BVA 501E Assignment 3

Due 15 January 2024

Submission Guidelines:

- Please submit a single python code file (your submission should have .py extension). You can use the template given to you with this assignment.
- You can work on your assignment and submit it through Ninova until the midnight of the due date.
- Any questions regarding the assignment and late submissions can be directed to mergun@itu.edu.tr.
- In case you cannot submit until midnight, you can submit you homework through email between 00:00 - 00:15 on January 16, 2024. Any email submissions before or after this period will be ignored.
- Please put "[BVA 501E] Assignment 3 Submission" (without quotes) to the subject of your email.
- This assignment is meant to be completed **individually**. Please **do not** share your codes with anyone for any reason.
- Enter your full name and student number into the relevant functions comes with the code template as directed.
- **All codes you submit must be in the designated function**. Writing code outside of the function scope will create errors when your file is imported.

Your task:

For this assignment you will be coding a function that implements a more advanced K-Means clustering algorithm. Your function from assignment 2 selects random starting points and creates a single cluster assignment. However, in some cases, if the starting points are close to each other, K-Means algorithm may yield to a sub-optimal partitioning. To alleviate this problem, modern implementations run K-Means with several times with different starting points and chooses the best output among them. One way to identify the best result is to choose the result with the lowest distortion/inertia value. Average inertia of a clustering result is calculated as follows:

$$Inertia = \sum_{k=0}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$$

where K is the number of clusters, N is the number of instances, x_i are individual instances, C_k is the set of points belong to cluster k and μ_k are the final cluster centers resulted from partitioning. In words, you need to calculate the distance between each instance and its assigned cluster center and then sum all distances up. As mentioned before, the result with the lowest inertia is preferred. A high level psuedocode for the K-Means algorithm is given below:

1. **Pre-step 1:** An appropriate k value is chosen. This value k specifies the number of groups to be identified in the data-set. This value is **specified by the user** and will be given as a **parameter**.

2. **Pre-step 2:** An appropriate n_init value is chosen. This value n_init specifies the number of times K-Means to be run with different starting points. This value is **specified by the user** and will be given as a **parameter**.

3. **Repeat** as many as n_init:

    a- **Basic K-Means:** Find the clustering result with basic K-Means by using the given k. Please note that each time you run basic k-means, you need to start with a different set of starting points.

b- **Calculate Inertia:** Calculate inertia of the clustering result. If inertia is lower than the previous iteration, update the best result.

4. **Return final clusters:** After the iterations are finished we will have the best partition over our data among the ones we tried. Return that result.

You can and will build upon your code from the Assignment 2. So all requirements for the Assignment 2 is valid for this assignment as well.

Additional Files:

1. **assignment3_template.py**: Write your code using this template. **DO NOT** change the function names or add new parameters to the functions. You may add extra functions if needed.

Inputs to your function:

1. **df_data**: A pandas dataframe that contains whole data. Columns correspond to individual features and rows correspond to data instances. **Required.**
2. **n_clusters**: An integer parameter that specifies the number of clusters. Corresponds to the k value in K-Means. **Required.**
3. **n_iter**: An integer parameter that specifies the number of iterations. Assignment and update steps need to be run as many as this parameter. Default value is 100. **Optional.**
4. **n_init**: An integer parameter that specifies the number of initiations. At each initiation a different set of starting points is chosen. Default value is 10. **Optional.**
5. **random_seed**: An integer value that needs to be used as the random seed if specified. **Optional.**

Outputs of your function:

1. If the number of instances is not enough (i.e. less than n_clusters) then throw an exception

2. If the number of instances is enough, continue with the clustering and return an updated pandas dataframe with an additional column named "Cluster". The column cluster should have the final cluster number (start from 0 and move upwards for enumerating the clusters)

For testing your function: Use **scikit-learn** library's **make_blobs** function to create artificial clustering data. This function will give you corresponding cluster labels as well. Compare the results of your function with the given cluster labels.