Due date: April 8, 2025
Return your code with the answer dictionaries as usual.
Return an assignment report together with your repository. The README.md file contains instructions on how to name your report.

Assignment 2: K-means and Hierarchical Clustering

At a high level, you will be building and evaluating different cluster algorithms. At a more granular level, you will be doing the following tasks:

1. **Evaluation of k-Means over Diverse Datasets:** In the first task, you will explore how k-Means perform on datasets with diverse structure.

   A. Load the following 5 datasets with 100 samples each: noisy_circles, noisy_moons, blobs with varied variances, Anisotropic distributed data, blobs. Use the parameters from (https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html) , with the `random state` set to 42.

   B. Write a function called `fit_kmeans` that takes dataset, i.e., pair of (data, label) Numpy arrays, and the number of clusters as arguments, and returns the predicted labels from k-means clustering. Use the random_init argument and make sure to standardize the data (see StandardScaler transform), prior to fitting the KMeans estimator.

   C. Make a big figure (4 rows x 5 columns) of scatter plots (where points are colored by predicted label) with each column corresponding to the datasets generated in part 1.A, and each row being k=[2,3,5,10] different number of clusters. Which datasets does k-means seem to produce correct clusters for (assuming the right number of k is specified) and for which datasets does k-means fail for all values of k?

D. Repeat 1.C a few times and comment on which (if any) datasets seem to be sensitive to the choice of initialization for the k=2,3 cases. You do not need to add the additional plots to your report.
Return a list of the datasets that are sensitive to the initialization.

2. **Comparison of Clustering Evaluation Metrics:** In this task you will explore different methods to find a good value for k

   A. Call the `make_blobs` function with following parameters: (center_box=(-20,20), n_samples=20, centers=5, random_state=12).

   B. Modify the fit_kmeans function to return the SSE (see Equations 8.1 and 8.2 in the book).

   C. Plot the SSE as a function of k for k=1,2,….,8, and choose the optimal k based on the elbow method.

   D. Repeat part 2.C for inertia (note this is an attribute in the kmeans estimator called _inertia). Do the optimal k's agree?

3. **Hierarchical Clustering:** Recall from lecture that agglomerative hierarchical clustering is a greedy iterative scheme that creates clusters, i.e., distinct sets of indices of points, by gradually merging the sets based on some cluster dissimilarity (distance) measure. Since each iteration merges a set of indices there are at most n-1 mergers until the all the data points are merged into a single cluster (assuming n is the total points). This merging process of the sets of indices can be illustrated by a tree diagram called a dendrogram. Hence, agglomerative hierarchal clustering can be simply defined as a function that takes in a set of points and outputs the dendrogram.

   A. Load the provided dataset "hierarchichal_toy_data.mat" using the *spicy.loadmat* function.

B. Create a linkage matrix Z, and plot a dendrogram using the *scipy.hierarchy.linkage and scipy.hierachy.dendrogram* functions, with "single" linkage.

C. Consider the merger of the cluster corresponding to points with index sets {I={8,2,13}} J={1,9}}. At what iteration (starting from 0) were these clusters merged? That is, what row does the merger of A correspond to in the linkage matrix Z?

D. Write a function that takes the data and the two index sets {I,J} above, and returns the dissimilarity given by single link clustering using the Euclidian distance metric. The function should output the same value as the 3rd column of the row found in problem 2.C.

E. In the actual algorithm, deciding which clusters to merge should consider all the available clusters at each iteration. List all the clusters as index sets, e.g., {{0,1,2}, {3,4}, {5}, {6}, …}, that were available when the two clusters in part 2.D were merged.

F. Single linked clustering is often criticized as producing clusters where "the rich get richer", that is, where one cluster is continuously merging with all available points. Does your dendrogram illustrate this phenomenon?

4. **Evaluation of Hierarchical Clustering over Diverse Datasets:** In this task, you will explore hierarchical clustering over different datasets. You will also evaluate different ways to merge clusters and good ways to find the cut-off point for breaking the dendrogram

   A. Repeat part 1.A and part 1.B with hierarchical clustering. That is, write a function called *fit_hierarchical_cluster* (an empty function is provided) that takes the dataset, the linkage type and the

number of clusters, that trains an *AgglomerativeClustering* sklearn estimator and returns the label predictions.

B. Apply your function from 4.A and make a plot similar to 1.C with the four linkage types (single, complete, ward, centroid), and use 2 clusters for all runs. Compare the results to problem 1, specifically, are there any datasets that are now correctly clustered that k-means could not handle?

C. There are essentially two main ways to find the cut-off point for breaking the diagram: specifying the number of clusters and specifying a maximum distance. The latter is challenging to optimize for without knowing and/or directly visualizing the dendrogram, however, sometimes simple heuristics can work well. The main idea is that since the merging of big clusters usually happens when distances increase, we can assume that a large distance change between clusters means that they should stay distinct. Modify the function from part 1.A to calculate a cut-off distance before classification. Specifically, estimate the cut-off distance as the maximum rate of change of the distance between successive cluster merges (you can use the *scipy.hierarchy.linkage* function to calculate the linkage matrix with distances). Apply this technique to all the datasets and make a plot similar to part 4.B.