Week 4
Clustering Techniques

In every technique covered in this class, the focused issues are

- The concept of the classification technique
    - This will be lectured early in the class
- Creation of the clustering models in Python 3 notebook
- Application of the model to a simple dataset

The dataset for this class :

- Mall_Customers.csv
- Two moons from sklearn.datasets to be generated by a small code segments

1) Upon mounting the google drive as usual, the follow code reads the dataset, presumably placed in the same folder as the notebook.

```
import pandas as pd
import numpy as np

dataset = pd.read_csv('Mall_Customers.csv')
```

2) The following code extract the two attributes of interest, the Annual Income and the Spending Score of each customer, from the dataset, as a numpy array.

```
X = dataset.iloc[:, [3, 4]].values
```

3) The following code will create a k-means model.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters, init)
```

   `init` can be `'k-means++'` (default) or `'random'`

4) Regression process to fit the model to the dataset is done by

```
kmeans.fit(X)
```

   Upon fitting, `kmeans.inertia_` will be the value of WCSS (Within-Cluster Sum of Square). Write a code to loop values of k from 2 to 11 and plot the curve of WCSS for each k.

   According to the Elbow Method, what should be an appropriate value of k ?

5) The prediction can be computed by `Y = kmeans.fit_predict(X)`

6) The following code will calculate Silhouette score for each fitted model.

```
from sklearn import metrics
score = metrics.silhouette_score(X, Y)
```

   According to Silhouette scores, what should be an appropriate value of k ?

7) Use the best value of k, create the final model with k-means clustering for the dataset. Calculate the output. Then the following code will create the scattered plot for the dataset, identifying each cluster with a distinct color.

```
def show_clusters(X, Y):
  plt.scatter(X[:, 0], X[:, 1], c=Y, cmap="plasma")
  plt.xlabel('Annual Income (k$)')
  plt.ylabel('Spending Score (1-100)')
  plt.show()
```

8) The following code will install Fuzzy C Means library into your Colab.

```
!pip install fuzzy-c-means
```

9) Then, the Fuzzy C Means model can be created in similar way to k-means model, and perform fitting regression, as the following.

```
from fcmeans import FCM

fcm = FCM(n_clusters, m)
fcm.fit(X)

Y = fcm.predict(X)
```

Experiment with the values of `m` to see how fuzziness can be tuned. The clustering result can be seen using the plotting function as before.

10) Both the k-means and the Fuzzy C Means libraries have variables that store the centroids' center. If interested, research to access them and add them on the scattered plot so you can see what the centers are. This is not required.

11) In the similar way, the following code will create a DBSCAN model and performing fitting regression.

```
from sklearn.cluster import DBSCAN
db = DBSCAN(eps, min_samples)
db.fit(X)

Y = db.fit_predict(X)
```

Experiment with the values of `eps` and `min_samples.` The clustering result can be seen using the plotting function as before.

12) The following code generates a sample set of Two Moons dataset.

```
from sklearn.datasets import make_moons
#moons_X: Data, moon_y: Labels
moons_X, moons_Y = make_moons(n_samples = 2000)
show_clusters(moons_X, moons_Y)
```

13) Try using k-means with the Two Moons dataset to see how close to the original labels i.e. moons_Y your model can match.

14) It is possible to tune the DBSCAN model to perfectly match the original labels of Two Moons dataset. Experiment on tuning its paramaters to achieve such result.

15) The following code will install Self-Organizing Map library.

```
!pip install sklearn-som
```

16) In the similar way, the following code will create a DBSCAN model and performing fitting regression.

```
from sklearn_som.som import SOM

som = SOM(m, n, dim)
som.fit(X)
```

Common value for `dim` is 2 (2-D relationship between neurons in the map)
The number of neurons (maximum number of clusters possible) is `m*n`

As in most models, the following code will compute outputs.

```
Y = som.fit_predict(X)
```

17) Experiment with values of SOM parameters to create a properly fit model for the Mall_Customers dataset.