

# Assignment-6

Github Link : <https://github.com/AayushSanghai/Machine-Learnin/tree/main/Assignment-6>

## Question 1:

The x coordinate and y coordinate is given in the question. For our ease, the distance matrix is also provided which makes calculations less.

We are asked to calculate and the clustering representation for Hierarchical clustering and also draw dendrograms for Single, Complete and Average links.

### Single Link:

We take the minimum of the points of the clusters that has to be merged and continue the process till only 1 cluster remains. Below are the screenshots"

Question1.pdf

Single Link

	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>5</sub>	P <sub>6</sub>	P <sub>7</sub>
P <sub>1</sub>	0.0						
P <sub>2</sub>	0.2357	0.0					
P <sub>3</sub>	0.2218	0.1483	0.0				
P <sub>4</sub>	0.3688	0.2042	0.1513	0.0			
P <sub>5</sub>	0.3921	0.1388	0.2843	0.2932	0.0		
P <sub>6</sub>	0.2317	0.2540	0.1100	0.2216	0.3921	0.0	

Now in the distance matrix, the min is:  
Pair {P<sub>3</sub>, P<sub>6</sub>} = 0.11

Updating distance matrix MIN [ dist (P<sub>3</sub>, P<sub>6</sub>), P<sub>1</sub> ]

$$\Rightarrow \text{MIN} (\text{dist} (P_3, P_1), (P_6, P_1))$$
$$= \min \{ 0.2218, 0.2347 \}$$
$$= 0.2218$$
$$\Rightarrow \text{MIN} (\text{dist} (P_3, P_6), P_2)$$
$$= \min \{ 0.1483, 0.2540 \}$$

Question1.pdf

Updated distance matrix for P<sub>3</sub>, P<sub>6</sub>.

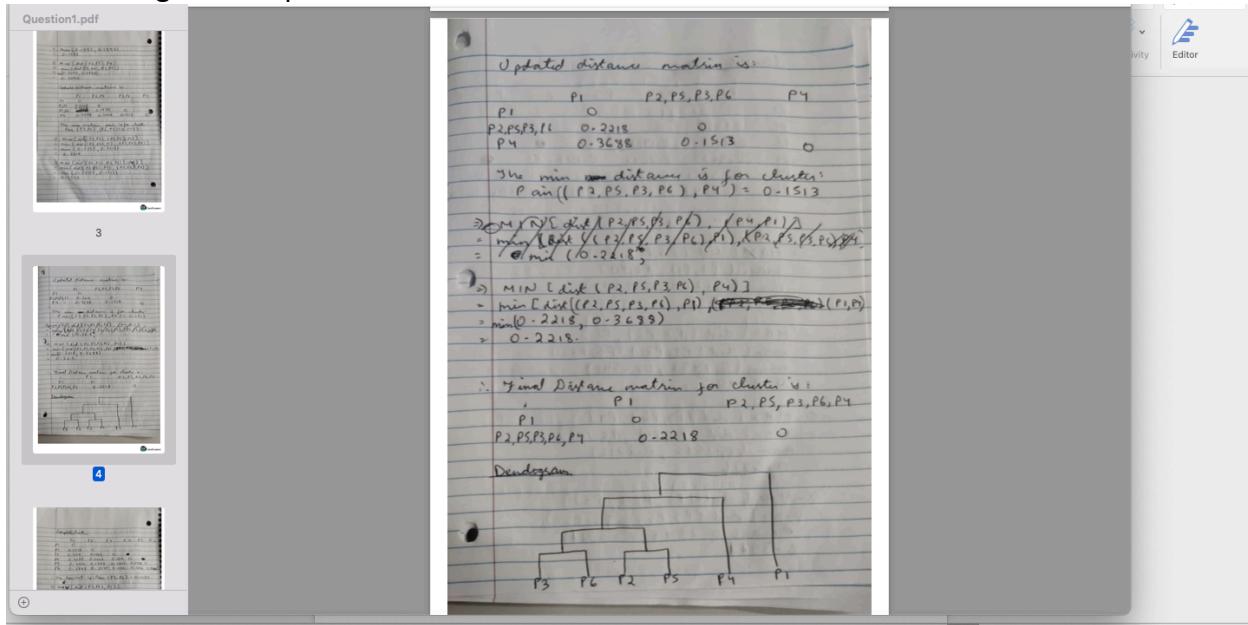
	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub> , P <sub>6</sub>	P <sub>4</sub>	P <sub>5</sub>
P <sub>1</sub>	0				
P <sub>2</sub>	0.2357	0			
P <sub>3</sub> , P <sub>6</sub>	0.2218	0.1483	0		
P <sub>4</sub>	0.3688	0.2042	0.1513	0	
P <sub>5</sub>	0.3921	0.1388	0.2843	0.2932	0

The min dist in distance matrix is:  
Pair {P<sub>2</sub>, P<sub>5</sub>} = 0.1388

$$\Rightarrow \text{MIN} [\text{dist} (P_2, P_5), P_1]$$
$$= \min (\text{dist} (P_2, P_1), (P_5, P_1))$$
$$= \min (0.2357, 0.3421)$$
$$= 0.2357$$
$$\Rightarrow \text{MIN} [\text{dist} (P_2, P_5), (P_3, P_6)]$$
$$= \min (\text{dist} (P_2, (P_3, P_6)), P_5 (P_3, P_6))$$

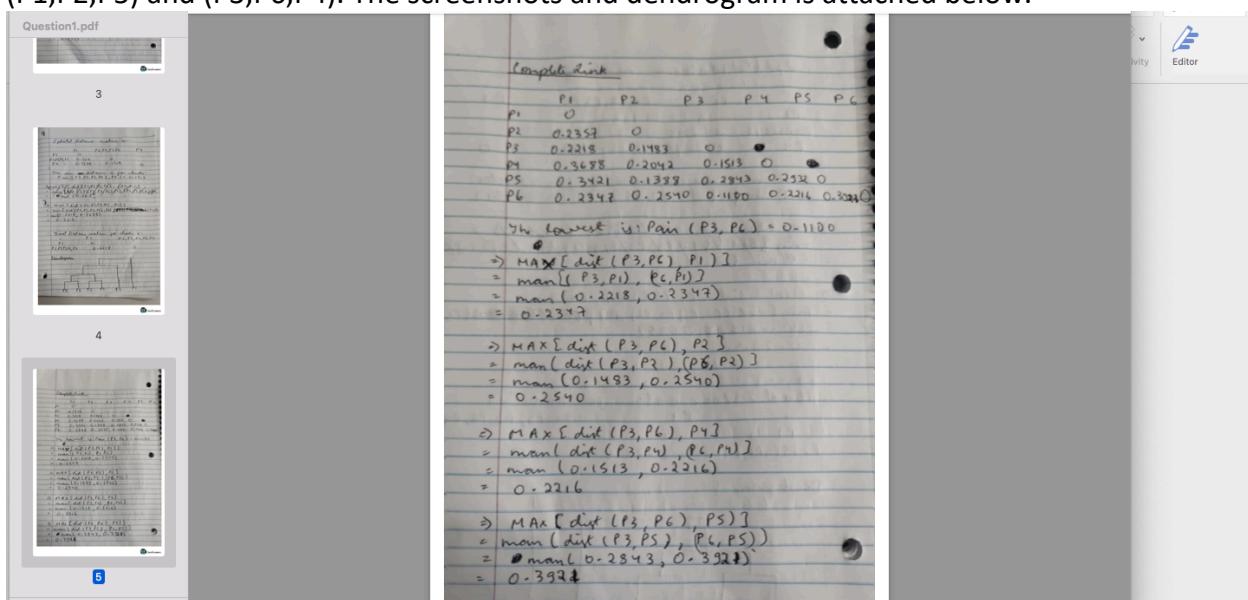
Here, 1<sup>st</sup> cluster is formed for pairs P3, P6, 2<sup>nd</sup> cluster is formed for P2, P5, 3<sup>rd</sup> cluster is formed between (P3,P6) and (P2,P5) and finally the final cluster is formed between P2,P3,P5,P5 and P4.

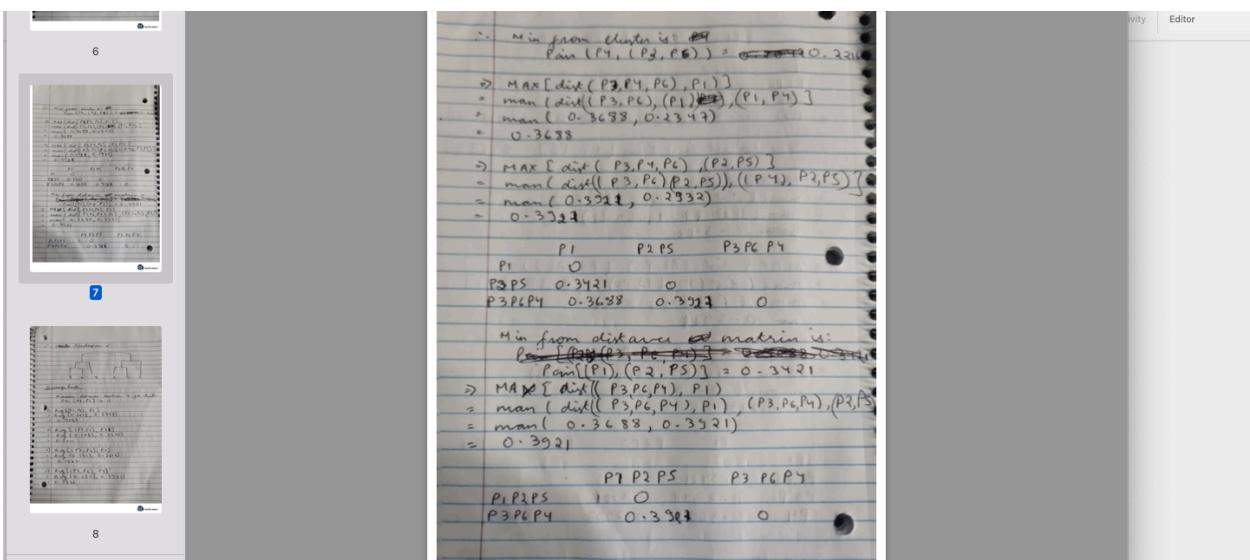
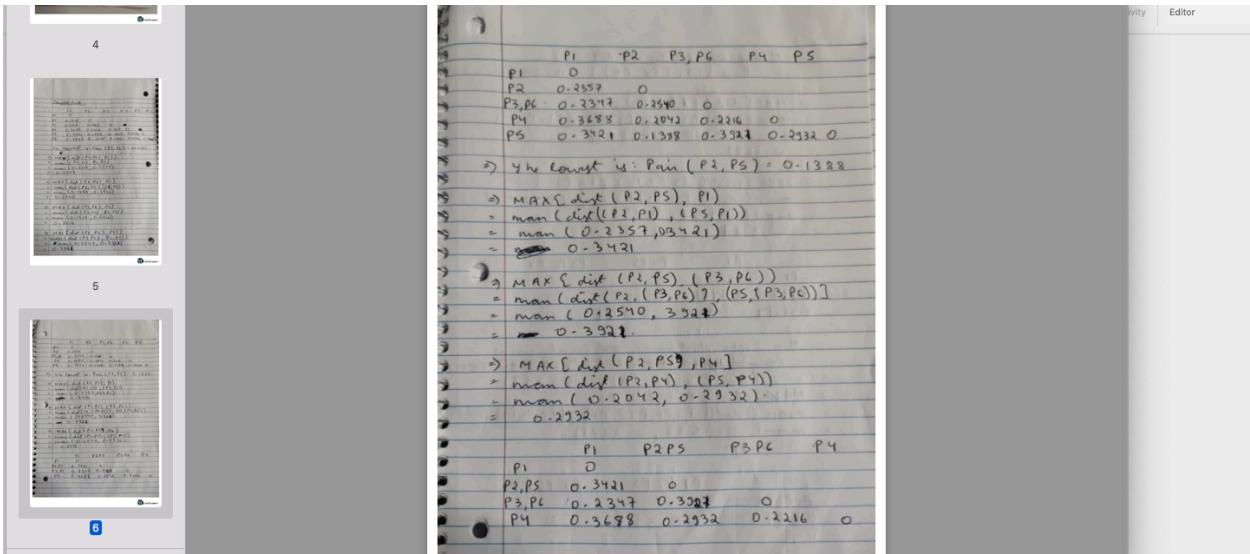
The dendrogram is represented as follows:



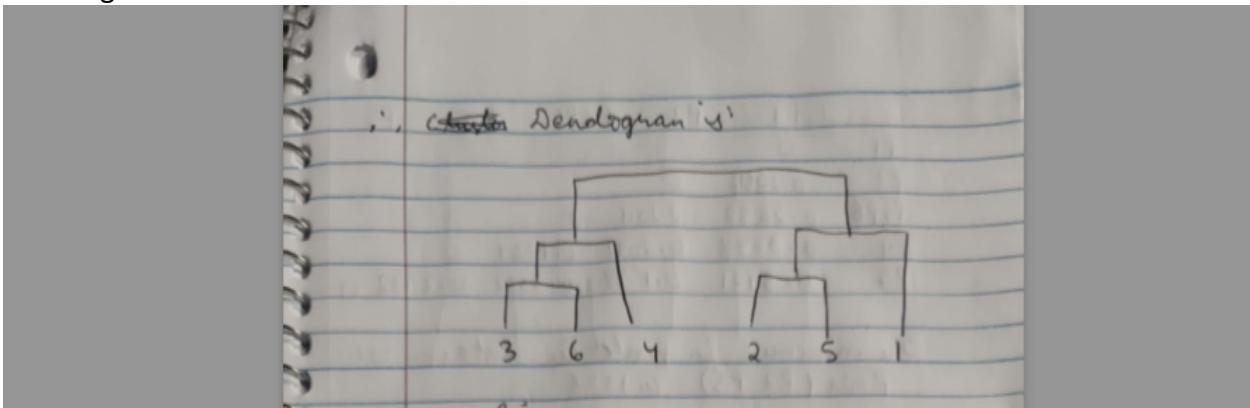
### Complete Link:

Here, we select the minimum point and then begin creating clusters. We take the maximum of the two minimum data points distances and then start creating clusters depending on the total no. of clusters we need. Here, the clusters formed for this assignment are P3 and P6, then P2,P5, then between P4 and (P3 and P5). Then between P1 and (P2,P5). Finally it ends with (P1,P2,P5) and (P3,P6,P4). The screenshots and dendrogram is attached below.





Dendrogram:



### Average Link:

Unlike single link and complete link, the average link takes the average between the distances of two minimum data points to form clusters. For this assignment, the clusters are formed as : P3 and P6, then between P2 and P5, then, (P3,P6) and P4. Then between (P5,P2) and (P3,P4,P6). Below are screenshots of cluster formation and the distance matrix:

Average Link

Minimum distance matrix is for cluster  
Pair (P3, P6) = 0.11

$$\Rightarrow \text{Avg} \{ (P3, P6), P1 \}$$

$$= \text{Avg} \{ 0.2218, 0.2317 \}$$

$$= 0.2282$$

$$\Rightarrow \text{Avg} \{ (P3, P6), P2 \}$$

$$= \text{Avg} \{ 0.1483, 0.2540 \}$$

$$= 0.2011$$

$$\Rightarrow \text{Avg} \{ (P3, P6), P4 \}$$

$$= \text{Avg} \{ 0.1513, 0.2216 \}$$

$$= 0.1864$$

$$\Rightarrow \text{Avg} \{ (P3, P6), P5 \}$$

$$= \text{Avg} \{ 0.2873, 0.3921 \}$$

$$= 0.3382$$

	P1	P2	P3 P6	P4	P5
P1	0				
P2	0.2357	0			
P3 P6	0.2282	0.2011	0		
P4	0.3688	0.2042	0.1864	0	
P5	0.3421	0.1388	0.3382	0.2932	0

- The minimum in matrix is distance is:  
Pair (P3, P5) = 0.1388.

$$\Rightarrow \text{Avg} \{ \text{dist} (P2, P5), P1 \}$$

$$= \text{Avg} \{ \text{dist} (P2, P1), (P5, P1) \}$$

$$= \text{avg} \{ 0.2357, 0.3921 \}$$

$$= 0.2989$$

$$\Rightarrow \text{Avg} \{ \text{dist} (P2, P5), (P3, P6) \}$$

$$= \text{avg} \{ \text{dist} [(P3, P6), P2], [(P3, P6), P5] \}$$

$$= \text{avg} \{ 0.2011, 0.3382 \}$$

$$= 0.2946$$

$$\Rightarrow \text{Avg} \{ \text{dist} (P2, P5), P4 \}$$

$$= \text{avg} \{ \text{dist} (P2, P4), (P5, P4) \}$$

$$= \text{avg} \{ 0.2042, 0.2932 \}$$

$$= 0.2487$$

	P1	P2 P3	P3 P6	P4
P1	0			
P2 P3	0.2889	0		
P3 P6	0.2282	0.2946	0	
P4	0.3688	0.2487	0.1864	0



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [105]: df.shape      #Getting the no. of attributes in the dataset
Out[105]: (8950, 18)

In [106]: df.isnull().sum()      #Checking for null values
Out[106]:
CUST_ID                 0
BALANCE                  0
BALANCE_FREQUENCY         0
PURCHASES                 0
ONEOFF_PURCHASES          0
INSTALLMENTS_PURCHASES    0
CASH_ADVANCE               0
PURCHASES_FREQUENCY        0
ONEOFF_PURCHASES_FREQUENCY 0
PURCHASES_INSTALLMENTS_FREQUENCY 0
CASH_ADVANCE_FREQUENCY     0
CASH_ADVANCE_TRX           0
PURCHASES_TRX                0
CREDIT_LIMIT                  1
PAYMENTS                   0
MINIMUM_PAYMENTS            313
PRC_FULL_PAYMENT             0
TENURE                      0
dtype: int64

In [107]: df = df.drop("CUST_ID", axis = 1)      #Dropping the categorical column

In [108]: df = df.fillna(df.mean())      #Filling the missing values

In [109]: df.head()      #Checking again after dropping customer ID
```

- b) Then, I have scaled the data using Standard Scaler and then normalized the raw data points using the normalize() function

(b)

```
In [111]: scaler = StandardScaler()
X_Scale = scaler.fit_transform(df)           # Scaling the dataset
print(X_Scale)

[[ -0.73198937 -0.24943448 -0.42489974 ... -0.31096755 -0.52555097
  0.36067954]
 [ 0.78696085  0.13432467 -0.46955188 ...  0.08931021  0.2342269
  0.36067954]
 [ 0.44713513  0.51808382 -0.10766823 ... -0.10166318 -0.52555097
  0.36067954]
 ...
 [-0.7403981 -0.18547673 -0.40196519 ... -0.33546549  0.32919999
 -4.12276757]
 [-0.74517423 -0.18547673 -0.46955188 ... -0.34690648  0.32919999
 -4.12276757]
 [-0.57257511 -0.88903307  0.04214581 ... -0.33294642 -0.52555097
 -4.12276757]]]

In [112]: normal = preprocessing.Normalizer().fit(X_Scale)          #Normalizing the raw input data
X_Norm = normal.transform(X_Scale)
print(X_Norm)

[[-0.31193826 -0.10629684 -0.1810716 ... -0.13251924 -0.22396426
  0.15370408]
 [ 0.21992533  0.03753859 -0.13122171 ...  0.02495877  0.06545742
  0.10079608]
 [ 0.12668203  0.14678317 -0.03050449 ... -0.02880315 -0.14889876
  0.10218749]
 ...
 [-0.1569743 -0.03932355 -0.085222 ... -0.07112317  0.0697948
 -0.87408185]
 [-0.15431961 -0.03841074 -0.09724043 ... -0.07184155  0.06817468
 -0.829537851]]]

In [113]: X_normalized = pd.DataFrame(X_Norm)           #Converting from array to panda
X_normalized.head()

Out[113]:
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	-0.311938	-0.106297	-0.181072	-0.152108	-0.148760	-0.198921	-0.343687	-0.289212	-0.301422	-0.287801	-0.202878	-0.217905	-0.409290	-0.225425	-1.32
1	0.219925	0.037539	-0.131222	-0.099749	-0.127037	0.728166	-0.341434	-0.189660	-0.256265	0.160401	0.030761	-0.165384	0.192448	0.228779	2.49
2	0.126682	0.146783	-0.030504	0.030850	-0.128790	-0.132249	0.359771	0.757440	-0.259802	-0.191339	-0.134880	-0.030888	0.234039	-0.108739	-2.88
3	0.020589	-0.426439	0.097309	0.229034	-0.190618	-0.154587	-0.425253	-0.167447	-0.384524	-0.108570	-0.138184	-0.231288	0.346393	-0.251048	-1.84
4	-0.151595	0.218909	-0.195238	-0.146744	-0.192075	-0.197234	-0.428504	-0.168727	-0.387463	-0.285359	-0.201157	-0.233056	-0.382591	-0.153959	-1.12

c) Now, I have used PCA and reduced the attributes to 2 dimensions.

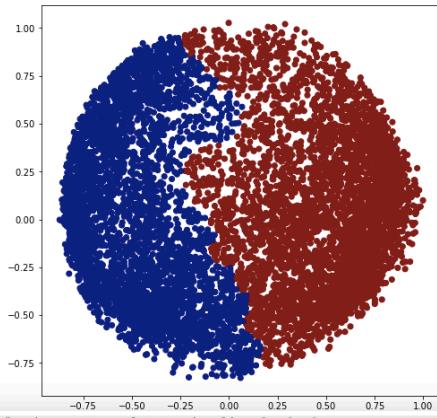
(c)

```
In [114]: pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X_Norm)      #Applying PCA and reducing the no. features to 2
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
principalDf.head()

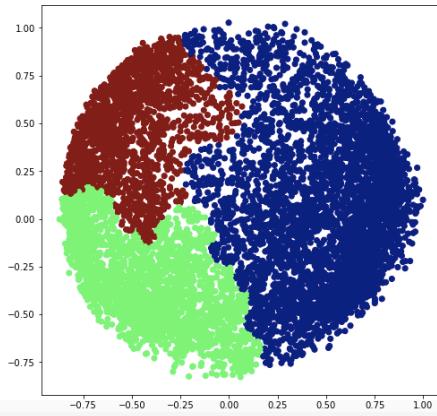
Out[114]:   principal component 1  principal component 2
0           -0.489826          -0.679678
1           -0.518792          0.545011
2            0.330885          0.268978
3           -0.482374         -0.092111
4           -0.563289          -0.481915
```

d) After applying PCA, I have now used scatter plot to visualize the agglomerative clustering using by taking k(no. of clusters) as 2,3,4,5

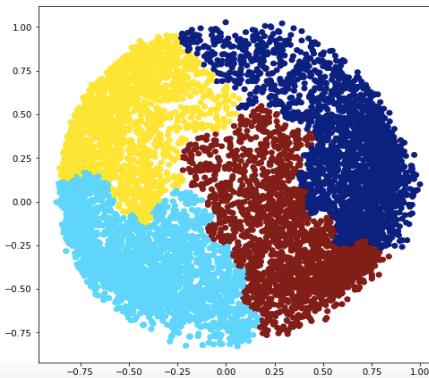
```
In [115]: # Using scatter plot to visualize for k =2
Aglo_Cluster2 = AgglomerativeClustering(n_clusters = 2)
plt.figure(figsize =(8,8))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
c = Aglo_Cluster2.fit_predict(principalDf), cmap ='jet')
plt.show()
```



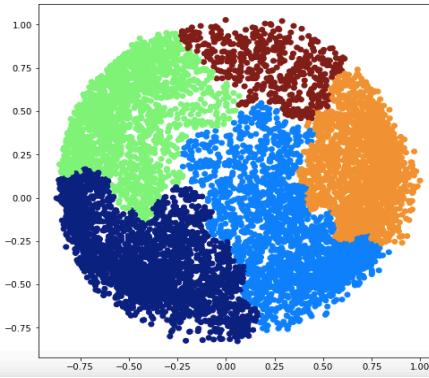
```
In [116]: # Using scatter plot to visualize for k =3
Aglo_Cluster3 = AgglomerativeClustering(n_clusters = 3)
plt.figure(figsize =(8,8))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
c = Aglo_Cluster3.fit_predict(principalDf), cmap ='jet')
plt.show()
```



```
In [18]: # Using scatter plot to visualize for k = 4
Aglo_Cluster4 = AgglomerativeClustering(n_clusters = 4)
plt.figure(figsize =(8,8))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
c = Aglo_Cluster4.fit_predict(principalDf), cmap ='jet')
plt.show()
```



```
In [18]: # Using scatter plot to visualize for k = 5
Aglo_Cluster5 = AgglomerativeClustering(n_clusters = 5)
plt.figure(figsize =(8,8))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
c = Aglo_Cluster5.fit_predict(principalDf), cmap ='jet')
plt.show()
```

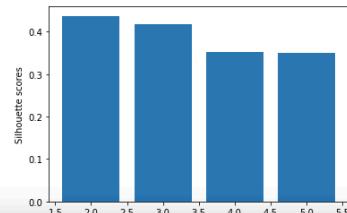


- e) Finally, I have calculated the silhouette score for each clusters and them compared them using a bar plot.

```
print("Silhouette score for 2 clusters is =", Silhouette[0])
Silhouette.append(silhouette_score(principalDf, Aglo_Cluster3.fit_predict(principalDf)))
print("Silhouette score for 3 clusters is =", Silhouette[1])
Silhouette.append(silhouette_score(principalDf, Aglo_Cluster4.fit_predict(principalDf)))
print("Silhouette score for 4 clusters is =", Silhouette[2])
Silhouette.append(silhouette_score(principalDf, Aglo_Cluster5.fit_predict(principalDf)))
print("Silhouette score for 5 clusters is =", Silhouette[3])
```

```
Silhouette score for 2 clusters is = 0.4373242834730189
Silhouette score for 3 clusters is = 0.4167229587664966
Silhouette score for 4 clusters is = 0.35310619213129274
Silhouette score for 5 clusters is = 0.35055082408170385
```

```
In [21]: # Plotting a bar graph to compare the results
plt.bar(k, Silhouette)
plt.xlabel('Number of clusters', fontsize = 10)
plt.ylabel('Silhouette scores', fontsize = 10)
plt.show()
```



Here I can conclude that the best silhouette score is when I use 2 cluster which is 43.7324%.