**MLPR Lab - Week 2**

Write a code that generates random data and cluster the datapoints using the K-Means algorithm. For a range of value of K from 1 to 20, compute the within-cluster sum of squares distance (WCSS) to find the optimal number of clusters using the Elbow method.

[K Means Clustering | Step-by-Step Tutorials For Data Analysis (analyticsvidhya.com)](https://www.analyticsvidhya.com/blog/2021/04/k-means-clustering-simplified-in-python/)

The output of the code should be:

a) Elbow Plot showing that 4 clusters are optimal for data generated below

b) The clustered data for 4 clusters

**Step 1:** Import Necessary Libraries

* Numpy
* Matplotlib
* Kmeans from Sklearn (***from sklearn.cluster import KMeans***)

**Step 2:** Generate synthetic/random data.

* Define the size of the data.
* Use ***np.random.seed(0)*** for reproducibility of random data generation. It would generate same random data on every code run thus provide consistency in the results to compare at different iterations.

**Step 3:** Define mean (mu) and covariance (Sigma) matrices for each cluster.

* Keep values of mean and covariance as given below for the four gaussian distribution.

*mu1 = [2, 2]*

*sigma1 = [[0.9, -0.0255], [-0.0255, 0.9]]*

*mu2 = [5, 5]*

*sigma2 = [[0.5, 0], [0, 0.3]]*

*mu3 = [-2, -2]*

*sigma3 = [[1, 0], [0, 0.9]]*

*mu4 = [-4, 8]*

*sigma4 = [[0.8, 0], [0, 0.6]]*

**Step 4:** Generate synthetic data by drawing samples from each distribution.

* Generate synthetic data using ***np.random.multivariate\_normal().***

[numpy.random.multivariate\_normal — NumPy v1.25 Manual](https://numpy.org/doc/stable/reference/random/generated/numpy.random.multivariate_normal.html)

* Stack data points drawn from the four gaussian distributions using ***np.vstack( ).***

**Step 5:** Initiate Within-Cluster Sum of Squares (WCSS) and store in empty list.

* Loop through a range of cluster counts from let’s say 1 to 20.
* Create a K-Means instance with the current cluster count (keep ***random\_state=0***) and fit it to generated data. Use ***Kmeans.fit()***
* Find WCSS for each iteration of K from 1 to 20.

**Step 6:** Perform K-Means Clustering

* Perform k-means clustering with optimal number of clusters that is given by the Elbow Method plot (should be 4)

**Step 7:** Visualize the Data and Clustering Results

* Data Plot
* Elbow Method Plot
* Clustered Data Plot

**Submission Instructions:**

* Items to be uploaded on LMS.
  + Main code file (.ipynb/.py)
  + Data Plot (PNG/JPG)
  + Elbow Method plot (PNG/JPG)
  + Clustered Data Plot (PNG/JPG)
* Due time and date are given on LMS. Submit it before the deadline.

**Output reference:**

