem

- Based on prior knowledge of conditions that may be related to an event, Bayes theorem describes the probability of the event
- conditional probability can be found this way
- Assume we have a Hypothesis(*H*) and evidence(*E*),
  According to Bayes theorem, the relationship between the probability of Hypothesis before getting the evidence represented as *P(H)* and the probability of the hypothesis after getting the evidence represented as *P(H|E)* is:

  *P(H|E) = P(E|H)*P(H)/P(E)*
  - **Prior probability** = *P(H)* is the probability before getting the evidence
    Posterior probability = *P(H|E)* is the probability after getting evidence
  - In general,

  *P(class|data) = (P(data|class) * P(class)) / P(data)*
  Bayes Theorem Example
  Assume we have to find the probability of the randomly picked card to be king given that it is a face card.
  There are *4* Kings in a Deck of Cards which implies that *P(King) = 4/52*
  as all the Kings are face Cards so *P(Face|King) = 1*
  there are *3* Face Cards in a Suit of *13 cards* and there are *4 Suits* in total so *P(Face) = 12/52*
  Therefore,
  *P(King|face) = P(face|king)*P(king)/P(face) = 1/3*

We can frame classification as a conditional classification problem with Bayes Theorem as follows:

- P(yi | x1, x2, …, xn) = P(x1, x2, …, xn | yi) * P(yi) / P(x1, x2, …, xn)
  The prior *P(yi)* is easy to estimate from a dataset, but the conditional probability of the observation based on the class *P(x1, x2, …, xn | yi)* is not feasible unless the number of examples is extraordinarily large, e.g. large enough to effectively estimate the probability distribution for all different possible combinations of values.

Example

```
#Weather Dataset
Outlook   Temp    Humidity   Windy  Play

Rainy     Hot     High       f      no

Rainy     Hot     High       t      no

Overcast  Hot     High       f      yes

Sunny     Mild    High       f      yes

Sunny     Cool    Normal     f      yes

Sunny     Cool    Normal     t      no

Overcast  Cool    Normal     t      yes

Rainy     Mild    High       f      no

Rainy     Cool    Normal     f      yes

Sunny     Mild    Normal     f      yes

Rainy     Mild    Normal     t      yes

Overcast  Mild    High       t      yes

Overcast  Hot     Normal     f      yes

Sunny     Mild    High       t      no
```

Calculate Prior Probability of Classes P(y)

```
#Frequency tableP
(Play=Yes) = 9/14 = 0.64
P(Play=No) = 5/14 = 0.36
```

Calculate the Likelihood Table for all features

```
#Likelihood Table#Outlook

Play Overcast Rainy Sunny

Yes  4/9      2/9   3/9

No   0/5      3/5   2/5

     ___      ___   ___

     4/14     5/14  5/14
#Temp

Play  Cool  Mild  Hot

Yes   3/9   4/9   2/9

No    1/5   2/5   2/5

      ___   ___   ___

      4/14  6/14  4/14
#Humidity
```

```
Play   High   Normal

Yes    3/9    6/9

No     4/5    1/5

       ___    ___

       7/14   7/14
#Windy

Play    f      t

Yes     6/9    3/9

No      2/5    3/5

        ___    ___

        8/14   6/14
```

Now, Calculate Posterior Probability for each class using the Naive Bayesian equation. The Class with maximum probability is the outcome of the prediction.

Query: Whether Players will play or not when the weather conditions are [Outlook=Rainy, Temp=Mild, Humidity=Normal, Windy=t]?

Calculation of Posterior Probability:

Since Conditional independence of two random variables, A and B gave C holds just in case
P(A, B | C) = P(A | C) * P(B | C)

$$P(y=Yes|x) = P(Yes|Rainy,Mild,Normal,t)$$

$$= \frac{P(Rainy,Mild,Normal,t|Yes) * P(Yes)}{P(Rainy,Mild,Normal,t)}$$

$$= \frac{P(Rainy|Yes)*P(Mild|Yes)*P(Normal|Yes)*P(t|Yes)*P(Yes)}{P(Rainy)*P(Mild)*P(Normal)*P(t)}$$

$$= \frac{(2/9) * (4/9) * (6/9) * (3/9) * (9/14)}{(5/14) * (6/14) * (7/14) * (6/14)}$$

```
        = 0.43

P(y=No|x) = P(No|Rainy,Mild,Normal,t)

        P(Rainy,Mild,Normal,t|No) * P(No)

        = ____________________________________

                P(Rainy,Mild,Normal,t)

          P(Rainy|No)*P(Mild|No)*P(Normal|No)*P(t|No)*P(No)

        = ____________________________________________________

                    P(Rainy)*P(Mild)*P(Normal)*P(t)

        (3/5) * (2/5) * (1/5) * (3/5) * (5/14)

        = ____________________________________

             (5/14) * (6/14) * (7/14) * (6/14)


        = 0.31

Now, P(Play=Yes|Rainy,Mild,Normal,t) has the highest Posterior probability.
```

Algorithm/steps:

1) Load the dataset and convert it into list.
2) Separating the data as per class(0,1)-mp[0], mp[1]
3) define the test input: test = [2,1,0,1]
4) find the prior probability of each class
5) Find the conditional probability of test for each class-[yes,no]

```
probYes = 1 // probNO = 1
count = 0
total = 0
//find the length of test
for i in range(len(test)):
count = 0
total = 0
for row in mp[1]: //mp[0] for NO
if(test[i] == row[i]):
  count += 1
total += 1
probYes *= count/total //probNO*= count/total
```

6) Display the posterior probability of each class and find maximization

Code with output:

Code:-

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import math


def accuracy_score(y_true, y_pred):

        """          score = (y_true - y_pred) / len(y_true) """

        return round(float(sum(y_pred == y_true))/float(len(y_true)) * 100 ,2)

def pre_processing(df):

        """ partioning data into features and target """

        X = df.drop([df.columns[-1]], axis = 1)
        y = df[df.columns[-1]]

        return X, y



class  NaiveBayes:

        """
          Bayes Theorem:

        Likelihood * Class prior probability
                                Posterior Probability = ------------------------------------

        Predictor prior probability

                                                                                        P(x|c)
* p(c)
                                                                P(c|x) = ------------------

          P(x)
        """

        def __init__(self):

          """
                        Attributes:
                                likelihoods: Likelihood of each feature per class
                                class_priors: Prior probabilities of classes
                                pred_priors: Prior probabilities of features
                                features: All features of dataset
          """
          self.features = list
          self.likelihoods = {}
          self.class_priors = {}
```

```
        self.pred_priors = {}

        self.X_train = np.array
        self.y_train = np.array
        self.train_size = int
        self.num_feats = int

    def fit(self, X, y):

        self.features = list(X.columns)
        self.X_train = X
        self.y_train = y
        self.train_size = X.shape[0]
        self.num_feats = X.shape[1]

        for feature in self.features:
                self.likelihoods[feature] = {}
                self.pred_priors[feature] = {}

                for feat_val in np.unique(self.X_train[feature]):
                        self.pred_priors[feature].update({feat_val: 0})

                        for outcome in np.unique(self.y_train):

    self.likelihoods[feature].update({feat_val+'_'+outcome:0})
                                        self.class_priors.update({outcome: 0})

        self._calc_class_prior()
        self._calc_likelihoods()
        self._calc_predictor_prior()

    def _calc_class_prior(self):

        """ P(c) - Prior Class Probability """

        for outcome in np.unique(self.y_train):
                outcome_count = sum(self.y_train == outcome)
                self.class_priors[outcome] = outcome_count / self.train_size

    def _calc_likelihoods(self):

        """ P(x|c) - Likelihood """

        for feature in self.features:

                for outcome in np.unique(self.y_train):
                        outcome_count = sum(self.y_train == outcome)
                        feat_likelihood = self.X_train[feature][self.y_train[self.y_train
== outcome].index.values.tolist()].value_counts().to_dict()
```

```
                        for feat_val, count in feat_likelihood.items():
                                self.likelihoods[feature][feat_val + '_' + outcome] =
count/outcome_count


        def _calc_predictor_prior(self):

          """ P(x) - Evidence """

          for feature in self.features:
                feat_vals = self.X_train[feature].value_counts().to_dict()

                for feat_val, count in feat_vals.items():
                        self.pred_priors[feature][feat_val] = count/self.train_size


        def predict(self, X):

          """ Calculates Posterior probability P(c|x) """

          results = []
          X = np.array(X)

          for query in X:
                probs_outcome = {}
                for outcome in np.unique(self.y_train):
                        prior = self.class_priors[outcome]
                        likelihood = 1
                        evidence = 1

                        for feat, feat_val in zip(self.features, query):
                                likelihood *= self.likelihoods[feat][feat_val + '_' +
outcome]
                                evidence *= self.pred_priors[feat][feat_val]

                        posterior = (likelihood * prior) / (evidence)

                        probs_outcome[outcome] = posterior

                result = max(probs_outcome, key = lambda x: probs_outcome[x])
                results.append(result)

          return np.array(results)



if __name__ == "__main__":

        #Weather Dataset
        print("\nWeather Dataset:")
```

```
df = pd.read_table("Weather_dataset.txt")
#print(df)

#Split fearures and target
X,y  = pre_processing(df)

nb_clf = NaiveBayes()
nb_clf.fit(X, y)

print("Train Accuracy: {}".format(accuracy_score(y, nb_clf.predict(X))))

#Query:
query = np.array([['Rainy','Mild', 'Normal', 't']])
print("Query 1:- {} ---> {}".format(query, nb_clf.predict(query)))
```

```
# Output:-
# Weather Dataset:
# Train Accuracy: 92.4
# Query 1:- [['Rainy' 'Mild' 'Normal' 't']] ---> ['Overcast  Cool    Normal    t     yes']
```

Part 2:

```python
import pandas as pd
from sklearn.naive_bayes import CategoricalNB
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

# Load Weather Dataset
df = pd.read_csv("Weather_dataset.txt")

# Encode the target variable (class labels)
le = LabelEncoder()
y = le.fit_transform(df['Play'])

# Encode the categorical features
X = df.drop('Play', axis=1).apply(le.transform)

# Create and train the Naive Bayes classifier
nb_clf = CategoricalNB()
nb_clf.fit(X, y)

# Predict and calculate accuracy on the training data
y_pred = nb_clf.predict(X)
accuracy = accuracy_score(y, y_pred)
```

```
print("Train Accuracy: {:.2f}%".format(accuracy * 100))

# Query:
query = pd.DataFrame({'Outlook': ['Rainy'], 'Temperature':
['Mild'], 'Humidity': ['Normal'], 'Windy': ['False']})

# Use the same LabelEncoder for query data
query_encoded = query.apply(lambda col: le.transform(col))

prediction = nb_clf.predict(query_encoded)
print("Query 1:", le.inverse_transform(prediction))
```

Output:-
Weather Dataset:
Train Accuracy: 92.4
Query 1:- [['Rainy' 'Mild' 'Normal' 't']] ---> ['Overcast Cool   Normal   t   yes']

References:

Naive Bayes Classification Program in Python from Scratch - japp.io
Naïve Bayes Algorithm -Implementation from scratch in Python. | by ranga_vamsi | Medium

ML | Naive Bayes Scratch Implementation using Python - GeeksforGeeks
How to Develop a Naive Bayes Classifier from Scratch in Python (machinelearningmastery.com)

Conclusion:
I am able to understand the navie bayes theorem and am able to implement the code

Experiment No: 06	TE AI&DS
Date of Performance:	Roll No: 9696
Aim: : To analyze and evaluate the performance of different classification techniques using WEKA tool	
Related CO5: To analyze and evaluate perform of data mining techniques applied on large dataset using open-source tool for data mining	
Objective: To learn WEKA(Data Mining tool) and analyze and evaluate classification algorithm performance.	

Rubrics for assessment of Experiment:

Sr. No	**Parameters**	**Exceed Expectations(EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

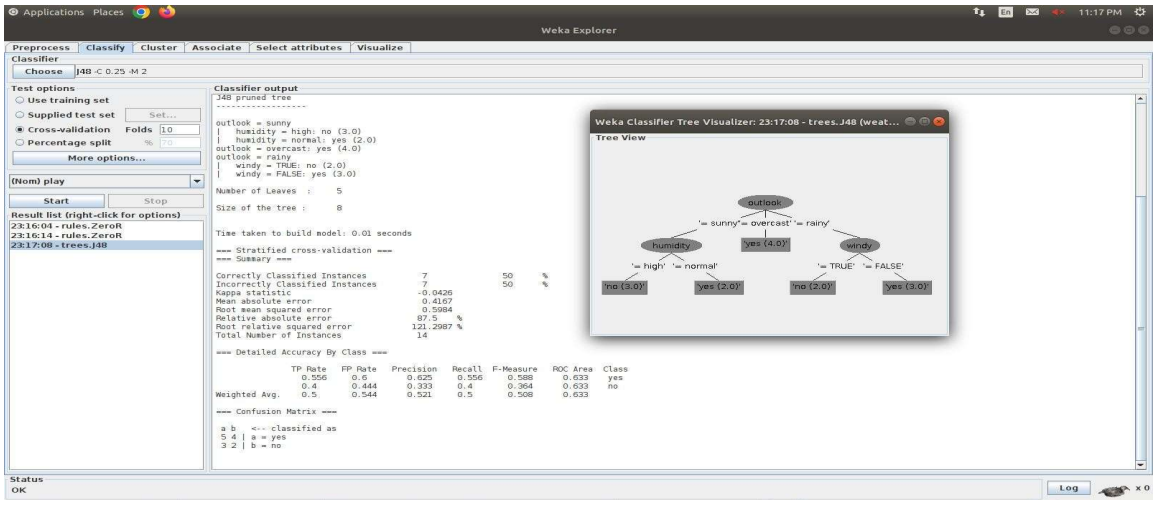
Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

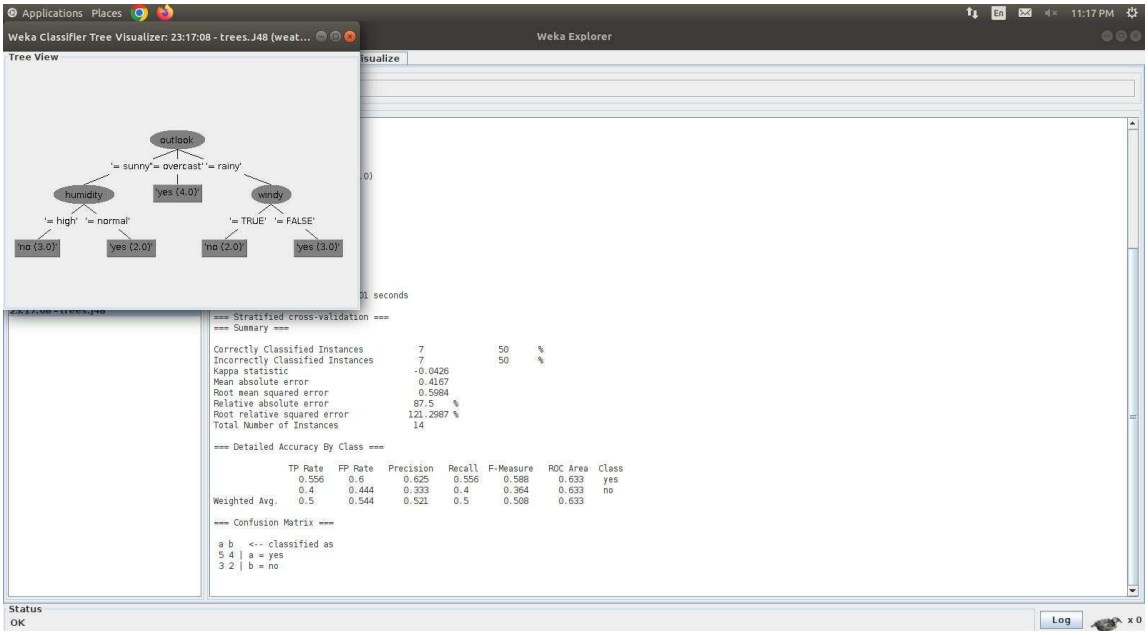
Theory:

WEKA is a data mining system developed by the University of Waikato in New Zealand that implements data mining algorithms. It is a collection of machine learning algorithms for data mining tasks. The algorithms are applied directly to a dataset.
WEKA implements algorithms for data preprocessing, classification, regression, clustering, association rules; it also includes a visualization tools. When 'WEKA GUI Chooser' window appears on the screen, we can select one of the four options.
The options are 1. Simple CLI : provides a simple command line interface and allows direct execution of WEKA commands. 2. Explorer : is an environment for exploring data. 3. Experimenter : is an environment for performing experiments and conducting statistical tests between learning schemes. 4. Knowledge Flow : is a Java-Beans-based interface for setting up and running machine learning experiments. Classifiers in WEKA are the models for predicting nominal or numeric quantities. The learning schemes available in WEKA include decision trees, Bayes net, neural network, support vector machine and so on. In this experiment analysis and evaluation of three classification algorithm: Naive Bayesian algorithm, C4.5 algorithm, zero R is done. Before running the classification algorithm it is required to set test options. The selected test options were : 1. Use training set : Evaluates the classifier on how well it predicts the class of the instances it was trained on. 2. Cross-validation : Evaluates the classifier by cross-validation, using the number of folds (10)
3. Percentage split : Evaluates the classifier on how well it predicts a certain percentage of the data, which is held out for testing. In the classifier evaluation options following options are checked : 1. Output model : The output is the classification model on the full training set. 2. Output per-class stats : The precision/recall and true/false statistics for each class output 3. Output confusion matrix : The confusion matrix of one classifier's prediction is included in the output When training set is complete, the 'classifier' output area on the right panel of the classifier window is filled with text describing the results of training and testing.

Practical Exercise:

Apply and evaluate the result for different classification techniques on various datasets using WEKA.





Post lab Questions:

1. **Explain with example different measures to check the performance of a classifier.**

To check how well a classifier is doing, we use different measures:

Accuracy: It's the percentage of correct predictions. For instance, if it's 80%, it means the classifier gets things right 80% of the time.

Precision: This tells us how many of the positive predictions are actually correct. If it's 90%, it means 90% of the times the classifier says something is positive, it's right.

Recall: This is about finding out how many of the actual positive cases the classifier gets right. A 70% recall means the classifier misses 30% of the actual positive cases.

F1 Score: This is a combination of precision and recall. It's useful when we want to balance getting things right and not missing out on positive cases.

AUC-ROC: This measures how well the classifier can distinguish between good and bad cases.

For example, let's say we have a classifier that predicts if a customer will cancel their subscription. When we tested it, we found:

Accuracy: 80% (It's right 80% of the time).
Precision: 90% (When it predicts cancellations, it's right 90% of the time).
Recall: 70% (It misses 30% of actual cancellations).
F1 Score: 80% (A good balance of precision and recall).
AUC-ROC: 0.90 (It's good at telling good from bad).
Overall, this classifier seems pretty good, but it's not perfect. It's great at predicting cancellations,

but it does miss some actual cancellations, which could be due to various reasons, like the types of customers or the test data. So, while it's good, it's not perfect.

2. Explain the result of any one classifier.

Here's an illustration of what happens when a classifier is trained to determine if an email is spam:

The following metrics can be computed using the confusion matrix:

- Accuracy: (TP + TN) / (TP + TN + FP + FN) = 80%
- Precision: TP / (TP + FP) = 90%
- Recall: TP / (TP + FN) = 70%
- F1 score: 2 * (Precision * Recall) / (Precision + Recall) = 80%

These metrics suggest that the classifier is performing quite well, with an accuracy of 80%, precision of 90%, and recall of 70%. However, it is important to keep in mind thatthe recall is slightly lower, so the classifier may be missing some of the actual spam emails.

Choosing the right metric

The particular problem we are attempting to solve will determine which metric is most appropriate to use when assessing a classifier.

To ensure that you don't overlook any fraudulent transactions, for instance, having a high recall is crucial when developing a classifier for fraud detection.

To avoid misclassifying any legitimate emails as spam, it is more crucial to have a high precision when developing a classifier to filter spam emails.

It is also important to consider the class imbalance of your data. If the data is imbalanced, then some metrics, such as accuracy, can be misleading. In this case, it is better to use metrics that take into account the class imbalance, such as the F1 score or AUC-ROC.

Tuple	class	Prob	TP	FP	TPR	FPR
1	P	0.9	1	0	0.2	0
2	P	0.8	2	0	0.4	0
3	N	0.7	2	1	0.4	0.2
4	P	0.6	3	1	0.6	0.2
5	P	0.55	4	1	0.8	0.2
6	N	0.54	4	2	0.8	0.4
7	N	0.53	4	3	0.8	0.6
8	N	0.51	4	4	0.8	0.8
9	P	0.5	5	4	1.0	0.8
10	N	0.4	5	5	1.0	1.0

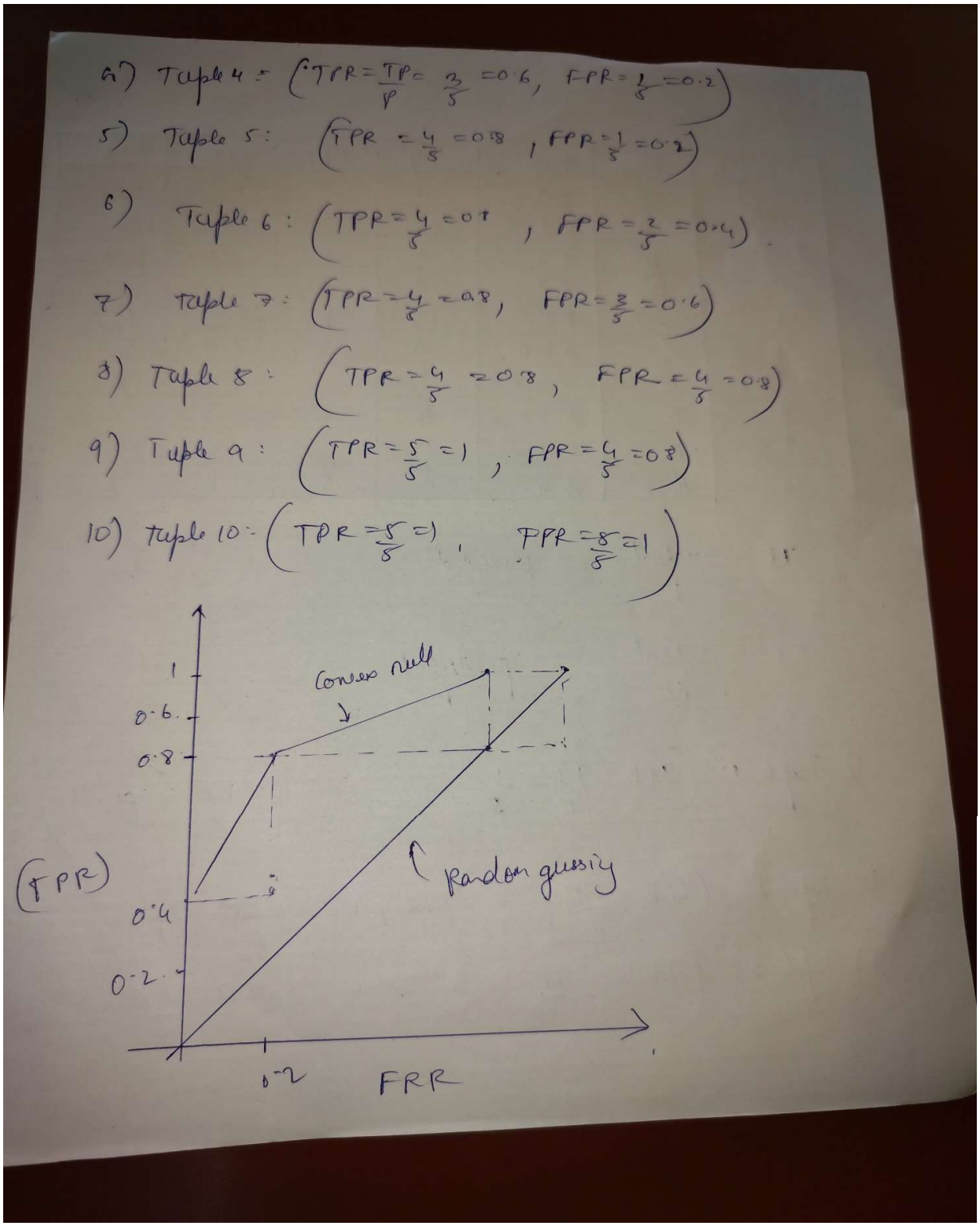
TP = actual P, predicted P

FP = actual N, predicted P

1) Tuple 1: $\left(TPR = \frac{TP}{P} = \frac{1}{5} = 0.2,\ FPR = \frac{FP}{F} = \frac{0}{5} = 0\right)$

2) Tuple 2: $\left(TPR = \frac{TP}{P} = \frac{2}{5} = 0.4,\ FPR = \frac{FP}{P} = 0\right)$

3) Tuple 3: $\left(TPR = \frac{TP}{P} = \frac{2}{5} = 0.4,\ FPR = \frac{FP}{P} = \frac{1}{5} = 0.2\right)$



Conclusion:

We learnt to analyze and evaluate the performance of different classification techniques using WEKA tool

Experiment No: 07	TE AI&DS
Date of Performance:	Roll No: 9696
Aim: To Implement K-mean clustering algorithm to group similar data objects	
Related CO4: Implement Classification, Clustering and Association mining techniques to extract knowledge	
Objective: To learn clustering techniques	

Rubrics for assessment of Experiment:

Sr. No	Parameters	Exceed Expectations(EE)	Meet Expectations (ME)	Below Expectations (BE)
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

Theory:

What is Clustering?

Clustering is dividing data points into homogeneous classes or clusters:

- Points in the same group are as similar as possible
- Points in different group are as dissimilar as possible

When a collection of objects is given, we put objects into group based on similarity.

Application of Clustering:

Clustering is used in almost all the fields. You can infer some ideas from Example 1 to come up with lot of clustering applications that you would have come across.
Listed here are few more applications, which would add to what you have learnt.

- Clustering helps marketers improve their customer base and work on the target areas. It helps group people (according to different criteria's such as willingness, purchasing power etc.) based on their similarity in many ways related to the product under consideration.
- Clustering helps in identification of groups of houses on the basis of their value, type and geographical locations.
- Clustering is used to study earth-quake. Based on the areas hit by an earthquake in a region, clustering can help analyse the next probable location where earthquake can occur.

Clustering Algorithms:

A Clustering Algorithm tries to analyse natural groups of data on the basis of some similarity. It locates the centroid of the group of data points. To carry out effective clustering, the algorithm evaluates the distance between each point from the centroid of the cluster.
The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data.

K-Means Clustering Algorithm-

K-Means Clustering Algorithm involves the following steps-

Step-01:

- Choose the number of clusters K.

Step-02:

- Randomly select any K data points as cluster centers.
- Select cluster centers in such a way that they are as farther as possible from each other.

Step-03:

- Calculate the distance between each data point and each cluster center.

- The distance may be calculated either by using given distance function or by using euclidean distance formula.

Step-04:

- Assign each data point to some cluster.
- A data point is assigned to that cluster whose center is nearest to that data point.

Step-05:

- Re-compute the center of newly formed clusters.
- The center of a cluster is computed by taking mean of all the data points contained in that cluster.

Step-06:

Keep repeating the procedure from Step-03 to Step-05 until any of the following stopping criteria is met-

- Center of newly formed clusters do not change
- Data points remain present in the same cluster
- Maximum number of iterations are reached

Implementation:

Part 1: WAP in Python/java to implement K-means clustering algorithm

Suppose That The Data Mining Task Is To Cluster Points (With *.X, Y/* Representing Location)
Into Three Clusters, Where The Points Are
A(2,10),B(2,5),C(8,4),D(5,8),*E(*7,5),*F(*6,4),G(1,2),*H(*4,9).

Read No of Objects
Read all Object Details
Read K
Give Output In Step Wise/Iteration Wise (distance matrix with cluster assignment)

Part 2: using standard libraries / packages

K-Means Clustering with scikit-learn - DataCamp

Post lab:

1. Differentiate K-means and K-medoid algorithms with one example
2. List the packages/ libraries of python used in experiments.

Other resources:
sklearn.cluster.KMeans — scikit-learn 0.24.1 documentation (scikit-learn.org)
Build K-Means from scratch in Python | by Rishit Dagli | Medium
K-Means Clustering Algorithm from Scratch - ML+ (machinelearningplus.com)
10 Clustering Algorithms With Python (machinelearningmastery.com)

Code with output:

Part 1:

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
# Points
# A(2,10),B(2,5),C(8,4),D(5,8),E(7,5),F(6,4),G(1,2),H(4,9).
style.use('ggplot')
X = np.array([[2, 10], # A
              [2, 5], # B
              [8, 4], # C
              [5, 8], # D
              [7, 5], # E
              [6, 4], # F
              [1, 2], # G
              [4, 9]  # H
               ])

plt.scatter(X[:,0], X[:,1], s=150)
plt.show()

#defining functions
class K_Means:
    def __init__(self, k=2, tol=0.001, max_iter=300):
        # Initialize K-Means parameters: number of clusters (k), tolerance, and maximum
iterations
        self.k = k
        self.tol = tol
        self.max_iter = max_iter

    def fit(self, data):
        # Initialize centroids for each cluster
        # Randomly assign the first k data points as centroids
        self.centroids = {}
        for i in range(self.k):
            self.centroids[i] = data[i]

        # Iterate to update centroids and cluster assignments
        for i in range(self.max_iter):
            self.classifications = {}  # Assignments of data points to clusters

            # Initialize classifications
            for i in range(self.k):
                self.classifications[i] = []

            # Assign each data point to the nearest centroid
            for featureset in data:
```

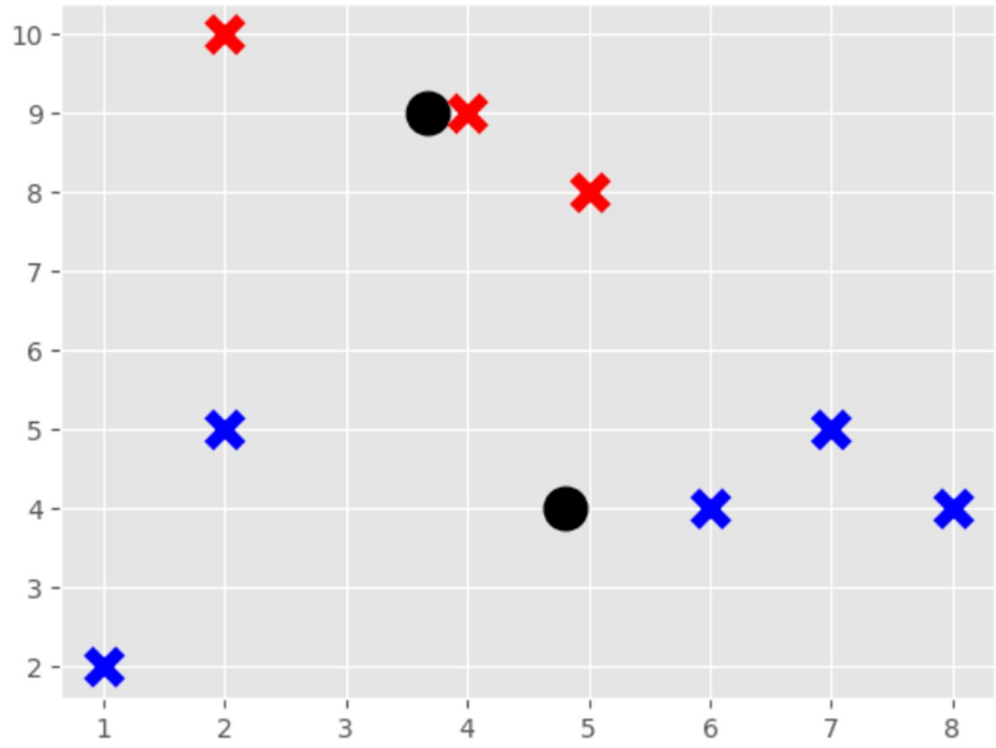
```
            distances = [np.linalg.norm(featureset - self.centroids[centroid]) for centroid in
self.centroids]
            classification = distances.index(min(distances))
            self.classifications[classification].append(featureset)

        prev_centroids = dict(self.centroids)

        # Update centroids based on the mean of points in each cluster
        for classification in self.classifications:
            self.centroids[classification] = np.average(self.classifications[classification],
axis=0)

        # Calculate the distance matrix for each centroid
        dist_matrix = np.zeros((self.k, len(data)))
        for centroid_idx in range (self.k):
            for i, feature_set in enumerate(data):
                dist_matrix[centroid_idx][i] = np.linalg.norm(feature_set -
self.centroids[centroid_idx])
                print(f"Distance Matrix for Iteration {i + 1}:")
                print(dist_matrix)

        # Print the distance matrix
#           print(f"Distance Matrix for Iteration {i + 1}:")
#           print(dist_matrix)

        # Check for convergence based on centroid movement
        optimized = True
        for c in self.centroids:
            original_centroid = prev_centroids[c]
            current_centroid = self.centroids[c]
            if np.sum((current_centroid - original_centroid) / original_centroid * 100.0) >
self.tol:
                optimized = False

        # Break if centroids have converged
        if optimized:
            break

    def predict(self, data):
        # Predict the cluster for a given data point (find the nearest centroid)
        distances = [np.linalg.norm(data - self.centroids[centroid]) for centroid in self.centroids]
        classification = distances.index(min(distances))
        return classification

#main function
model = K_Means()  # Create a K-Means model instance
model.fit(X)  # Fit the model to the data

# Plot the centroids of each cluster
for centroid in model.centroids:
```

```
    plt.scatter(model.centroids[centroid][0], model.centroids[centroid][1],
                marker="o", color="k", s=150, linewidths=5)

# Plot the data points for each cluster
for classification in model.classifications:
    # Assign a different color to each cluster
    color = 'r' if classification == 0 else 'b'
    for featureset in model.classifications[classification]:
        plt.scatter(featureset[0], featureset[1], marker="x", color=color, s=150, linewidths=5)

plt.show()


# Output:
# distance matrix calculation for each iterations
Distance Matrix for Iteration 1:
[[1.94365063 0.         0.         0.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 2:
[[1.94365063 4.33333333 0.         0.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 3:
[[1.94365063 4.33333333 6.61647775 0.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 4:
[[1.94365063 4.33333333 6.61647775 1.66666667 0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 5:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 6:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 7:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
```

```
Distance Matrix for Iteration 8:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 1:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 2:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 3:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 4:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        4.00499688 0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 5:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        4.00499688 2.41660919 0.
  0.         0.        ]]
Distance Matrix for Iteration 6:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        4.00499688 2.41660919 1.2
  0.         0.        ]]
Distance Matrix for Iteration 7:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        4.00499688 2.41660919 1.2
  4.29418211 0.        ]]
Distance Matrix for Iteration 8:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        4.00499688 2.41660919 1.2
  4.29418211 5.06359556]]
Distance Matrix for Iteration 1:
[[1.94365063 0.         0.         0.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
```

```
Distance Matrix for Iteration 2:
[[1.94365063 4.33333333 0.         0.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 3:
[[1.94365063 4.33333333 6.61647775 0.         0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 4:
[[1.94365063 4.33333333 6.61647775 1.66666667 0.         0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 5:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 0.
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 6:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 7:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 8:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [0.         0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 1:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 0.         0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 2:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 0.         0.         0.         0.
  0.         0.        ]]
Distance Matrix for Iteration 3:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2        0.         0.         0.
  0.         0.        ]]
```

Distance Matrix for Iteration 4:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2   4.00499688 0.   0.
  0.   0.   ]]
Distance Matrix for Iteration 5:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2   4.00499688 2.41660919 0.
  0.   0.   ]]
Distance Matrix for Iteration 6:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2   4.00499688 2.41660919 1.2
  0.   0.   ]]
Distance Matrix for Iteration 7:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2   4.00499688 2.41660919 1.2
  4.29418211 0.   ]]
Distance Matrix for Iteration 8:
[[1.94365063 4.33333333 6.61647775 1.66666667 5.20683312 5.51764845
  7.49073502 0.33333333]
 [6.62117814 2.97321375 3.2   4.00499688 2.41660919 1.2
  4.29418211 5.06359556]]

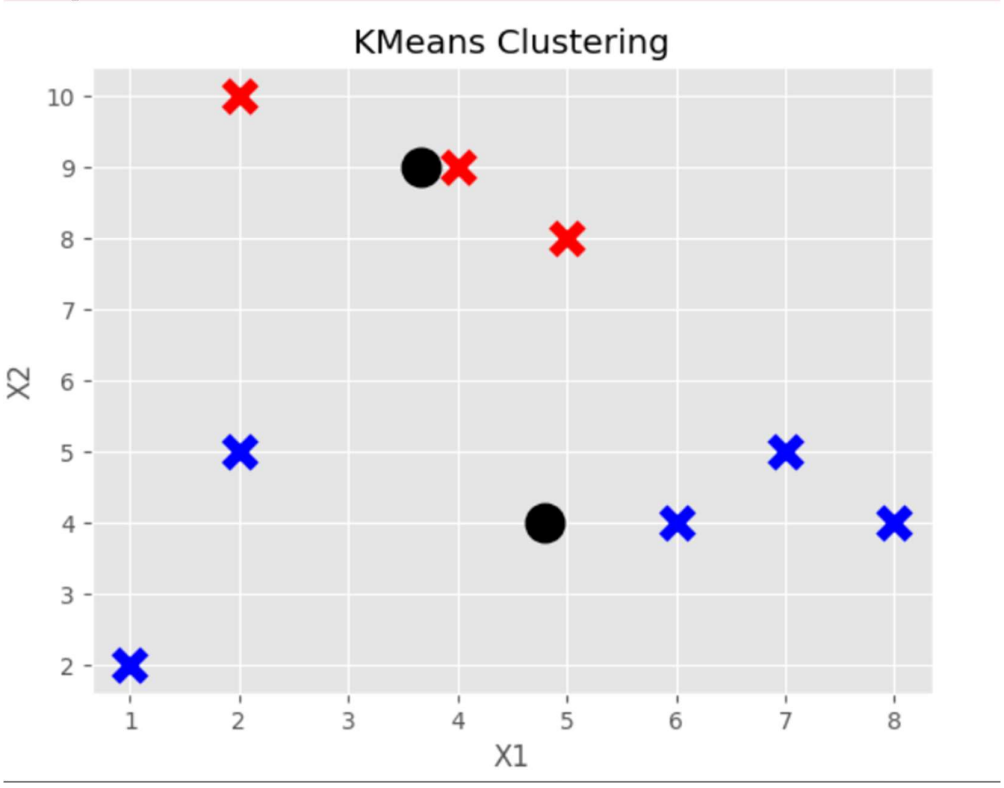


Part 2:

Code:

```
# Using scikit-learn's KMeans
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

# Plot the centroids of each cluster
plt.scatter(centroids[:, 0], centroids[:, 1], marker="o", color="k", s=150, linewidths=5)

# Plot the data points for each cluster
for cluster_num in range(kmeans.n_clusters):
    color = 'r' if cluster_num == 0 else 'b'
    plt.scatter(X[labels == cluster_num][:, 0], X[labels == cluster_num][:, 1],
                marker="x", color=color, s=150, linewidths=5)

plt.xlabel('X1')
plt.ylabel('X2')
plt.title('KMeans Clustering')
plt.show()
```

Output:



Post lab:

1. Differentiate K-means and K-medoid algorithms with one example

Ans:

1. K-Means:

• Centroid-Based Clustering: In order to reduce the sum of squared distances, K-Means divides data into K clusters according to the centroids.
• Centroid as Representative: The centroid of each cluster—which might not actually be a data point—serves as its representative.
• Sensitive to Outliers: Because squared distances are used, it is sensitive to outliers.
• Computational Efficiency: For large datasets, generally faster and more scalable.
One example would be to identify market segments by classifying customers based on t heir past purchases.

2. K-Medoid:

• Medoids-Based Clustering: Using real data points (medoids) as cluster representatives, K-Medoid clusters data into K groups.
• Using a Medium as a Representative: Real data points, which are more resilient to outliers, are used to represent each cluster.
• Robust to Outliers: Because absolute distances are used, it is less susceptible to outliers.
• Computational Complexity: When dealing with large datasets, computational complexity is higher than with K-Means.
• As an illustration, consider grouping cities according to travel times, with the medoid designating the location with the shortest overall distance to other cities.

3. List the packages/ libraries of python used in experiments.

Ans:

The following packages and libraries for Python were used in the code that you submitt ed:

1. matplotlib.pyplot: This programme is used to create plots and data visualisations, incl uding scatter plots.
2. numpy (np): Used to manipulate arrays and perform numerical operations.
3. matplotlib.style: This specifies the plot style, which is 'ggplot' in this example.
4. KMeans from sklearn.cluster: This K-Means clustering tool was imported from Scikit-Learn.

Conclusion:

We learnt to Implement K-mean clustering algorithm to group similar data objects.

Experiment No: 08	TE AI&DS
Date of Performance:	Roll No: 9696
Aim: : To analyze and evaluate the performance of different Clustering techniques using WEKA tool	
Related CO5: To analyze and evaluate perform of data mining techniques applied on large dataset using open-source tool for data mining	
Objective: To learn WEKA(Data Mining tool) and analyze and evaluate clustering algorithm performance.	

Rubrics for assessment of Experiment:

Sr. No	Parameters	Exceed Expectations(EE)	Meet Expectations (ME)	Below Expectations (BE)
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

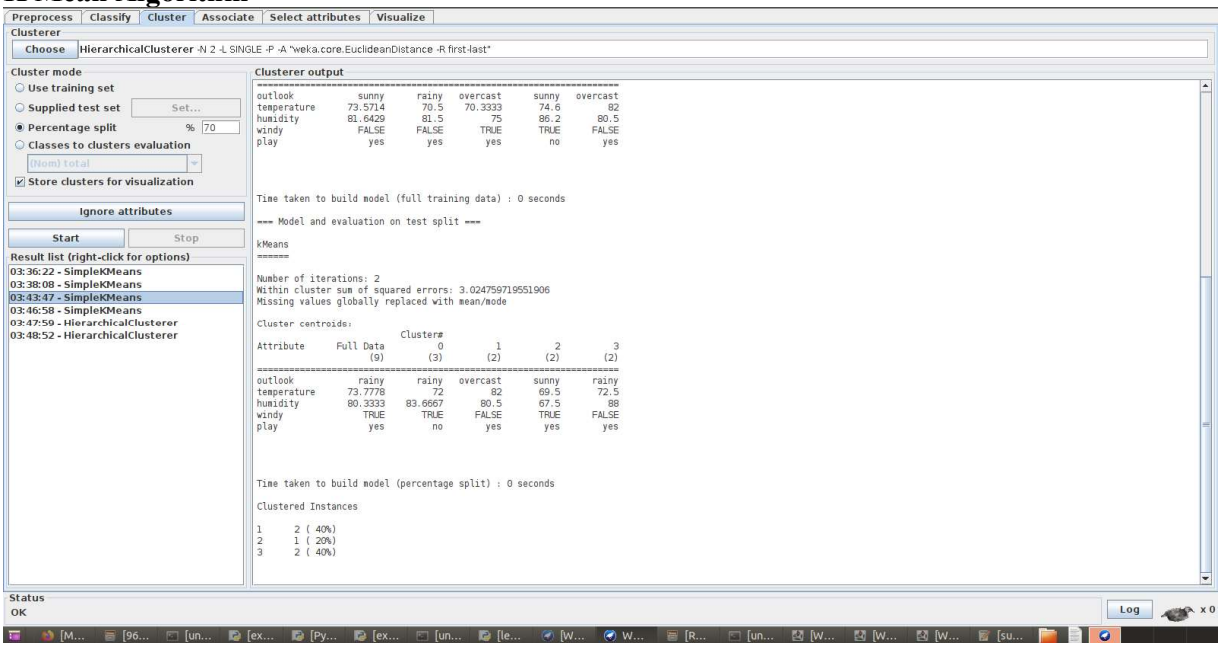
Theory:

WEKA is a data mining system developed by the University of Waikato in New Zealand that implements data mining algorithms. It is a collection of machine learning algorithms for data mining tasks. The algorithms are applied directly to a dataset.
WEKA implements algorithms for data preprocessing, classification, regression, clustering, association rules; it also includes a visualization tools. When 'WEKA GUI Chooser' window appears on the screen, we can select one of the four options.
The options are 1. Simple CLI : provides a simple command line interface and allows direct execution of WEKA commands. 2. Explorer : is an environment for exploring data. 3. Experimenter : is an environment for performing experiments and conducting statistical tests between learning schemes. 4. Knowledge Flow : is a Java-Beans-based interface for setting up and running machine learning experiments. Classifiers in WEKA are the models for predicting nominal or numeric quantities. The learning schemes available in WEKA include decision trees, Bayes net, neural network, support vector machine and so on. In this experiment analysis and evaluation of three classification algorithm: Naive Bayesian algorithm, C4.5 algorithm, zero R is done. Before running the classification algorithm it is required to set test options. The selected test options were : 1. Use training set : Evaluates the classifier on how well it predicts the class of the instances it was trained on. 2. Cross-validation : Evaluates the classifier by cross-validation, using the number of folds (10) 3. Percentage split : Evaluates the classifier on how well it predicts a certain percentage of the data, which is held out for testing. In the classifier evaluation options following options are checked : 1. Output model : The output is the classification model on the full training set. 2. Output per-class stats : The precision/recall and true/false statistics for each class output 3. Output confusion matrix : The confusion matrix of one classifier's prediction is included in the output When training set is complete, the 'classifier' output area on the right panel of the classifier window is filled with text describing the results of training and testing.

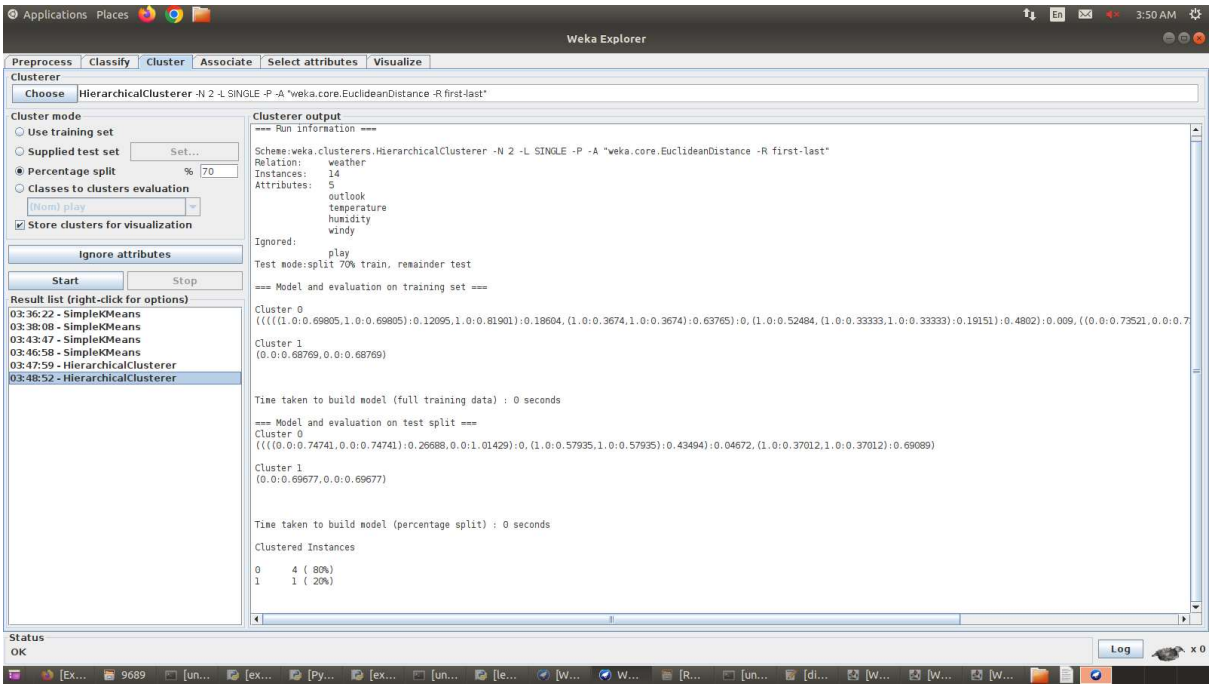
Practical Exercise:

Apply and evaluate the result for different classification techniques on various datasets using WEKA.

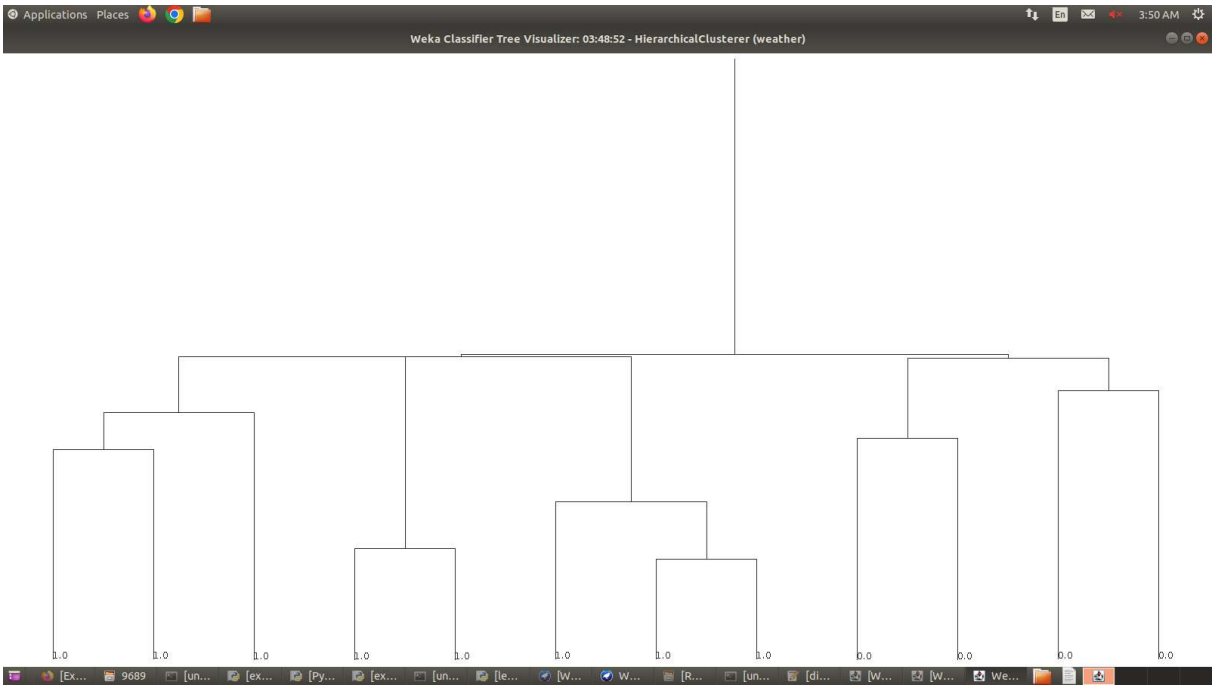
K Mean Algorithm



Hierarchical Clustring:



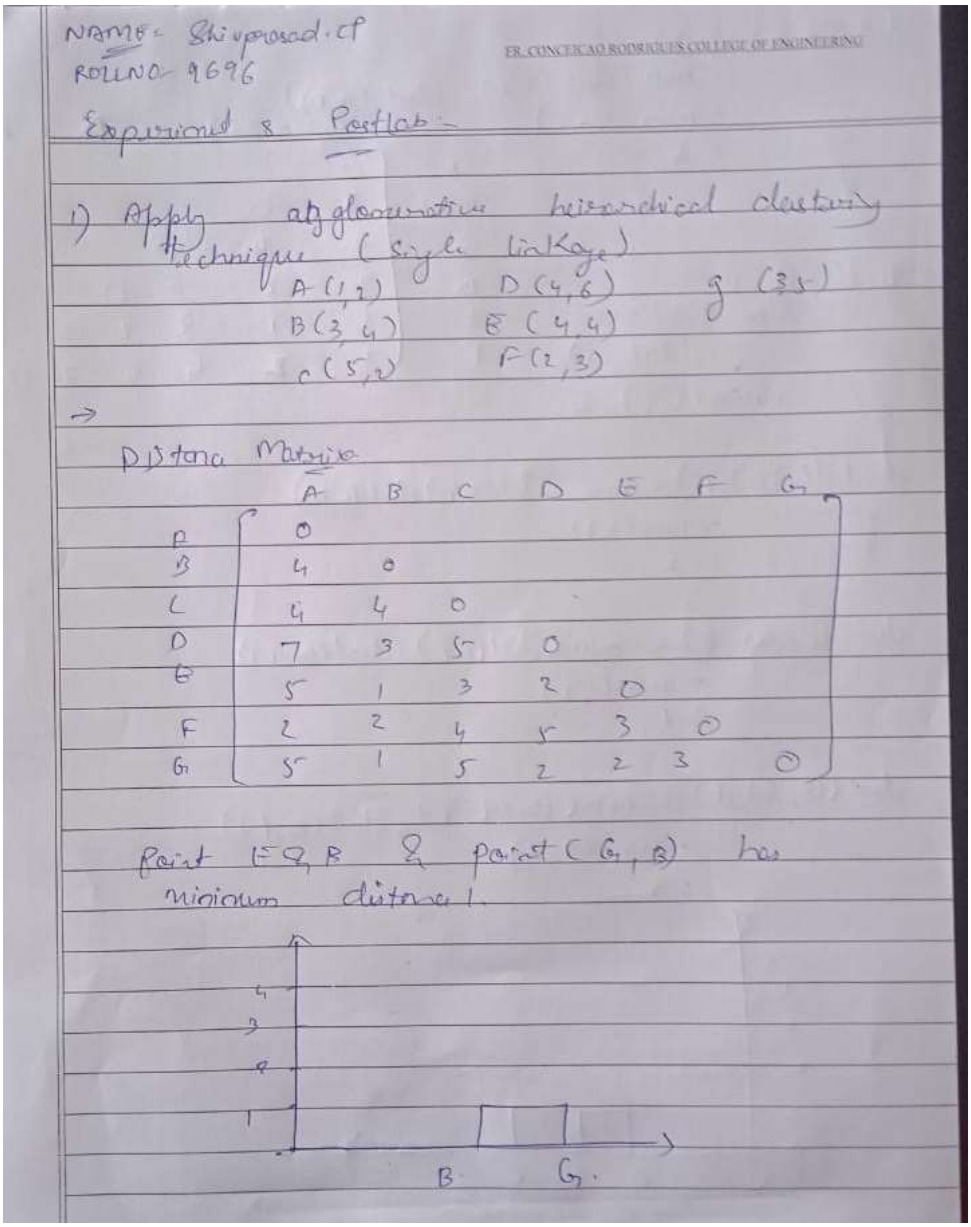
Dendogram:



Post lab Questions:

1. Apply agglomerative hierarchical clustering technique (single linkage) on following dataset.

   A(1,2)
   B(3,4)
   C(5,2)
   D(4,6)
   E(4,4)
   F(2,3)
   G(3,5)



NAME: Shivprasad.cP
ROLLNO: 9696
FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Experiment 8 Postlab:-

1) Apply agglomerative hierarchical clustering technique (Single Linkage)

A (1,2) D (4,6) g (3,5)
B (3,4) E (4,4)
C (5,2) F (2,3)

→

Distance Matrix

	A	B	C	D	E	F	G
A	0						
B	4	0					
C	4	4	0				
D	7	3	5	0			
E	5	1	3	2	0		
F	2	2	4	5	3	0	
G	5	1	5	2	2	3	0

Point E & B & point (G, B) has minimum distance 1.

dist((B,G), A) =
min (dist(B,A), dist(G,A))
= min (4,5)
= 4.

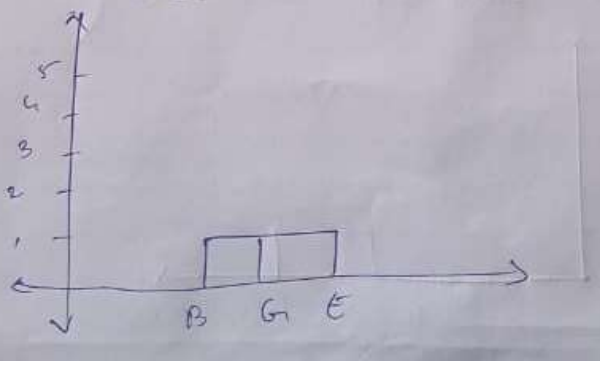
	A	(B,G)	C	D	E	F
A	0					
(B,G)	4	0				
G	4	4	0			
D	7	2	5	0		
E	5	1	3	2	0	
F	2	2	4	5	3	0

dist((B,G), C)
= min (dist(B,C), dist(G,C))
= min(4,5) = 4.

dist((B,G), D) = min (dist(B,D) dist(G,D))
= min(3,2)
= 2.

dist ((B,G), E) = min (dist(B,E), dist(G,E))
= min(1,2)
= 1.

dist((B,G),F) = min (dist(B,F), dist(G,F))
= min(2,3)
= 2.



dist(((B,G),E),A)

= min(dist(B,A), dist(G,A), dist(E,A))

= min(4,5,5)

= 4.

	A	(B,G),E)	C	D	F
A	0				
((B,G),E)	4	0			
C	4	3	0		
D	7	2	5	0	
F	2	2	4	5	0

dist(((B,G),E),C) = min(dist(B,C), dist(G,C), dist(E,C))

= min(4,5,3)

= 3

dist(((B,G),E),D) = min(dist(B,D), dist(G,D), dist(E,D))

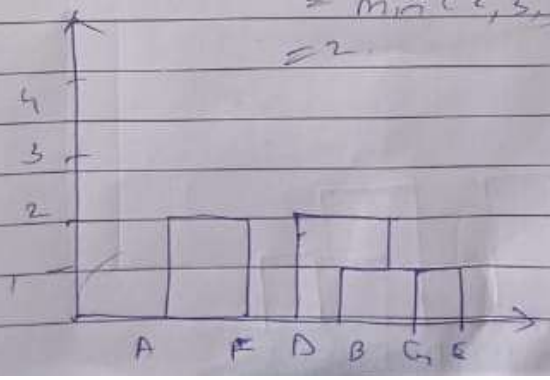
= min(3,5,2)

= 2

dist(((B,G),E),F) = min(dist(B,F), dist(G,F), dist(E,F))

= min(2,3,3)

= 2



dist ((A,F), C)

= min (dist (A,C), dist (F,C))

= min (4, 2)

= 2.

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

dist (A, F), (((B,G), E), D))

= dist min (dist (A,D), dist (F, D), dist (A, B),
dist (A,G), dist (A,E), dist (F, B)
dist (F, G), dist (F, E))

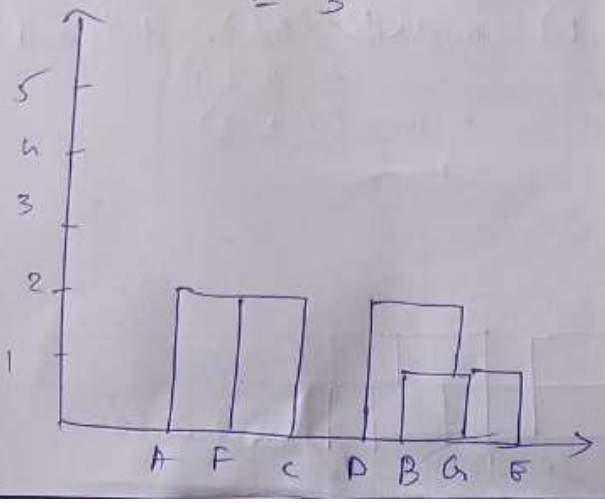
= min (7, 5, 4, 5, 5, 2, 3, 3)

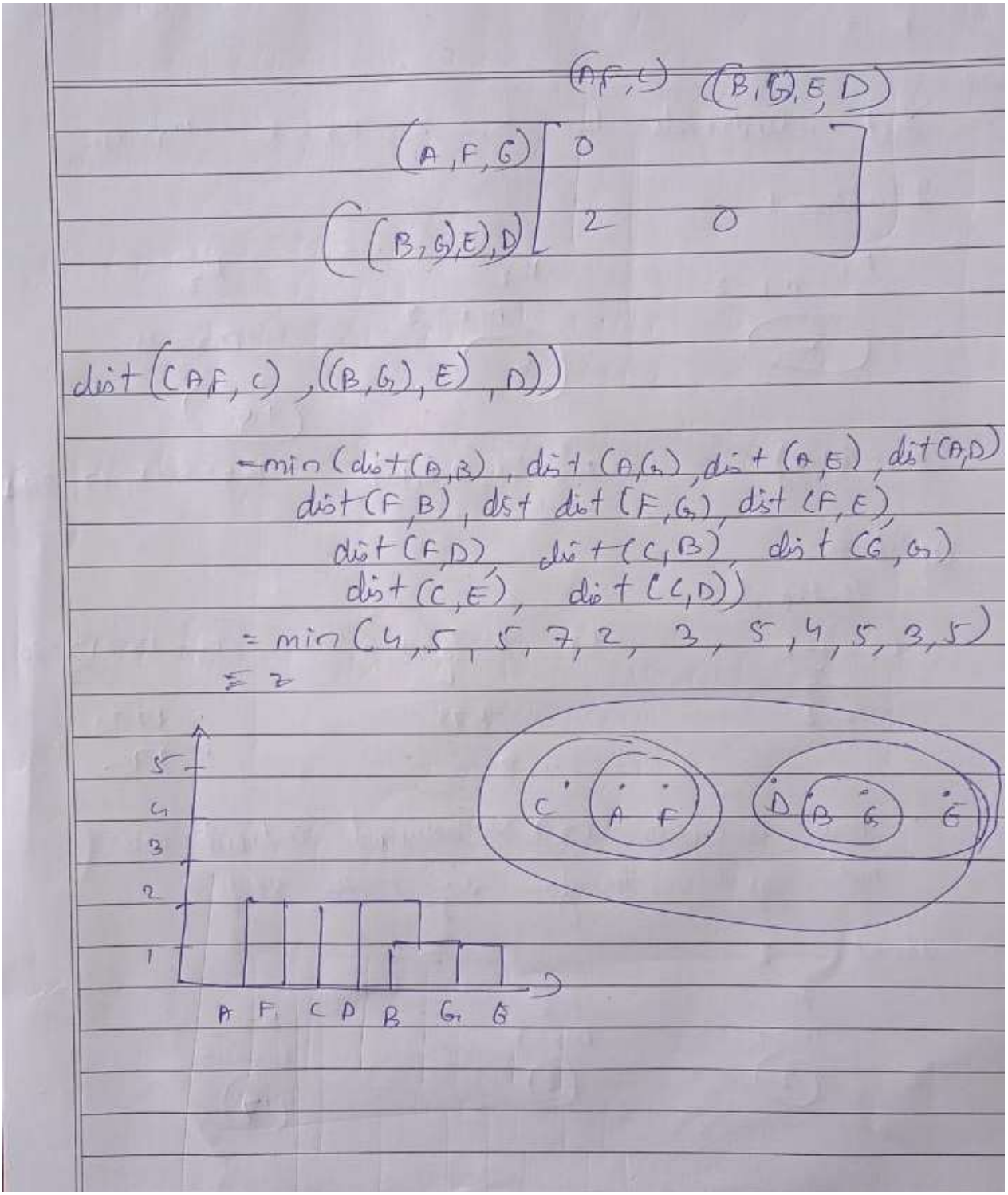
= 2.

dist (C, (((B,G), E), D)) = min (dist (C, D), dist (C B), dist (C, G)
dist (C, E)).

= min (5, 4, 5, 3)

= 3





2. Apply k-mean clustering technique on following dataset.
   1,2,6,7,8,10,15,17,20
   K=3

2) 1, 2, 6, 7, 8, 10, 15, 17, 20

K = 3

→

Let Centroids be $K_1 = 2$, $K_2 = 8$, $K_3 = 15$

Iteration 1

K_1 $m_1 = 2$ (2)

K_2 $m_2 = 8$ (8)

K_3 $m_3 = 15$ (15)

$K_1 = \{1, 2\}$ $K_2 = \{6, 7, 8, 10\}$, $K_3 = \{15, 17, 20\}$

Update the Centroid

$C_1 = \frac{1+2}{2}$

$= 2.$

$C_2 = \frac{6+7+8+10}{4}$

$= 7.75$

$= 8$

$C_3 = 15 + \frac{17}{3} + 20$

$= 17.33$

≈ 17

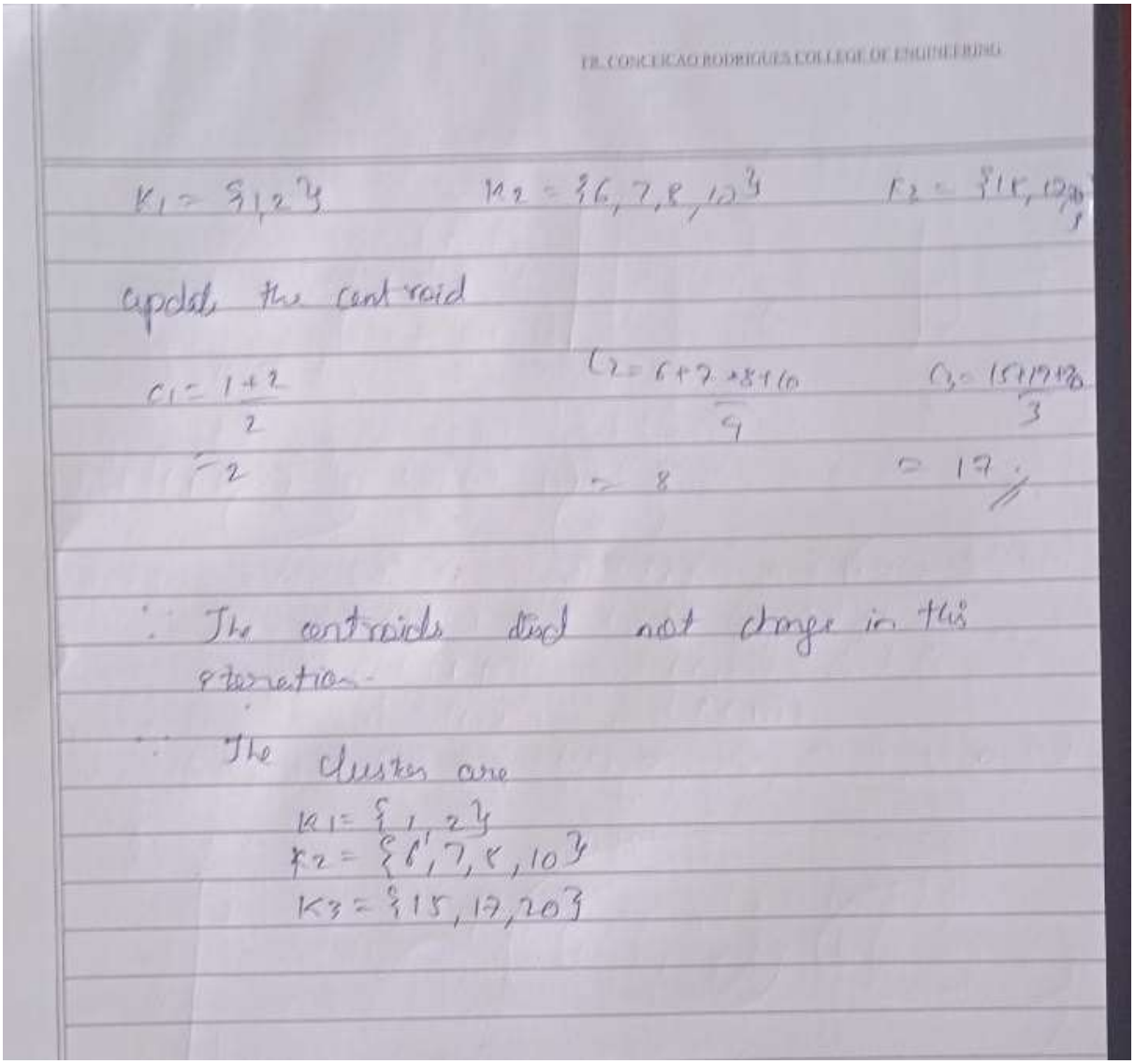
∵ new centroids (C_3) & assumed centroid are not equal, therefore repeat the step.

Iteration 2

K_1 $m_1 = 2$ (2)

K_2 $m_2 = 8$ (8)

K_3 $m_3 = 17$ (17)



Conclusion:

We are able to understand different clustering algorithm and am able to implement the algorithm using WEKA tool.

Experiment No: 09	TE AI&DS
Date of Performance: 17/10/20233	Roll No: 9696
Aim: To Implement Apriory algorithm to find frequent itemsets and strong association rules.	
Related CO4: Implement Classification, Clustering and Association mining techniques to extract knowledge	
Related LO 5. Perform exploratory analysis of the data to be used for mining. **LO6**: Implement the appropriate data mining methods like classification, clustering or Frequent Pattern mining on large data sets. **Objective:** To learn and implement Association Mining techniques.	

Rubrics for assessment of Experiment:

Sr. No	Parameters	Exceed Expectations(EE)	Meet Expectations (ME)	Below Expectations (BE)
1	Timeline (2)	Early or on time (2)	One session late (1)	More than one session late (0)
2	Preparedness (2)	Knows the basic theory related to the experiment very well. (2)	Managed to explain the theory related to the experiment. (1)	Not aware of the theory to the point. (1)
3	Effort (3)	Done expt on their own. (3)	Done expt with help from other. (2)	Just managed. (1)
4	Documentation(2)	Lab experiment is documented in proper format and maintained neatly. (2)	Documented in proper format but some formatting guidelines are missed. (1)	Experiments not written in proper format (0.5)
5	Result (1)	Specific conclusion.(1)	Partially specific conclusion. (0.5)	Not specific at all. (0)

Assessment Marks:

Timeline(2)	Preparedness(2)	Effort(3)	Documentation(2)	Result(1)	Total(10)

Theory:

Theory : Frequent patterns are patterns (such as itemsets, subsequences or substructures) that appear in a data set frequently. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a frequent itemset. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a frequent sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructures occurs frequently, it is called a frequent structured pattern. Finding such frequent patterns plays an essential role in mining associations, correlations and many other interesting relationships among data. A typical example of frequent itemset mining is **market basket analysis**. This process analyzes customer buying habits by finding association between different items that customers place in their shopping basket. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. In general, association rule mining can be viewed as a two-step process:

1. Find all frequent itemsets : By definition, each of these itemsets will occur atleast as frquently as a predetermined minimum support count.

2. Generate strong association rules fro the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

An itemset X is closed in a data set S if there exists no proper super-itemset Y such that Y has the same support count as X in S. An itemset X is a closed frequent itemset in set S if X is both closed and frequent in S. An itemset X is a maximal frequent itemset in set S if X is frequent, and there exists no super-itemset Y such that X is subset of Y and Y is frequent in S.

Apriori is a seminal algorithm proposed by R. Agrawal and R.Srikant in 1994 for mining frequent itemsets for boolean association rule. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties.

In data mining, **Apriori** is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).
Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing).
Overview
The whole point of the algorithm (and data mining, in general) is to extract useful information from large amounts of data. For example, the information that a customer who purchases a keyboard also tends to buy a mouse at the same time is acquired from the association rule below:
Support: The percentage of task-relevant data transactions for which the pattern is true.

Support (Keyboard -> Mouse)

$$= \frac{\text{No. of transacti ons containing both Keyboard and Mouse}}{\text{No. of total transacti ons}}$$

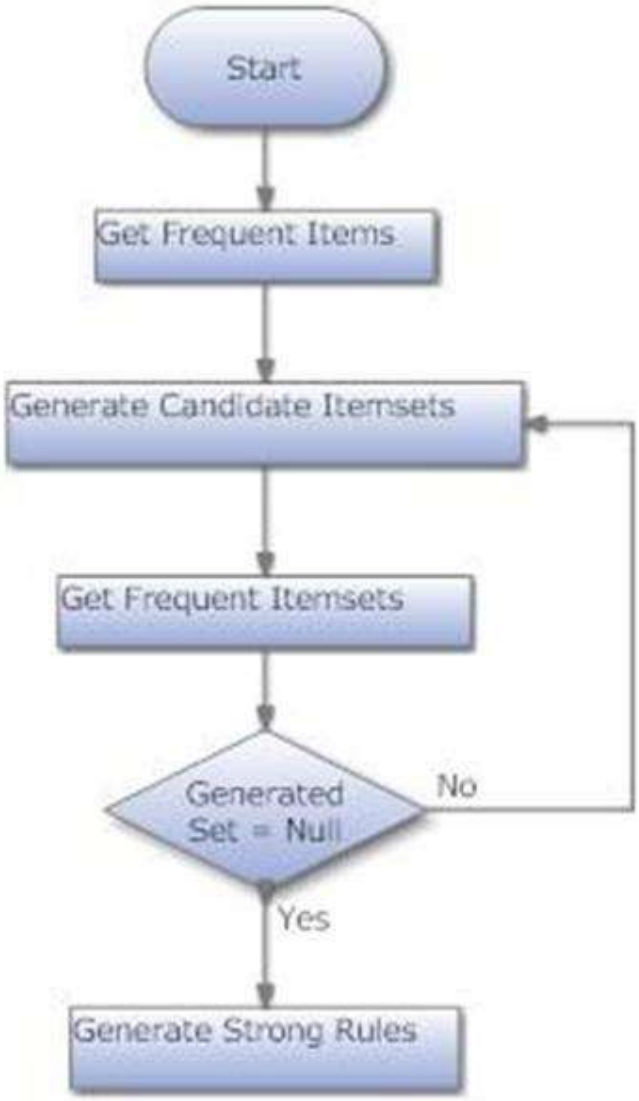
Confidence: The measure of certainty or trustworthiness associated with each discovered pattern.
Confidence (Keyboard -> Mouse) =
The algorithm aims to find the rules which satisfy both a minimum support threshold and a minimum confidence threshold (Strong Rules).

- Item: article in the basket.
    - Itemset: a group of items purchased together in a single transaction.

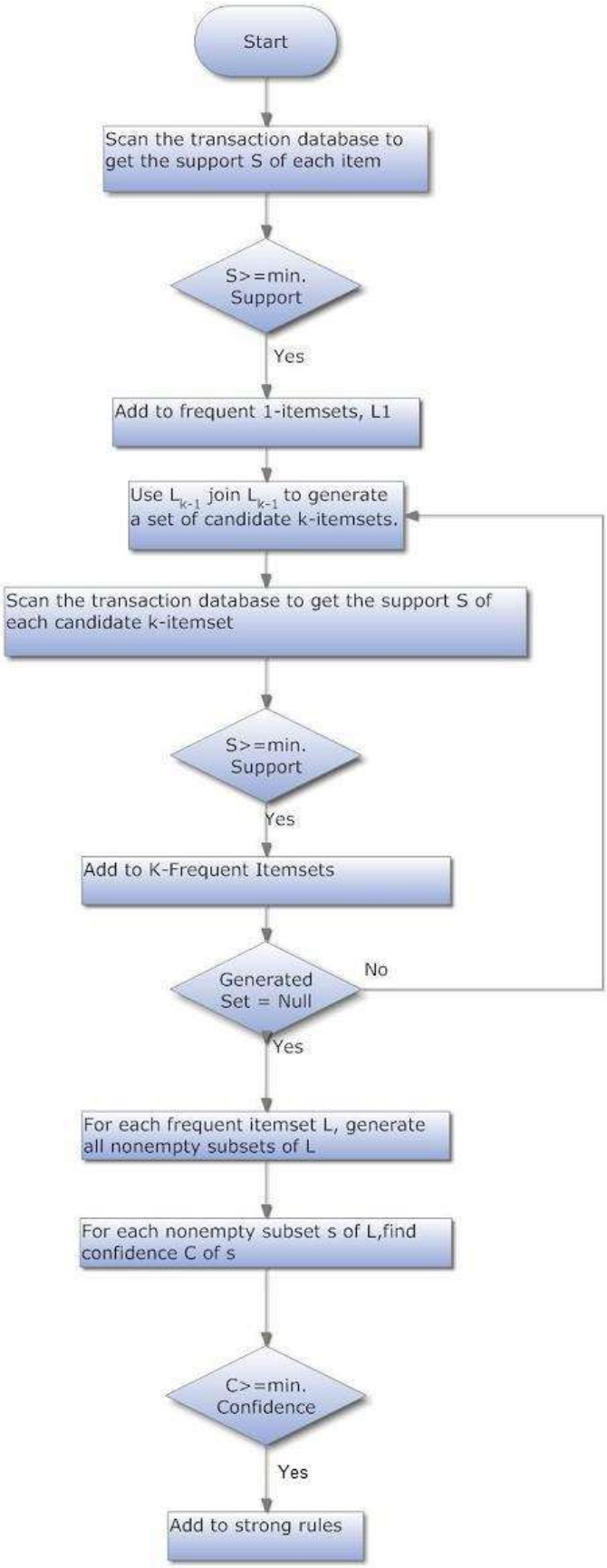
How Apriori Works

1. Find all frequent itemsets:
    - Get frequent items:
        - Items whose occurrence in database is greater than or equal to the min.support threshold.
    - Get frequent itemsets:
        - Generate candidates from frequent items.
        - Prune the results to find the frequent itemsets.
2. Generate strong association rules from frequent itemsets
    - Rules which satisfy the min.support and min.confidence threshold.

High Level Design



Low Level Design

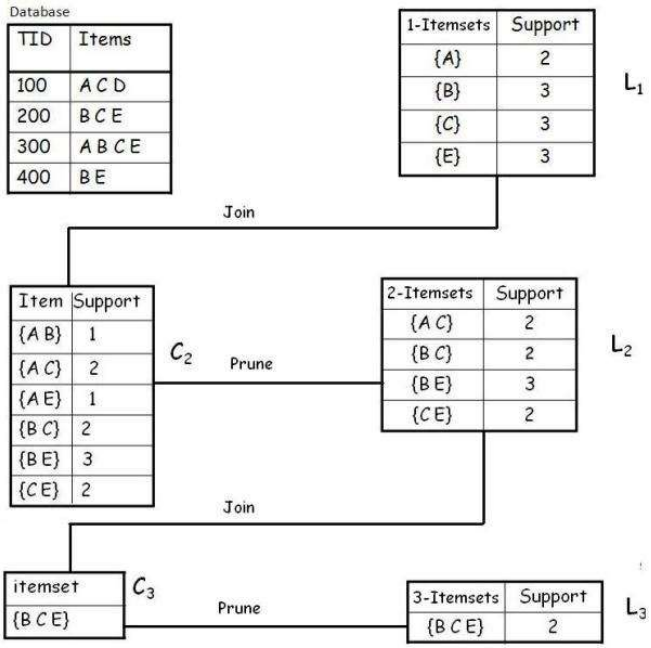

Start
Scan the transaction database to get the support S of each item
S>=min. Support
Yes
Add to frequent 1-itemsets, L1
Use L_{k-1} join L_{k-1} to generate a set of candidate k-itemsets.
Scan the transaction database to get the support S of each candidate k-itemset
S>=min. Support
Yes
Add to K-Frequent Itemsets
Generated Set = Null
No
Yes
For each frequent itemset L, generate all nonempty subsets of L
For each nonempty subset s of L,find confidence C of s
C>=min. Confidence
Yes
Add to strong rules

Example
A database has five transactions. Let the min sup = 50% and min con f = 80%.

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Solution
Step 1: Find all Frequent Itemsets



Frequent Itemsets
Hide Copy Code
{A} {B} {C} {E} {A C} {B C} {B E} {C E} {B C E}
Step 2: Generate strong association rules from the frequent itemsets

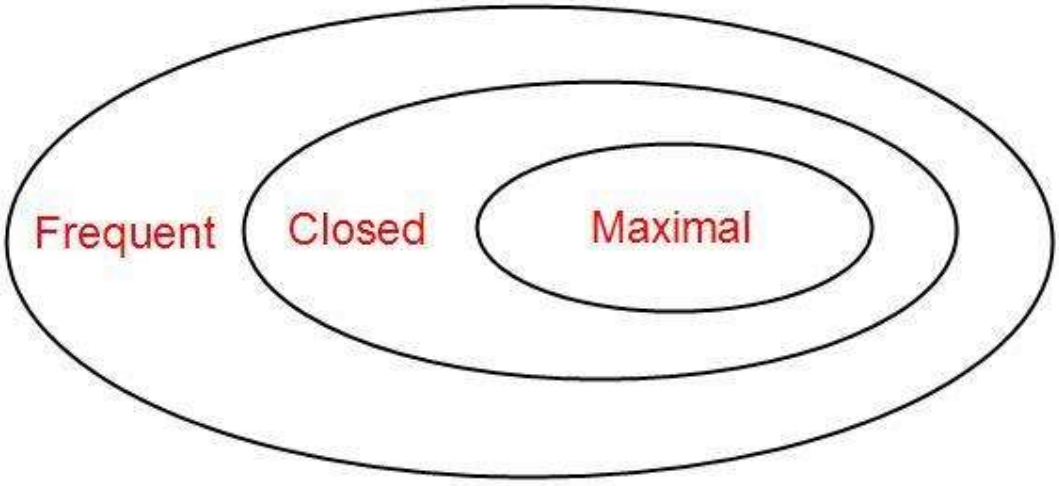
Rules	Support (X Y)	Support(X)	Confidence
{A} -> {C}	2	2	100
{B} -> {C}	2	3	66.66666667
{B} -> {E}	3	3	100
{C} -> {E}	2	3	66.66666667
{B} -> {C E}	2	3	66.66666667
{C} -> {B E}	2	3	66.66666667
{E} -> {B C}	2	3	66.66666667
{C} -> {A}	2	3	66.66666667
{C} -> {B}	2	3	66.66666667
{E} -> {B}	3	3	100
{E} -> {C}	2	3	66.66666667
{C E} -> {B}	2	2	100
{B E} -> {C}	2	3	66.66666667
{B C} -> {E}	2	2	100

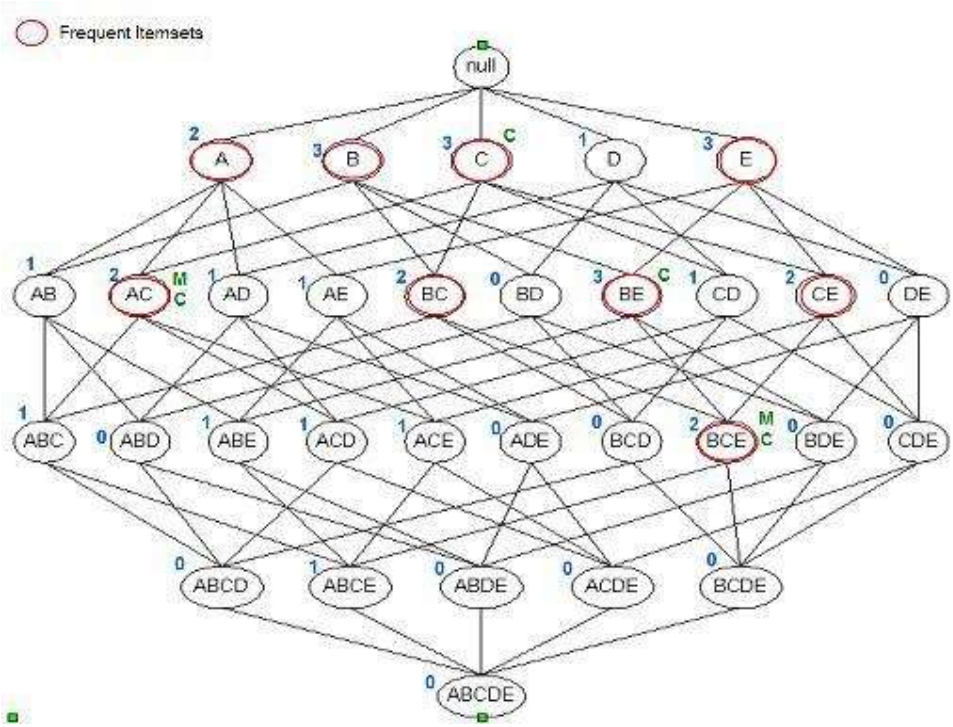
Lattice

Closed Itemset: support of all parents are not equal to the support of the itemset.

Maximal Itemset: all parents of that itemset must be infrequent.

Keep in mind:





Itemset {c} is closed as support of parents (supersets) {A C}:2, {B C}:2, {C D}:1, {C E}:2 not equal support of {c}:3.
And the same for {A C}, {B E} & {B C E}.
Itemset {A C} is maximal as all parents (supersets) {A B C}, {A C D}, {A C E} are infrequent.
And the same for {B C E

Implementation:

Part1 : WAP in Python/java to implement Apriori –Attach code and output
Apriori Algorithm (Python 3.0) - A Data Analyst
Apriori Algorithm from Scratch - Python (vucreations.com)
Apriori: Association Rule Mining In-depth Explanation and Python Implementation | by Chonyy | Towards Data Science)

Part2: using Std libraries- Attach code and output
Link to download dataset(Grocery Store Data Set | Kaggle)
Implementing Apriori algorithm in Python - GeeksforGeeks
Give summary table Comparison of Pattern Evaluation Measures Using Contingency Tables for a Variety of Data Sets(refer text book table 6.9)(CHI-SQAURE, *lift, all conf, max con,. Kulc, cosine)*

Other resources:
Beginner's Guide To Apriori Algorithm With Implementation In Python (analyticsindiamag.com)
Introduction to Market Basket Analysis in Python - Practical Business Python (pbpython.com)
How to Create Data Visualization for Association Rules in Data Mining - Machine Learning Applications (intelligentonlinetools.com)
Apriori - mlxtend (rasbt.github.io)

Code with output:

Part 1:

Code:

```
from itertools import combinations
from collections import Counter

data = [
        ['T100',['A', 'C', 'D']],
        ['T200',['B', 'C', 'E']],
        ['T300',['A', 'B', 'C', 'E']],
        ['T400',['B', 'E']],
        ]

# printing dataset
init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)

# min support
sp = 0.4
s = int(sp*len(init))
s

c = Counter()
for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
print("C1:")
for i in c:
    print(str([i])+": "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+=c[i]
print("L1:")
for i in l:
    print(str(list(i))+": "+str(l[i]))
print()
pl = l
pos = 1
for count in range (2,1000):
    nc = set()
```

```
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    c = Counter()
    for i in nc:
        c[i] = 0
        for q in data:
            temp = set(q[1])
            if(i.issubset(temp)):
                c[i]+=1
    print("C"+str(count)+":")
    for i in c:
        print(str(list(i))+": "+str(c[i]))
    print()
    l = Counter()
    for i in c:
        if(c[i] >= s):
            l[i]+=c[i]
    print("L"+str(count)+":")
    for i in l:
        print(str(list(i))+": "+str(l[i]))
    print()
    if(len(l) == 0):
        break
    pl = l
    pos = count
print("Result: ")
print("L"+str(pos)+":")
for i in pl:
    print(str(list(i))+": "+str(pl[i]))
print()

# generate association rules
for l in pl:
    c = [frozenset(q) for q in combinations(l,len(l)-1)]
    mmax = 0

    # calculate confidence and print association rules
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
```

```
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp > mmax):
            mmax = temp
        temp = sab/sb*100
        if(temp > mmax):
            mmax = temp
        print(str(list(a))+" -> "+str(list(b))+" = "+str(sab/sa*100)+"%")
        print(str(list(b))+" -> "+str(list(a))+" = "+str(sab/sb*100)+"%")
    curr = 1
    print("choosing:", end=' ')
    for a in c:
        b = l-a
        ab = l
        sab = 0
        sa = 0
        sb = 0
        for q in data:
            temp = set(q[1])
            if(a.issubset(temp)):
                sa+=1
            if(b.issubset(temp)):
                sb+=1
            if(ab.issubset(temp)):
                sab+=1
        temp = sab/sa*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
        temp = sab/sb*100
        if(temp == mmax):
            print(curr, end = ' ')
        curr += 1
    print()
    print()
```

Output:

```
['A', 'B', 'C', 'D', 'E']
C1:
['A']: 2
['B']: 3
['C']: 3
['D']: 1
['E']: 3
```

L1:
['A']: 2
['B']: 3
['C']: 3
['E']: 3

C2:
['E', 'C']: 2
['E', 'A']: 1
['A', 'C']: 2
['E', 'B']: 3
['B', 'A']: 1
['B', 'C']: 2

L2:
['E', 'C']: 2
['A', 'C']: 2
['E', 'B']: 3
['B', 'C']: 2

C3:
['E', 'B', 'C']: 2
['E', 'A', 'C']: 1
['B', 'A', 'C']: 1

L3:
['E', 'B', 'C']: 2

C4:

L4:

Result:
L3:
['E', 'B', 'C']: 2

['E', 'B'] -> ['C'] = 66.66666666666666%
['C'] -> ['E', 'B'] = 66.66666666666666%
['E', 'C'] -> ['B'] = 100.0%
['B'] -> ['E', 'C'] = 66.66666666666666%
['B', 'C'] -> ['E'] = 100.0%
['E'] -> ['B', 'C'] = 66.66666666666666%
choosing: 3 5

Part 2:

Code:

```
from apyori import apriori

# Define a list of transactions
transactions = [
    ['MILK', 'BREAD', 'BISCUIT'],
    ['MILK', 'BISCUIT'],
    ['MILK', 'CEREAL'],
    ['BREAD', 'BISCUIT'],
    ['BREAD', 'CEREAL'],
    ['BISCUIT', 'CEREAL'],
]

# Adjusted parameters
min_support = 0.1  # Adjusted for demonstration
min_confidence = 0.1  # Adjusted for demonstration
min_lift = 1
min_length = 2

# Apply Apriori algorithm
rules = apriori(transactions, min_support=min_support, min_confidence=min_confidence)

# Print the rules and details
for rule in rules:
    print("Rule:", rule.items)
    print("Support:", rule.support)
    print("Confidence:", rule.ordered_statistics[0].confidence)
    print("Lift:", rule.ordered_statistics[0].lift)
    print()
```

Output:

```
Rule: frozenset({'BISCUIT'})
Support: 0.6666666666666666
Confidence: 0.6666666666666666
Lift: 1.0

Rule: frozenset({'BREAD'})
Support: 0.5
Confidence: 0.5
Lift: 1.0

Rule: frozenset({'CEREAL'})
Support: 0.5
Confidence: 0.5
Lift: 1.0
```

Rule: frozenset({'MILK'})
Support: 0.5
Confidence: 0.5
Lift: 1.0

Rule: frozenset({'BISCUIT', 'BREAD'})
Support: 0.3333333333333333
Confidence: 0.3333333333333333
Lift: 1.0

Rule: frozenset({'CEREAL', 'BISCUIT'})
Support: 0.16666666666666666
Confidence: 0.16666666666666666
Lift: 1.0

Rule: frozenset({'BISCUIT', 'MILK'})
Support: 0.3333333333333333
Confidence: 0.3333333333333333
Lift: 1.0

Rule: frozenset({'CEREAL', 'BREAD'})
Support: 0.16666666666666666
Confidence: 0.16666666666666666
Lift: 1.0

Rule: frozenset({'BREAD', 'MILK'})
Support: 0.16666666666666666
Confidence: 0.16666666666666666
Lift: 1.0

Rule: frozenset({'CEREAL', 'MILK'})
Support: 0.16666666666666666
Confidence: 0.16666666666666666
Lift: 1.0

Rule: frozenset({'BISCUIT', 'BREAD', 'MILK'})
Support: 0.16666666666666666
Confidence: 0.16666666666666666
Lift: 1.0

Conclusion: I understand apriori algorithm and able to implement the code