MACHINE LEARNING ASSIGNMENT-6

NAME: PRUDHVI MAHESH MEKA

ID: 700738978

CRN: 12664

GITHUB REPO LINK: https://github.com/mpm941/Assignment_6.git

Link to the Video:
https://drive.google.com/file/d/1Nhole7Eej4h4EL6cDeUkpweiJHOrGE3j/view?usp=share_link

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.**

- In **Single Linkage,** the distance between two clusters is the minimum distance between members of the two clusters.

	p1	p2	p3	p4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

Smallest distance from above data is 0.11 and so p3 & p6 forms first cluster.

	p1	p2	p36	p4	p5
p1	0	0.2357	0.2218	0.3688	0.3421
p2	0.2357	0	0.1483	0.2042	0.1388
p36	0.2218	0.1483	0	0.1513	0.2843
p4	0.3688	0.2042	0.1513	0	0.2932
p5	0.3421	0.1388	0.2843	0.2932	0

Smallest distance from above data is 0.1388 so p2 & p5 forms 2nd cluster

	p1	p25	p36	p4
p1	0	0.2357	0.2218	0.3688
p25	0.2357	0	0.1483	0.2042
p36	0.2218	0.1483	0	0.1513

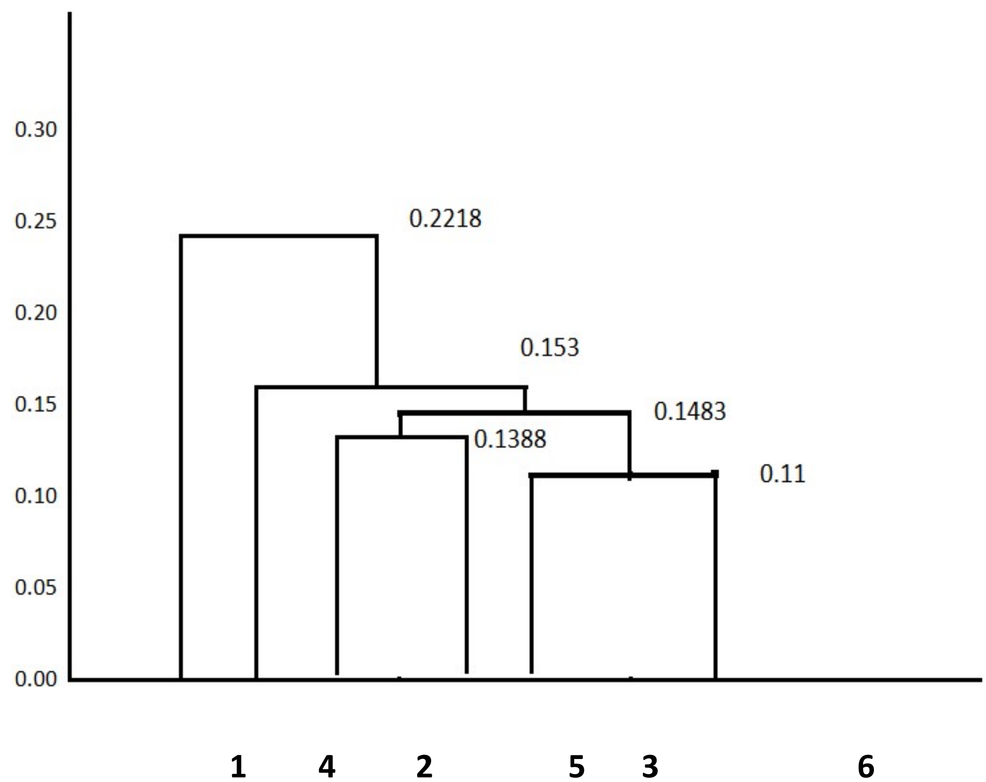
p4	0.3688	0.2042	0.1513	0

Smallest distance from above data is 0.1483 so p25 & p36 forms 3rd cluster.

	p1	**p(25)(36)**	**p4**
p1	0	0.2218	0.3688
p(25)(36)	0.2218	0	0.1513
p4	0.3688	0.1513	0

Smallest distance from above data is 0.1513 so p (25)(36) & p4 forms 4th cluster.

	p1	**p4(25)(36)**
p1	0	0.2218
p4(25)(36)	0.2218	0



- In **Complete Linkage,** the distance between two clusters is the maximum distance between members of the two clusters.

	p1	p2	p3	p4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

Smallest distance from above data is 0.11. So, p3 & p6 forms first cluster.

	p1	p2	p36	p4	p5
p1	0	0.2357	0.2347	0.3688	0.3421
p2	0.2357	0	0.254	0.2042	0.1388
p36	0.2347	0.254	0	0.2216	0.3921
p4	0.3688	0.2042	0.2216	0	0.2932
p5	0.3421	0.1388	0.3921	0.2932	0

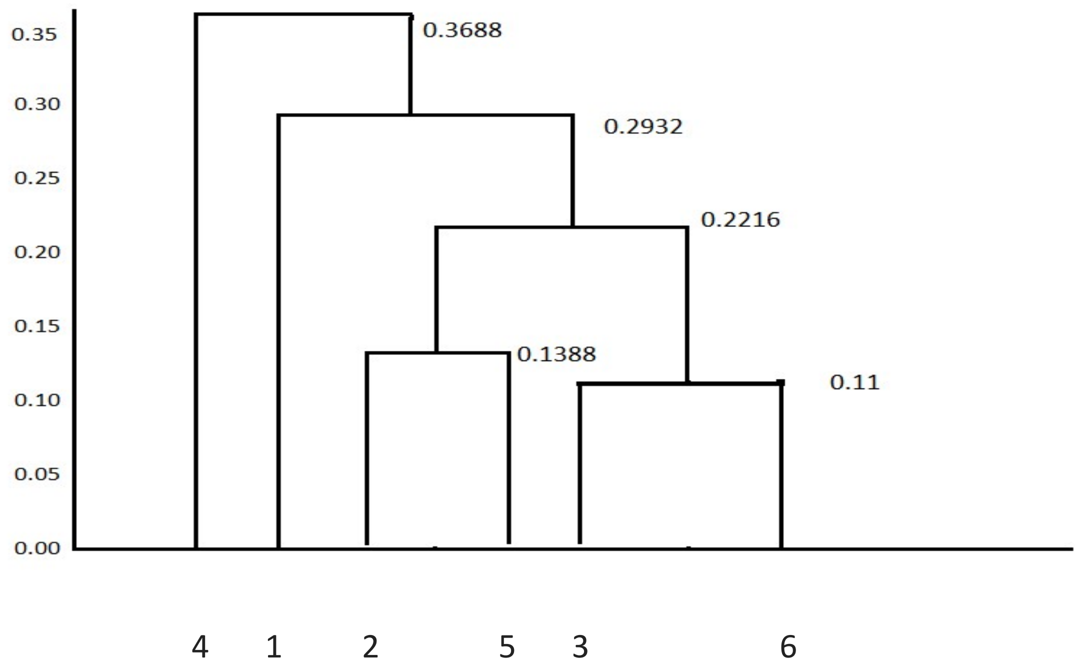
Smallest distance from above data is 0.1388. So, p2 & p5 forms 2nd cluster.

	p1	p25	p36	p4
p1	0	0.3421	0.2347	0.3688
p25	0.3421	0	0.3921	0.2932
p36	0.2347	0.3921	0	0.2216
p4	0.3688	0.2932	0.2216	0

Smallest distance from above data is 0.2216. So, p25 & p36 forms 3rd cluster.

	p1	p (25)(36)	p4
p1	0	0.3421	0.3688
P (25)(36)	0.3421	0	0.2932
p4	0.3688	0.2932	0

Smallest distance from above data is 0.2932. So, p (25)(36) & p1 forms 4th cluster.



- In **Average Linkage,** the distance between two clusters is the average of all distances between members of the two clusters.

	p1	p2	p3	p4	p5	p6
p1	0	0.2357	0.2218	0.3688	0.3421	0.2347
p2	0.2357	0	0.1483	0.2042	0.1388	0.254
p3	0.2218	0.1483	0	0.1513	0.2843	0.11
p4	0.3688	0.2042	0.1513	0	0.2932	0.2216
p5	0.3421	0.1388	0.2843	0.2932	0	0.3921
p6	0.2347	0.254	0.11	0.2216	0.3921	0

Smallest distance from above data is 0.11. So, p3 & p6 forms 1st cluster.

	p1	p2	p36	p4	p5
p1	0	0.2357	0.22825	0.3688	0.3421
p2	0.2357	0	0.20115	0.2042	0.1388
p36	0.22825	0.20115	0	0.18645	0.3382
p4	0.3688	0.2042	0.18645	0	0.2932
p5	0.3421	0.1388	0.3382	0.2932	0

Smallest distance from above data is 0.1388. So, p2 & p5 forms 2nd cluster.

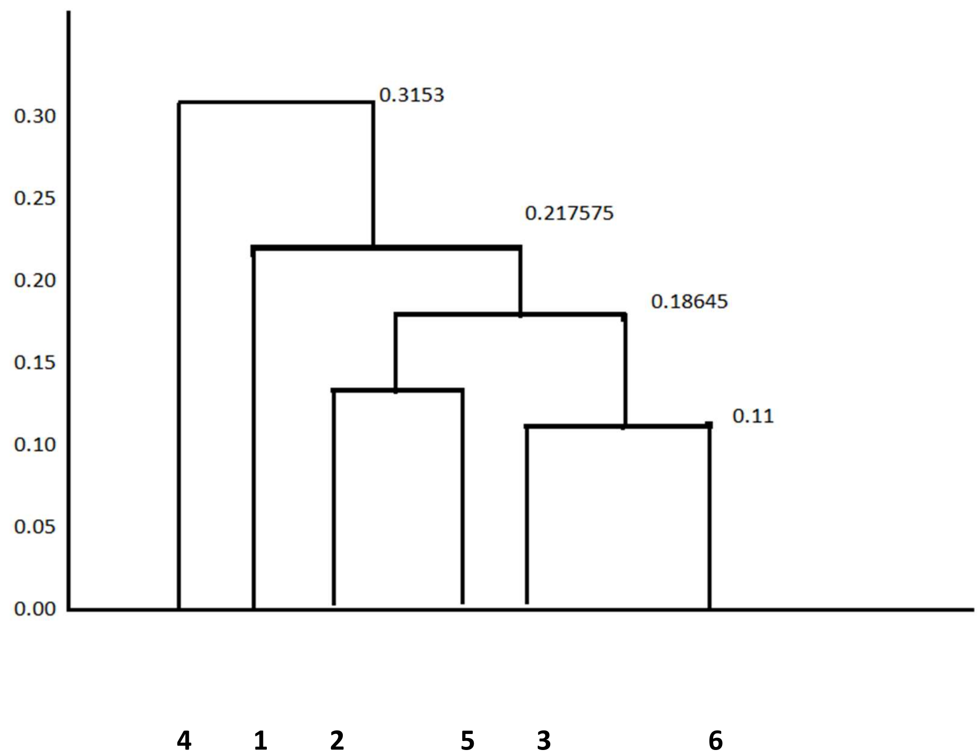
	p1	p25	p36	p4
p1	0	0.2889	0.2347	0.3688
p25	0.2889	0	0.269675	0.2487
p36	0.2347	0.269675	0	0.18645
p4	0.3688	0.2487	0.18645	0

Smallest distance from above data is 0.18645. So, p25 & p36 forms 3^{rd} cluster.

	p1	P (25)(36)	p4
p1	0	0.2618	0.3688
P (25)(36)	0.2618	0	0.217575
p4	0.3688	0.217575	0

Smallest distance from above data is 0.2175. So, p (25)(36) and p1 forms 4^{th} cluster.

	p1(25)(36)	p4
p1(25)(36)	0	0.3153
p4	0.3153	0



---

2) Use CC_GENERAL.csv given in the folder and apply:

a) Pre-process 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.

In [18]:

```
#IMPORTING ALL THE REQUIRED LIBRARIES
import seaborn as sns
from sklearn import preprocessing, metrics
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as shc
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")
```

In [19]:

```
df = pd.read_csv('CC GENERAL.csv')
df.head()
```

Out[19]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FRE(
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	

Here, we imported all the required libraries and then read "CC GENERAL.CSV" Data set. Then obtained five entries from the data as shown in the above illustration.

In [20]:

```
df.isnull().any()
```

Out[20]:

```
CUST_ID                              False
BALANCE                              False
BALANCE_FREQUENCY                    False
PURCHASES                            False
ONEOFF_PURCHASES                     False
INSTALLMENTS_PURCHASES               False
CASH_ADVANCE                         False
PURCHASES_FREQUENCY                  False
ONEOFF_PURCHASES_FREQUENCY           False
PURCHASES_INSTALLMENTS_FREQUENCY     False
CASH_ADVANCE_FREQUENCY               False
CASH_ADVANCE_TRX                     False
PURCHASES_TRX                        False
CREDIT_LIMIT                          True
PAYMENTS                             False
MINIMUM_PAYMENTS                      True
PRC_FULL_PAYMENT                     False
TENURE                               False
dtype: bool
```

Then it verified for any null values from the table.

In [36]:
```
x = df.drop('CUST_ID', axis = 1)
print(x)
```

OUTPUT:

```
BALANCE  BALANCE_FREQUENCY  PURCHASES  ONEOFF_PURCHASES  \
0       40.900749           0.818182      95.40              0.00
1     3202.467416           0.909091       0.00              0.00
2     2495.148862           1.000000     773.17            773.17
3     1666.670542           0.636364    1499.00           1499.00
4      817.714335           1.000000      16.00             16.00
...           ...                ...        ...               ...
8945    28.493517           1.000000     291.12              0.00
8946    19.183215           1.000000     300.00              0.00
8947    23.398673           0.833333     144.40              0.00
8948    13.457564           0.833333       0.00              0.00
8949   372.708075           0.666667    1093.25           1093.25

      INSTALLMENTS_PURCHASES  CASH_ADVANCE  PURCHASES_FREQUENCY  \
0                      95.40      0.000000             0.166667
1                       0.00   6442.945483             0.000000
2                       0.00      0.000000             1.000000
3                       0.00    205.788017             0.083333
4                       0.00      0.000000             0.083333
...                      ...           ...                  ...
8945                  291.12      0.000000             1.000000
8946                  300.00      0.000000             1.000000
8947                  144.40      0.000000             0.833333
8948                    0.00     36.558778             0.000000
8949                    0.00    127.040008             0.666667

      ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
0                       0.000000                          0.083333
1                       0.000000                          0.000000
2                       1.000000                          0.000000
3                       0.083333                          0.000000
4                       0.083333                          0.000000
...                          ...                               ...
8945                    0.000000                          0.833333
8946                    0.000000                          0.833333
8947                    0.000000                          0.666667
8948                    0.000000                          0.000000
8949                    0.666667                          0.000000

      CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX  CREDIT_LIMIT  \
0                   0.000000                 0              2        1000.0
1                   0.250000                 4              0        7000.0
2                   0.000000                 0             12        7500.0
3                   0.083333                 1              1        7500.0
4                   0.000000                 0              1        1200.0
...                      ...               ...            ...           ...
8945                0.000000                 0              6        1000.0
8946                0.000000                 0              6        1000.0
8947                0.000000                 0              5        1000.0
8948                0.166667                 2              0         500.0
8949                0.333333                 2             23        1200.0

         PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT  TENURE
0      201.802084        139.509787          0.000000      12
1     4103.032597       1072.340217          0.222222      12
2      622.066742        627.284787          0.000000      12
```

```
3         0.000000        864.206542          0.000000      12
4       678.334763        244.791237          0.000000      12
...            ...               ...               ...     ...
8945    325.594462         48.886365          0.500000       6
8946    275.861322        864.206542          0.000000       6
8947     81.270775         82.418369          0.250000       6
8948     52.549959         55.755628          0.250000       6
8949     63.165404         88.288956          0.000000       6

[8950 rows x 17 columns]
```

In [23]:
```
#SCALING
scaler = StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
```

In [24]:
```
#NORMALIZING THE DATA
X_normalized = normalize(X_scaled_array)
X_normalized = pd.DataFrame(X_normalized)
```

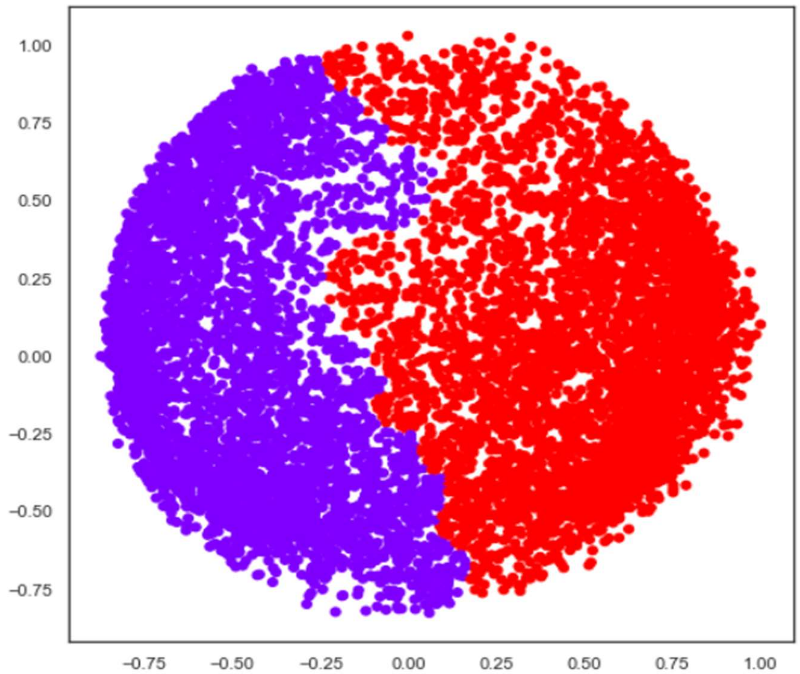
In [25]:
```
#REDUCING THE DIMENSIONALITY OF DATA
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(X_normalized)
principalDf =  pd.DataFrame(data = X_principal, columns = ['principal component-1', 'principal component-2'])
finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)
finalDf.head()
```

Out[25]:

	principal component-1	principal component-2	TENURE
0	-0.489826	-0.679679	12
1	-0.518790	0.545014	12
2	0.330886	0.268984	12
3	-0.482376	-0.092122	12
4	-0.563289	-0.481914	12

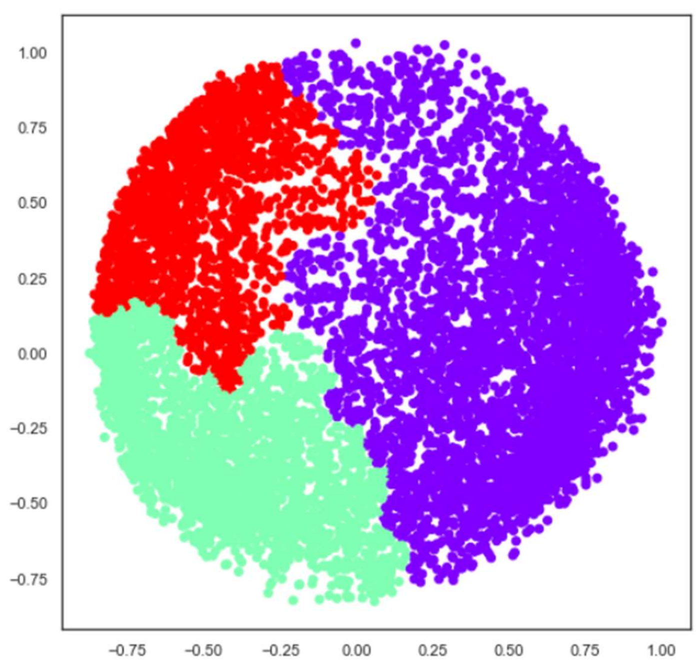
- We used PCA with K=2 to reduce the input dimensions to two features.
- Now, we applied the scaling and normalized the data and then reduced the dimensionality of data and applied principal component analysis.

In [27]:
```
ac2 = AgglomerativeClustering(n_clusters = 2)

# VISUALIZING THE CLUSTERING
plt.figure(figsize =(8, 8))
plt.scatter(principalDf['principal component-1'], principalDf['principal component-2'],
           c = ac2.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```



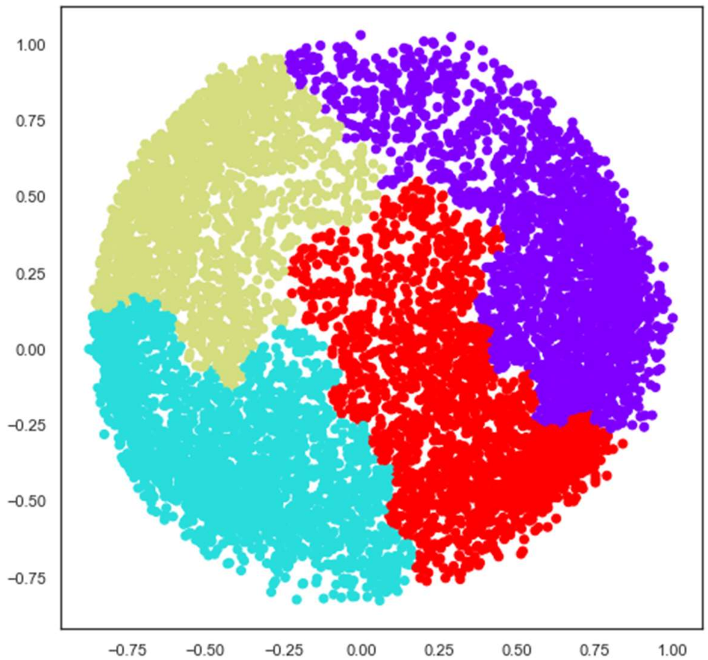
Here, we used the agglomerative clustering with K=2 and on reduced features and illustrated results for each k value in above scatter plot.

In [28]:
```
ac3 = AgglomerativeClustering(n_clusters = 3)

# VISUALIZING THE CLUSTERING
plt.figure(figsize =(8, 8))
plt.scatter(principalDf['principal component-1'], principalDf['principal component-2'],
           c = ac3.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```



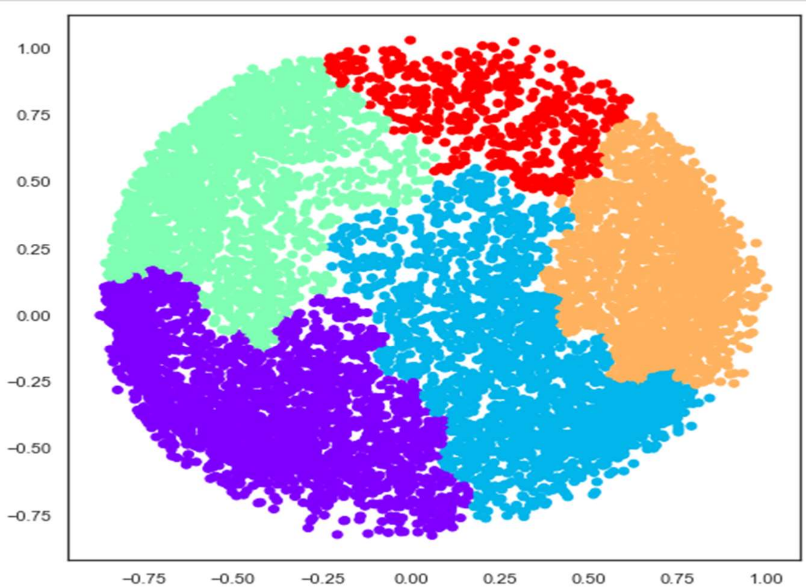
Here, we used the agglomerative clustering with K=4 and on reduced features and illustrated results for each k value in above scatter plot.

```
In [29]: ac4 = AgglomerativeClustering(n_clusters = 4)

         #VISUALIZING THE CLUSTERING
         plt.figure(figsize =(8, 8))
         plt.scatter(principalDf['principal component-1'], principalDf['principal component-2'],
                    c = ac4.fit_predict(principalDf), cmap ='rainbow')
         plt.show()
```



Here, we used the agglomerative clustering with K=4 and on reduced features and illustrated results for each k value in above scatter plot.

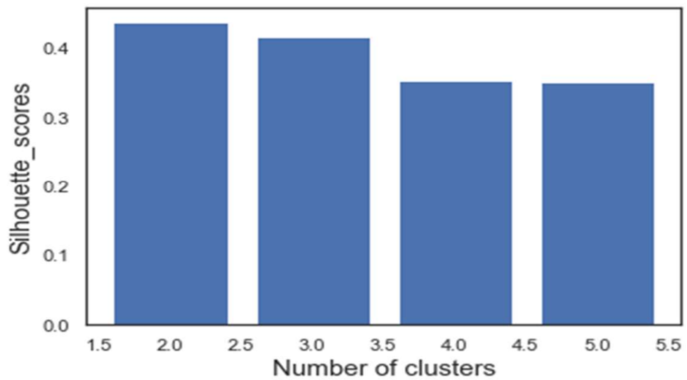
```
In [33]: ac5 = AgglomerativeClustering(n_clusters = 5)

         # Visualizing the clustering
         plt.figure(figsize =(8, 8))
         plt.scatter(principalDf['principal component-1'], principalDf['principal component-2'],
                    c = ac5.fit_predict(principalDf), cmap ='rainbow')
         plt.show()
```



Here, we used the agglomerative clustering with K=5 and on reduced features and illustrated results for each k value in above scatter plot.

```
In [34]:  k = [2, 3, 4, 5]

          # APPENDING THE SILHOUETTE SCORE OF ALL MODELS TO THE LIST.
          silhouette_scores = []
          silhouette_scores.append(
                  silhouette_score(principalDf, ac2.fit_predict(principalDf)))
          silhouette_scores.append(
                  silhouette_score(principalDf, ac3.fit_predict(principalDf)))
          silhouette_scores.append(
                  silhouette_score(principalDf, ac4.fit_predict(principalDf)))
          silhouette_scores.append(
                  silhouette_score(principalDf, ac5.fit_predict(principalDf)))
```

```
In [35]:  # BAR GRAPH TO ILLUSTRATE THE RESULTS
          plt.bar(k, silhouette_scores)
          plt.xlabel('Number of clusters', fontsize = 15)
          plt.ylabel('Silhouette_scores', fontsize = 15)
          plt.show()
```



Here, we append all the silhouette scores of all models and visualized the results with a bar chart.
