MACHINE LEARNING
ASSIGNMENT -06

Name :Thandarupalli Naveen Goud

UCM ID :nxt46830, 700# :700734683

Video Link: https://drive.google.com/file/d/1hMFfBpYE7_Zj5Do6Vs-d9qFKgBDHaU4c/view?usp=share_link

Github Link:https://github.com/nxt46830/ML-Assignment-1

Naveen Goud Thandarupalli
NXT46830, 700734683.

Question :- 1

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0	0.2357	0.2218	0.3698	0.3421	0.2347
P_2	0.2357	0	0.1483	0.2042	0.1388	0.2540
P_3	0.2218	0.1413	0	0.1513	0.2843	0.1100
P_4	0.3688	0.2042	0.1513	0	0.2932	0.2216
P_5	0.3421	0.1388	0.2843	0.2932	0	0.3921
P_6	0.2347	0.2540	0.1100	0.2216	0.3921	0

In single linkage the distance between two clusters is the minimum distance between the members of two clusters.

So, here P_3 & P_6 forms the first cluster.

	P_1	P_2	P_3P_6	P_4	P_5
P_1	0	0.2357	0.2218	0.3688	0.3427
P_2	0.2357	0	0.1483	0.2042	0.1388
P_3P_6	0.2218	0.1483	0	0.1513	0.2843
P_4	0.3688	0.2042	0.1513	0	0.2932
P_5	0.3421	0.1388	0.2843	0.2932	0

So, here P_2 & P_5 forms the second cluster.

	P_1	P_2P_5	P_3P_6	P_4
P_1	0	0.2357	0.2218	0.3688
P_2P_5	0.2357	0	0.1483	0.2042
P_3P_6	0.2218	0.1483	0	0.1513
P_4	0.3688	0.2042	0.1513	0

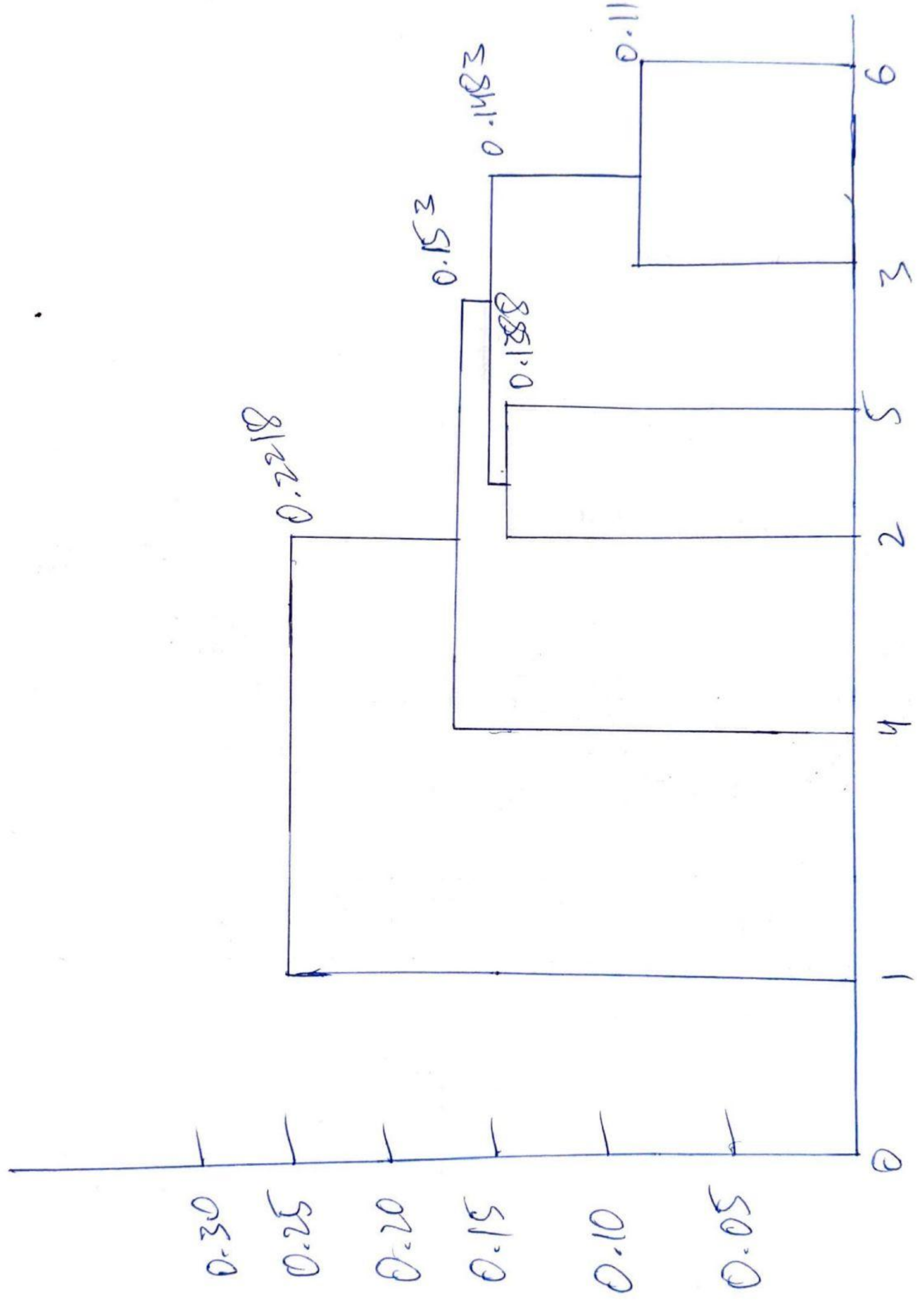
So, here P_2P_5 & P_3P_6 forms the third cluster.

	P_1	$P_2P_5P_3P_6$	P_4
P_1	0	0.2218	0.3688
$P_2P_5P_3P_6$	0.2218	0	0.1513
P_4	0.3688		0

So, here $P_2P_5P_3P_6$ & P_4 forms the fourth cluster.

	P_1	$P_2P_5P_3P_6P_4$
P_1	0	0.2218
$P_2P_5P_3P_6P_4$	0.2218	0

Single Link Proximity



	P1	P2	P3	P4	P5	P6
P1	0	0.2357	0.2218	0.3688	0.3421	0.2347
P2	0.2357	0	0.1483	0.2043	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

Complete Linkage is the maximum distance between the members of two clusters.
Here P3 & P6 forms the first cluster.

	P1	P2	P3P6	P4	P5
P1	0	0.2357	0.2347	0.3633	0.3421
P2	0.2357	0	0.254	0.2042	0.1388
P3P6	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

Here P2 & P5 form the second cluster.

	P_1	P_2P_5	P_3P_6	P_4
P_1	0			
P_2P_5	0.3421	0	0.3921	0.2932
P_3P_6	0.2347	0.3921	0	
P_4	0.3688	0.2932	0.2216	0

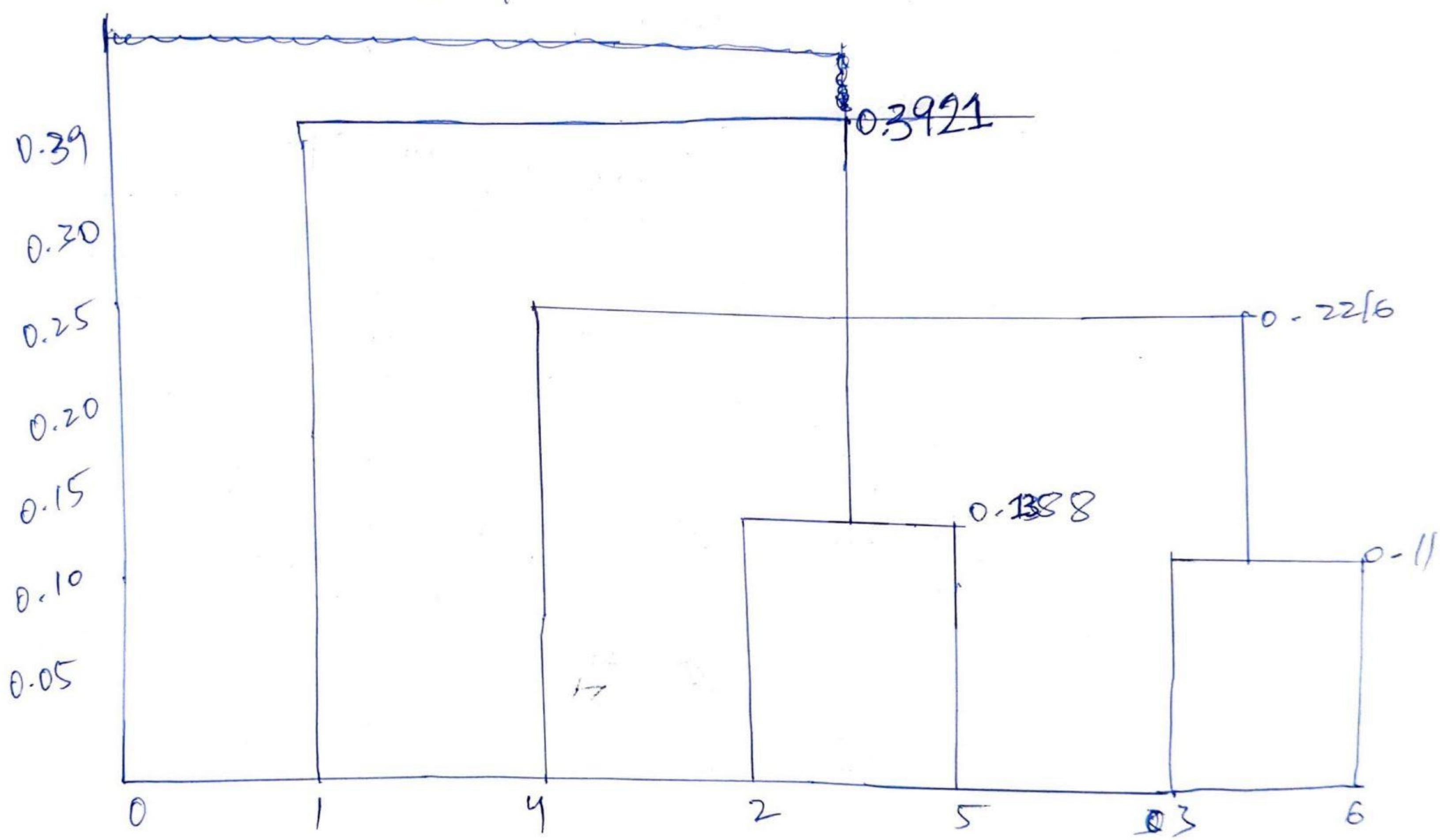
Here $P_3 P_6$ & P_4 forms third cluster.

	P_1	P_2P_5	$P_3P_6P_4$
P_1	0	0.3421	0.3688
P_2P_5	0.3421	0	0.3921
$P_3P_6P_4$	0.3688	0.3921	0

Here P_1 & P_2P_5 forms fourth cluster

	$P_1P_2P_5$	$P_3P_6P_4$
$P_1P_2P_5$	0	0.3928
$P_3P_6P_4$	0.3928	0

Complete link proximity.



In Average Link Proximity we use the average of the distance between members of two clusters.

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0	0.2357	0.2218	0.3688	0.3421	0.2347
P_2	0.2357	0	0.1483	0.2042	0.1388	0.254
P_3	0.2218	0.1483	0	0.1513	0.2843	0.11
P_4	0.3688	0.2042	0.1513	0	0.2932	0.2216
P_5	0.3421	0.1388	0.2843	0.2932	0	0.3921
P_6	0.2347	0.254	0.11	0.2216	0.3921	0

These P_3 & P_6 forms the first cluster.

	P_1	P_2	P_3P_6	P_4	P_5
P_1	0				
P_2	0.2357	0	0.20115	0.2042	
P_3P_6	0.22825	0.20115	0	0.18645	0.3382
P_4	0.3688	0.2042	0.18645	0	
P_5	0.3421	0.1388	0.3382		0

These P_2 & P_5 forms the second clusters.

	P_1	P_2P_5	P_3P_6	P_4
P_1	0			
P_2P_5	0.2889	0	0.269675	0.2487
P_3P_6	0.2282	0.26965	0	0.18645
P_4	0.3421	0.2487	0.18645	0

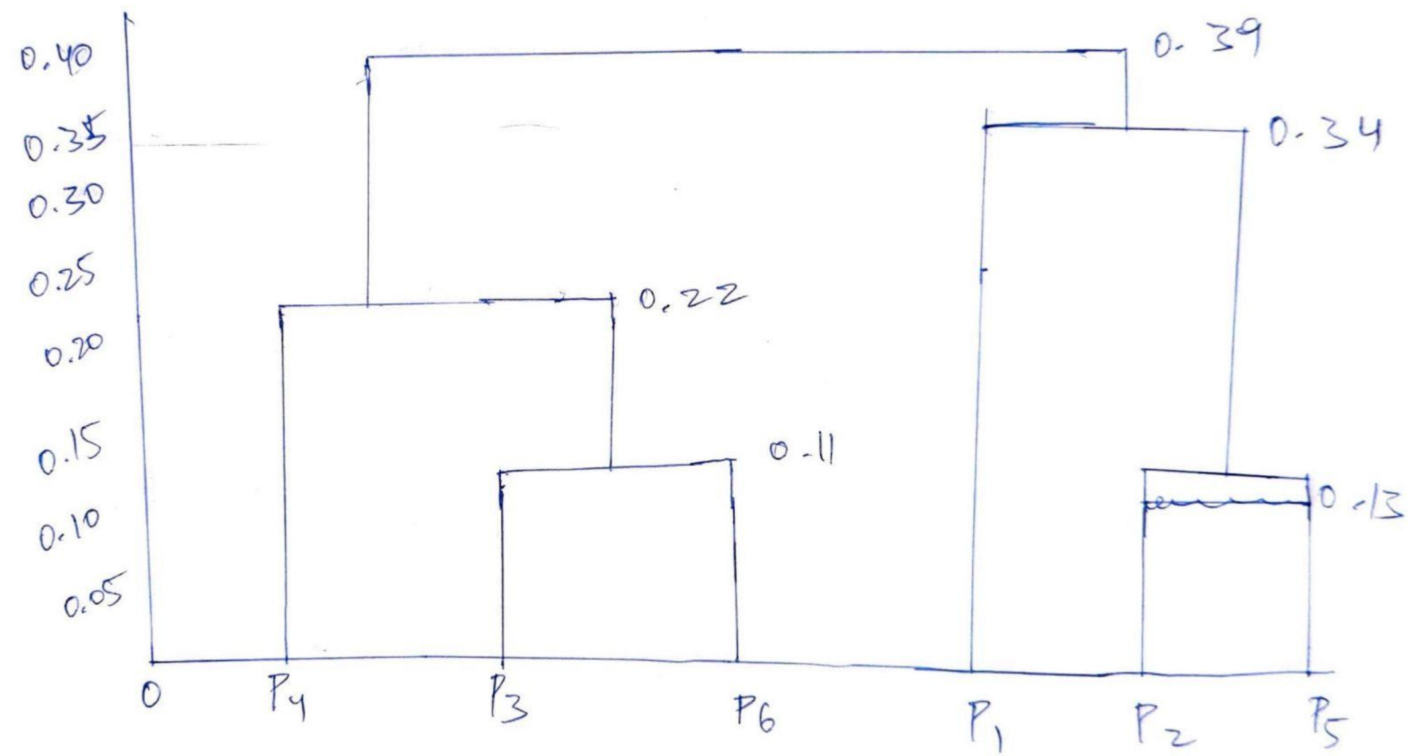
Here P_3P_6 & P_4 will form a cluster.

	P_1	P_2P_5	$P_3P_6P_4$
P_1	0		0.2815
P_2P_5	0.2889	0	0.291895
$P_3P_6P_4$	0.2813	0.259185	0

Here P_2P_5 & $P_3P_6P_4$ forms a cluster

	P_1	$P_2P_5P_3P_6P_4$
P_1	0	0.285
$P_2P_5P_3P_6P_4$	0.285	0

Complete Linkage



5

```
## 2) Use CC_GENERAL.csv given in the folder and apply:
# a) Preprocess the data by removing the categorical column and filling the missi
# b) Apply StandardScaler() and normalize() functions to scale and normalize raw
# 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 vi
# result for each k value using scatter plot.
# e) Evaluate different variations using Silhouette Scores and Visualize results
```

```
#importing all libraries here for assignment
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")
```

```
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):
 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
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

```
dataframe.head()
```

	CUST_ID object	BALANCE float64	BALANCE_FREQ...	PURCHASES floa...	ONEOFF_PURC
0	C10001	40.900749	0.818182	95.4	
1	C10002	3202.467416	0.909091	0.0	
2	C10003	2495.148862	1.0	773.17	7
3	C10004	1666.670542	0.636364	1499.0	1.
4	C10005	817.714335	1.0	16.0	

5 rows, showing 10 per page Page 1 of 1

```
dataframe.describe()
```

	BALANCE float64	BALANCE_FREQ...	PURCHASES floa...	ONEOFF_PURC...	INSTALLMENTS
count	8950.0	8950.0	8950.0	8950.0	8
mean	1564.4748276781006	0.8772707255865921	1003.2048335195531	592.4373709497207	411.067644
std	2081.5318794565546	0.23690400268476855	2136.6347818728887	1659.887917437811	904.338115
min	0.0	0.0	0.0	0.0	
25%	128.2819155	0.888889	39.635	0.0	
50%	873.385231	1.0	361.28	38.0	
75%	2054.1400355	1.0	1110.13	577.405	468.

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

	BALANCE float64	BALANCE_FREQ...	PURCHASES floa...	ONEOFF_PURC...	INSTALLMENT:
0	40.900749	0.818182	95.4	0.0	
1	3202.467416	0.909091	0.0	0.0	
2	2495.148862	1.0	773.17	773.17	
3	1666.670542	0.636364	1499.0	1499.0	
4	817.714335	1.0	16.0	16.0	

Visualize

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

```
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
```

```python
df.fillna(dataframe.mean(), inplace=True)
df.isnull().any()
```

```
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                        False
PAYMENTS                            False
MINIMUM_PAYMENTS                    False
PRC_FULL_PAYMENT                    False
TENURE                              False
```

```python
df.corr().style.background_gradient(cmap="Greens")
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES
BALANCE	1.000000	0.322412	0.181261
BALANCE_FREQUENCY	0.322412	1.000000	0.133674
PURCHASES	0.181261	0.133674	1.000000
ONEOFF_PURCHASES	0.164350	0.104323	0.916845
INSTALLMENTS_PURCHASES	0.126469	0.124292	0.679896
CASH_ADVANCE	0.496692	0.099388	-0.051474
PURCHASES_FREQUENCY	-0.077944	0.229715	0.393017
ONEOFF_PURCHASES_FREQUENCY	0.073166	0.202415	0.498430
PURCHASES_INSTALLMENTS_FREQUENCY	-0.063186	0.176079	0.315567
CASH_ADVANCE_FREQUENCY	0.449218	0.191873	-0.120143
CASH_ADVANCE_TRX	0.385152	0.141555	-0.067175

```
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)
```

```
#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-p
X_normalized = preprocessing.normalize(X_scaled_df)
# Converting the numpy array into a pandas DataFrame
X_normalized = pd.DataFrame(X_normalized)
```

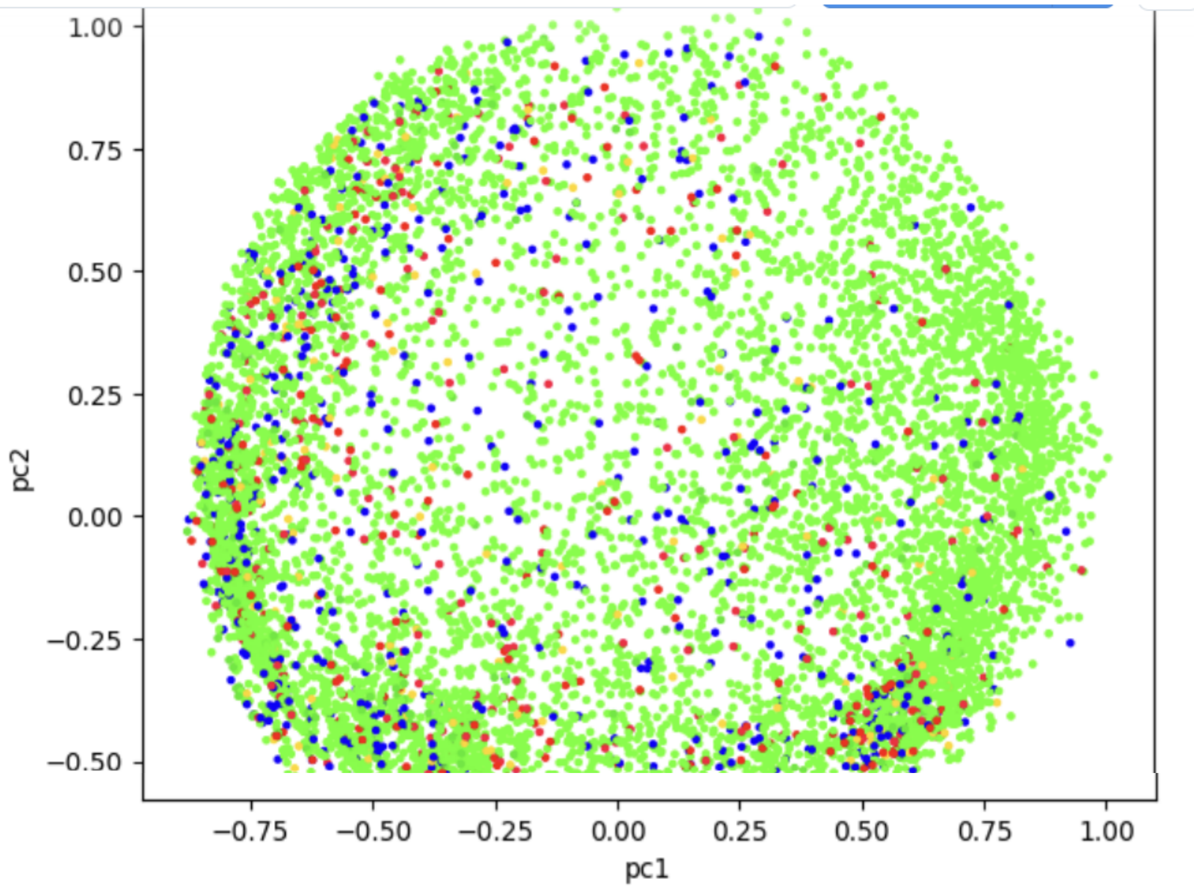
```
pca2 = PCA(n_components=2)
principalComponents = pca2.fit_transform(X_normalized)

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

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

Visualize

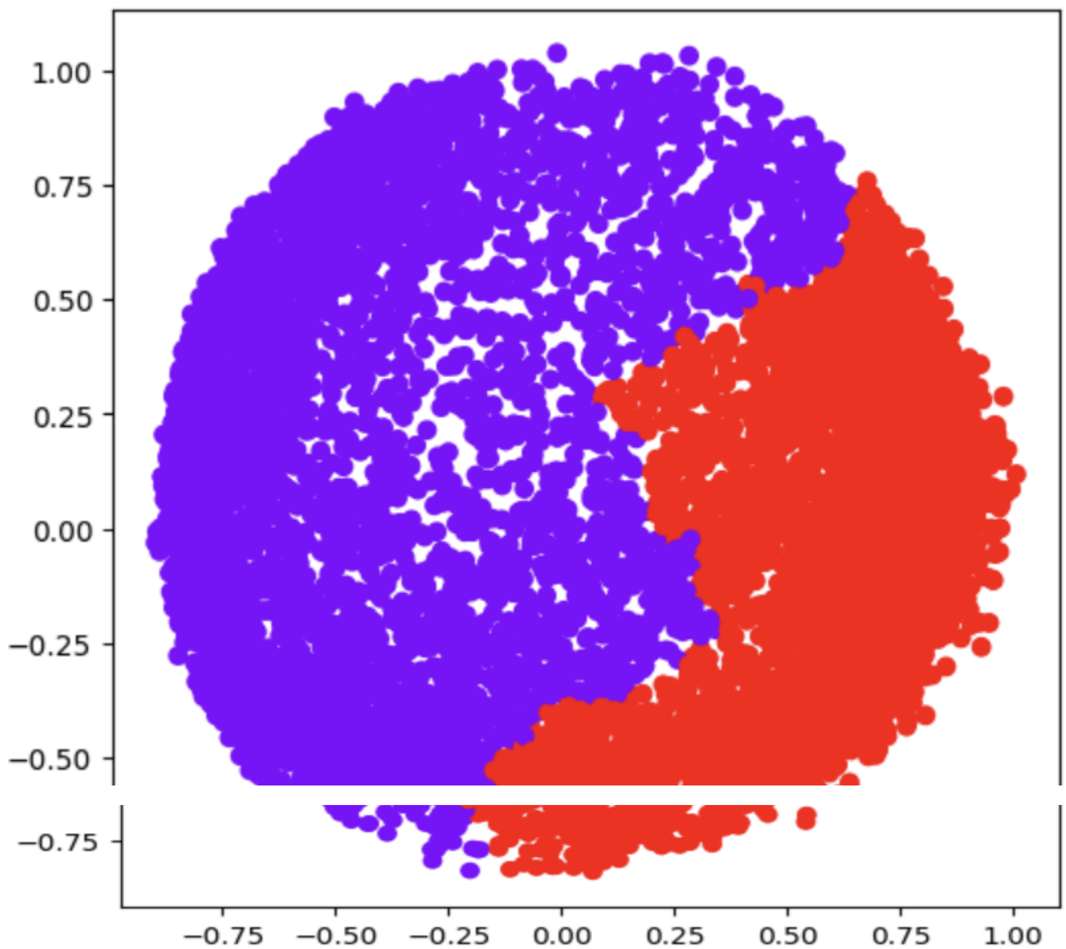
	P1 float64	P2 float64	TENURE int64
0	-0.48818609086980813	-0.67723317016894	12
1	-0.5172943272612286	0.5560740969628902	12
2	0.3343842270171463	0.28731279359066886	12
3	-0.4866163782550769	-0.08078196874324063	12

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

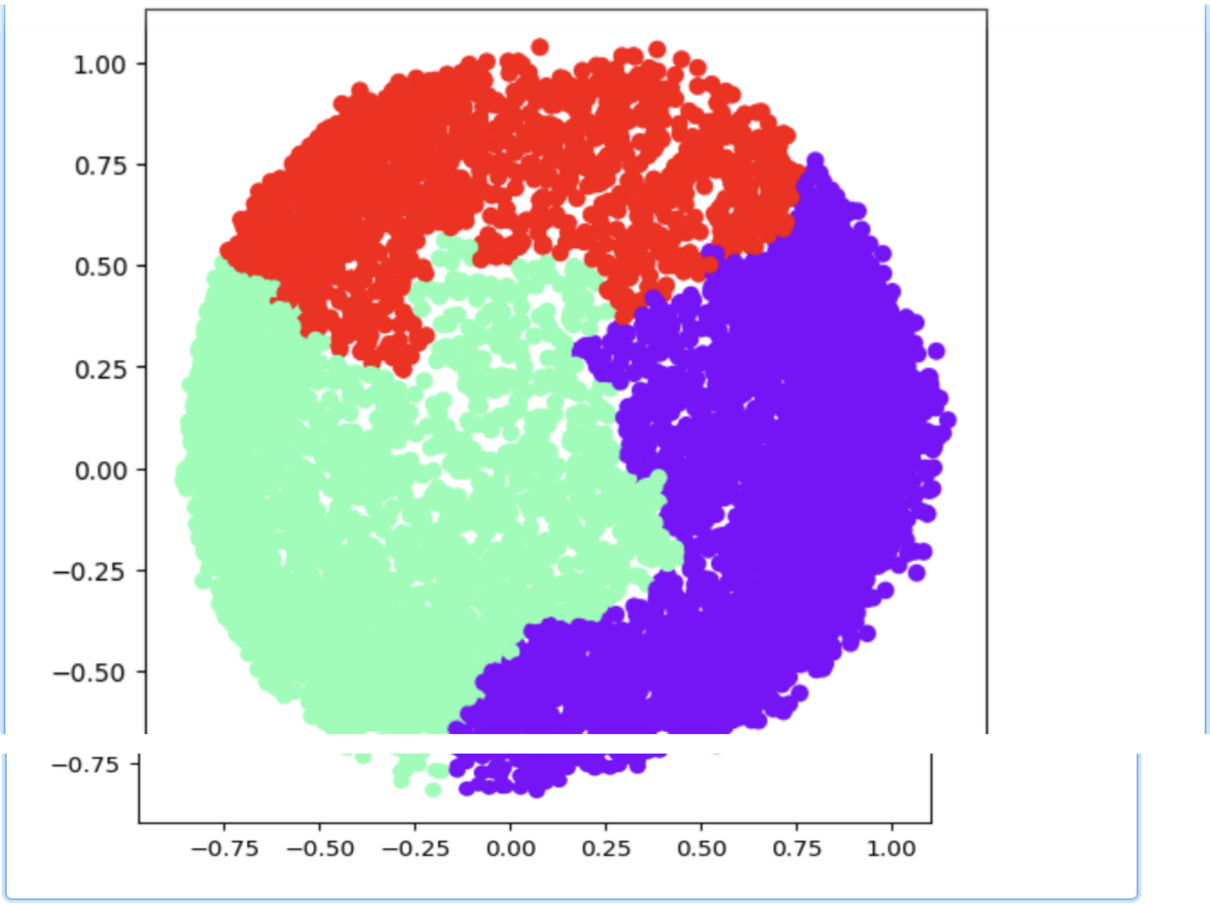
```
Text(0, 0.5, 'pc2')
```



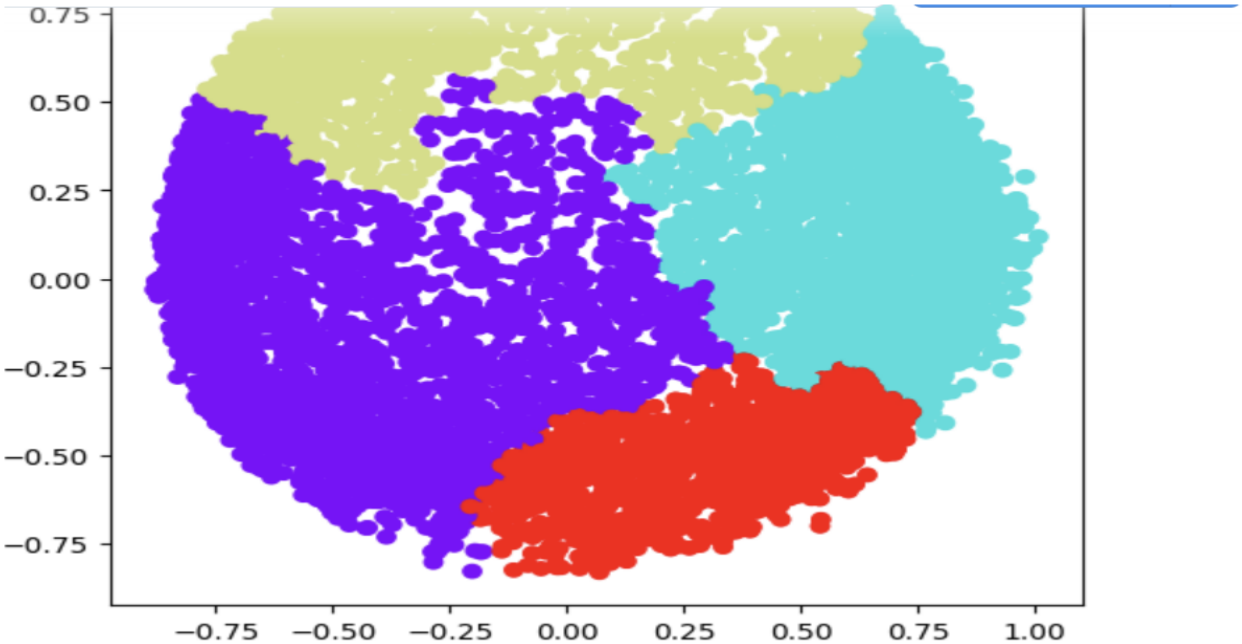

```
ac2 = AgglomerativeClustering(n_clusters = 2)

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



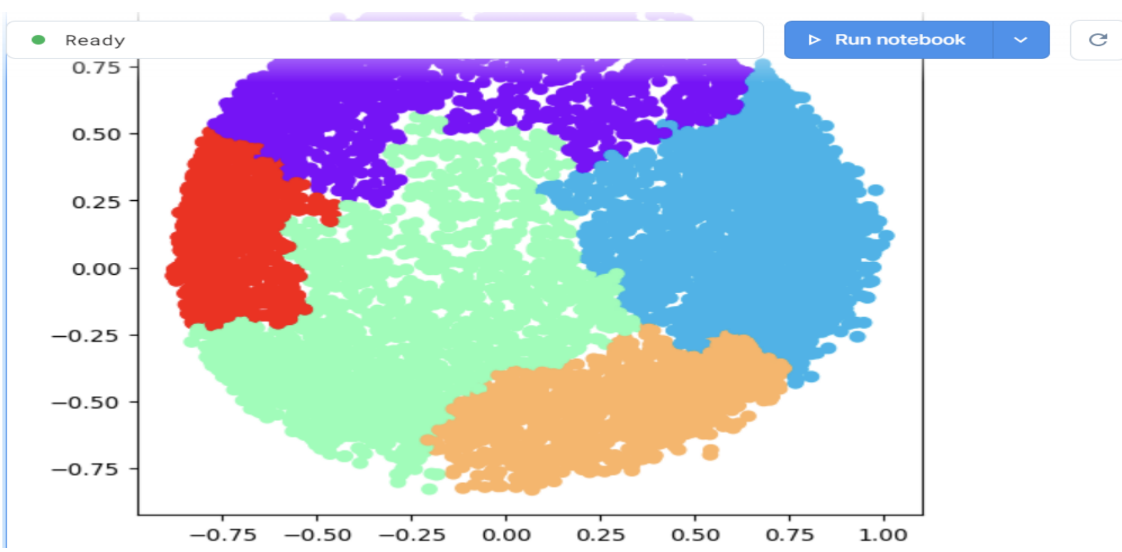

```
ac3 = AgglomerativeClustering(n_clusters = 3)

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




```
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()
```




```
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()
```



```python
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()
```