Assignment-6

Kiran Kumar Kongari

700734361

GitHub Link: https://github.com/KiranKumarKongari/MachineLearning-Assignment-6

2) Use CC_GENERAL.csv given in the folder and apply:
   a) Preprocess the data by removing the categorical column and filling the missing values.
   b) Apply StandardScaler() and normalize() functions to scale and normalize raw input data.
   c) Use PCA with K=2 to reduce the input dimensions to two features.
   d) Apply Agglomerative Clustering with k=2,3,4 and 5 on reduced features and visualize result for each k value using scatter plot.
   e) Evaluate different variations using Silhouette Scores and Visualize results with a bar chart.

Question-2.py

```python
# 2) Use CC_GENERAL.csv given in the folder and apply:
#    a) Preprocess the data by removing the categorical column and filling the missing values.
#    b) Apply StandardScaler() and normalize() functions to scale and normalize raw input data.
#    c) Use PCA with K=2 to reduce the input dimensions to two features.
#    d) Apply Agglomerative Clustering with k=2,3,4 and 5 on reduced features and visualize
#       result for each k value using scatter plot.
#    e) Evaluate different variations using Silhouette Scores and Visualize results with a bar chart.

import math
import warnings
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler, normalize

warnings.filterwarnings('ignore')


# cc_general is a dataframe where we load the csv data.
cc_general = pd.read_csv("C:/Users/Kiran Kumar Kongari/PycharmProjects/ML-Assignment-6/Dataset/CC GENERAL.csv")
print("\nThe Original Dataframe is : \n", cc_general)
```

```
"C:\Users\Kiran Kumar Kongari\PycharmProjects\ML-Assignment-6\venv\Scripts\python.exe

The Original Dataframe is :
      CUST_ID      BALANCE  ...  PRC_FULL_PAYMENT  TENURE
0     C10001    40.900749  ...          0.000000      12
1     C10002  3202.467416  ...          0.222222      12
2     C10003  2495.148862  ...          0.000000      12
3     C10004  1666.670542  ...          0.000000      12
4     C10005   817.714335  ...          0.000000      12
...      ...          ...  ...               ...     ...
8945  C19186    28.493517  ...          0.500000       6
8946  C19187    19.183215  ...          0.000000       6
8947  C19188    23.398673  ...          0.250000       6
8948  C19189    13.457564  ...          0.250000       6
8949  C19190   372.708075  ...          0.000000       6

[8950 rows x 18 columns]
```

```python
# a) Preprocess the data by removing the categorical column and filling the missing values.
# dropping the categorical column i.e., CUST_ID column
customerDf = cc_general.drop(['CUST_ID'], axis='columns')

# Checking the columns having null values and displaying the resultant columns.
columnsWithNullValues = customerDf.isna().any()

# a. Replacing the null values with the mean
customerDf['CREDIT_LIMIT'] = customerDf['CREDIT_LIMIT'].fillna(customerDf['CREDIT_LIMIT'].mean())
customerDf['MINIMUM_PAYMENTS'] = customerDf['MINIMUM_PAYMENTS'].fillna(customerDf['MINIMUM_PAYMENTS'].mean())

# Verifying the dataframe again for null values
f = customerDf[customerDf.isna().any(axis=1)]
print('\nVerifying customer dataframe for null values again : ', f)
```

```
Verifying customer dataframe for null values again :  Empty DataFrame
Columns: [BALANCE, BALANCE_FREQUENCY, PURCHASES, ONEOFF_PURCHASES, INSTALLMENTS_PURCHASES, CASH_ADVANCE, PURCHASES_FREQUENCY, ONEOFF_PURCHASES_FREQUENCY, PURCHASES_INSTALLMENTS_F
Index: []
```

```python
# b) Apply StandardScaler() and normalize() functions to scale and normalize raw input data.
# Performing Scaling
scaler = StandardScaler()
X_Scale = scaler.fit_transform(customerDf)

# Normalizing the data so that the data approximately
X_normalize = normalize(X_Scale)

# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalize)
print("\n The dataframe after performing the Scaling and normalizing is : \n ", X_normalized)
```

```
 The dataframe after performing the Scaling and normalizing is :
              0         1         2   ...            14        15        16
0     -0.311938 -0.106297 -0.181072  ... -1.325192e-01 -0.223964  0.153704
1      0.219925  0.037539 -0.131222  ...  2.495877e-02  0.065457  0.100796
2      0.126682  0.146783 -0.030504  ... -2.880315e-02 -0.148899  0.102187
3      0.020589 -0.426439  0.097309  ...  2.045620e-17 -0.220379  0.151244
4     -0.151595  0.218909 -0.195238  ... -1.123064e-01 -0.222064  0.152400
...         ...       ...       ...  ...           ...       ...       ...
8945 -0.146893  0.103128 -0.066344  ... -6.964046e-02  0.235672 -0.820660
8946 -0.151521  0.105735 -0.067173  ...  9.956102e-18 -0.107259 -0.841413
8947 -0.156974 -0.039324 -0.085222  ... -7.112317e-02  0.069795 -0.874082
8948 -0.154320 -0.038411 -0.097240  ... -7.184155e-02  0.068175 -0.853792
8949 -0.115207 -0.178881  0.008480  ... -6.699181e-02 -0.105746 -0.829538

[8950 rows x 17 columns]
```

```python
# c) Use PCA with K=2 to reduce the input dimensions to two features.
# Applying PCA (k=2)
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X_normalized)
principalDf = pd.DataFrame(data=principalComponents,
                           columns=['principal component 1', 'principal component 2'])
print("\nThe Dataframe after applying PCA : \n", principalDf)
```

```
The Dataframe after applying PCA :
       principal component 1  principal component 2
0                  -0.489825              -0.679678
1                  -0.518791               0.545010
2                   0.330886               0.268981
3                  -0.482374              -0.092113
4                  -0.563289              -0.481914
...                      ...                    ...
8945                0.328718              -0.198545
8946                0.259862              -0.167657
8947                0.188798              -0.248498
8948               -0.313018              -0.171384
8949                0.012929               0.097872

[8950 rows x 2 columns]
```

```python
# d) Apply Agglomerative Clustering with k=2,3,4 and 5 on reduced features and visualize
#    result for each k value using scatter plot.

# With k=2
ac2 = AgglomerativeClustering(n_clusters=2)
# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
            c=ac2.fit_predict(principalDf), cmap='rainbow')
plt.show()
```

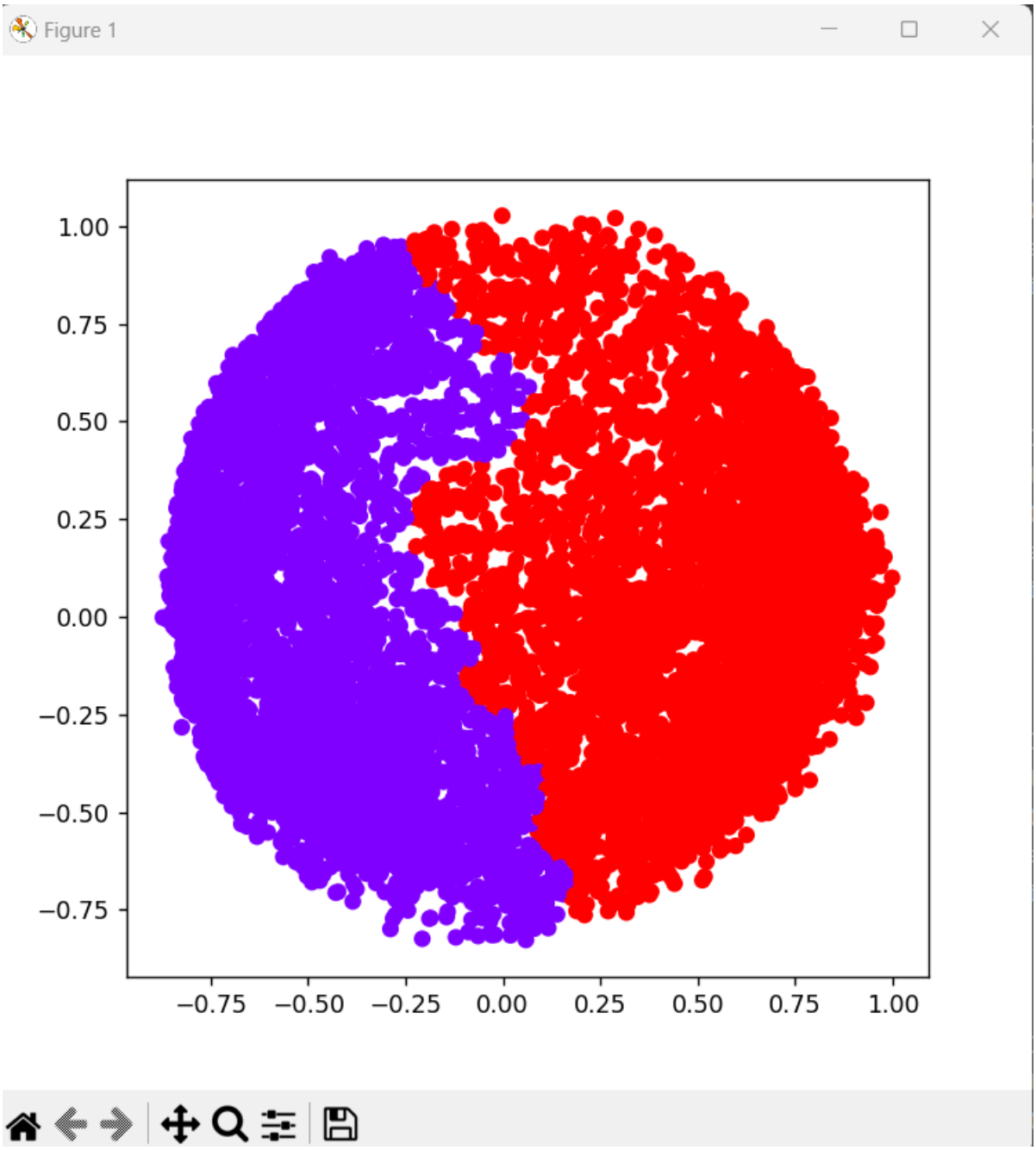
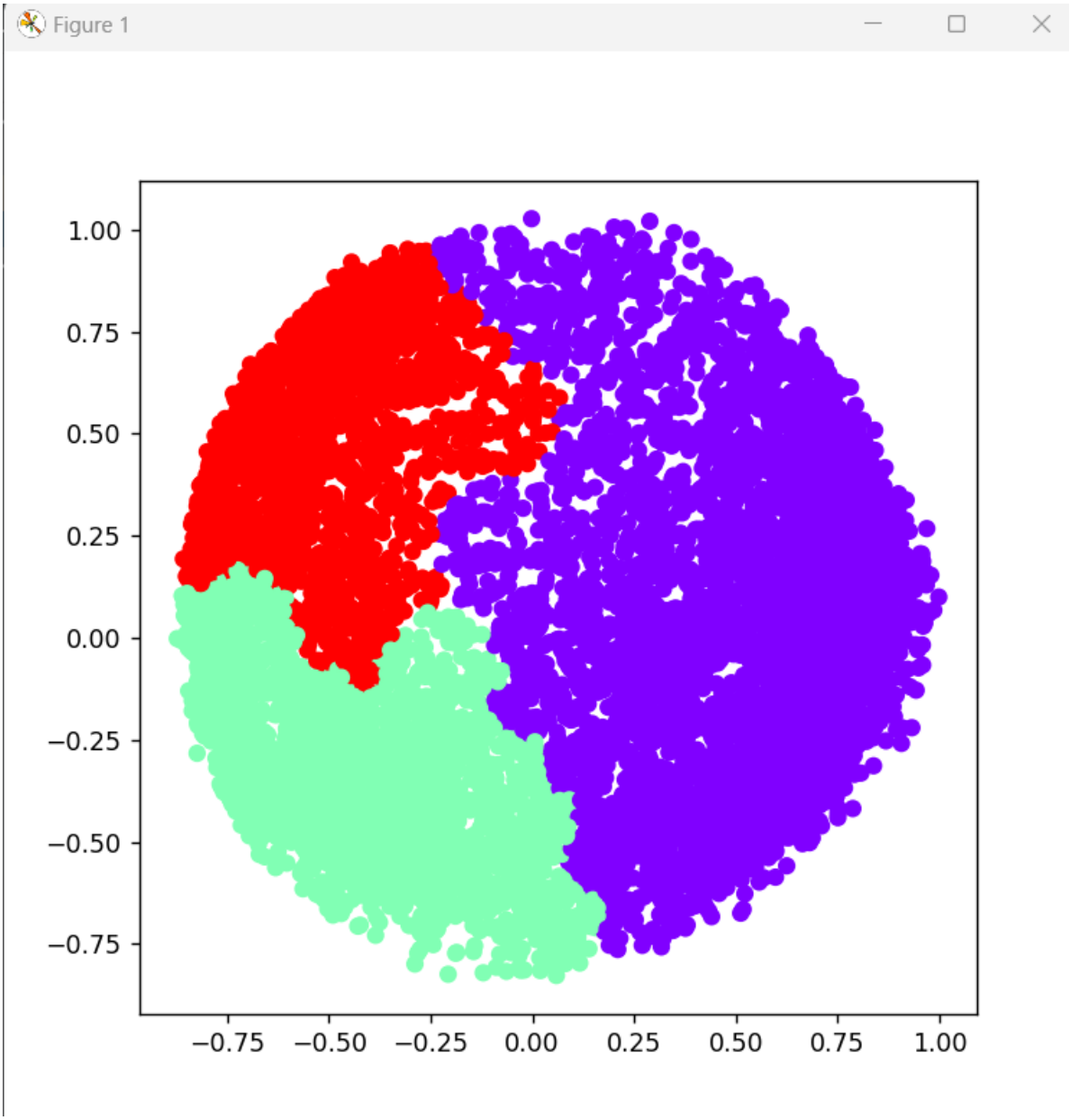
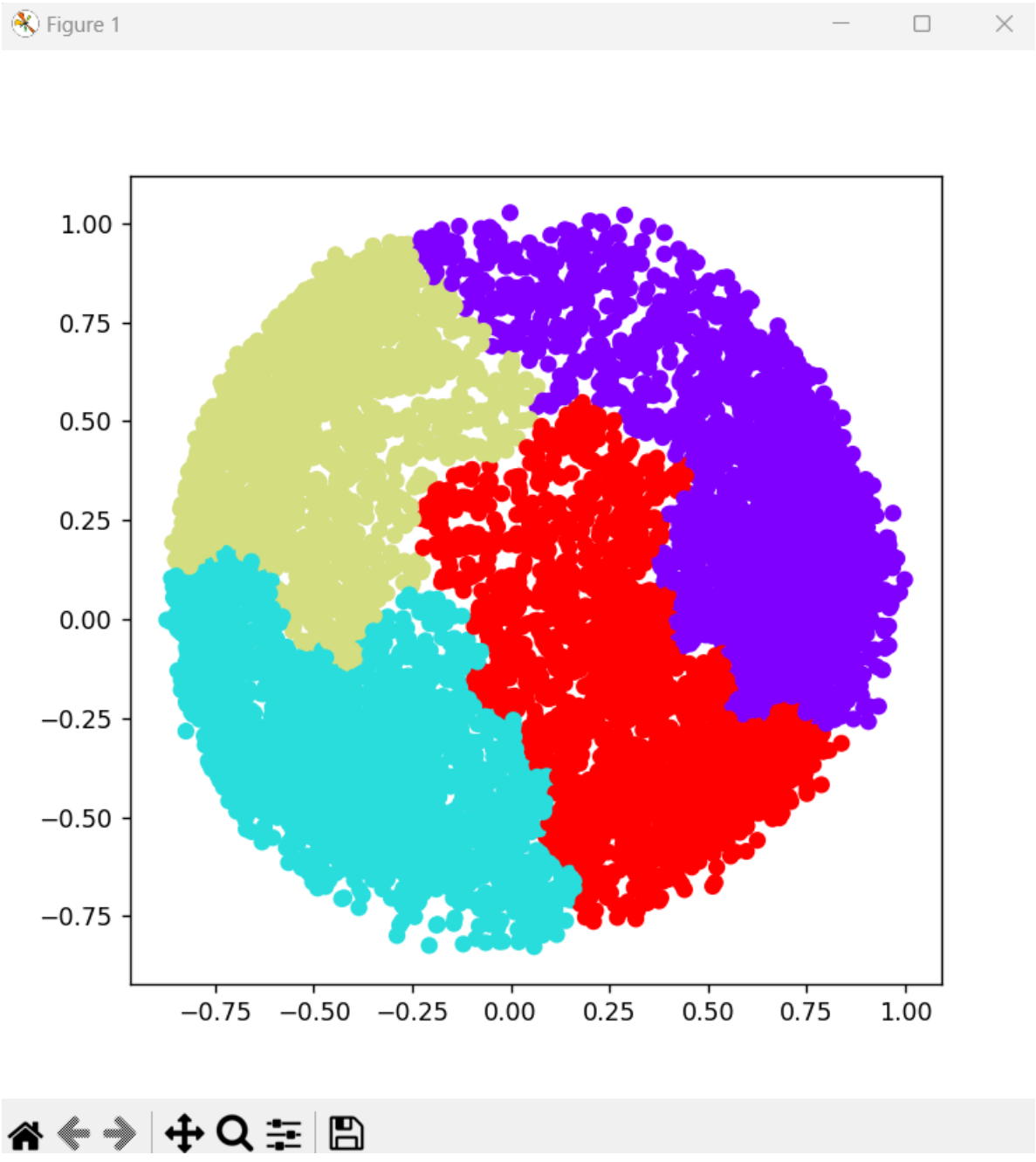


Figure 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

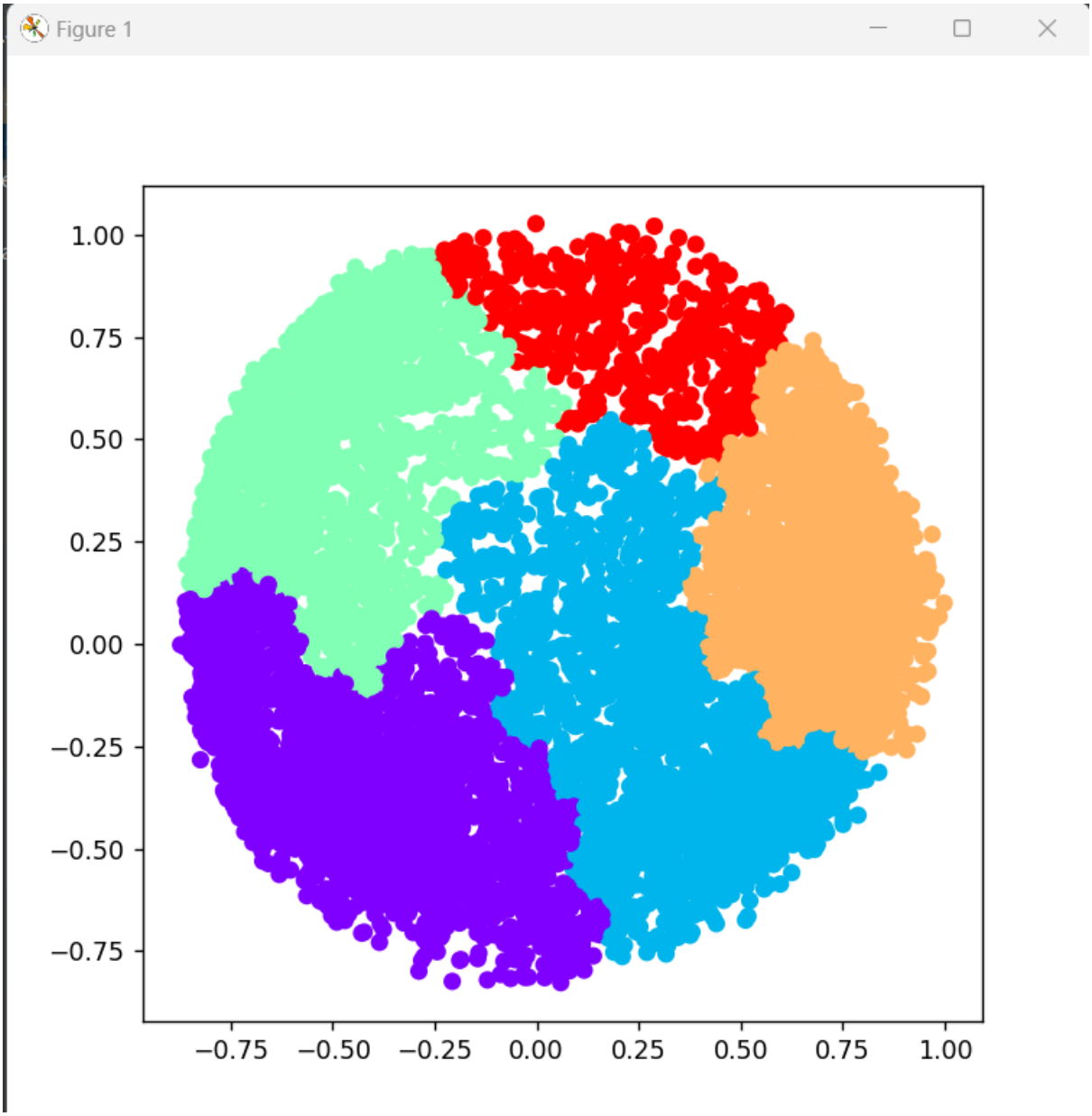
```
# With k=3
ac3 = AgglomerativeClustering(n_clusters=3)
# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
            c=ac3.fit_predict(principalDf), cmap='rainbow')
plt.show()
```



```
# With k=4
ac4 = AgglomerativeClustering(n_clusters=4)
# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
            c=ac4.fit_predict(principalDf), cmap='rainbow')
plt.show()
```



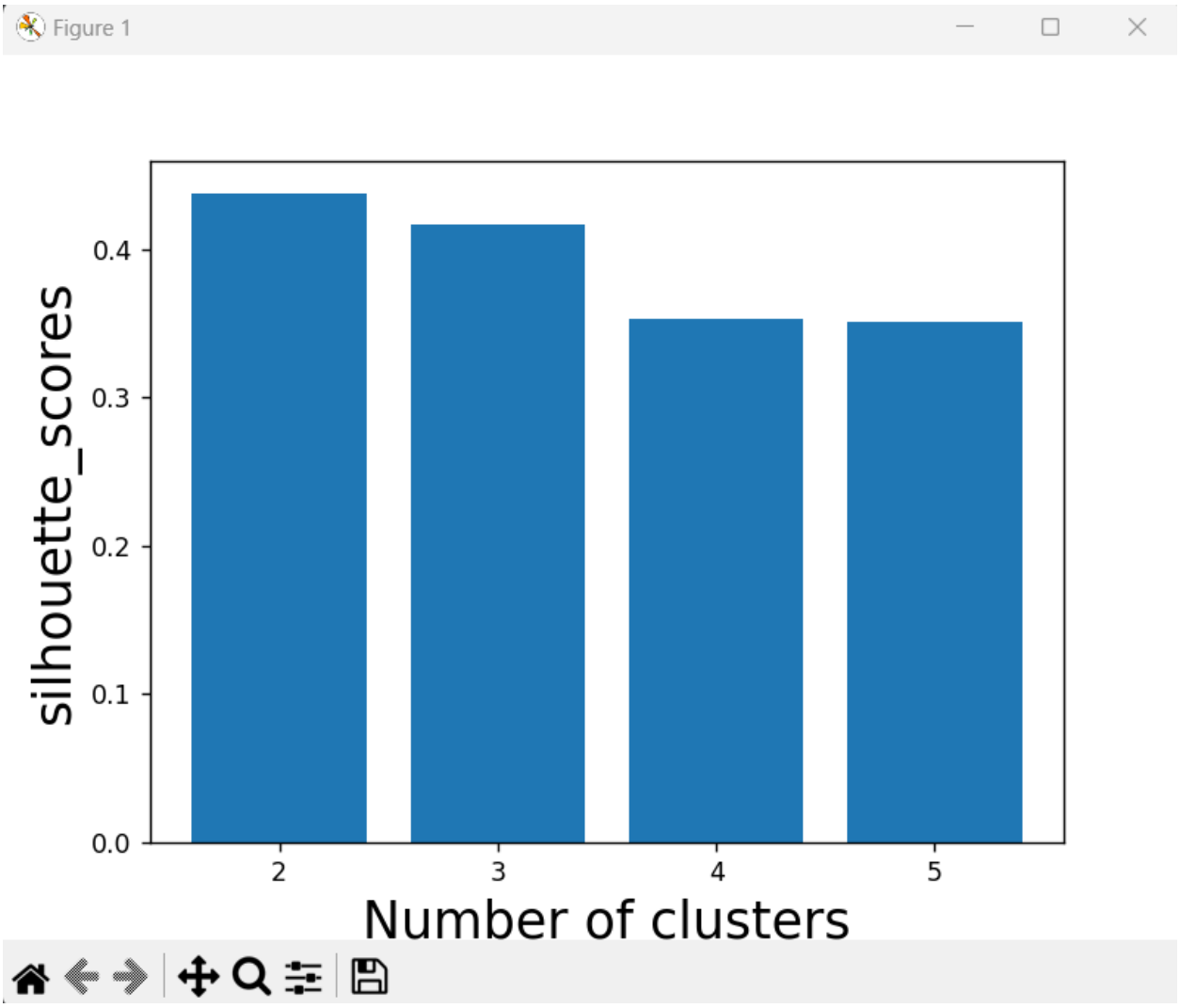
```
# With k=5
ac5 = AgglomerativeClustering(n_clusters=5)
# Visualizing the clustering
plt.figure(figsize=(6, 6))
plt.scatter(principalDf['principal component 1'], principalDf['principal component 2'],
            c=ac5.fit_predict(principalDf), cmap='rainbow')
plt.show()
```



```python
# e) Evaluate different variations using Silhouette Scores and Visualize results with a bar chart.
k = [2, 3, 4, 5]
# To display the number of clusters as integers in the bar plot
new_list = range(math.floor(min(k)), math.ceil(max(k))+1)
plt.xticks(new_list)

# Creating a list with the silhouette scores of the different models
silhouette_scores = [silhouette_score(principalDf, ac2.fit_predict(principalDf)),
                     silhouette_score(principalDf, ac3.fit_predict(principalDf)),
                     silhouette_score(principalDf, ac4.fit_predict(principalDf)),
                     silhouette_score(principalDf, ac5.fit_predict(principalDf))]

# Plotting a bar graph to compare the results
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize=20)
plt.ylabel('silhouette_scores', fontsize=20)
plt.show()
```



Programming elements:
Hierarchical Clustering

In class programming:

1) (Provide only mathematical solutions for this question) Six points with the following attributes are given, calculate and find out clustering representations and dendrogram using Single, complete, and average link proximity function in hierarchical clustering technique.

point	x coordinate	y coordinate
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022

Table : X-Y coordinates of six points.

	p1	p2	p3	p4	p5	p6
p1	0.0000	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0.0000	0.1483	0.2042	0.1388	0.2540
p3	0.2218	0.1483	0.0000	0.1513	0.2843	0.1100
p4	0.3688	0.2042	0.1513	0.0000	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0.0000	0.3921
p6	0.2347	0.2540	0.1100	0.2216	0.3921	0.0000

Table : Distance Matrix for Six Points

1 Ans :- Given Distance Matrix, here in this solution we are not writing the values in the upper diagonal as it is reflection of lower triangle [as distance between x & y is same as distance between y & x].

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0.0000					
P_2	0.2357	0.0000				
P_3	0.2218	0.1483	0.0000			
P_4	0.3688	0.2042	0.1513	0.0000		
P_5	0.3421	0.1388	0.2843	0.2932	0.0000	
P_6	0.2347	0.2540	0.1100	0.2216	0.3921	0.0000

→ Now will start clustering using Single link proximity

* The Smallest distance is between P_6 & P_3, so they get linked up (or) merged first into the cluster $P_3 P_6$.

* To obtain the new distance matrix, we need to remove the P_3 & P_6 entries and replace it by an entry "$P_3 P_6$". Since we are using Single link proximity, the distance between

"P_3P_6" and every other item is the minimum of of the distance between this item and P_3 and this item and P_6.

Min [Distance $d(P_1, P_3) = 0.2218$ & $d(P_1, P_6) = 0.2347$]

$\Rightarrow$ 0.2218

Same way, $D("P_3P_6", P_2)$

$= \text{Min}\left[d(P_2, P_3) \text{ \& } d(P_2, P_6)\right]$

$= \text{Min}\left[0.1483 \text{ \& } 0.2540\right]$

$= 0.1483$

$\Rightarrow D("P_3P_6", P_4) = \text{Min}\left[d(P_4, P_3) \text{ \& } d(P_4, P_6)\right]$

$= 0.1513$

$\Rightarrow D("P_3P_6", P_5) = \text{Min}\left[d(P_5, P_3) \text{ \& } d(P_5, P_6)\right]$

$= 0.2843$

The new Distance Matrix is -

	P_3P_6	P_1	P_2	P_4	P_5
P_3P_6	0.0000				
P_1	0.2218	0.0000			
P_2	0.1483	0.2357	0.0000		
P_4	0.1513	0.3688	0.2042	0.0000	
P_5	0.2843	0.3421	0.1388	0.2932	0.0000

→ The next smallest distance is between P_2 & P_5. So they get merged into a new cluster i.e., "P_2P_5"

→ Finding distance between P_2P_5 & every other item.

$\Rightarrow D(\text{"}P_2P_5\text{"}, \text{"}P_3P_6\text{"}) = \text{Min}[d(\text{"}P_3P_6\text{"}, P_2) \& d(\text{"}P_3P_6\text{"}, P_5)]$

$= \text{Min}(0.1483 \& 0.2843)$

$= 0.1483$

$\Rightarrow D(\text{"}P_2P_5\text{"}, P_1) = \text{Min}[d(P_1, P_2) \& d(P_1, P_5)]$

$= 0.2357$

$\Rightarrow D(\text{"}P_2P_5\text{"}, P_4) = \text{Min}[d(P_4, P_2) \& d(P_4, P_5)]$

$= 0.2042$

The new Distance Matrix is

	P_3P_6	P_2P_5	P_1	P_4
P_3P_6	0.0000			
P_2P_5	0.1483	0.0000		
P_1	0.2218	0.2357	0.0000	
P_4	0.1513	0.2042	0.3688	0.0000

→ The next Smallest distance is between "P_2P_5" & "P_3P_6", So they merged into a new cluster "$P_2P_3P_5P_6$"

→ Finding distance between "$P_2P_3P_5P_6$" & every other item.

$$D\left["P_2P_3P_5P_6", P_1\right] = \text{Min}\left[d("P_2P_5", P_1) \text{ & } d("P_3P_6", P_1)\right]$$

$$= \text{Min}\left[0.2357 \text{ & } 0.2218\right]$$

$$= 0.2218$$

$$D\left["P_2P_3P_5P_6", P_4\right] = \text{Min}\left[d("P_2P_5", P_4) \text{ & } d("P_3P_6", P_4)\right]$$

$$= \text{Min}\left[0.2042 \text{ & } 0.1513\right].$$

$$= 0.1513$$

The New Distance Matrix is:

	$P_2P_3P_5P_6$	P_1	P_4
$P_2P_3P_5P_6$	0.0000		
P_1	0.2218	0.0000	
P_4	0.1513	0.3688	0.0000

→ The next Smallest distance is between P_4 & "$P_2P_3P_5P_6$"

So they form a new cluster "$P_2P_3P_4P_5P_6$"

$$D["P_2P_3P_4P_5P_6", P_1] = Min[d("P_2P_3P_5P_6", P_1) \& d("P_4", P_1)]$$

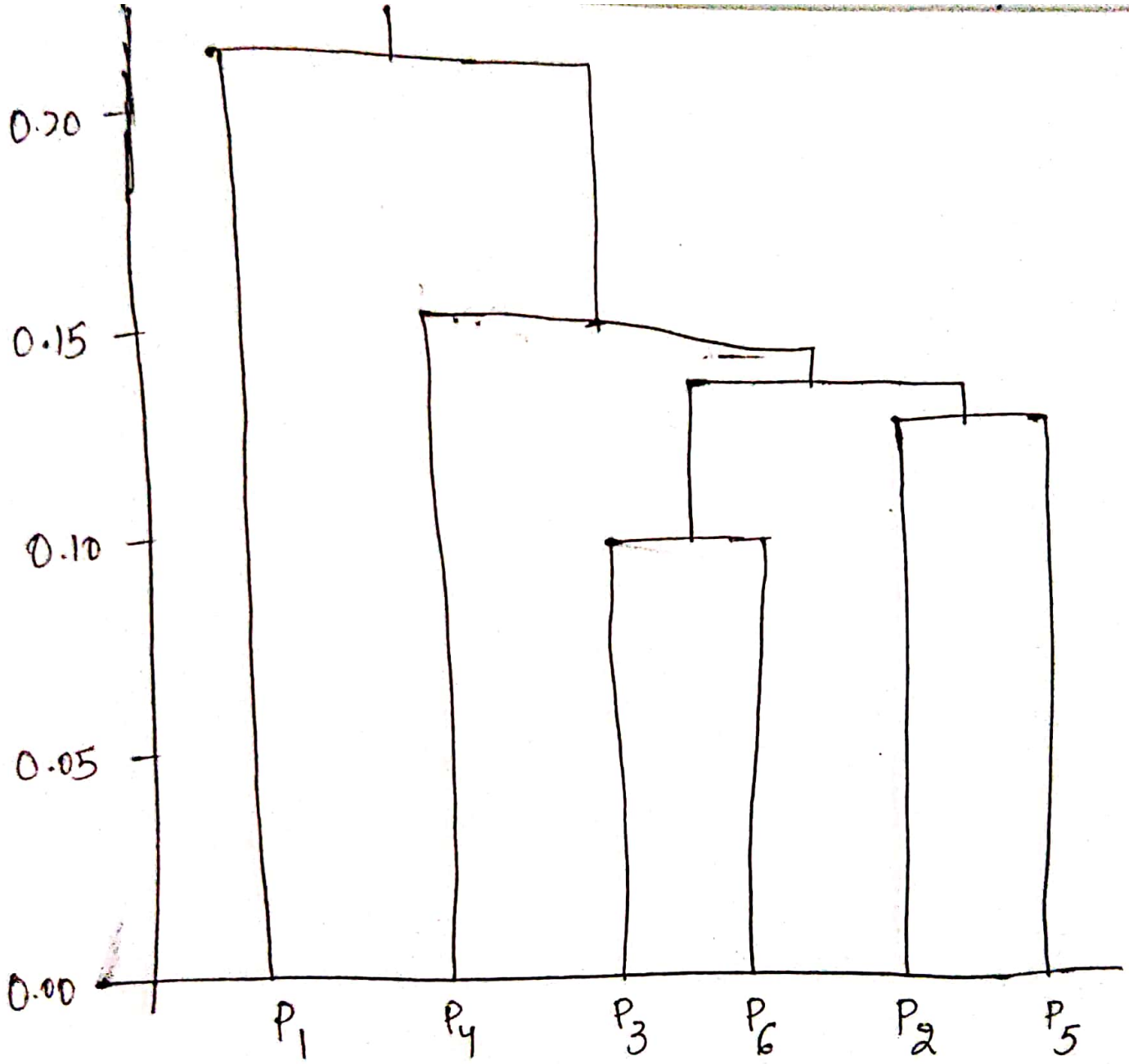
$$= 0.2218$$

New Distance Matrix is.

	$P_2P_3P_4P_5P_6$	P_1
$P_2P_3P_4P_5P_6$	0.0000	
P_1	0.2218	0.0000

Finally P_1 merges with $P_2P_3P_4P_5P_6$ to form the new cluster "$P_1P_2P_3P_4P_5P_6$"

dendrogram is :-



Complete Link Proximity

Since we got the distance matrix provided will start clustering using Complete Link Proximity.

→ The smallest Distance is between P_3 & P_6, So they get linked up & merged first into the cluster "P_3P_6".

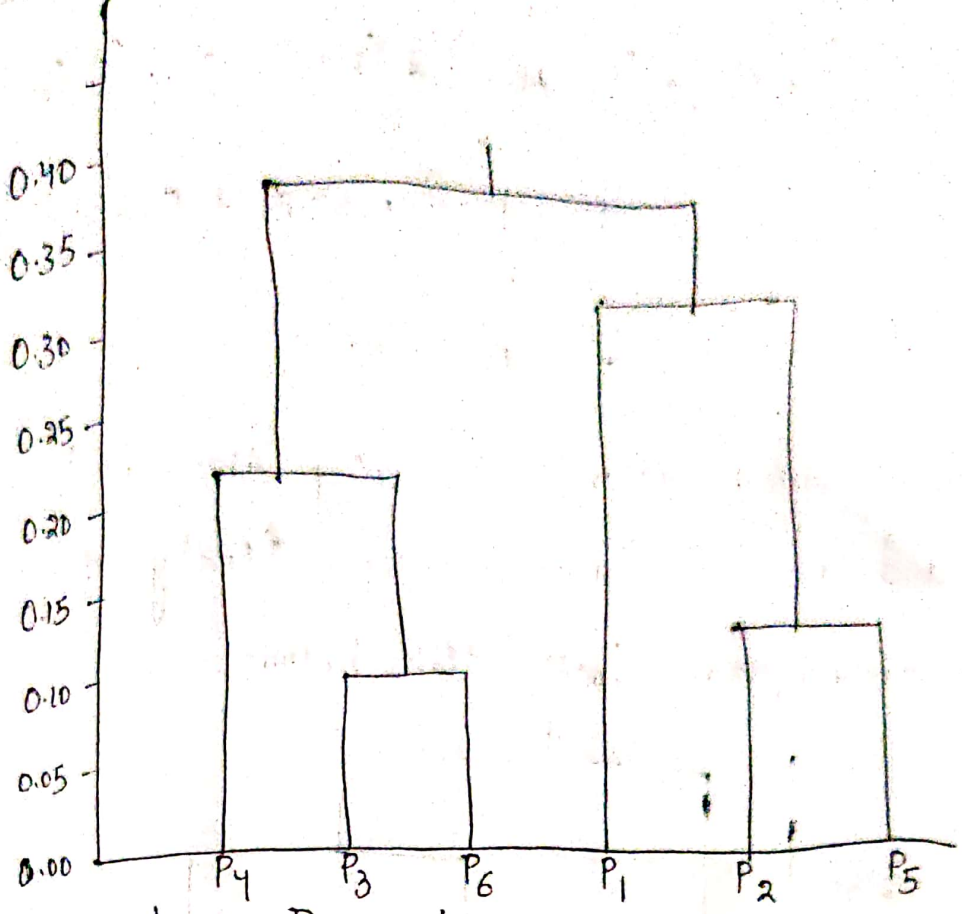
→ To get the new distance matrix, we need to remove the P_3 & P_6 entries and replace it by an entry "P_3P_6". Since we are using Complete Link Proximity, the distance between "P_3P_6" and every other item is the maximum of the distance between this item & P_3 and this item & P_6.

$$\Rightarrow D(\text{"}P_3P_6\text{"}, P_1) = \text{Max}[d(P_1, P_3) \& d[P_1, P_6)]$$

$$= \text{Max}[0.2218, 0.2347]$$

$$= 0.2347$$

Same way after performing all the steps like finding the distance & new distance matrix. The dendrogram looks like the below



Average Link Proximity

Since we got the distance matrix provided will start clustering using Average Link Proximity.

→ The Smallest Distance is between P_3 & P_6. So they get linked up & merged first into the cluster "P_3P_6"

→ To get the new distance matrix; we need to remove the P_3 & P_6 entries and replace it by an entry "P_3P_6". Since we are using Average Link Proximity, the distance between "P_3P_6" and every other item is the Average of the distances between this item & P_3 and this item & P_6.

$$\Rightarrow D(\text{"}P_3P_6\text{"}, P_1) = \text{Avg}\left[d(P_1,P_3) \;\&\; d(P_1,P_6)\right]$$

$$= \frac{1}{2}\left[0.2218 + 0.2347\right]$$

$$= 0.22825$$

Same way after calculating the distances and new distance matrices, finally the dendrogram looks like below.