EXPERIMENT - 7

AIM: To implement different clustering algorithms.
Problem Statement: a) Clustering algorithm for unsupervised classification (K-means, density based
(DBSCAN), Hierarchical clustering)
b) Plot the cluster data and show mathematical steps.

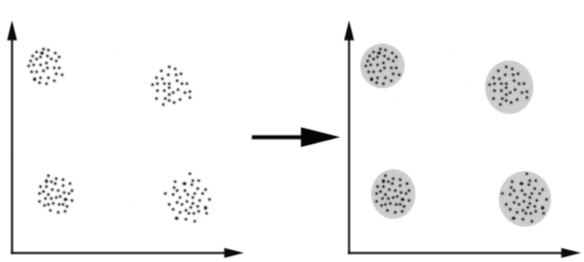
ABOUT DATASET:

Link to our dataset:
https://drive.google.com/file/d/1PYbyyI1d4Im3StrsQMts94Xg0ZuaRA-b/view?usp=sharing

The data set contains basic flight data information about people. It consists of columns such as "age","no_of_flight_taken","spending score" etc.

THEORY:

Clustering is an unsupervised machine learning task. You might also hear this referred to as cluster analysis because of the way this method works.
Using a clustering algorithm means you're going to give the algorithm a lot of input data with no labels and let it find any groupings in the data it can.
Those groupings are called *clusters*. A cluster is a group of data points that are similar to each other based on their relation to surrounding data points. Clustering is used for things like feature engineering or pattern discovery.



K-means clustering algorithm

K-means clustering is the most commonly used clustering algorithm. It's a centroid-based algorithm and the simplest unsupervised learning algorithm.

This algorithm tries to minimize the variance of data points within a cluster. It's also how most people are introduced to unsupervised machine learning. K-means is best used on smaller data sets because it iterates over *all* of the data points. That means it'll take more time to classify data points if there are a large amount of them in the data set. Since this is how k-means clusters data points, it doesn't scale well.

DBSCAN clustering algorithm

DBSCAN stands for density-based spatial clustering of applications with noise. It's a density-based clustering algorithm, unlike k-means. This is a good algorithm for finding outliers in a data set. It finds arbitrarily shaped clusters based on the density of data points in different regions. It separates regions by areas of low-density so that it can detect outliers between the high-density clusters. This algorithm is better than k-means when it comes to working with oddly shaped data.

DBSCAN uses two parameters to determine how clusters are defined: *minPts* (the minimum number of data points that need to be clustered together for an area to be considered high-density) and *eps* (the distance used to determine if a data point is in the same area as other data points). Choosing the right initial parameters is critical for this algorithm to work.

Hierarchical Clustering Algorithms: Given a set of N items to be clustered, and an N*N distance (or similarity) matrix, the basic process of hierarchical clustering is this:

- Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters the same as the distances (similarities) between the items they contain.
- Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.
- Compute distances (similarities) between the new cluster and each of the old clusters.
- Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

IMPLEMENTATION:

- **Importing dataset**

```
[38] import numpy as np # to handle numeric data
     import matplotlib.pyplot as plt # for visualization
     import pandas as pd # for handling dataframe
     #Import the libraries
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sb
```

```
ourData = pd.read_csv('Flight_dataset.csv') # read the data
ourData.head() # print the
```

	ID	Gender	Age	No_of_Flight_taken	Spending_Score(1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

- **K-means clustering**

```
!pip install scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (1.2.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (3.1.0)
```

```
[66] from sklearn.cluster import KMeans
```

```
[67] X = ourData[['Age', 'Spending_Score(1-100)']].copy()
```

```
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=0)
    kmeans.fit(X)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
```

```
[69] wcss= []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++',random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

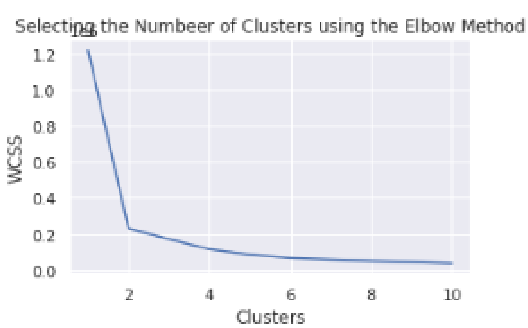
```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
  warnings.warn(
```

```
[70] import matplotlib.pyplot as plt
import seaborn as sns
```

- ➢ Thus we selected the number of clusters using the elbow method.

```
[71] sns.set()
```

```
plt.plot(range(1, 11), wcss)
plt.title('Selecting the Numbeer of Clusters using the Elbow Method')
plt.xlabel('Clusters')
plt.ylabel('WCSS')
plt.show()
```



➢ Using the gaussian mixture to identify and display the cluster

```
[77] from sklearn.mixture import GaussianMixture
```

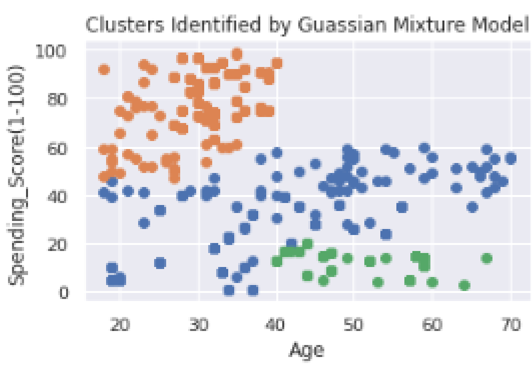
```
from sklearn.mixture import GaussianMixture
n_clusters = 3
gmm_model = GaussianMixture(n_components=n_clusters)
gmm_model.fit(X)
```

```
GaussianMixture
GaussianMixture(n_components=3)
```

```
[79] cluster_labels = gmm_model.predict(X)
X = pd.DataFrame(X)
X['cluster'] = cluster_labels
```

```
for k in range(0,n_clusters):
    data = X[X["cluster"]==k]
    plt.scatter(data["Age"],data["Spending_Score(1-100)"])

plt.title("Clusters Identified by Guassian Mixture Model")
plt.ylabel("Spending_Score(1-100)")
plt.xlabel("Age")
plt.show()
```



➢ Thus the above cluster shows the amount of spending score of each age group

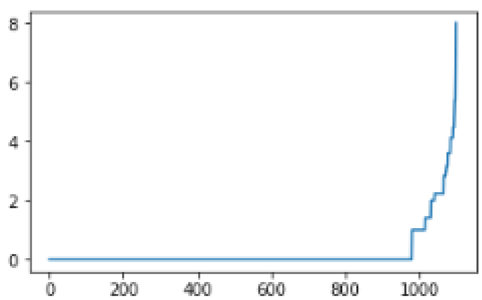
- **DBSCAN**

DBSCAN

```
[49] x = ourData.loc[:, ['Age',
                  'Spending_Score(1-100)']].values
```

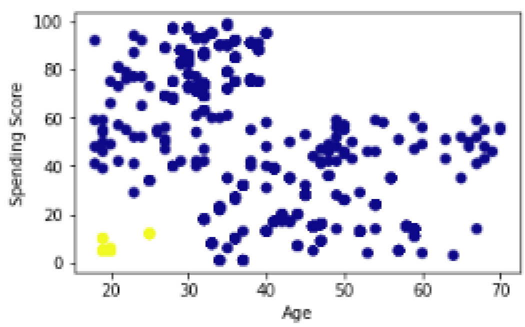
```
[50] from sklearn.neighbors import NearestNeighbors # importing the library
     neighb = NearestNeighbors(n_neighbors=2)
     nbrs=neighb.fit(x) # fitting the data to the object
     distances,indices=nbrs.kneighbors(x) #
```

```
# Sort and plot the distances results
distances = np.sort(distances, axis = 0) # sorting the distances
distances = distances[:, 1] # taking the second column of the sorted distances
plt.rcParams['figure.figsize'] = (5,3) # setting the figure size
plt.plot(distances) # plotting the distances
plt.show() # show
```



```
[52] from sklearn.cluster import DBSCAN
     # cluster the data into five clusters
     dbscan = DBSCAN(eps = 8, min_samples = 4).fit(x) # fitting the model
     labels = dbscan.labels_ # getting the labels
```

```
# Plot the clusters
plt.scatter(x[:, 0], x[:,1], c = labels, cmap= "plasma") # plotting the clusters
plt.xlabel("Age") # X-axis label
plt.ylabel("Spending Score") # Y-axis label
plt.show() # showing the plot
```

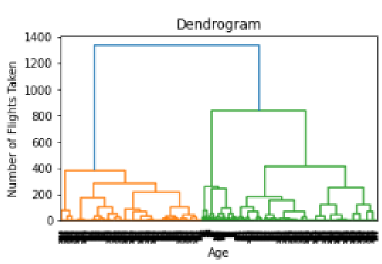


- **Hierarchical clustering**

➢ First we chose two features (ie. Age and Number of flights taken)
➢ Then we created a dendrogram to find the optimal number of clusters.

```
[40] newData = ourData.iloc[:, [3, 4]].values # extract the two features from our dataset
```

```
import scipy.cluster.hierarchy as sch # importing scipy.cluster.hierarchy for dendrogram
dendrogram = sch.dendrogram(sch.linkage(newData, method = 'ward')) # finding the optimal number of clusters using dendrogram
plt.title('Dendrogram') # title of the dendrogram
plt.xlabel('Age') # label of the x-axis
plt.ylabel('Number of Flights Taken') # label of the y-axis
plt.show()
```



➢ According to our dendrogram, we can say the optimal number of clusters would be around 5-6 clusters.

➢ After that, we need to create an AgglomerativeClustering object, and in it, we pass the following parameters:

★ n_cluster= 5, the number of clusters our model should return

★ affinity=euclidean, specify metric to be used to calculate distances

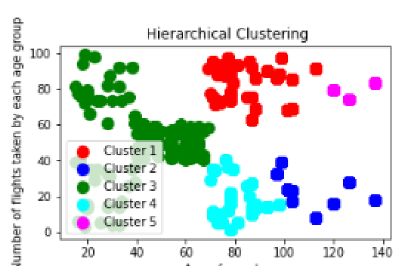
★ linkage= ward to regulate how distance calculation will be carried out between different clusters.

```
from sklearn.cluster import AgglomerativeClustering # this line of code imports AgglomerativeClustering model from sk-learn
Agg_hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = Agg_hc.fit_predict(newData) # model fitting on the dataset
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_agglomerative.py:983: FutureWarning: Attribute `affinity` was deprecated i
  warnings.warn(
```

➢ After that, we will simply display our cluster.

```
# plotting cluster 1
plt.scatter(newData[y_hc == 0, 0], newData[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1') # plotting cluster 2
plt.scatter(newData[y_hc == 1, 0], newData[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2') # plotting cluster 3
plt.scatter(newData[y_hc == 2, 0], newData[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3') # plotting cluster 4
plt.scatter(newData[y_hc == 3, 0], newData[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')  # plotting cluster 5
plt.scatter(newData[y_hc == 4, 0], newData[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
# plot title addition
plt.title(' Hierarchical Clustering')
# labelling the x-axis
plt.xlabel('Age of people')
# label of the y-axis
plt.ylabel('Number of flights taken by each age group')
# printing the legend
plt.legend()
# show the plot
plt.show()
```



CONCLUSION: Thus we successfully implemented different clustering algorithms. We showed the clustering for Age and their respective spending score using K Means and DBSCAN. We showed the clustering for Number of flights taken per age group using hierarchical clustering. In. K Means we used the elbow method and used the gaussian mixture method to display clusters. In Hierarchical clustering , we used dendrograms to show the optimal number of clusters.