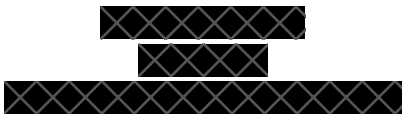
COMP40370 Practical 5



November 12th 2020

Question 1

Question 1.1

Read in the CSV file and then Run $KMeans$ on it

```
q1_df = pd.read_csv("./specs/question_1.csv")
kmeans = KMeans(n_clusters=3, random_state=0).fit(q1_df.values)
```

Question 1.2

```
q1_df["cluster"] = kmeans.labels_
```

Question 1.3

If we have a look at ***Figure 1*** We can see the Clusters that were generated. This seems like a generally good fit for when we ask for 3 clusters with KMeans. I personally found this plot a bit useless so I took the liberty in plotting the cluster centers (***Figure 2***) and the lines from the Cluster Centers (***Figure 3***). If We look at the Second Figure we can see that the cluster centers are in where I myself might put them if I were to guess at some centers. If we Look at Figure 3 then we get a very clear picture of what is going on. Two Clusters deal with all the points where as the Third Cluster deals with the one outlier.

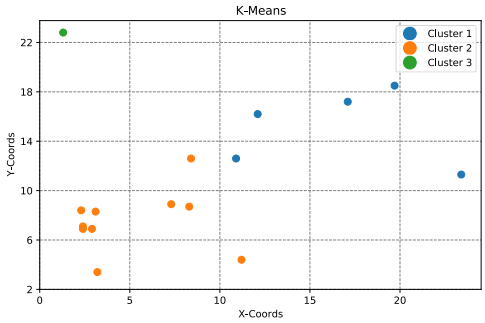


Figure 1: Question1.3 PDF

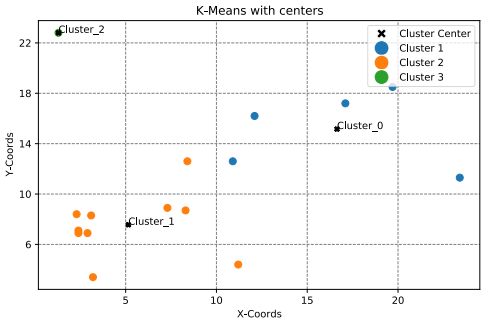


Figure 2: Question1.3 Complimentary PDF

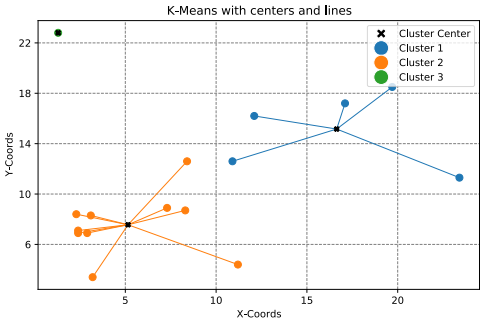


Figure 3: Question1.3 Complimentary PDF

Question 2

Question 2.1

```
discard = ["NAME", "MANUF", "TYPE", "RATING"]
q2_df = q2_df[[c for c in q2_df.columns if c not in discard]]
```

Question 2.2

```
clusters, runs, iterations = 5, 5, 100
kmeans1 = KMeans(n_clusters=clusters, n_init=runs, max_iter=iterations, random_state=0)
kmeans1 = kmeans1.fit(q2_df[columns].values)
q2_df["config1"] = kmeans1.labels_
```

Question 2.3

```
clusters, runs, iterations = 5, 100, 100
kmeans2 = KMeans(n_clusters=clusters, n_init=runs, max_iter=iterations, random_state=0)
kmeans2 = kmeans2.fit(q2_df[columns].values)
q2_df["config2"] = kmeans2.labels_
```

Question 2.4

To compare The Clustering results we will look at A plot of the First Configuration ***Figure 4*** and a plot of the Second Configuration ***Figure 5*** **(Note to get this Plots we had to run a PCA to reduce the dimensions down to 2 Dimensions. it is a good 2-D Analogue)**

We can see that Both the KMeans differ in the 5 Clusters they came up with. However there are still similarities, the left most cluster is Red in the Second Configuration and this Cluster is a Sub Cluster of the First Configuration. Also, the Red Cluster in the First Configuration is the same as the Orange Configuration in the Second Configuration. Besides these two Similarities there is then a tri-cluster in the right side on both Configurations but they differ slightly in both.

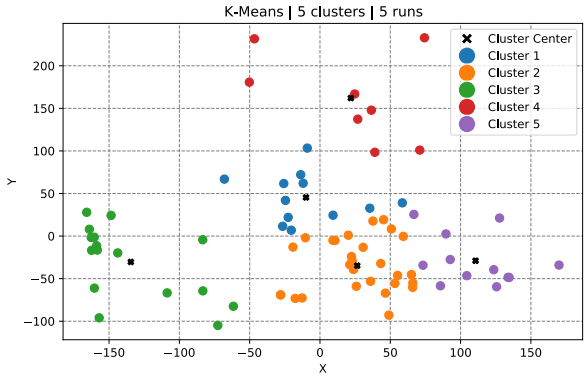


Figure 4: Question2.4 1st Config PDF

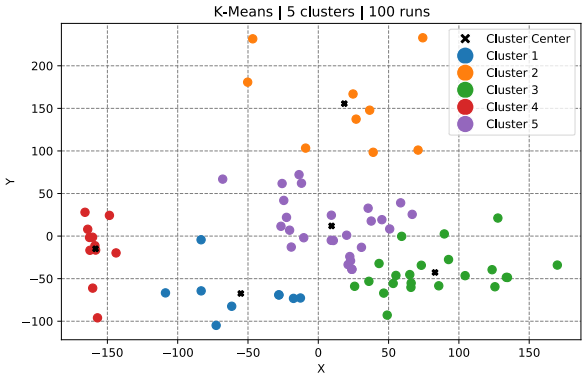


Figure 5: Question2.4 2nd Config PDF

Question 2.5

```
clusters, runs, iterations = 3, 100, 100
kmeans3 = KMeans(n_clusters=clusters, n_init=runs, max_iter=iterations, random_state=0)
kmeans3 = kmeans3.fit(q2_df[columns].values)
q2_df["config3"] = kmeans3.labels_
```

Question 2.6

(Note to get this Plots we had to run a PCA to reduce the dimensions down to 2 Dimensions. it is a good 2-D Analogue)

We can see the Kmeans Carried out on 3 Clusters instead of 7 in ***Figure 6***. Since we Don't have actual Labels and this is meant to be an Unsupervised task we cannot say exactly what the Accuracy of the Clusters are. However, we do have a set of measures and I will choose the Silhouette Score ($S = \frac{b-a}{max(a,b)}$) as the score to measure the performance of the Clustering Methods.

Figure	Silhouette Score
4	0.3218013975918788
5	0.3601346535452705
6	0.46430924739274937

For Silhouette Scores "The best value is 1" So we can Conclude the 3 Cluster K-Means is the best. Which matches up nicely with our plots. The First and Second Configurations in **Figures 4 & 5** do give use coherent clusters they are still a bit all over the Place and Messy while the Kmeans with 3 Clusters has Three Clearly defined regions with high intra-Cluster similarity and Low Inter Cluster similarity.

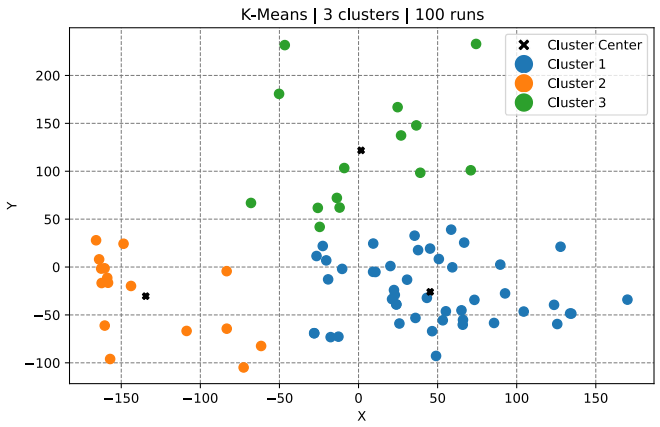


Figure 6: Question2.6 3 Cluster PDF

Question 2.7

```
q2_df.to_csv("output/question_2.csv")
```

Question 3

Question 3.1 & 3.2

```
q3_df = pd.read_csv("./specs/question_3.csv")[["x", "y"]]
q3_km1 = KMeans(n_clusters=7, n_init=5, max_iter=100, random_state=0)
q3_km1 = q3_km1.fit(q3_df[columns].values)
q3_df["kmeans"] = q3_km1.labels_
```

We can see that when we Run Kmeans with 7 Clusters (**Figure 7**) we get some clearly defined clusters but due to the nature of the data it is quite difficult to say that this cluster is good. The Silhouette Score is 0.5423344622046449 so we know that its at least somewhat of a good fit But we can see areas where it is lacking. For example even though there are clearly defined Clusters in the Plot there are a great many outliers. Due of the nature of Kmeans these outliers Interfere with our Overall Score

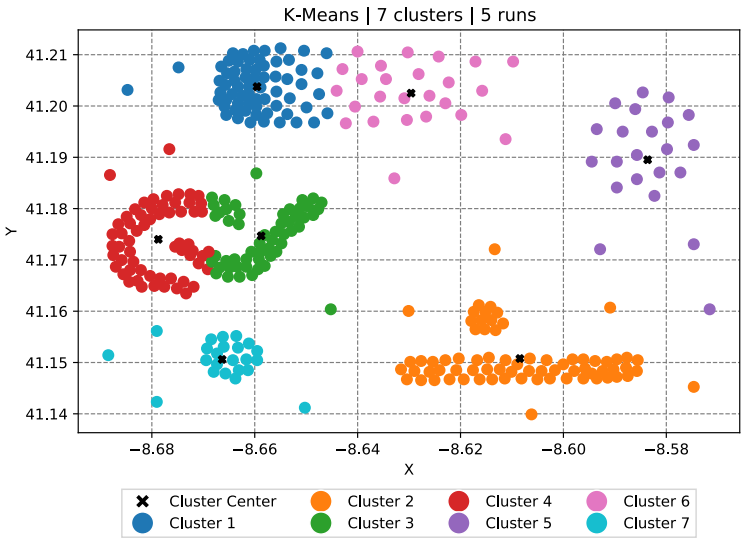


Figure 7: Question3 Plot 1 PDF

Question 3.3

```
scaler = MinMaxScaler()
scaler.fit(q3_df[columns].values)
trans_data = scaler.transform(q3_df[columns].values)
dbscan_cluster1 = DBSCAN(eps=0.04, min_samples=4).fit(trans_data)
q3_df["dbscan1"] = dbscan_cluster1.labels_
```

Question 3.4

```
dbscan_cluster2 = DBSCAN(eps=0.08, min_samples=4).fit(trans_data)
q3_df["dbscan2"] = dbscan_cluster2.labels_
```

Question 3.5

```
q3_df.to_csv("output/question_3.csv")
```

Question 3.6

We are going to Discuss the Different Clustering Solutions in relation to (**Figure 7, 8 & 9**) and use their Silhouette Scores as listed below.

Figure	Silhouette Score
7	0.5423344622046449
8	0.32655065512772047
9	0.5272621059483141

I propose we Examine the Figures and then use the Silhouette Scores to see if what we derive from the plots is on the right track. The First plot (**Figure 7**) Would Appear to be the least coherent. and it really highlights the constraints of KMeans. since kmenas functions by distance we end up getting clusters that are only ever circular(or Elliptical) in shape. which is why we see the harsh differences Between the Red and Green. The one advantage this has is that points close together are clustered together but in more abstract Data this doesn't work. Now we switch to the Second plot (**Figure 8**) which uses DBSCAN we can now see the advantages offered, Our clusters can now be represented by polygons. We can see that This first DBSCAN done a good job clustering and dealt with outliers by assigning all to the same cluster. the Second DBSCAN done the same but it chose different Clusters and instead of assigning all the outliers to one of the clusters it just assigned all the outliers to their own cluster. Now that we have looked at the Plots We see that the Silhouette Score of figure 7 = 0.5423344622046449, 8 = 0.32655065512772047, and 9 = 0.5272621059483141. We see the **Figure 8** has the lowest Silhouette Score And its easy to see why. since the outliers are part of a cluster their score is skewed. I think its interesting that the Silhouette Score for The kMeans is the highest. It makes sense due to how its calculated But I Would still say the 2nd DBSCAN done the best at clustering the Data

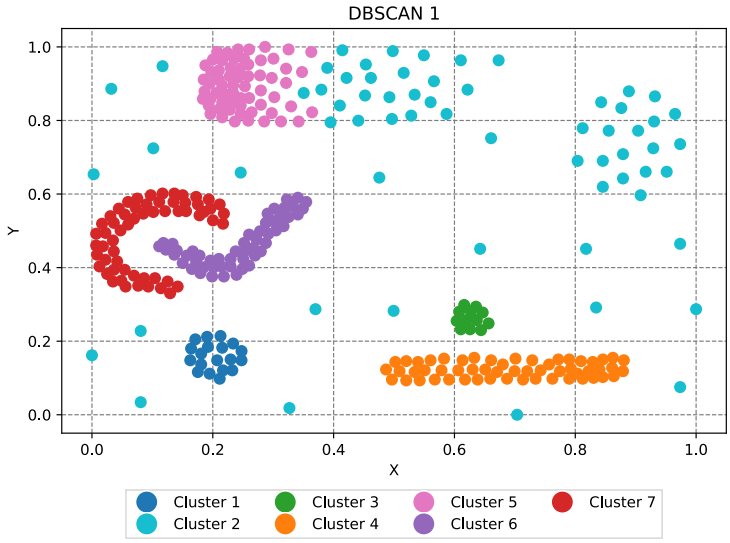


Figure 8: Question3 Plot 2 PDF

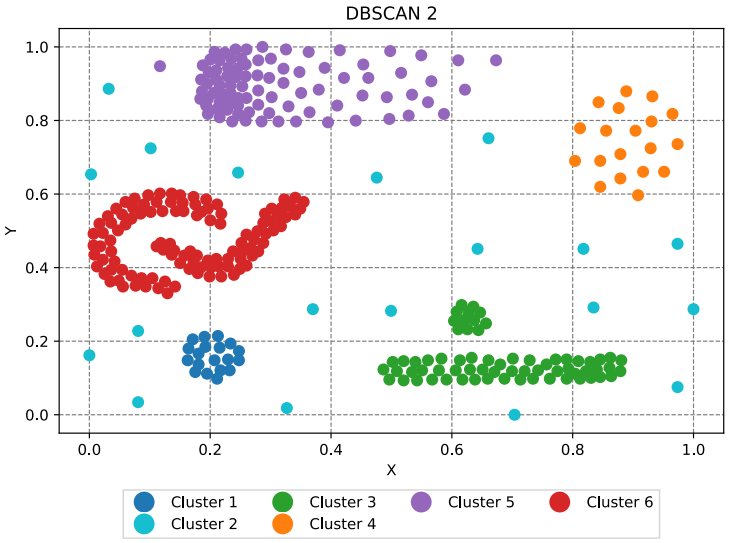


Figure 9: Question3 Plot 3 PDF