

# IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: Adikari Adikari

Student Reference Number: 10749174

Module Code: PUSL3123	Module Name: AI and Machine Learning								
Coursework Title: Coursework 02									
Deadline Date: 23 <sup>rd</sup> January 2023	Member of staff responsible for coursework:								
Programme: BSc (Hons) Software Engineering									
Please note that University Academic Regulations are available under Rules and Regulations on the University website <a href="http://www.plymouth.ac.uk/studenthandbook">www.plymouth.ac.uk/studenthandbook</a> .									
Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.									
<p><i>We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.</i></p> <p>Signed on behalf of the group:</p>									
<p>Individual assignment: <i>I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.</i></p> <p>Signed : </p>									
<p>Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.</p> <p>I *have used/not used translation software.</p> <p>If used, please state name of software.....</p>									
<table border="1"> <thead> <tr> <th>Overall mark</th> <th>%</th> <th>Assessors Initials</th> <th>Date</th> </tr> </thead> <tbody> <tr> <td colspan="4"> </td> </tr> </tbody> </table>		Overall mark	%	Assessors Initials	Date				
Overall mark	%	Assessors Initials	Date						

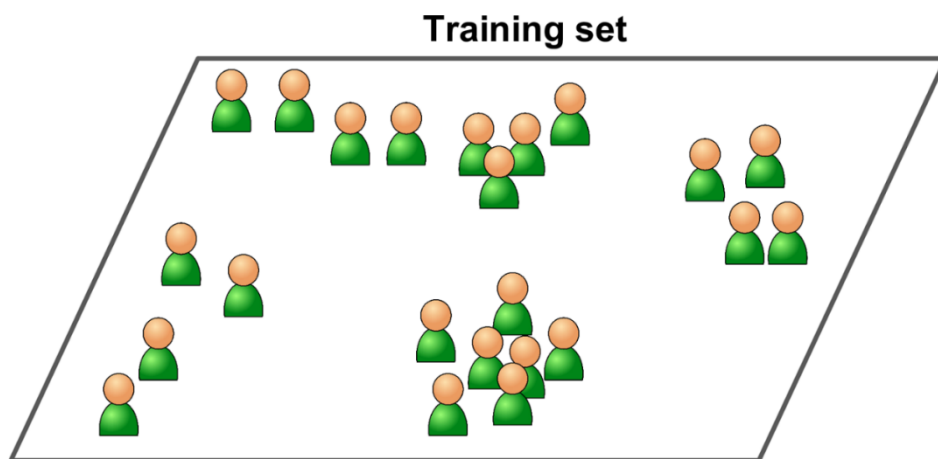
\*Please delete as appropriate(s) [d:\students\cw\frontcover\2013\14

# INTRODUCTION

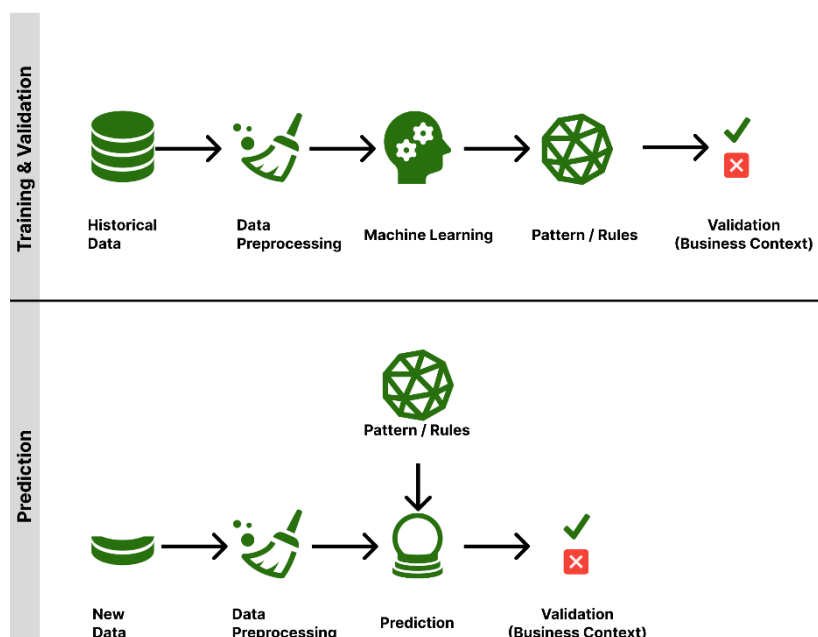
The focal point of this report is **unsupervised machine learning** and its application in the context of **K-means clustering**. Unsupervised machine learning is a sort of machine learning that involves **training a model** to identify patterns and relationships in **data without being supplied with labeled samples** or explicit instruction. K-means clustering is a specific type of unsupervised machine learning algorithm that is utilised to group data into **clusters** based on similarities between the data points. And the initial section of the report furnishes the utmost precise delineation of unsupervised machine learning (UML). The section then delves into the fundamentals of K-means clustering, encompassing the algorithm's operation and the **indispensable parameters that can be modified to maximize its performance**. In the report's latter section, MATLAB code is exerted to implement the k-means clustering technique. The report concludes with a discussion of **the key takeaways and drawbacks** of k-means clustering technique. By examining this report, the examiner will be able to determine the level of comprehension and expertise on the aforementioned spheres.

## Overview of Unsupervised Machine Learning and K-Means Clustering.

One of the most prevalent techniques for **imbuing a machine** with **artificial intelligence** is through the utilisation of **machine learning algorithms**. And **Unsupervised machine learning** is one of the most omnipresent subfields of ML. In the broadest sense, unsupervised learning is the process of studying data **without labels** by drawing assumptions about the data's structure (e.g., algebraic, combinatorial, or probabilistic) (**Jordan & Mitchell, 2015**). In other words, there is a lack of precision or clarity regarding the meaning or significance of the data, as well as the expected or desired results. Instead, it is anticipated that the model will **identify patterns or correlations** in the data that can facilitate a more comprehensive comprehension of it or enable predictions about additional data. Unsupervised learning is often known as "learning what often occurs" (**Barlow, 1989**).

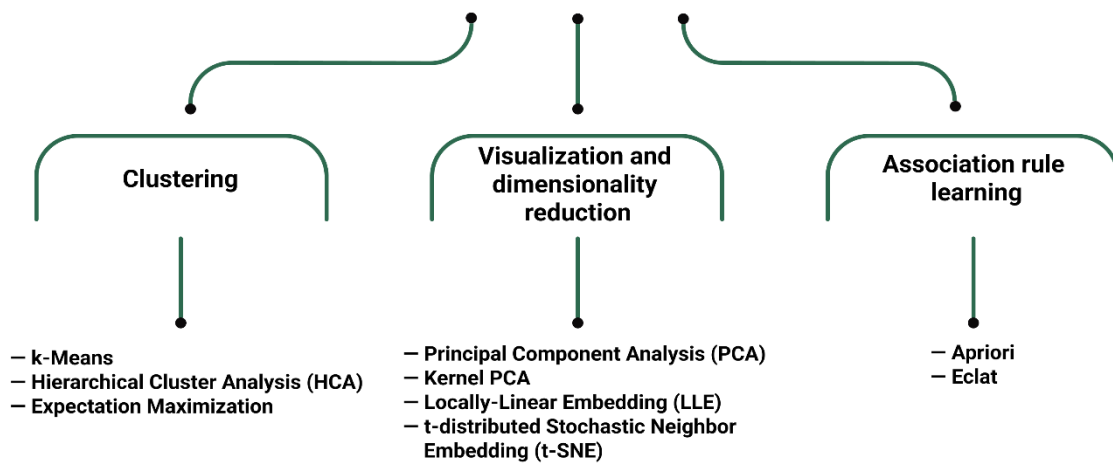


As of late, there has been a proliferation of utilizing unsupervised machine learning on unstructured raw network data to enhance network performance and furnish services such as **traffic engineering**, **anomaly detection** and **QOS optimization**. There has been a surge in the utilization of unsupervised learning methodologies in networking due to their demonstrated efficacy in various other domains, including **computer vision**, **natural language processing**, **speech recognition**, and **autonomous vehicle control** (Usama et al., 2019).

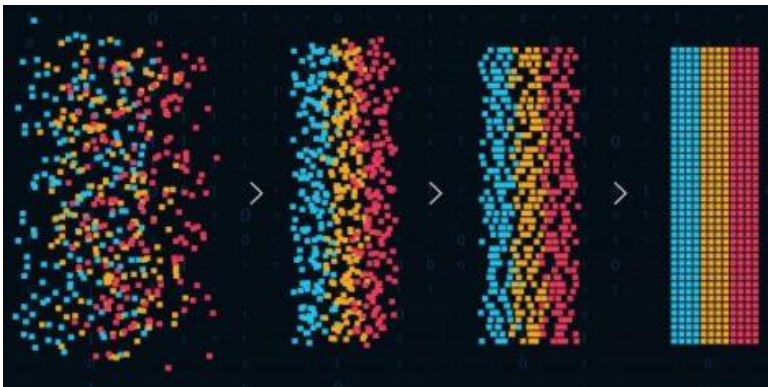


Listed below are some of the most widely-used algorithms in the field of unsupervised learning.

## Unsupervised Machine Learning

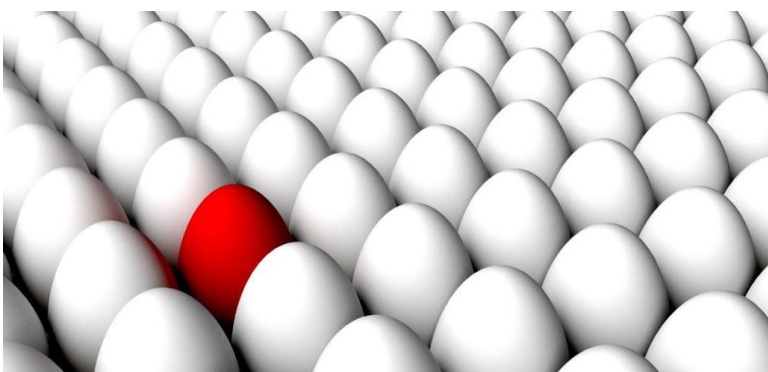


In **Supervised ML**, training data can be utilised to categorize customer comments according to sentiment (favorable, adverse, impartial), as well as according to context. For instance, In the airline industry, the context may include factors such as timeliness, cuisine, amenities, and leisure options. By means of this study, a business proprietor can identify the areas upon which his enterprise should concentrate its efforts. For example, if he notices that the majority of negative feedback pertains to the culinary offerings, he will prioritize improving the culinary experience for patrons.



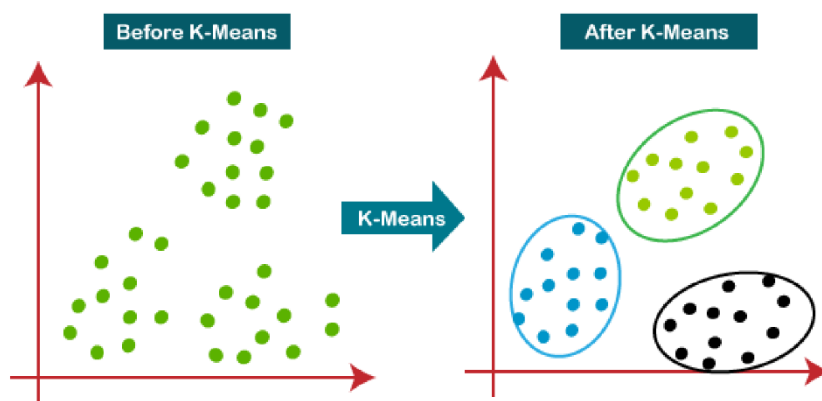
In some circumstances, though, business proprietors may be **uncertain about the context**. Moreover, there may be instances **where training data is inaccessible**. Additionally, **the frame of reference may evolve** over time. It is not possible to utilize supervised machine learning algorithms (Classification algorithms) in situations where the **target categories are unknown**. **In these cases, a clustering technique is utilized.**

A **common application of clustering** is in the **winemaking industry**, where **each cluster denotes a wine brand** and the wines are sorted according to the **proportions of their constituents**. Additionally, in the realm of Supervised Machine Learning, **classification algorithms** can be utilized to identify and **distinguish various types of images**. **However, there may be instances where a single image contains multiple, distinct shapes, requiring the implementation of an algorithm capable of segregating the various figures within the image.** In this instance, **clustering algorithms** are utilized to facilitate the desired outcome. Clustering categorizes objects into groups according to their resemblance or distance measure (Mukhopadhyay, 2018).



As illustrated in figure 02, clustering is a common type of unsupervised learning.





### One common technique for Clustering is K-means clustering.

K-means clustering is a technique for dividing a dataset into a **predetermined number of clusters** or groups based on data patterns. The objective of K-means clustering is to **split the data** into **distinct groups** such that the points within each cluster are as homogenous as possible, while being as dissimilar as possible from those in other clusters.

In order to implement K-means clustering, it is **necessary to specify the number of**

**clusters** that you wish to discern within the dataset. Typically, the letter "**K**" is utilized to **denote this number**. Once the value of K has been ascertained, the algorithm commences by **randomly selecting K points** from the dataset and utilizing them as the **initial centroids**, or centers, of the clusters. The algorithm subsequently assigns each data point to the cluster whose centroid is most proximate to it, based on some metric of similarity or distance.

Upon completing the process of assigning all data points to their respective clusters, the centroids are recalculated based on the **mean position of all points within the cluster**. The algorithm subsequently **repeats the steps** of assigning points to clusters and recalculating centroids **until convergence is achieved**, at which point the **clusters are considered to be in a stable state**.

An essential factor to consider when employing K-means clustering is **determining the most suitable value of K**. There exist various approaches that can be employed to ascertain the most suitable quantity of clusters.

E.g.: Elbow method, the silhouette method, and the gap statistic (Matt.0, 2019).

One of the primary benefits of utilizing K-means clustering is **its computational efficiency and simplicity of implementation**, particularly for datasets with a high number of features or dimensions. Additionally, the results of K-means clustering are relatively straightforward to comprehend, as each cluster can be interpreted as a collection of data points that are similar to one another.

There are several parameters that can be modified to potentially improve the performance of the K-means algorithm. Some of the most important ones are:

1. **The quantity of clusters:** This is one of the most essential K-means parameters, since it **defines the number of clusters** the algorithm will attempt to locate in the data. Selecting the optimal quantity of clusters is **frequently a trade-off** between **simplicity and complexity**, and can be **influenced by the attributes of the data** and the objectives of the analysis.
2. **The initialization method:** The K-means algorithm initiates by randomly selecting an initial set of cluster centers. The selection of these initial centers can significantly affect the resultant clusters that are generated. Various techniques can be utilized to initialize the centers of clusters, including randomly selecting the initial centers from the data points or employing a method like **k-means++** to select the initial centers in a more strategic manner.
3. **The distance measure:** The K-means algorithm employs a **distance measure** to ascertain the proximity of data points to cluster centers. **Euclidean distance** is a widely utilized metric, though alternative measures such as **Manhattan distance** and **cosine similarity** may be more suitable depending on the inherent traits of the data being examined.
4. **The convergence criteria:** The K-means algorithm utilizes an **iterative process** to continually adjust the centers of the clusters until **convergence is achieved**, typically defined as a point at which there is minimal alteration in the cluster centers between successive iterations. The convergence criteria can be altered to regulate the number of iterations the algorithm executes before ceasing, and can influence the ultimate clusters that are generated.
5. **The scaling of the data:** The K-means algorithm is prone to sensitivity with regards to the magnitude of the features in the dataset, and it may be advantageous to perform scaling on the data prior to implementing the K-means algorithm. **Utilizing the process of scaling on the data** can aid in ensuring that the method for **calculating distance is interpretable** and can prevent certain characteristics from **disproportionately influencing** the calculation of distance.

## Implementation of K-means clustering.

```

1  clear;close all;clear all, clc;
2  % Generate data matrix using gen_clusterdata function
3  ID = 10749174;
4  X = gen_clusterdata(ID);
5
6
7  % Get number of rows (objects or cases) in data matrix
8  N = size(X,1);
9  fprintf('\nTotal Number of Rows = %f\n\n',N);
10
11
12 % Loop through each column (feature) in data matrix
13 for i = 1:size(X,2)
14     % Get mean and standard deviation of column
15     mu = mean(X(:,i));
16     stdDev = std(X(:,i));
17
18     % Print mean and standard deviation
19     fprintf('Column %d: Mean = %f, Standard deviation = %f\n', i, mu, stdDev);
20
21     % Plot histogram of column
22     figure
23     histogram(X(:,i))
24     title(sprintf('Histogram of column %d', i))
25
26     % Add labels and title
27     xlabel('Data');
28     ylabel('Frequency');
29     title(sprintf('Column %d: Mean = %.2f, Std = %.2f', i, mu, stdDev));
30 end
31
32
33 % Get covariance matrix of data matrix
34 C = cov(X);
35
36 % Print covariance matrix
37 fprintf('\nCovariance matrix:\n')
38 disp(C)
39
40 % Get correlation matrix of data matrix
41 R = corr(X);
42
43 % Print correlation matrix
44 fprintf('Correlation matrix:\n')
45 disp(R)
46
47 % Set range of values for K
48 K_range = 3:5;
49
50 % Preallocate variable to store silhouette scores
51 s_all = zeros(length(K_range), 1);
52
53 % Loop over values of K
54 for i = 1:length(K_range)
55     % Perform K-means clustering
56     [IDX, C] = kmeans(X, K_range(i));
57
58     % Compute silhouette score
59     s = silhouette(X, IDX);
60
61     % Store silhouette score
62     s_all(i) = mean(s);
63
64     % Plot silhouette scores for each cluster
65     figure
66     silhouette(X, IDX)
67     title(sprintf('Silhouette scores for K = %d', K_range(i)))
68
69     % Plot clusters and cluster centroids
70     figure
71     gscatter(X(:,1), X(:,2), IDX)
72     hold on
73     plot(C(:,1), C(:,2), 'kx', 'MarkerSize', 15, 'LineWidth', 3)
74     if K_range(i) < 3
75         legend(sprintf('K = %d', K_range(i)), 'Location','best')
76     elseif K_range(i) == 3
77         legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids', 'Location', 'best')
78     elseif K_range(i) == 4
79         legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Centroids', 'Location', 'best')
80     else
81         legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5', 'Centroids', 'Location', 'best')
82     end
83     title(sprintf('K-means clustering for K = %d', K_range(i)))
84 end
85
86 % Find index of K with highest silhouette score
87 [~, idx] = max(s_all);
88
89 % Print mean Silhouette scores for each K value
90 fprintf('Mean silhouette scores:\n')
91 disp(s_all)
92
93 % Print optimal number of clusters
94 fprintf('Optimal number of clusters: %d\n', K_range(idx))

```

## TASK 2.1 – Data Preparation

## DATA:



```
1 clear;close all;clear all, clc;
2 % Generate data matrix using gen_clusterdata function
3 ID = 10749174;
4 X = gen_clusterdata(ID);
```

CODE	Elucidation
<b>Clear; close all; clear all, clc;</b>	•Clear the command window, close all figures, and clear all variables.
<b>ID = 10749174</b>	•Assigns the value 10749174 to the variable <b>ID</b> .
<b>X = gen_clusterdata(ID)</b>	•Generates a data matrix <b>X</b> using the function <code>gen_clusterdata</code> and the input variable <b>ID</b> .
<p>The <b>gen_clusterdata</b> is custom function that generates a data matrix with four columns representing features and a specified number of rows representing objects. The input variable <b>ID</b> specifies the number of rows in the matrix; in this case, it is set to <b>10749174</b>.</p>	

	1	2	3	4	5	6	7	8	9	10	11	12
1	11.2910	13.1264	9.2263	10.0412								
2	5.2238	4.0102	6.4104	3.8844								
3	5.7549	1.6096	4.9398	4.7003								
4	8.4842	7.1552	7.4273	8.4436								
5	14.2743	13.2179	15.1657	12.2117								
6	3.9136	3.7212	4.2565	4.2370								
7	9.8929	9.9136	11.5610	10.8193								
8	10.7717	10.9471	10.2003	10.4440								
9	14.4588	14.8007	14.7175	14.1391								
10	8.4422	9.3464	9.1622	8.1020								
11	15.4528	14.6529	13.3880	13.9691								
12	5.0812	5.0708	4.0076	6.1516								
13	14.0599	15.1633	13.3920	14.1297								
14	12.9832	11.2753	12.2820	12.2166								

Data Analysis:

a).

```
1 % Get number of rows (objects or cases) in data matrix
2 N = size(X,1);
3 fprintf('\nTotal Number of Rows = %f\n\n',N);
```

CODE	Elucidation
N = size(X,1)	obtains the number of rows in the data matrix <b>X</b> and assigns it to the variable <b>N</b> . The function size returns a matrix of size 1-by-2, where the first element is the number of rows and the second element is the number of columns. The input <b>1</b> specifies that the function should return the number of rows
fprintf('\nTotal Number of Rows = %f\n\n',N)	Output <b>N</b> to the command window.



N							
1x1 double							
	1	2	3	4	5	6	7
1	2088						
2							
3							



b).

```

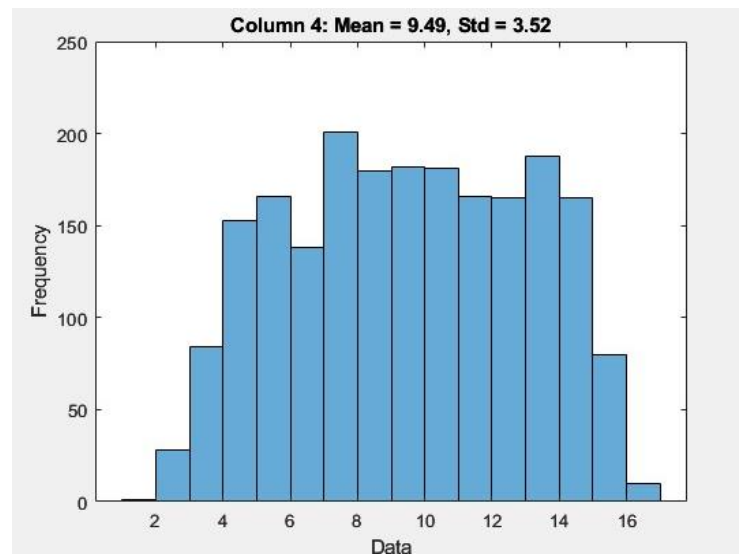
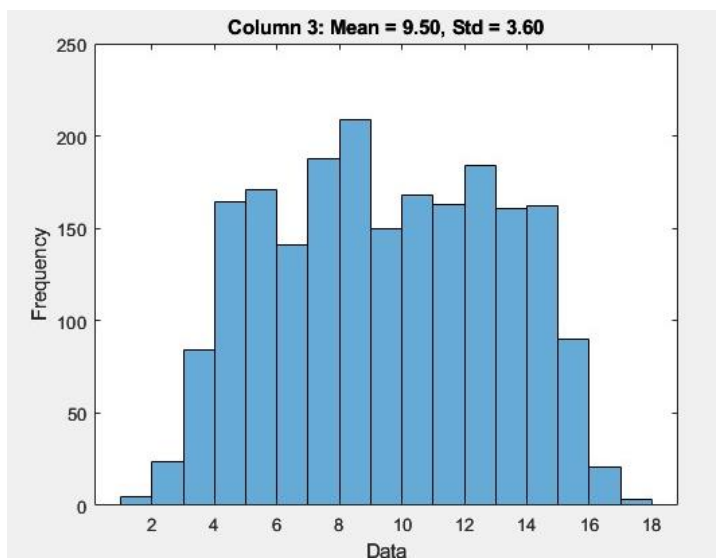
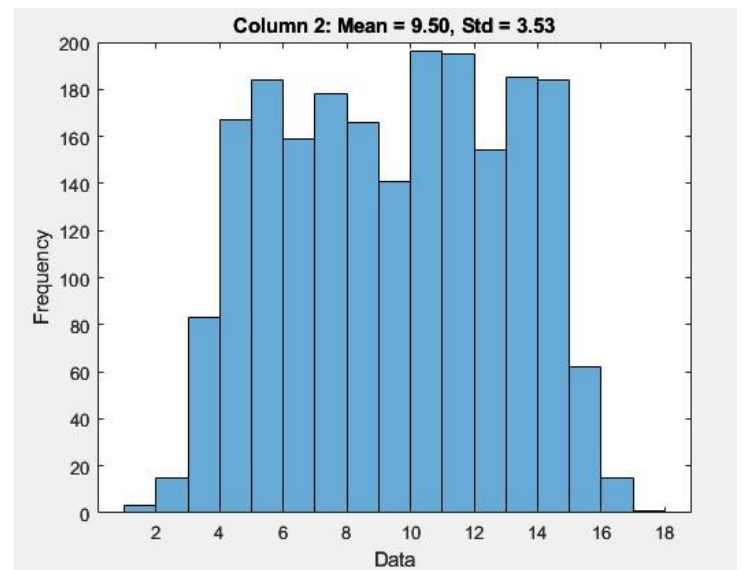
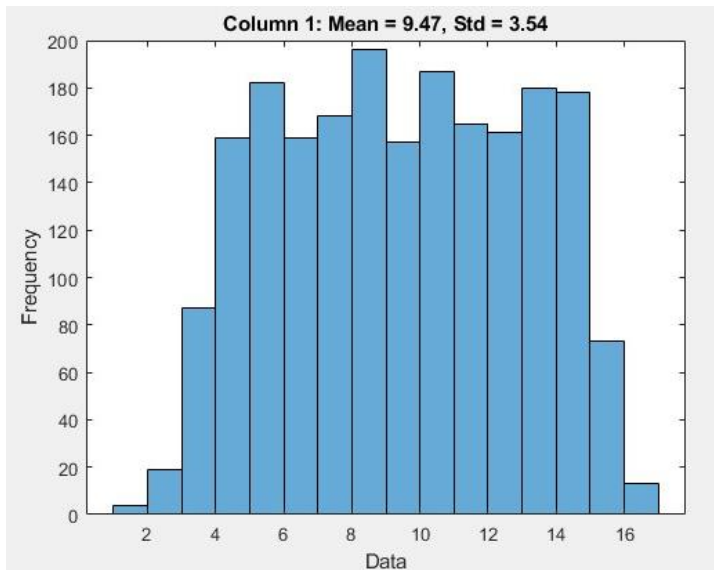
1  % Loop through each column (feature) in data matrix
2  for i = 1:size(X,2)
3      % Get mean and standard deviation of column
4      mu = mean(X(:,i));
5      stdDev = std(X(:,i));
6
7      % Print mean and standard deviation
8      fprintf('Column %d: Mean = %f, Standard deviation = %f\n', i, mu, stdDev);
9
10     % Plot histogram of column
11     figure
12     histogram(X(:,i))
13
14     % Add labels and title
15     xlabel('Data');
16     ylabel('Frequency');
17     title(sprintf('Column %d: Mean = %.2f, Std = %.2f', i, mu, stdDev));
18 end

```

CODE	Elucidation
<b>for i = 1:size(X,2)</b>	Initiates a loop that will iterate through each column in the data matrix <b>X</b> . The function size returns a matrix of size 1-by-2, where the first element is the number of rows and the second element is the number of columns. The input <b>2</b> specifies that the function should return the number of columns. The loop variable <b>i</b> takes on values from <b>1</b> to the number of columns in <b>X</b> .
<b>mu = mean(X(:,i))</b>	Calculates the mean of the <b>i</b> -th column of the data matrix <b>X</b> and assigns it to the variable <b>mu</b>
<b>stdDev = std(X(:,i))</b>	Calculates the standard deviation of the <b>i</b> -th column of the data matrix <b>X</b> and assigns it to the variable <b>stdDev</b> .
<b>fprintf('Column %d: Mean = %f, Standard deviation = %f\n', i, mu, stdDev)</b>	Prints the column index <b>i</b> , mean <b>mu</b> , and standard deviation <b>stdDev</b> to the command window.
<b>figure</b>	Creates a new figure window.
<b>histogram(X(:,i))</b>	Plots a histogram of the <b>i</b> -th column of the data matrix <b>X</b> .
<b>xlabel('Data');</b> <b>ylabel('Frequency');</b>	Adds x-axis and y-axis labels to the histogram plot using the <b>xlabel</b> and <b>ylabel</b> functions, respectively.
<b>title(sprintf('Column %d: Mean = %.2f, Std = %.2f', i, mu, stdDev));</b>	Adds a title to the histogram plot using the title function. The function <b>sprintf</b> is utilized to format the title string, which includes the column index <b>i</b> , mean <b>mu</b> , and standard deviation <b>stdDev</b> .

## Command Window

```
Column 1: Mean = 9.466589, Standard deviation = 3.538398  
Column 2: Mean = 9.499758, Standard deviation = 3.534954  
Column 3: Mean = 9.496809, Standard deviation = 3.601679  
Column 4: Mean = 9.491418, Standard deviation = 3.520630
```



c)

```
1 % Get covariance matrix of data matrix
2 C = cov(X);
3
4 % Print covariance matrix
5 fprintf('\nCovariance matrix:\n')
6 disp(C)
7
8 % Get correlation matrix of data matrix
9 R = corr(X);
10
11 % Print correlation matrix
12 fprintf('Correlation matrix:\n')
13 disp(R)
```

#### Elucidation

This code computes the covariance and correlation matrices of the data matrix **X** using the **cov** and **corr** functions, respectively. The resulting matrices are stored in the variables **C** and **R**, and are subsequently printed to the command window using the **fprintf** and **disp** functions."

#### Command Window

##### Covariance matrix:

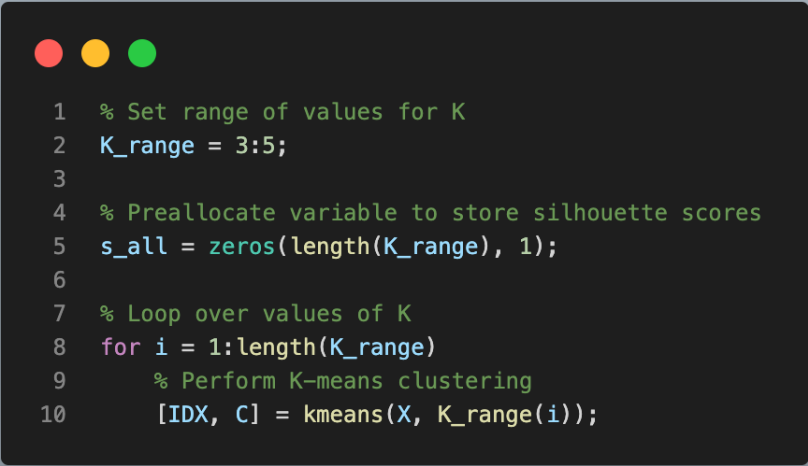
12.5203	11.4568	11.6101	11.4251
11.4568	12.4959	11.5435	11.4014
11.6101	11.5435	12.9721	11.4641
11.4251	11.4014	11.4641	12.3948

##### Correlation matrix:

1.0000	0.9160	0.9110	0.9171
0.9160	1.0000	0.9067	0.9161
0.9110	0.9067	1.0000	0.9041
0.9171	0.9161	0.9041	1.0000

## TASK 2.2 – K-Means

1.



```
1 % Set range of values for K
2 K_range = 3:5;
3
4 % Preallocate variable to store silhouette scores
5 s_all = zeros(length(K_range), 1);
6
7 % Loop over values of K
8 for i = 1:length(K_range)
9     % Perform K-means clustering
10    [IDX, C] = kmeans(X, K_range(i));
```

## Elucidation

This code establishes a loop to evaluate various values of **K** for the K-means clustering algorithm. The range of values for **K** is specified using the **K\_range** array, which consists of the values **3**, **4**, and **5**. A variable **s\_all** is preallocated to store the silhouette scores for each value of **K**. The loop iterates over the values in the **K\_range** array and performs K-means clustering for each value of **K**. The kmeans function returns the cluster indices and cluster centroids, which are stored in the variables **IDX** and **C**, respectively.

2.

```

1 % Compute silhouette score
2     s = silhouette(X, IDX);
3
4 % Store silhouette score
5     s_all(i) = mean(s);
6
7 % Plot silhouette scores for each cluster
8     figure
9     silhouette(X, IDX)
10    title(sprintf('Silhouette scores for K = %d', K_range(i)))

```

#### Elucidation

This code computes the silhouette score for each cluster using the **silhouette function** and stores the mean silhouette score in the **s\_all** array. The silhouette scores for each cluster are also plotted using the **silhouette function**, and a title is added to the plot using the **title** function. The title includes the **value of K** for the current iteration of the loop.

```

1 % Find index of K with highest silhouette score
2     [~, idx] = max(s_all);
3
4 % Print mean Silhouette scores for each K value
5     fprintf('Mean silhouette scores:\n')
6     disp(s_all)

```

CODE	Elucidation
<b>[~, idx] = max(s_all)</b>	Identifies the index of the maximum value in the vector <b>s_all</b> and assigns it to the variable <b>idx</b> . The <b>~</b> symbol is used to ignore the output of the <b>function max</b> , which returns the maximum value and its index.
<b>fprintf('Mean silhouette scores:\n')</b>	Print mean silhouette score to the command window.
<b>disp(s_all)</b>	Display in command window



## Command Window

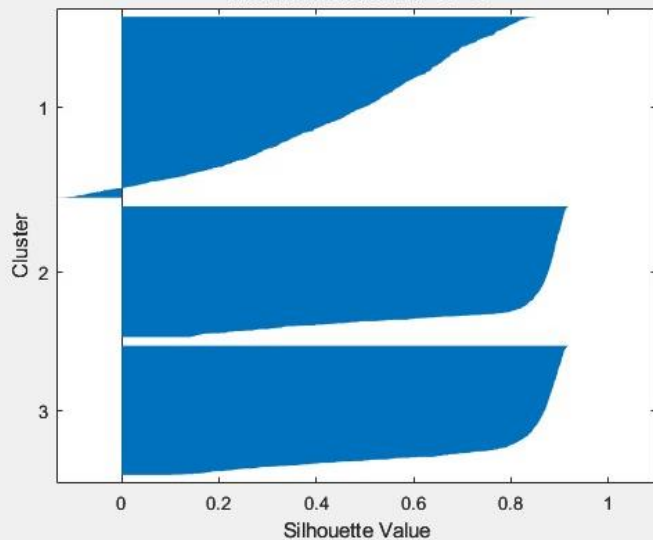
Mean silhouette scores:

0.6538

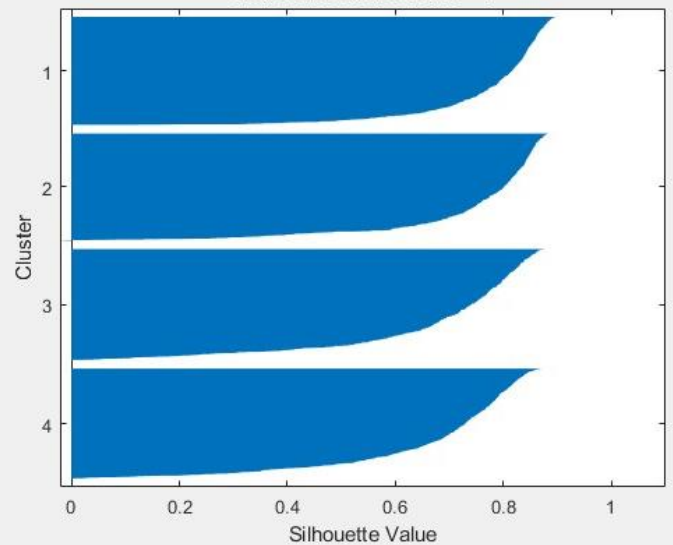
0.7196

0.5984

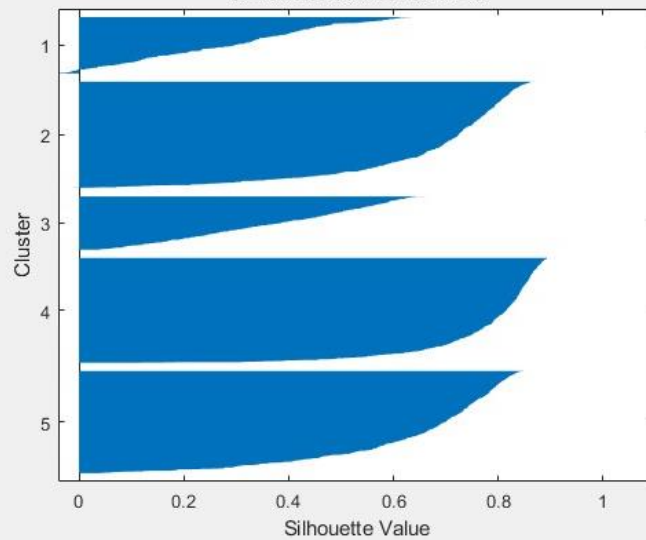
Silhouette scores for K = 3



Silhouette scores for K = 4



Silhouette scores for K = 5



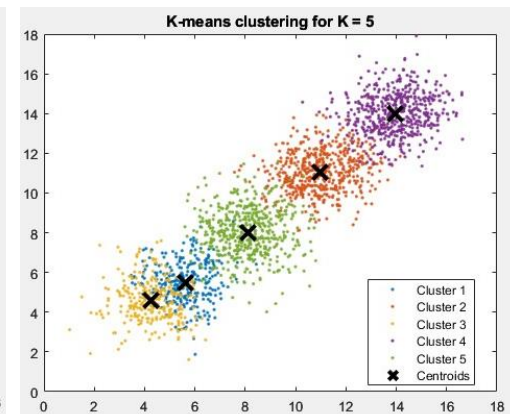
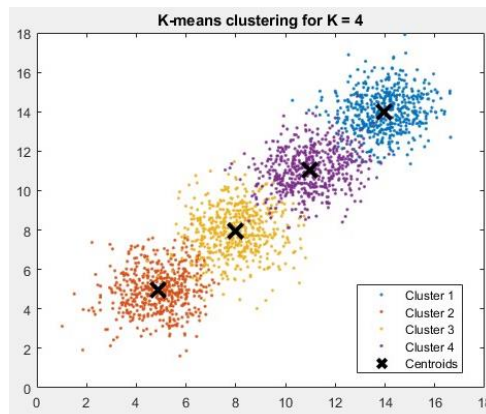
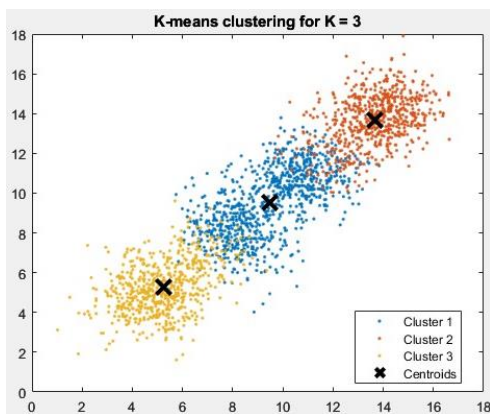
3. The termination condition for K-means clustering is satisfied when the assignments of data points to clusters remains static, or when the maximum number of iterations has been reached. The maximum number of iterations is determined by the `MaxIter` parameter in the `kmeans` function. The default value for `MaxIter` is 10, but this can be modified to suit the requirements of the analysis.

```

1 % Plot clusters and cluster centroids
2 figure
3 gscatter(X(:,1), X(:,2), IDX)
4 hold on
5 plot(C(:,1), C(:,2), 'kx', 'MarkerSize', 15, 'LineWidth', 3)
6 if K_range(i) < 3
7     legend(sprintf('K = %d', K_range(i)), 'Location', 'best')
8 elseif K_range(i) == 3
9     legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids', 'Location', 'best')
10 elseif K_range(i) == 4
11     legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Centroids', 'Location', 'best')
12 else
13     legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5', 'Centroids', 'Location', 'best')
14 end
15 title(sprintf('K-means clustering for K = %d', K_range(i)))
16 end

```

CODE	Elucidation
<code>gscatter(X(:,1), X(:,2), IDX)</code>	Plot the data points in the data matrix <b>X</b> using the first and second columns as the x- and y-axes, respectively. The assignments of each data point to a cluster are depicted by diverse colors through the use of the <b>gscatter</b> function.
<code>plot(C(:,1), C(:,2), 'kx', 'MarkerSize', 15, 'LineWidth', 3)</code>	Depicts the cluster centroids as black crosses using the <b>plot</b> function. The coordinates for the cluster centroids are derived from the first and second columns of matrix <b>C</b> . The <b>MarkerSize</b> and <b>LineWidth</b> properties are utilized to specify the size and thickness of the crosses, respectively.
Additionally, it appends a <b>legend</b> to the plot based on the <b>value of K</b> . The legend consists of one entry for each cluster and one entry for the centroids. The <b>if</b> and <b>elseif</b> statements are utilized to determine the number of entries in the legend based on the value of <b>K</b> . The <b>sprintf</b> function is employed to format the legend string. The <b>Location</b> property is set to <b>'best'</b> to position the legend in the most appropriate location. Finally, the code adds a title to the plot through the use of the <b>title</b> function.	



## 4.



```
1 % Print optimal number of clusters
2 fprintf('Optimal number of clusters: %d\n', K_range(idx))
```

### Elucidation

This code outputs the optimal number of clusters to the command window by identifying the value of K that corresponds to the highest silhouette score and printing it using the **fprintf** function

```
Command Window

Mean silhouette scores:
    0.6538
    0.7196
    0.5984

Optimal number of clusters: 4
>>
```

### Why k=4 is the best number of clusters?

To determine the best number of clusters, the silhouette score can be utilized as a measure of the compactness and separation of the clusters. The silhouette score is a metric that ranges from -1 to 1, with higher scores indicating more optimal cluster performance. Thus, taking these factors into consideration, it can be inferred that k=4 represents the best number of clusters.

## 5. Limitations and drawbacks of K-means Clustering.

- K-means clustering postulates that the clusters are spherical and uniformly sized, which may not always be true for real-world data.
- The outcomes of K-means clustering are delicate to the initial centroid positions. If the initial centroids are not selected with care, the algorithm may converge to a suboptimal solution.
- K-means clustering may not be effective for data that is not distinctly separated or that exhibits complex shapes.

## Discussions and Conclusions

In this analysis, the K-means clustering algorithm was utilized to determine the optimal number of clusters for a given data matrix. The silhouette measure was employed to assess the performance of the algorithm for various values of K. Based on the results of the analysis, it was concluded that the optimal number of clusters was 4. This determination was made by identifying the value of K that corresponded to the highest mean silhouette score.

In conclusion, K-means clustering is a widely used unsupervised machine learning algorithm that can be useful for uncovering patterns in data. However, it has a number of limitations and drawbacks.

One of the drawbacks of K-means clustering is that it necessitates the user to specify the number of clusters beforehand. This can be challenging if the true number of clusters is unknown or if the data does not clearly segregate into distinct clusters. Another limitation of K-means clustering is that it is sensitive to the initial placement of the cluster centroids, which can influence the final cluster assignments.

It is important to consider these limitations when determining if K-means clustering is the appropriate method for a given problem.

## REFERENCES

Jordan, M.I. and Mitchell, T.M. (2015) "Machine learning: Trends, Perspectives, and prospects," *Science*, 349(6245), pp. 255–260. Available at: <https://doi.org/10.1126/science.aaa8415>.

Barlow, H.B. (1989) "Unsupervised learning," *Neural Computation*, 1(3), pp. 295–311. Available at: <https://doi.org/10.1162/neco.1989.1.3.295>.

Usama, M. et al. (2019) "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE Access*, 7, pp. 65579–65615. Available at: <https://doi.org/10.1109/access.2019.2916648>.

Mukhopadhyay, S. (2018) *Advanced Data Analytics using python: With machine learning, deep learning and NLP examples*. Berkeley, CA: Apress.

Matt.O (2019) 10 tips for choosing the optimal number of clusters, Medium. Towards Data Science. Available at: <https://towardsdatascience.com/10-tips-for-choosing-the-optimal-number-of-clusters-277e93d72d92> (Accessed: December 26, 2022).



## Appendix

Matlab Code:

```
clear;close all;clear all, clc;
% Generate data matrix using gen_clusterdata function
ID = 10749174;
X = gen_clusterdata(ID);

% Get number of rows (objects or cases) in data matrix
N = size(X,1);
fprintf('\nTotal Number of Rows = %f\n\n',N);

% Loop through each column (feature) in data matrix
for i = 1:size(X,2)
    % Get mean and standard deviation of column
    mu = mean(X(:,i));
    stdDev = std(X(:,i));

    % Print mean and standard deviation
    fprintf('Column %d: Mean = %f, Standard deviation = %f\n', i, mu, stdDev);

    % Plot histogram of column
    figure
    histogram(X(:,i))

    % Add labels and title
    xlabel('Data');
    ylabel('Frequency');
    title(sprintf('Column %d: Mean = %.2f, Std = %.2f', i, mu, stdDev));
end

% Get covariance matrix of data matrix
C = cov(X);

% Print covariance matrix
fprintf('\nCovariance matrix:\n')
disp(C)

% Get correlation matrix of data matrix
R = corr(X);

% Print correlation matrix
fprintf('Correlation matrix:\n')
disp(R)

% Set range of values for K
K_range = 3:5;

% Preallocate variable to store silhouette scores
s_all = zeros(length(K_range), 1);

% Loop over values of K
for i = 1:length(K_range)
    % Perform K-means clustering
    [IDX, C] = kmeans(X, K_range(i));

    % Compute silhouette score
    s = silhouette(X, IDX);
```

```
% Store silhouette score
s_all(i) = mean(s);

% Plot silhouette scores for each cluster
figure
silhouette(X, IDX)
title(sprintf('Silhouette scores for K = %d', K_range(i)))

% Plot clusters and cluster centroids
figure
gscatter(X(:,1), X(:,2), IDX)
hold on
plot(C(:,1), C(:,2), 'kx', 'MarkerSize', 15, 'LineWidth', 3)
if K_range(i) < 3
    legend(sprintf('K = %d', K_range(i)), 'Location','best')
elseif K_range(i) == 3
    legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids', 'Location', 'best')
elseif K_range(i) == 4
    legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Centroids', 'Location',
'best')
else
    legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5', 'Centroids',
'Location', 'best')
end

title(sprintf('K-means clustering for K = %d', K_range(i)))
end

% Find index of K with highest silhouette score
[~, idx] = max(s_all);

% Print mean Silhouette scores for each K value
fprintf('Mean silhouette scores:\n')
disp(s_all)

% Print optimal number of clusters
fprintf('Optimal number of clusters: %d\n', K_range(idx))
```