**Kmeans clustering**

Download [dataset](https://iubedubd-my.sharepoint.com/:u:/g/personal/akmmrahman_iub_edu_bd/EbJtzPTvamRItr8t-O0_pNMBgC60rBh008qyNc-bAx_-Zg?e=qytqK7) from here. Use iris.dat for your data file.

Task: **TODO**

1. You have to implement K-means clustering algorithm with the iris data with 3 dimensions.
2. A sample [implementation file](https://iubedubd-my.sharepoint.com/:u:/g/personal/akmmrahman_iub_edu_bd/EXaEzfDaS25PpRUem-StwxYBVp8UyU171ZVDSW6lgASD_g?e=pS8ZrZ) is provided. Please download it. You can use notebook to complete the **TODOs**. The implementation file is actually a skeleton or pseudocodes. You need to fill up your code blocks where asked. Numpy array has been used to store the datapoints, centroids, indices in this skeleton.
3. At first, data are loaded. You can use the load function or panda function to load numerical data in numpy matrix. Use K = 3
4. Use initialize\_centroids\_simple() to initialize your centroids. This is the simple assignment function you need to implement. Randomly select **K points** from the sampled data and assign them as initialized centroids.
5. Then, in the kmeans function,
   1. You have to calculate the cluster\_affiliation array based on the distance of each data points from all centroids.
   2. you have to write your own code to count the number of points assigned for each cluster based on the cluster\_affiliation and store in the defined structure clutser\_point\_count
   3. You have to re-compute the centroids based on the cluster\_affiliation and clutser\_point\_count
   4. Then write your own code to terminate the process based on the termination criteria discussed in the class. We evaluate the quality of the clustering using the clustering objective



Where N is the total number of sampled points. ***xi*** is the ith data point. zk is the centroid for kth cluster. The algorithm is terminated when J is nearly equal in two successive iterations (e.g., we terminate when |J − Jprev| ≤ 10−5J, where Jprev is the value of J after the previous iteration, flag = False).

* 1. In the main function, draw a 3D plot where:
     1. plot the data points. Points in different clusters will have different colors. Such as for cluster 0: use blue, for cluster 1: red, cluster 2: yellow
     2. Using the ClassLabel data, determine the shape of the each data point (triangle, square or circle).
     3. Visually calculate how many data points are wrongly clustered where most of the data points are same in a cluster.