Machine Learning Assignment 6

Hierarchical clustering

Single Link: Distance between two clusters is the distance between the closest points. Also called “neighbor joining.”

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

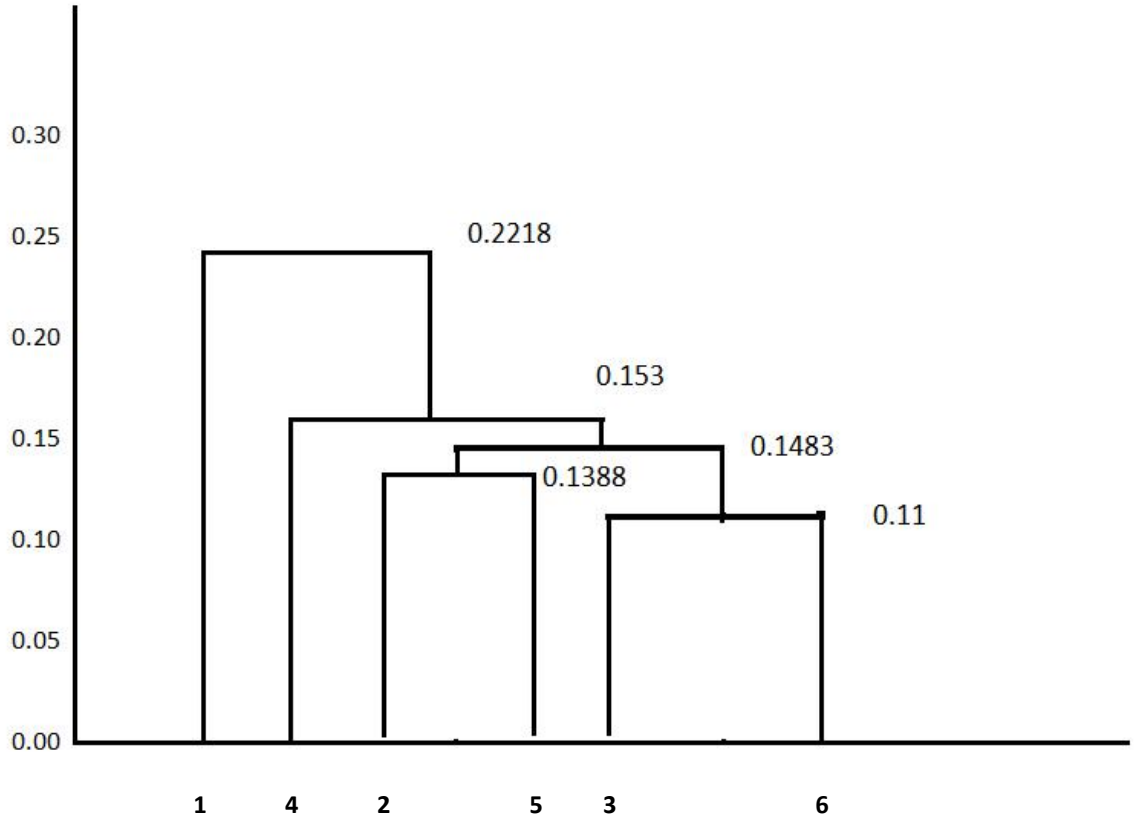
Distance[(x,y),(a,b)] = sqrt((x-a)^2+(y-b)^2)

	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

	p1	**p2**	**p3,p6**	**p4**	**p5**
p1	0	0.2357	0.2218	0.3688	0.3421
p2	0.2357	0	0.1483	0.2042	0.1388
p3,p6	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

We are taking minimum distance value which is 0.11 and calculating minimum distance between min[(p3,p6),p1], min[(p3,p6),p2], min[(p3,p6),p4], min[(p3,p6),p5] updating in the table. Then the minimum distance is 0.1388. and repeating same for next cluster

so p2 and p5 forms 2nd cluster					
	p1	**p2,p5**	**p3,p6**	**p4**	
p1	0	0.2357	0.2218	0.3688	
p2,p5	0.2357	0	0.1483	0.2042	
p3,p6	0.2218	0.1483	0	0.1513	
p4	0.3688	0.2042	0.1513	0	
smallest distance from above data is					0.1483
so (p2,p5) and (p3,P6) forms 3rdcluster					
	p1	**p2,p5,p3,p6**	**p4**		
p1	0	0.2218	0.3688		
p2,p5,p3,p6	0.2218	0	0.1513		
p4	0.3688	0.1513	0		
smallest distance from above data is					0.153
so (**p2,p5,p3,p6)**and p4 forms 4thcluster					
	p1	**p4,p2,p5,p3,p6**			
p1	0	0.2218			
p4,p2,p5,p3,p6	0.2218	0			



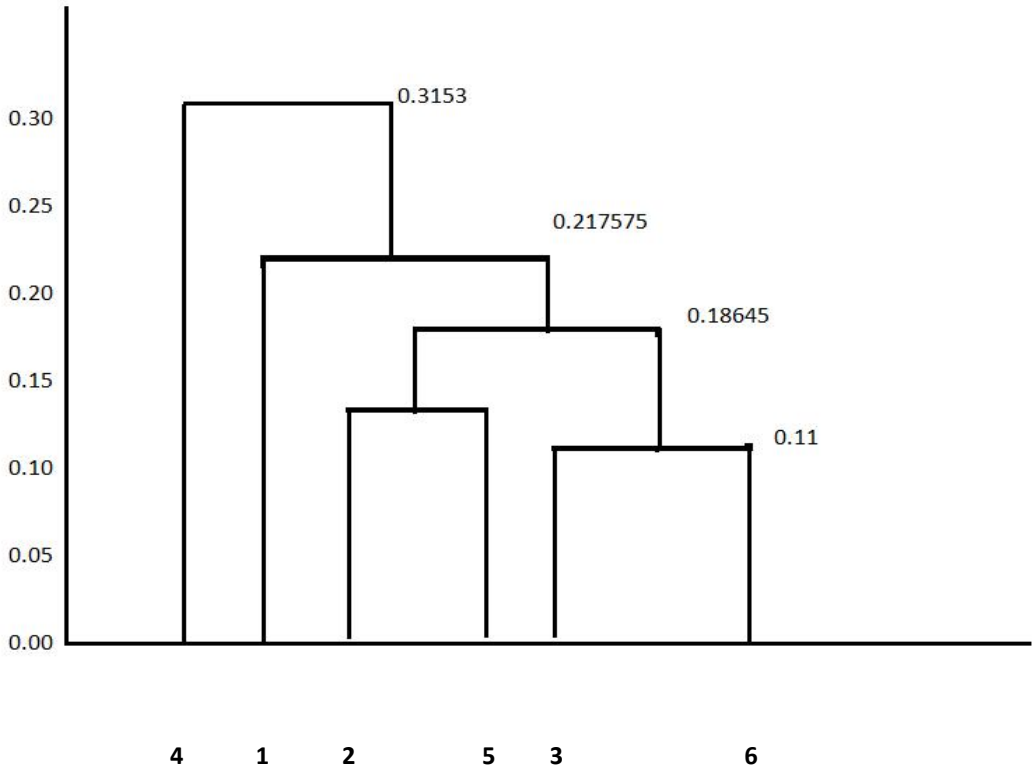
Average Link: Distance between clusters is distance between the cluster centroids.

Distance[(x,y),(a,b)] = sqrt((x-a)^2+(y-b)^2)

We are taking minimum distance value which is 0.11 and calculating average distance between AVG[dis(p3,p6),p1]), AVG[dis(p3,p6),p2]), AVG[dis(p3,p6),p4]),AVG[dis(p3,p6),p5]) updating in the table. Then the minimum distance is 0.1388. and repeating same for next cluster

	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 and p6 forms first cluster						
	p1	**p2**	**p3,p6**	**p4**	**p5**	
p1	0	0.2357	0.22825	0.3688	0.3421	
p2	0.2357	0	0.20115	0.2042	0.1388	
p3,p6	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 and p5 forms 2nd cluster						
	p1	**p2,p5**	**p3,p6**	**p4**		
p1	0	0.2889	0.2347	0.3688		
p2,p5	0.2889	0	0.269675	0.2487		
p3,p6	0.2347	0.269675	0	0.18645		
p4	0.3688	0.2487	0.18645	0		
smallest distance from above data is					0.18645	
so (p2,p5) and (p3,p6) forms 3rdcluster						
	p1	**p2,p5,p3,p6**	**p4**			
p1	0	0.2618	0.3688			
p2,p5,p3,p6	0.2618	0	0.217575			
p4	0.3688	0.217575	0			
smallest distance from above data is					0.217575	
so (**p2,p5,p3,p6)**and p1 forms 4thcluster						
	p1,p2,p5,p3,p6	**p4**				
p1,p2,p5,p3,p6	0	0.3153				
p4	0.3153	0				



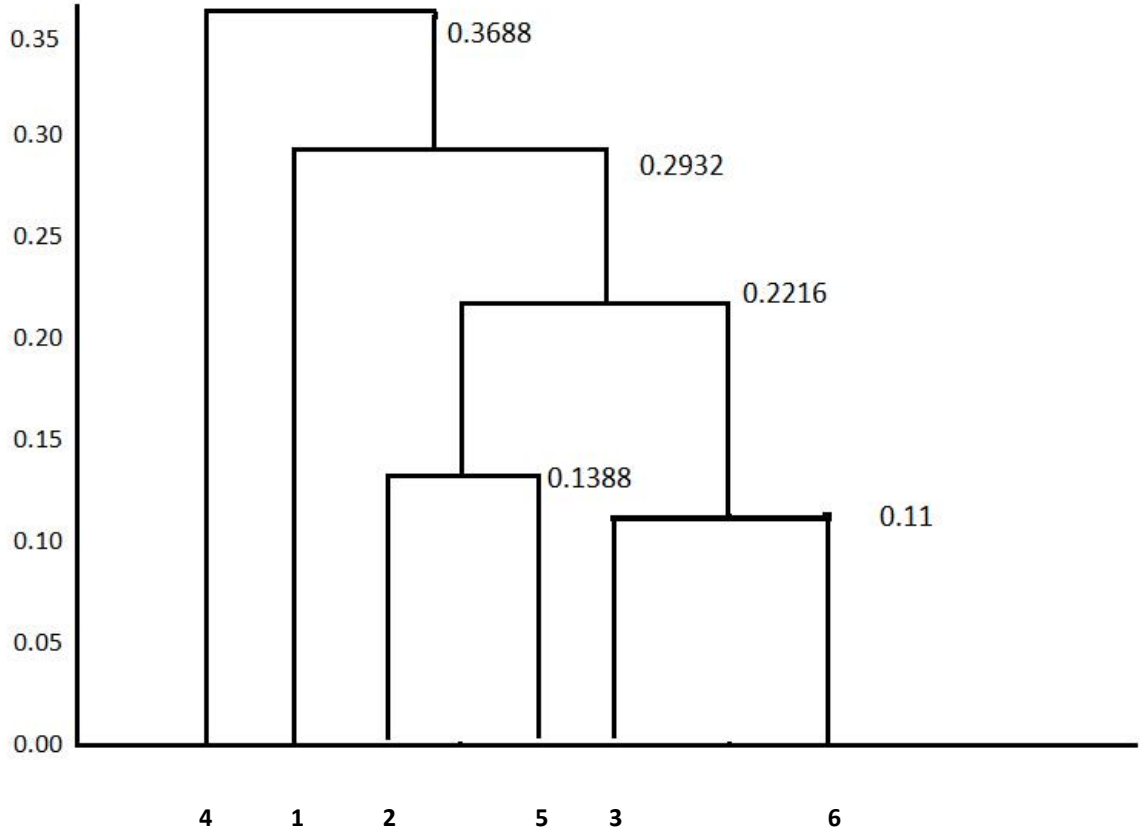
Complete Link: Distance between clusters is distance between farthest pair of points

Distance[(x,y),(a,b)] = sqrt((x-a)^2+(y-b)^2)

We are taking minimum distance value which is 0.11 and calculatin maximum distance between max[(p3,p6),p1], max[(p3,p6),p2], max[(p3,p6),p4], max[(p3,p6),p5] updating in the table. Then the minimum distance is 0.1388. and repeating same for next cluster

	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 and p6 forms first cluster						
	p1	**p2**	**p3,p6**	**p4**	**p5**	
p1	0	0.2357	0.2347	0.3688	0.3421	
p2	0.2357	0	0.254	0.2042	0.1388	
p3,p6	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 and p5 forms 2nd cluster						
	p1	**p2,p5**	**p3,p6**	**p4**		
p1	0	0.3421	0.2347	0.3688		
p2,p5	0.3421	0	0.3921	0.2932		
p3,p6	0.2347	0.3921	0	0.2216		
p4	0.3688	0.2932	0.2216	0		
smallest distance from above data is					0.2216	
so (p2,p5) and (p3,p6) forms 3rdcluster						
	p1	**p2,p5,p3,p6**	**p4**			
p1	0	0.3421	0.3688			
p2,p5,p3,p6	0.3421	0	0.2932			
p4	0.3688	0.2932	0			
smallest distance from above data is					0.2932	
so (**p2,p5,p3,p6)**and p1 forms 4thcluster						

	p1,p2,p5,p3,p6	p4				
p1,p2,p5,p3,p6	0	0.1483				
p4	0.3688	0				



jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [6]:
```
## 2) Use CC_GENERAL.csv given in the folder and apply
```

In [7]:
```
#importing all libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score

import warnings
warnings.filterwarnings("ignore")
```

In [8]:
```
dataframe = pd.read_csv('CC GENERAL.csv')
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
10   CASH_ADVANCE_FREQUENCY            8950 non-null   float64
```


jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [9]:
```
dataframe.head()
```

Out[9]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUEN
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083

In [10]:
```
dataframe.describe()
```

Out[10]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871112	0.490351
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877	0.401371
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000000	0.083333
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000	0.500000
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139	0.916667
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760	1.000000

In [11]:
```
df = dataframe.drop(['CUST_ID'], axis=1)
df.head()
```

Out[11]:

jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

Out[11]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEC
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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 [13]:
```
df.fillna(dataframe.mean(), inplace=True)
df.isnull().sum()
```


jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

Out[13]:
```
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                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [14]:
```
x = df.iloc[:,0:-1]
y = df.iloc[:,-1]


scaler = preprocessing.StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
X_scaled_df = pd.DataFrame(X_scaled_array, columns = x.columns)
```

In [15]:
```
#Normalization is the process of scaling individual samples to have unit norm.
#This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the simi
X_normalized = preprocessing.normalize(X_scaled_df)
# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalized)
```

jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [15]:
```
#Normalization is the process of scaling individual samples to have unit norm.
#This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the simil
X_normalized = preprocessing.normalize(X_scaled_df)
# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalized)
```

In [16]:
```
pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X_normalized)

principalDf = pd.DataFrame(data = principalComponents, columns = ['P1', 'P2'])

finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)
finalDf.head()
```

Out[16]:

	P1	P2	TENURE
0	-0.488186	-0.677233	12
1	-0.517294	0.556074	12
2	0.334384	0.287313	12
3	-0.486617	-0.080780	12
4	-0.562175	-0.474770	12

In [17]:
```
plt.figure(figsize=(7,7))
plt.scatter(finalDf['P1'],finalDf['P2'],c=finalDf['TENURE'],cmap='prism', s =5)
plt.xlabel('pc1')
plt.ylabel('pc2')
```

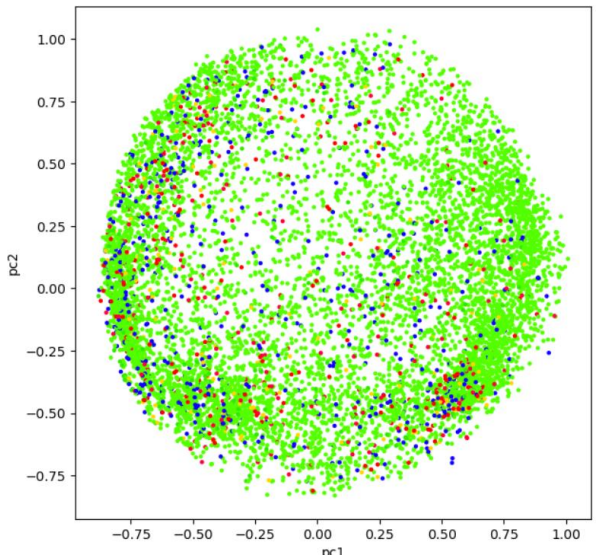
jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Out[17]: Text(0, 0.5, 'pc2')



jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

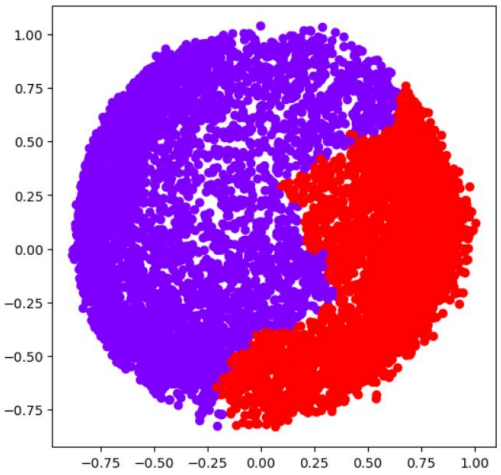
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [18]:
```
ac2 = AgglomerativeClustering(n_clusters = 2)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
           c = ac2.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```



jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

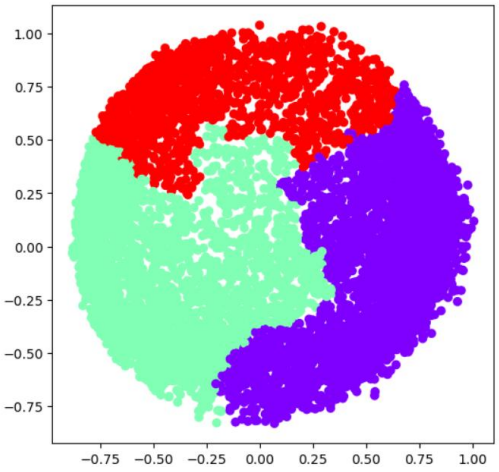
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [19]:
```
ac3 = AgglomerativeClustering(n_clusters = 3)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
           c = ac3.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```



jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

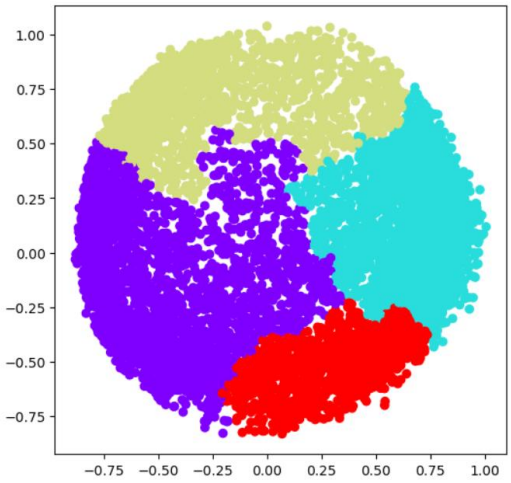
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [20]:
```
ac4 = AgglomerativeClustering(n_clusters = 4)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
           c = ac4.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```



jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

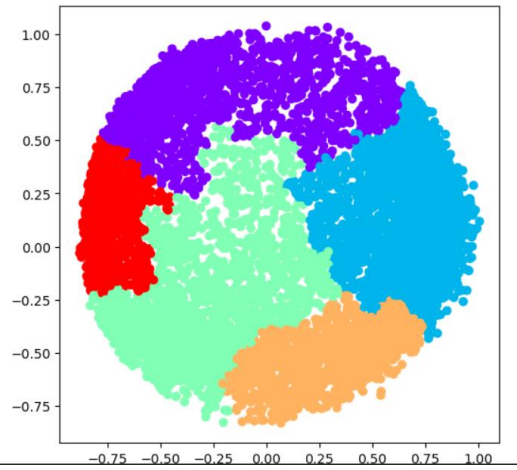
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [21]:
```
ac5 = AgglomerativeClustering(n_clusters = 5)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(principalDf['P1'], principalDf['P2'],
           c = ac5.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```



jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

In [22]:
```
k = [2, 3, 4, 5]

# Appending the silhouette scores of the different 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)))


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

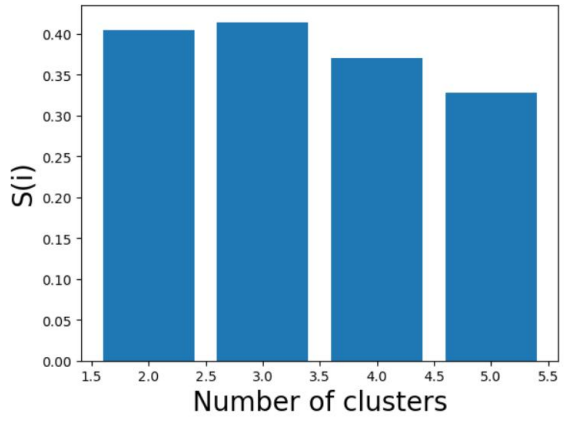
jupyter Assignment6 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code



GIT-HUB LINK:

nimmalapudisriram/ML_Assignment6 (github.com)