**Q1. Clustering**

The file specs/question 1.csv contains coordinates of 2-dimensional points.

Write a Python script that:

• Using all the attributes, performs the k-means algorithm for three clusters. If using sklearn, set a fixed random state of 0.

• Save the input data with an extra column that contains the labels generated by KMeans into a file called output/question 1.csv. The new column should be called cluster.

• Plots the clustering results into output/question 1.pdf. Make sure that clusters are marked with different colors.

Discuss the obtained clustering results in your report.

Ans:

* KMeans Clustering:It is a simple unsupervised learning algorithm that is used to solve clustering problems. It follows a simple procedure of classifying a given data set into a number of clusters, defined by the letter "k," which is fixed beforehand.
* Read the csv file ***question\_1.csv*** using pandas.read\_csv() and stored it into a data frame.
* We can apply kmeans clustering on our data using a inbuilt package inside ***sklearn.cluster*** called as ***KMeans***. While creating clusters we gave the cluster parameter ***n\_clusters*** as value equals ***3.*** This parameter is nothing but the k value. Specifying its value as 3 implies that we want to create 3 clusters out of our data-set.
* Saved the dataframe by adding a new column ***cluster*** in it.
* To plot the clustering result we are using scatter plot. Scatter plot is more suitable and easy to understand the formation of clusters. We have the color label assigned to each data point through which we can see which point belongs to which cluster.
* Looking at the result it can be seen that the data was correctly classified into two cluster but the third cluster contains just one data point that is very far away from rest of the points. According to me data point inside the the third cluster is nothing but a noise. We can remove that data point for better cluster formation.

**Q2.**

The file specs/question 2.csv contains data related to nutritional content of

several cereal brands.

• Discard the columns NAME, MANUF, TYPE, and RATING.

• Run the k-means algorithm using 5 clusters as target, 5 maximum runs,

and 100 maximum optimization steps. Keep the random state to 0. Save

the cluster labels in a new column called config1.

• Run k-means again, but this time use 100 maximum runs and 100 maximum optimization steps. Again, use a random state of 0. Save the cluster

labels in a new column called config2.

• Are the clustering results obtained with the first configuration different from the results obtained with the second configuration? Explain your

answer in your report.

• Run the clustering algorithm again, but this time use only 3 clusters. Save the generated cluster labels in a new column called config3.

• Which clustering solution is better? Discuss it in your report.

• Save the input data with the newly generated columns into a file called

output/question 2.csv

Ans

* Read the csv file ***question\_2.csv*** using pandas.read\_csv() and stored it into a data frame.
* To Discard the columns NAME, MANUF, TYPE, and RATING we can use dataframe.drop([“column Names”]).
* **max\_iter:** It is maximum number of iterations of the k-means algorithm for a single run.

**- n\_init:** Number of time the k-means algorithm will be run with different centroid seeds. That is for every value of n\_init, a new set of centroid would be selected and that would run max\_iter times.

**- For eg:** The default values are n\_init=10 and max\_iter=300. This means the initial centroids will be chosen 10 times, and each run will use up to 300 iterations. The best out of those 10 runs will be the final result.

- After running the kmeans algorithm 5 times and keeping n\_init=100 and max\_itr=5 we classified the data into five clusters. Each cluster is marked by different color and color code(integer value).

- We saved cluster labels in a new column called ***config1***

* Again we changed the configuration as:

n\_init=100

max\_iter=100

We saved the new cluster label in new column called ***config2***

* The results from both the configuration are same. The reason could be that, the convergence point would have been reached in 5 iteration and after that no change in cluster is being observed.
* Ran the algorithm again but this time with **n\_cluster=3** and saved the new labels in a new column ***config3.***
* Looking at the data it seems cluster with k=3 is better than k=5. It is observed from the data that the columns ***Sodium*** and ***Potass*** are two major factors that are affecting the cluster formation. When we plot a cluster between sodium and potass with k=3 is the inter-cluster distance is more clearly visible.
* Saved the input data with newly created columns into a file called **question\_2.csv**

**Q3. Question 3**

The file specs/question 3.csv contains coordinates of 2-dimensional points.

Write a Python script to perform the following tasks.

• Discard the ID column, the use the X and Y coordinates to run the k-means algorithm to detect 7 clusters. Use 5 maximum runs, and 100 maximum

optimization steps. Keep a random state of 0. Save the cluster labels into a new column called kmeans. Discuss the cluster results in your report.

• Plot the generated clusters in a file called ./output/question 3 1.pdf.

• Normalize the X and Y columns in a range between 0 and 1, then use the DBSCAN algorithm to cluster the points again. Use a value of 0.04 for

epsilon, and use 4 minimum points for neighborhood evaluation. Save the generated plot in a file called ./output/question 3 2.pdf, and save the

cluster labels into a new column called dbscan1.

• Execute DBSCAN again, but this time use a value of 0.08 for epsilon. Plot the generated clusters in a file called ./output/question 3 3.pdf,

and save the cluster labels into a new column called dbscan2.

• Save the data with the cluster labels in a file called ./output/question 3.csv

• Discuss the different clustering solutions in your report. Which solution

is the best? What is the reason behind the differennces in the results?

**Ans:**

* Read the csv file question\_3.csv using pandas.read\_csv() and stored it into a data frame.
* To Discard the column ID we can use dataframe.drop([“column Name”]). Now again we apply kmeans alorithm and set n\_cluster=7, max\_iter= 5 and n\_init = 100. Saved the cluster label to a new column called kmeans.

- There are lot of noise that is being considered as a part of cluster and hence the cluster boundaries are not clearly defined.

* Saved the generated cluster into the file ***question\_3\_1.pdf.***
* Now we have to normalize the X,Y value in a range of 0-1. To do that we have used ***min\_max\_scaler*** from ***sklearn.preprocessing.***
* **DBSCAN:** The main concept of DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density[1].

-**Density at a point P:** Number of points within a circle of Radius Eps (ϵ) from point P[1].

-**Dense Region:** For each point in the cluster, the circle with radius ϵ contains at least minimum number of points (min\_samples)[1].

* Taking epsilon=0.04 and min\_samples= 4. The plot is better than kmeans but it miss-classifies data points as noise. As the neighbourhood(epsilon) value is small, the data points which should actually form a cluster doesn’t lie in the radius of point P hence couldn’t form a cluster.
* Taking epsilon=0.08 and min\_samples= 4. The plot classifies the data points accurately into 5 clusters and remaining noisy data are marked as -1. The data points are in the neighbourhood of point P hence are able to form a better cluster.
* Kmeans is distance based clustering hence it could not classify the arbitary shaped data points properly.
* DBSCAN is better than kmeans if there are lot of noise present in our data. In DBSCAN we can specify the radius and minimum neighbour to look before considering a data point into the cluster. However it is important to choose correct value of epsilon and min\_samples to have a good classification.

**References:**

[1] Saptashwa, “DBSCAN Algorithm: Complete Guide and Application with Python Scikit-Learn”, 9, June 2019 [Online]. Available: <https://towardsdatascience.com/dbscan-algorithm-complete-guide-and-application-with-python-scikit-learn-d690cbae4c5d>. [Accessed Nov. 2, 2019]