II - ASSIGNMENT

(Start Writing From Here)

① Discuss K-medoids clustering with an example.

Ⓐ K-medoids is an unsupervised method with unlabelled data to be clustered. It is an improvised version of the K-Means algorithm mainly designed to deal with other outlier data sensitivity. Compared to other partioning algorithms, the algorithm is simple, fast and easy to implement. A medoid can be defined as a point in the cluster whose dissimilarities with all the other points in the cluster are minimum. The dissimilarity of the medoid(c_i) and object(p_i) is calculated by using $E = |p_i - c_i|$.

The cost in K-Medoids algorithm is given as

$$C = \sum_{c_i} \sum_{p_i \in c_i} |p_i - c_i|$$

Ex:-

	x	y
0	8	7
1	3	7
2	4	9
3	9	6
4	8	5
5	5	8
6	7	3
7	8	4
8	7	5
9	4	5

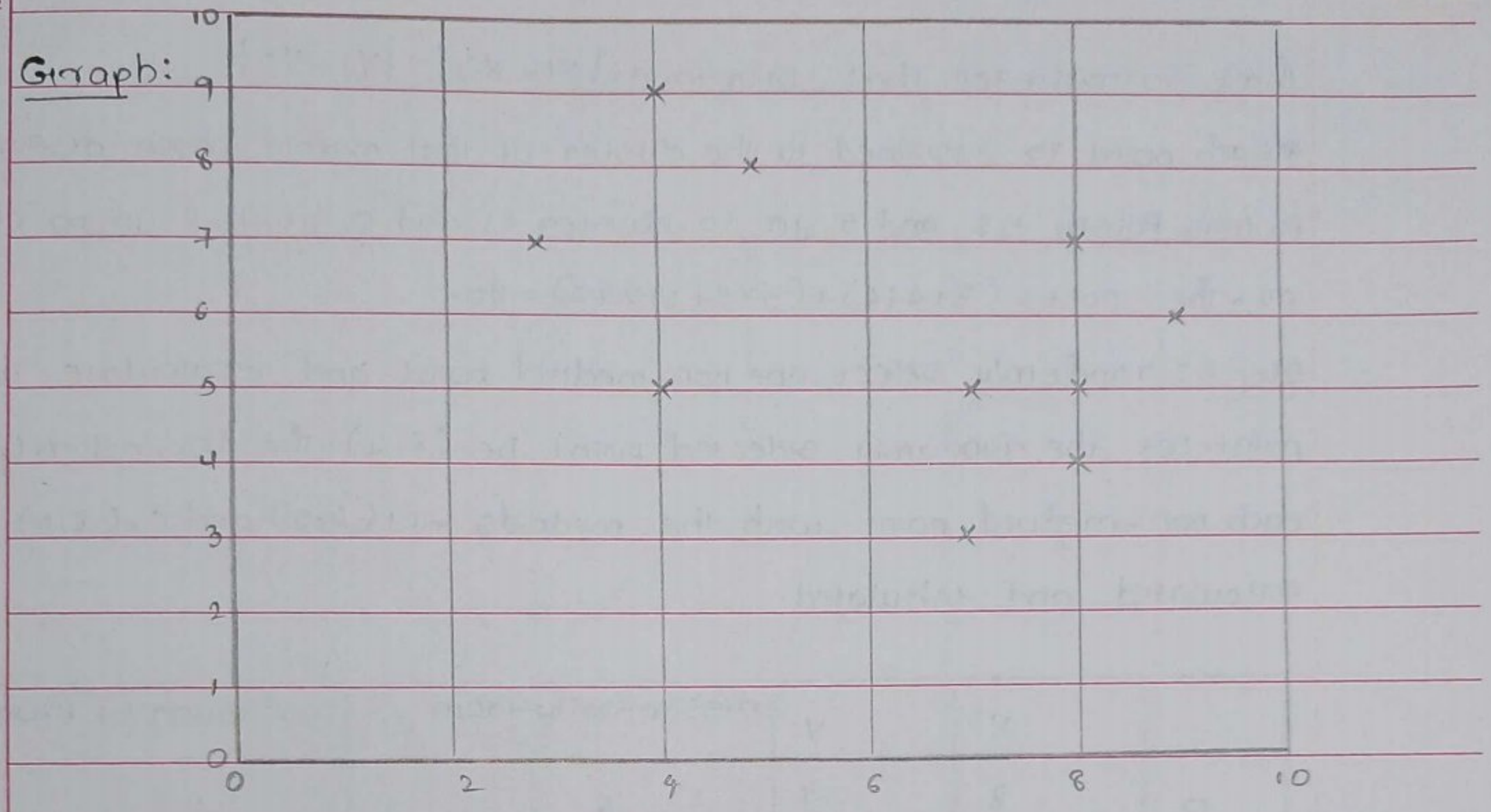
Let's consider the following example:

If a graph is drawn using the above data points, we obtain the following:

<u>Step 1</u>:- Let the randomly selected 2 medoids, so select k=2, and let $c_1 = (4,5)$ and $c_2 = (8,5)$ are two medoids.

<u>Step 2</u>:- Calculating cost. The dissimilarity of each non-medoid point with the medoids is calculated and tabulated.

Graph:



	x	y	Dissimilarity from C1	Dissimilarity from C2	
0	8	7	6	2	
1	3	7	3	7	
2	4	9	4	8	
3	9	6	6	2	
4	8	5	–	–	
5	5	8	4	6	
6	7	3	5	3	
7	8	4	5	1	
8	7	5	3	1	
9	4	5	–	–	

Here we used Manhattan distance formula to calculate the distance matrices between medoid and non-medoid points.

That formula tell that, Distance = $|x_1 - x_2| + |y_1 - y_2|$.

* Each point is assigned to the cluster of that medoid whose dissimilarity is less. Points 1, 2 and 5 go to cluster c1 and 0, 3, 6, 7, 8 go to cluster c2. The cost = (3+4+4) + (3+1+1+2+2) = 20.

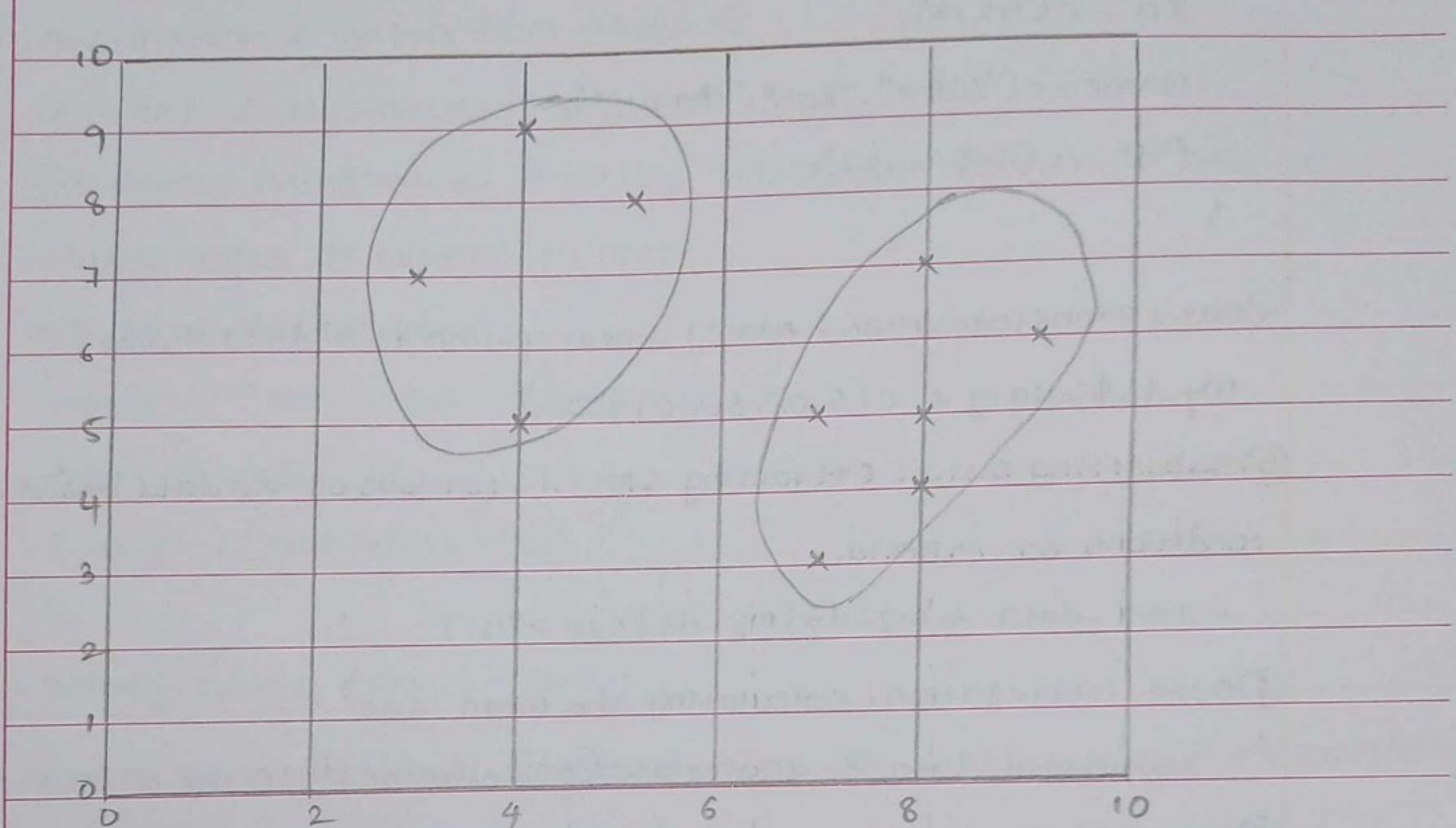
Step 3: randomly select one non-medoid point and recalculate the point. Let the randomly selected point be (8,4). The dissimilarity of each non-medoid point with the medoids - c1(4,5) and c2(8,4) is calculated and tabulated.

	X	Y	Dissimilarity from c_1	Dissimilarity from c_2
0	8	7	6	3
1	3	7	3	8
2	4	9	4	9
3	9	6	6	3
4	8	5	4	1
5	5	8	4	7
6	7	3	5	2
7	8	4	—	—
8	7	5	3	2
9	4	5	—	—

Each point is assigned to that cluster whose dissimilarity is less. So, points (1,2) and 5 go to cluster c1 and 0, 3, 6, 7, 8 go to cluster c2.

The new cost = (3+4+4) + (2+2+1+3+3) = 22, swap cost = New cost - previous cost = 22 - 20 and 2 > 0. As the swap cost is not less than zero, we undo the swap. Hence (4,5) and (8,5) are final medoids.

The clustering would be in the following way. The time complexity is $O(K * (D-K)^2)$.



(3) Explain about data management and indexing in R with suitable examples.

(A) Data management in R involves the organization, storage, manipulation and analysis of data using the R programming language. Here are some aspects of data management in R with examples:

(1) Data Import/Export: 'read.table' or 'read.csv' functions to import data from a csv file into a data frame.

```
my-data ← read.csv("path/to/your/file.csv")
```

(2) Data cleaning: Removing missing values from a data frame.

```
clean-data ← na.omit(my-df)
```

③ Data structures: creating a data frame

```
my_df <- data.frame(
  ID = c(1,2,3),
  Name = c("Alice","Bob","charlie"),
  Age = c(25,30,22)
)
```

④ Data Transformation: Adding a new column to a data frame.

```
my_df$salary <- c(5000,6000,4500)
```

⑤ Subsetting Data: Extracting specific portions of the data based on conditions or criteria.

```
subset_data <- my_df[my_df$Age > 25,]
```

⑥ Data Aggregation: calculating the mean age

```
aggregate_data <- aggregate(Age ~ Name, data = my_df, FUN = mean)
```

⑦ Data Merging: Merging two data frames

```
merged_data <- merge(df1, df2, by = "ID")
```

⑧ Data Visualisation: Visualising the data using plots and charts

```
plot(my_df$Age, my_df$salary, main = "Age vs salary", xlab = "Age",
```

⑨ Data storage: Saving the data for future use.

```
write.csv(my_df, file = "path/to/save/data.csv", row.name = FALSE)
```

Indexing :- Indexing is the process of accessing or extracting specific elements, rows or columns from data structures like vectors, matrices and data frames.

① Indexing vectors: Vectors in R can be indexed using square brackets '[]'.

```
my_vector <- c(10,20,30,40,50)
second_element <- my_vector[2]
```

② Indexing Matrices: Matrices in R are two-dimensional and you can use row and column indices to access elements.

```
my-matrix ← matrix (1:9, nrow=3)
element ← my-matrix [2,3]
```

③ Indexing DataFrames: these are similar to matrices. You can use column names or numeric indices.

```
my-df ← data.frame (
Name = c("Alice", "Bob", "charlie"),
Age = c(25,30,22),
Salary = c(5000,6000,4500)
)
salary-column ← my-df$salary.
```

③ ~~Diss~~ a Discuss about different types of operators in R programming with example.

Ⓐ In R programming there are different types of operators and each operator performs a different task.

Arithmetic Operators: These are the symbols which are used to represent arithmetic math (operators) operations.

+ → This operator is used to add two vectors in R. a ← c(2,3.3,4)

~~b ← c(2,3.3,4)~~ b ← c (11,5,3)

print (a+b)

- → used to subtract two vectors. a ← c(2,3.3,4)

b ← c(11,5,3)

print (a-b)

* → used to multiply two vectors

print (a * b)

/ → used to divides the vector from another one.

print (a/b)

%% %% → used to find remainder of the first vector with second vector.

print(a %% b)

%/% %/% → used to find the division of first vector with second.

print(a %/% b)

Relational operators: It is a symbol which defines some kind of relation between two entities.

> → will return TRUE when every element in the first vector is greater than the corresponding element of second vector.

< → will return TRUE when every element in the first vector is less than the corresponding element of second vector.

<= → will return TRUE when first vector element is less than or equal to second element.

>= → will return TRUE when first vector element is greater than or equal to the second element.

== → return TRUE when two elements are equal.

!= → return TRUE when two elements are not equal.

Logical operators: & → takes the first element of both the vector and return TRUE if both the elements are TRUE.

| → takes the first element of both the vector and returns TRUE if one of them is TRUE.

Assignment operators:

<- or = or <<- ⇒ these operators are known as left assignment operators.

-> or ->> ⇒ known as right assignment operators.

Miscellaneous operators:

: → used to create the series of numbers in sequence for a vector.

```
v <- 1:8
print(v)        o/p: [1] 1 2 3 4 5 6 7 8
```

%in% → used when we want to identify if an element belongs to a vector.

%*% → used to multiply a matrix with its transpose.

(4) How to estimate the parameters of a model using maximum likelihood method?

(A) Maximum likelihood estimation (MLE) is a statistical method used to estimate the parameters of a probability distribution that best describe a given dataset. The fundamental idea behind MLE is to find the values of the parameters that maximize the likelihood of the observed data, assuming that the data are generated by the specified distribution. A parameter is a numerical characteristic of a distribution. Mean(μ), variance (σ^2) as parameters, number of trials (n) & probability of success (p) as parameters. Gamma distributions have shape (k) and scale (θ) as parameters. Exponential distributions have the inverse mean (λ) as the parameter.

*things aren't always that simple. Sometimes, you may encounter problems involving estimating parameters that do not have a simple one-to-one correspondance with common numerical characteristics. For instance,

If I give you the following distribution:

$$f_\theta(x) = \theta x^{-\theta-1}$$

the above equation shows the probability density function of a Pareto distribution with scale=1. It's not easy to estimate parameter θ of the distribution using simple estimators based because the numerical characteristics of the distribution vary as a function of the range of the parameter. For instance, the mean of the above distribution is expressed as follows:

$$\text{Mean} = \begin{cases} \infty & \text{if } \theta \leq 1 \\ \frac{\theta}{1-\theta} & \text{if } \theta > 1 \end{cases}$$

(5) @ Explain how the F-value can be estimated in statistical learning.

(A) F-Test is any test that utilises the F-Distribution table to fulfill its purpose. It compares the ratio of the variances of two populations and determines if they are statistically similar or not. We can use this test when:

* The population is normally distributed.
* The samples are taken at random and are independent samples.

Formula:

$$F_{calc} = \frac{\sigma_1^2}{\sigma_2^2}$$

Steps involved:

① Use standard deviation (σ) and find variance(σ^2) of the data.

② Determine the null and alternative hypothesis.

Ho → no difference in variances, Ha → difference in variancies.

(3) Find Fcalc using eq-1.

(4) Find degrees of freedom of the two samples.

(5) Find Ftable value using d1 and d2 obtained in step-4 from the F-distribution table. Take $\alpha = 0.05$

(6) Interpret the results using Fcalc and Ftable.

Fcalc < Ftable → can't reject null hypothesis.

Fcalc > Ftable → reject null hypothesis.

(b) Illustrate the levels of measurement of data.

there are 4 levels of measurement.

(1) Nominal level: You can categorize your data by labelling them in mutually exclusive groups, but there is no order blw the categories.

(2) Ordinal level: You can categorize and rank your data in an order, but you cannot say anything about the intervals between the rankings.

(3) Interval level: You can categorize, rank and infer equal intervals between neighbouring data points, but there is no true zero point.

(4) Ratio level: You can categorize, rank, and infer equal intervals between neighbouring data points, and there is a true zero point.