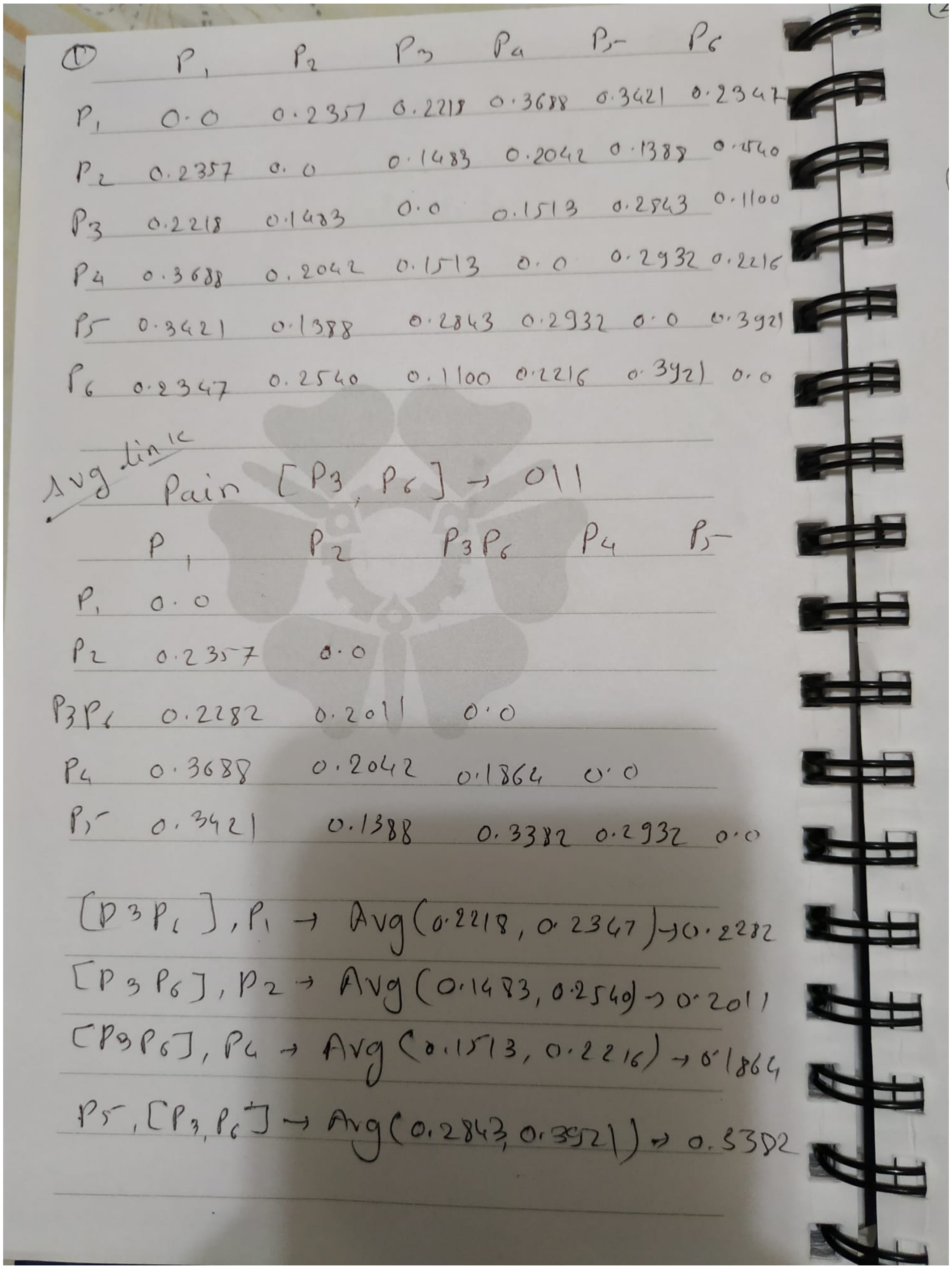
Assignment 6 ML

Pallavi Meher

700727681

Question 1:



①	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0.0	0.2357	0.2218	0.3688	0.3421	0.2347
P_2	0.2357	0.0	0.1483	0.2042	0.1388	0.2540
P_3	0.2218	0.1483	0.0	0.1513	0.2843	0.1100
P_4	0.3688	0.2042	0.1513	0.0	0.2932	0.2216
P_5	0.3421	0.1388	0.2843	0.2932	0.0	0.3921
P_6	0.2347	0.2540	0.1100	0.2216	0.3921	0.0

Avg link

Pair $[P_3, P_6] \rightarrow$ 011

	P_1	P_2	P_3P_6	P_4	P_5
P_1	0.0				
P_2	0.2357	0.0			
P_3P_6	0.2282	0.2011	0.0		
P_4	0.3688	0.2042	0.1864	0.0	
P_5	0.3421	0.1388	0.3382	0.2932	0.0

$[P_3P_6], P_1 \rightarrow \text{Avg}(0.2218, 0.2347) \rightarrow 0.2282$

$[P_3P_6], P_2 \rightarrow \text{Avg}(0.1483, 0.2540) \rightarrow 0.2011$

$[P_3P_6], P_4 \rightarrow \text{Avg}(0.1513, 0.2216) \rightarrow 0.1864$

$P_5, [P_3, P_6] \rightarrow \text{Avg}(0.2843, 0.3921) \rightarrow 0.3382$

② $[P_2, P_5] \rightarrow 0.1388$

	P_1	$P_5 P_2$	$P_3 P_6$	P_4
P_1	0.0			
$P_2 P_5$	0.2889	0.0		
$P_3 P_6$	0.2282	0.2946	0.0	
P_4	0.3688	0.2487	0.1864	0.0

$[P_2 P_5], P_1 \rightarrow \text{Avg} = (0.2357, 0.3421) \rightarrow 0.2889$

$[P_3 P_6][P_2 P_5] \rightarrow \text{Avg} = (0.2011, 0.3382) \rightarrow 0.2946$

$P_4 [P_5 P_2] \rightarrow \text{Avg} = (0.2042, 0.2932) \rightarrow 0.2487$

$[P_4 \; P_3 P_6] \rightarrow 0.1864$

	P_1	$P_5 P_2$	$P_3 P_4 P_6$
P_1	0.0		
$P_2 P_5$	0.2889	0.0	
$P_3 P_4 P_6$	0.2958	0.2591	0.0

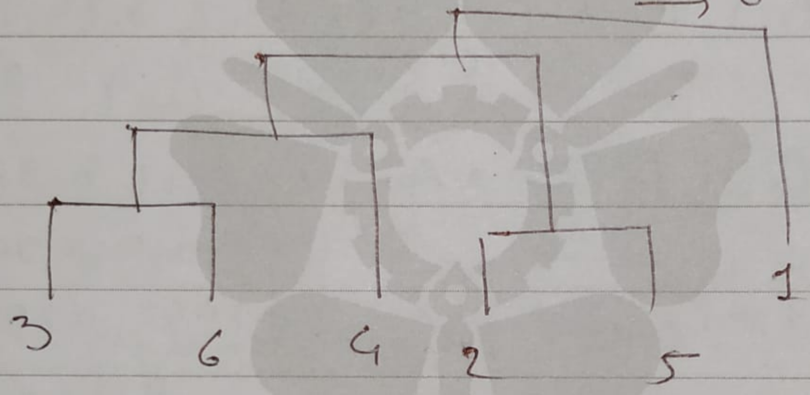
$P_1 [P_3 P_4 P_6] \rightarrow \text{Avg} = (0.2282, 0.3688) = 0.2988$

$[P_3 P_4 P_6][P_5 P_2] \rightarrow \text{Avg} = (0.2946, 0.2487) = 0.291$

③ [$P_5 P_2$] [$P_3 P_4 P_6$] → 0.2591

	P_1	$P_2 P_3 P_4 P_5 P_6$
P_1	0.0	
$P_2 P_3 P_4 P_5 P_6$	0.2923	0.0

P_1 [$P_2 P_3 P_4 P_5 P_6$] → Avg(0.2889, 0.2958)
→ 0.2923.



MAX - complete

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0.0					
P_2	0.2357	0.0				
P_3	0.2218	0.1483	0.0			
P_4	0.3688	0.2042	0.1513	0.0		
P_5	0.3421	0.1388	0.2843	0.2932	0.0	
P_6	0.2347	0.2540	0.1100	0.2216	0.3921	0.0

(4) Pair [P_3 P_6] → 0·1100

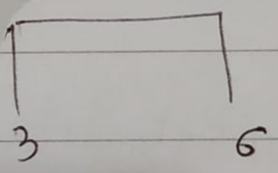
	P_1	P_2	P_3P_6	P_4	P_5
P_1	0·0				
P_2	0·2357	0·0			
P_3P_6	0·2347	0·2540	0·0		
P_4	0·3688	0·2042	0·2216	0·0	
P_5	0·3421	0·1388	0·3921	0·2932	0·0

P_1[P_3 P_6] → MAX(0·2218, 0·2347) = 0·2347

P_2[P_3 P_6] → MAX(0·1483, 0·2540) = 0·2540

P_4[P_3 P_6] → MAX(0·1513, 0·2216) = 0·2216

P_5[P_3P_6] → MAX(0·2843, 0·3921) = 0·3921



Pair [P_2 P_5]

	P_1	P_2P_5	P_3P_6	P_4
P_1	0·0			
P_2P_5	0·3421	0·0		
P_3P_6	0·2347	0·3921	0·0	
P_4	0·3688	0·2932	0·2216	0·0

⑧

$P_1 [P_2 P_5] \rightarrow MAX[0.2357, 0.3421) = 0.3421$

$(P_3 P_6)(P_2 P_5) \rightarrow MAX(0.2540, 0.3921) = 0.3921$

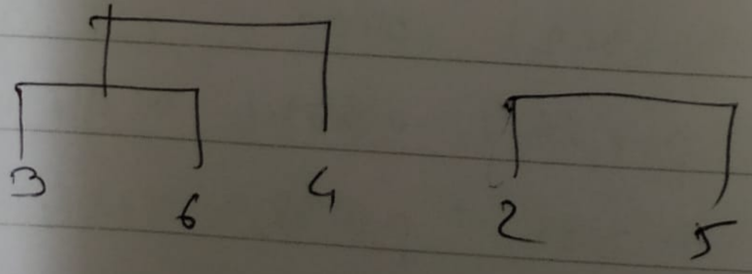
$P_4 (P_2 P_5] \rightarrow MAX(0.2042, 0.1388) = 0.2932$

$Pair (P_4, (P_3, P_6)) \rightarrow 0.2216$

	P_1	$P_2 P_5$	$P_3 P_6 P_4$
P_1	0.0		
$P_2 P_5$	0.3421	0.0	
$P_3 P_6 P_4$	0.3688	0.3921	0.0

$P_1 [P_4 P_3 P_6] \rightarrow MAX(0.3688, 0.2347) = 0.3688$

$(P_2 P_5)[P_4 P_3 P_6] \rightarrow MAX(0.2932, 0.3921) = 0.3921$



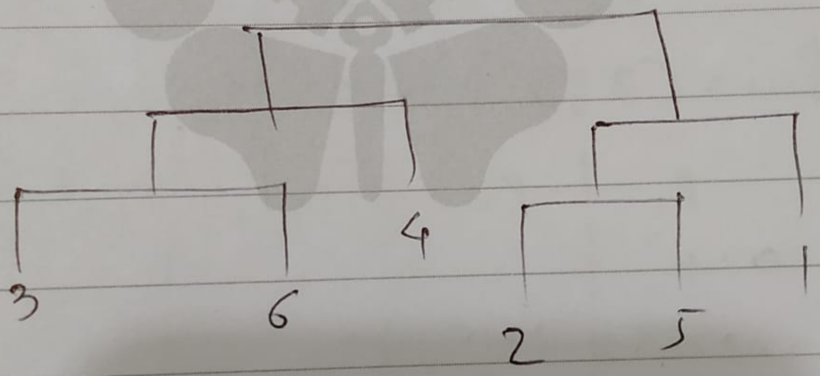
(6) $P_1[P_2 P_5] \rightarrow 0.3421$

$P_1[P_2 P_5] \rightarrow 0.3421$

	$P_1 P_2 P_5$	$P_3 P_6 P_4$
$P_1 P_2 P_5$	0.0	
$P_3 P_6 P_4$	0.3921	0.0

$[P_1 P_2 P_5][P_3 P_6 P_4] \rightarrow \text{MAX}(0.3688, 0.3421)$

$= 0.3921$



Single link

7

Single link

	P1	P2	P3	P4	P5	P6
P1	0.0					
P2	0.2357	0.0				
P3	0.2218	0.1483	0.0			
P4	0.3688	0.2042	0.1513	0.0		
P5	0.3421	0.1388	0.2843	0.2932	0.0	
P6	0.2347	0.2540	0.1100	0.2216	0.3921	0.0

Pair (3, 6) → 0.1100

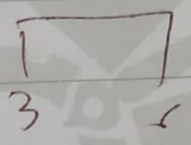
	P1	P2	P3P6	P4	P5
P1	0.0				
P2	0.2357	0.0			
P3P6	0.2218	0.1483	0.0		
P4	0.3688	0.2042	0.1513	0.0	
P5	0.3421	0.1388	0.2843	0.2932	0.0

$P_1[P_3P_6] \rightarrow \min(0.2218, 0.2347) \rightarrow$ 0.2218

$P_2[P_3P_6] \rightarrow \min(0.1483, 0.2540) \rightarrow$ 0.1483

$P_4[P_3P_6] \rightarrow \min(0.1513, 0.2216) \rightarrow 0.1513$

$P_5[P_3P_6] \rightarrow \min(0.2843, 0.3921) \rightarrow$ 0.2843



$[P_2P_5] \rightarrow 0.1388$

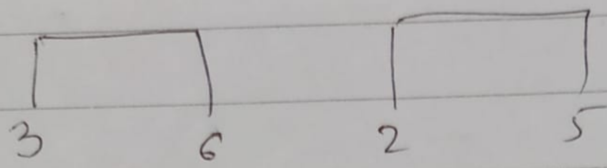
	P_1	P_2P_5	P_3P_6	P_5
P_1	0.0			
P_2P_5	0.2357	0.0		
P_3P_6	0.2218	0.1483	0.0	
P_4	0.3684	0.2042	0.1513	0.0

$P_1[P_2P_5] \rightarrow \min[0.2357, 0.3421) \rightarrow 0.2357$

$(P_3P_6)[P_2P_5] \rightarrow \min(0.1483, 0.2843) \rightarrow$ 0.1483

$P_4[P_2P_5] \rightarrow \min(0.2042, 0.2932) \rightarrow 0.2042$

9



Pair [P3P6] [P2P5] = 0.1483

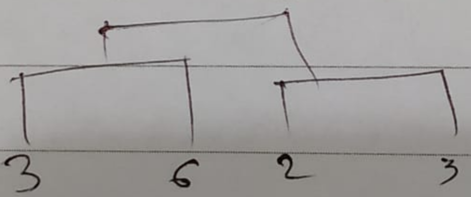
	P1	P2P5-P3P6	P4
P1	0.0		
P2P5P3P6	0.2218	0.0	
P4	0.3688	0.1513	0.0

P1 [P2 P5-P3P6] → min(0.2357, 0.2218)

= 0.2218

P4[P2P5-P3P6] → min (0.2042, 0.1511)

= 0.1513



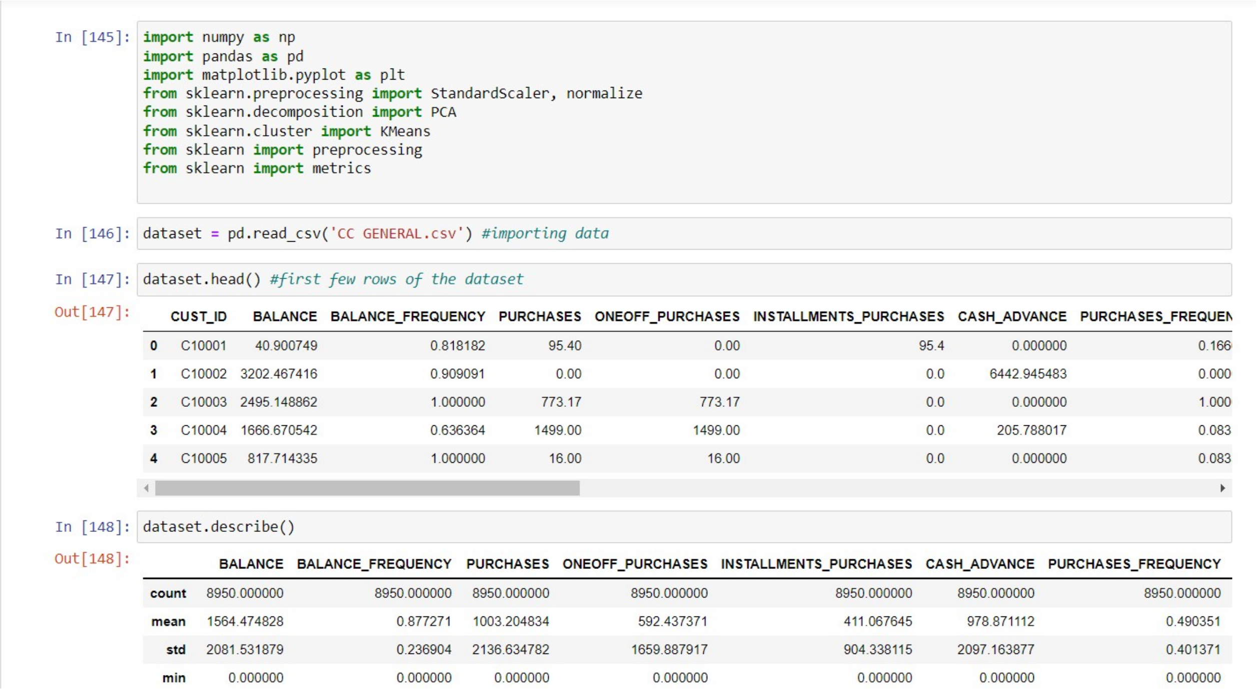
Pair [P2 P5P3 P6] P4 = 0.1513

	P1	P2P5-P3P6 P4
P1	0.0	
P2P5P3P6P4	0.2218	0.0

P1[P2P5P3P6P4] = min(0.2218, 0.3688)

= 0.2218

Question 2:



In [145]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn import metrics
```

In [146]:
```
dataset = pd.read_csv('CC GENERAL.csv') #importing data
```

In [147]:
```
dataset.head() #first few rows of the dataset
```

Out[147]:

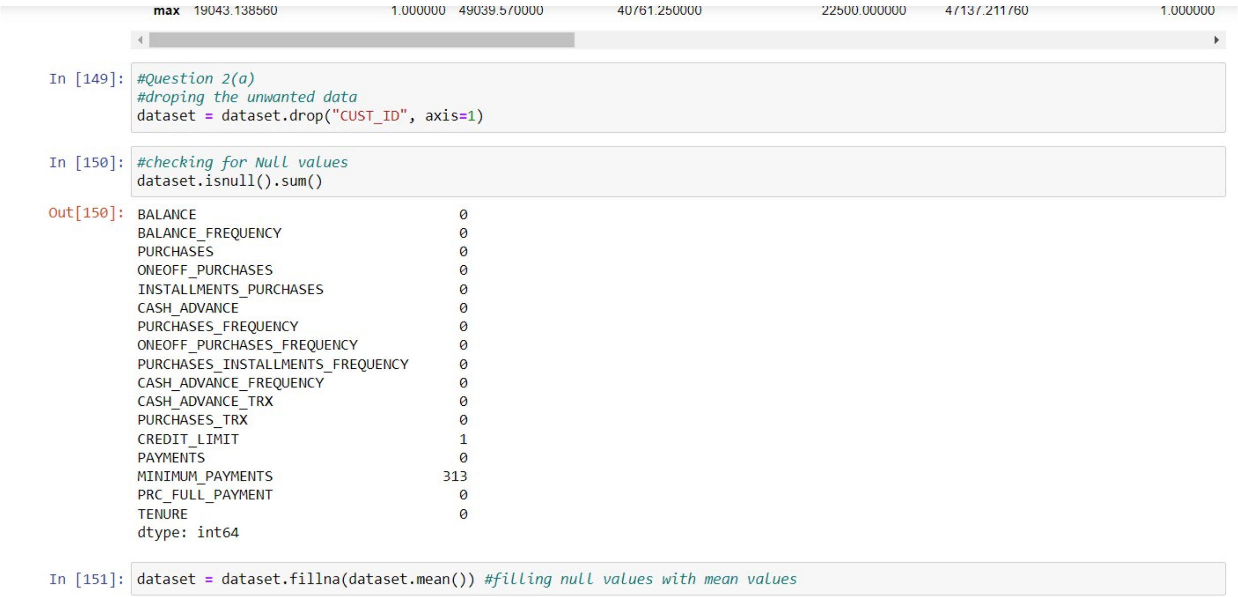
	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUEN
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000
3	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 [148]:
```
dataset.describe()
```

Out[148]:

	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

First, I have imported the required libraries. Then imported the dataset ‘CC GENERAL.csv’. Dataset is also displayed using head () function and there is description of the dataset.



```
max   19043.138560        1.000000   49039.570000        40761.250000        22500.000000    47137.211760        1.000000
```

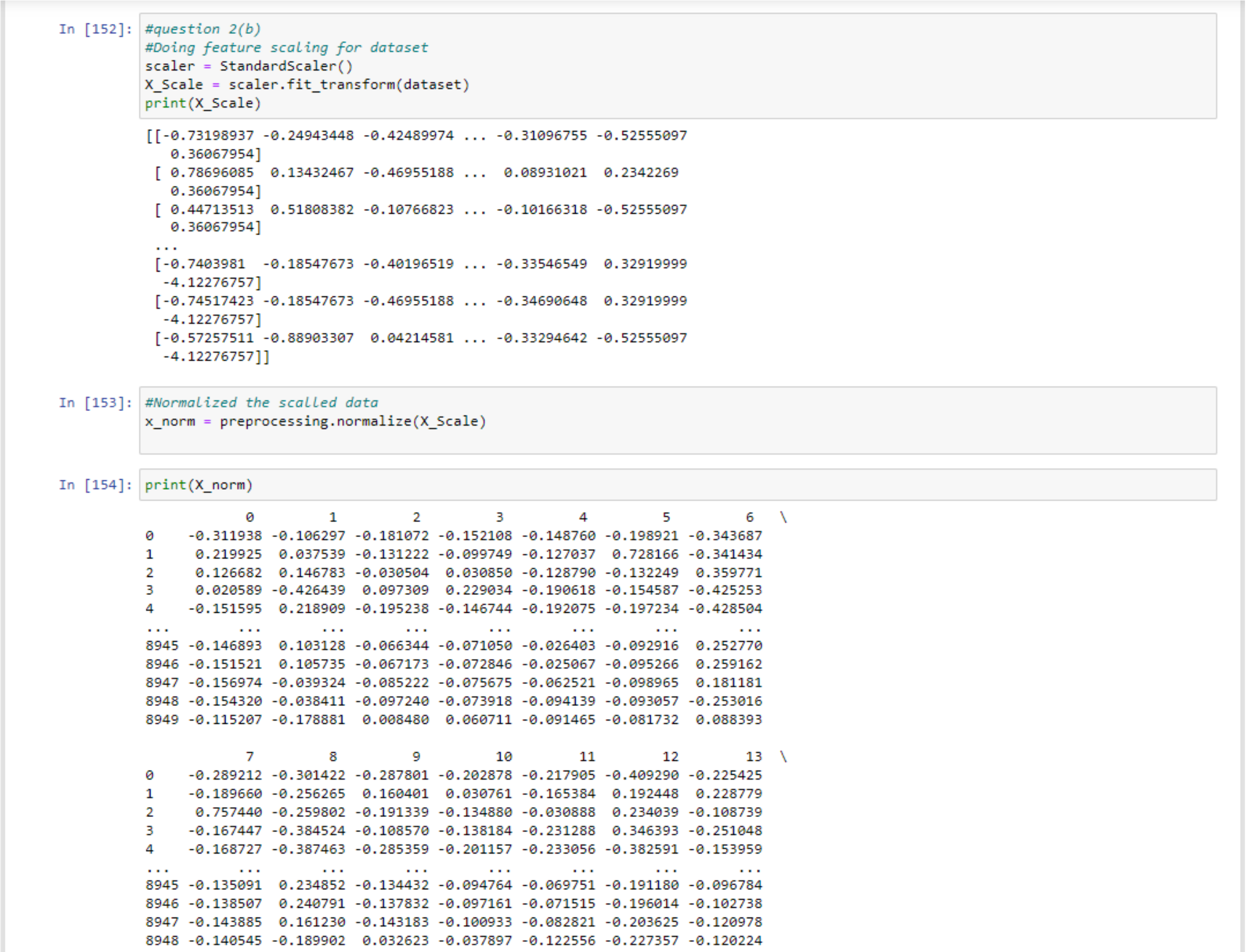
In [149]:
```
#Question 2(a)
#droping the unwanted data
dataset = dataset.drop("CUST_ID", axis=1)
```

In [150]:
```
#checking for Null values
dataset.isnull().sum()
```

Out[150]:
```
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
```

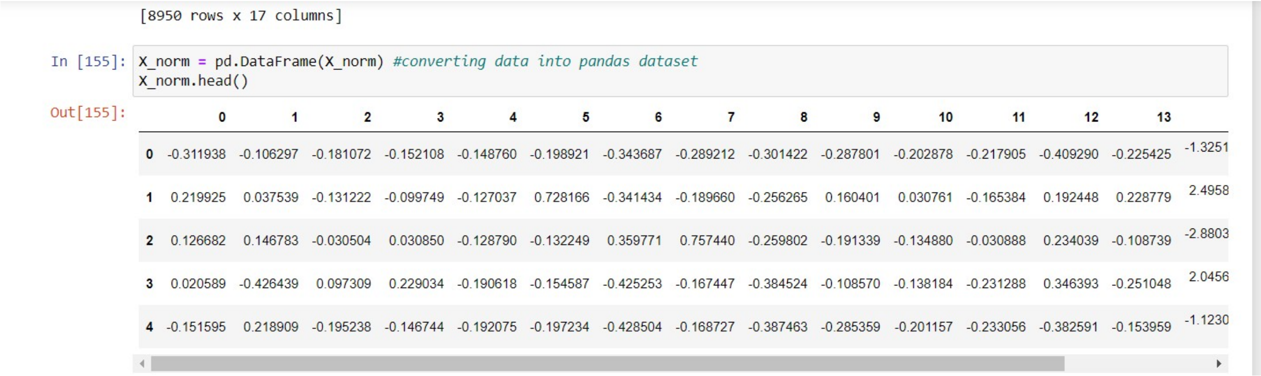
In [151]:
```
dataset = dataset.fillna(dataset.mean()) #filling null values with mean values
```

For Question 2(a), I have deleted the first column which is ‘CUST_ID’. I have checked for the null values in the dataset there are 2 attributes with the null values. I have used the mean values to fill the null values of those two attributes.



```
In [152]: #question 2(b)
          #Doing feature scaling for dataset
          scaler = StandardScaler()
          X_Scale = scaler.fit_transform(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 [153]: #Normalized the scalled data
          x_norm = preprocessing.normalize(X_Scale)

In [154]: print(X_norm)

             0         1         2         3         4         5         6  \
0    -0.311938 -0.106297 -0.181072 -0.152108 -0.148760 -0.198921 -0.343687
1     0.219925  0.037539 -0.131222 -0.099749 -0.127037  0.728166 -0.341434
2     0.126682  0.146783 -0.030504  0.030850 -0.128790 -0.132249  0.359771
3     0.020589 -0.426439  0.097309  0.229034 -0.190618 -0.154587 -0.425253
4    -0.151595  0.218909 -0.195238 -0.146744 -0.192075 -0.197234 -0.428504
...        ...       ...       ...       ...       ...       ...       ...
8945 -0.146893  0.103128 -0.066344 -0.071050 -0.026403 -0.092916  0.252770
8946 -0.151521  0.105735 -0.067173 -0.072846 -0.025067 -0.095266  0.259162
8947 -0.156974 -0.039324 -0.085222 -0.075675 -0.062521 -0.098965  0.181181
8948 -0.154320 -0.038411 -0.097240 -0.073918 -0.094139 -0.093057 -0.253016
8949 -0.115207 -0.178881  0.008480  0.060711 -0.091465 -0.081732  0.088393

             7         8         9        10        11        12        13  \
0    -0.289212 -0.301422 -0.287801 -0.202878 -0.217905 -0.409290 -0.225425
1    -0.189660 -0.256265  0.160401  0.030761 -0.165384  0.192448  0.228779
2     0.757440 -0.259802 -0.191339 -0.134880 -0.030888  0.234039 -0.108739
3    -0.167447 -0.384524 -0.108570 -0.138184 -0.231288  0.346393 -0.251048
4    -0.168727 -0.387463 -0.285359 -0.201157 -0.233056 -0.382591 -0.153959
...        ...       ...       ...       ...       ...       ...       ...
8945 -0.135091  0.234852 -0.134432 -0.094764 -0.069751 -0.191180 -0.096784
8946 -0.138507  0.240791 -0.137832 -0.097161 -0.071515 -0.196014 -0.102738
8947 -0.143885  0.161230 -0.143183 -0.100933 -0.082821 -0.203625 -0.120978
8948 -0.140545 -0.189902  0.032623 -0.037897 -0.122556 -0.227357 -0.120224
```

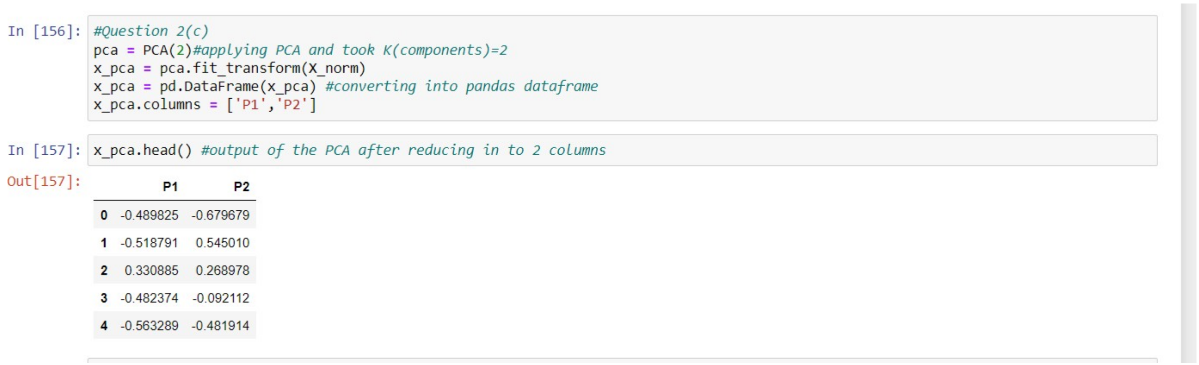
For question 2(b), first I have applied the standard scaler. And then I have normalized the data using normalize () function. In above screenshot I have displayed the dataset after the standard scaler and after normalizing to see how dataset changes.



```
[8950 rows x 17 columns]

In [155]: X_norm = pd.DataFrame(X_norm) #converting data into pandas dataset
          X_norm.head()

Out[155]:
```

	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	-1.3251
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	2.4958
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	-2.8803
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	2.0456
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	-1.1230

After applying normalizing we get array as an output so I have converted the array into panda dataframe and displayed the dataset named ‘x_norm’.



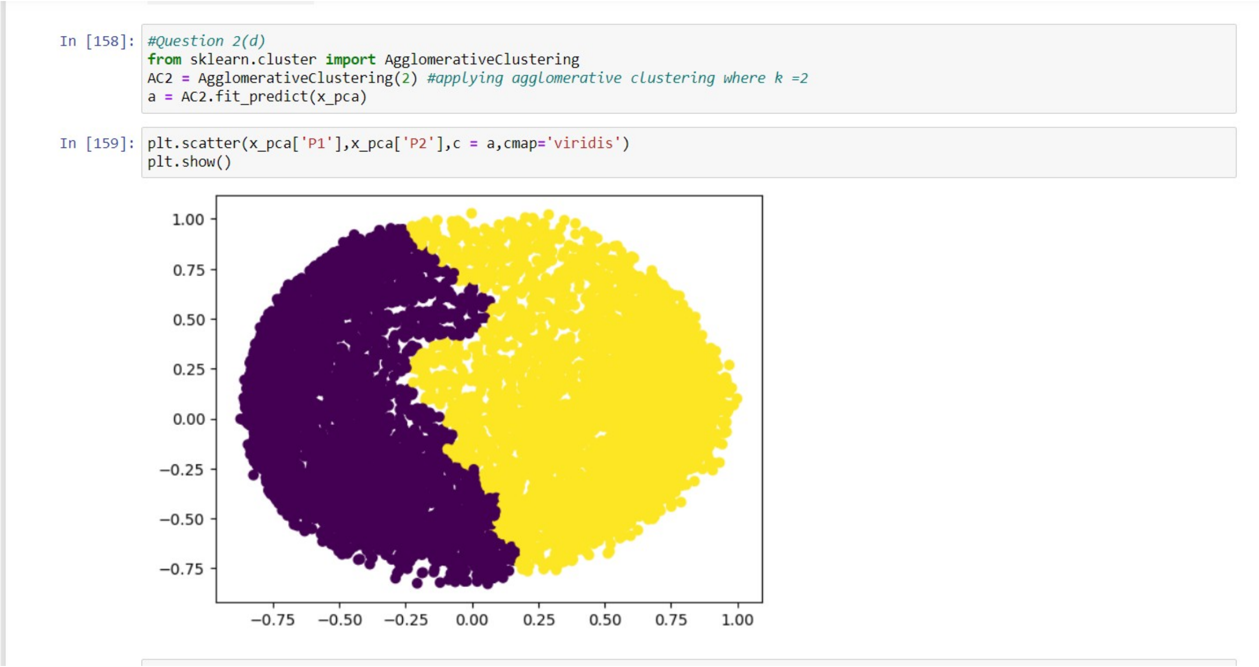
In [156]:
```
#Question 2(c)
pca = PCA(2)#applying PCA and took K(components)=2
x_pca = pca.fit_transform(X_norm)
x_pca = pd.DataFrame(x_pca) #converting into pandas dataframe
x_pca.columns = ['P1','P2']
```

In [157]:
```
x_pca.head() #output of the PCA after reducing in to 2 columns
```

Out[157]:

	P1	P2
0	-0.489825	-0.679679
1	-0.518791	0.545010
2	0.330885	0.268978
3	-0.482374	-0.092112
4	-0.563289	-0.481914

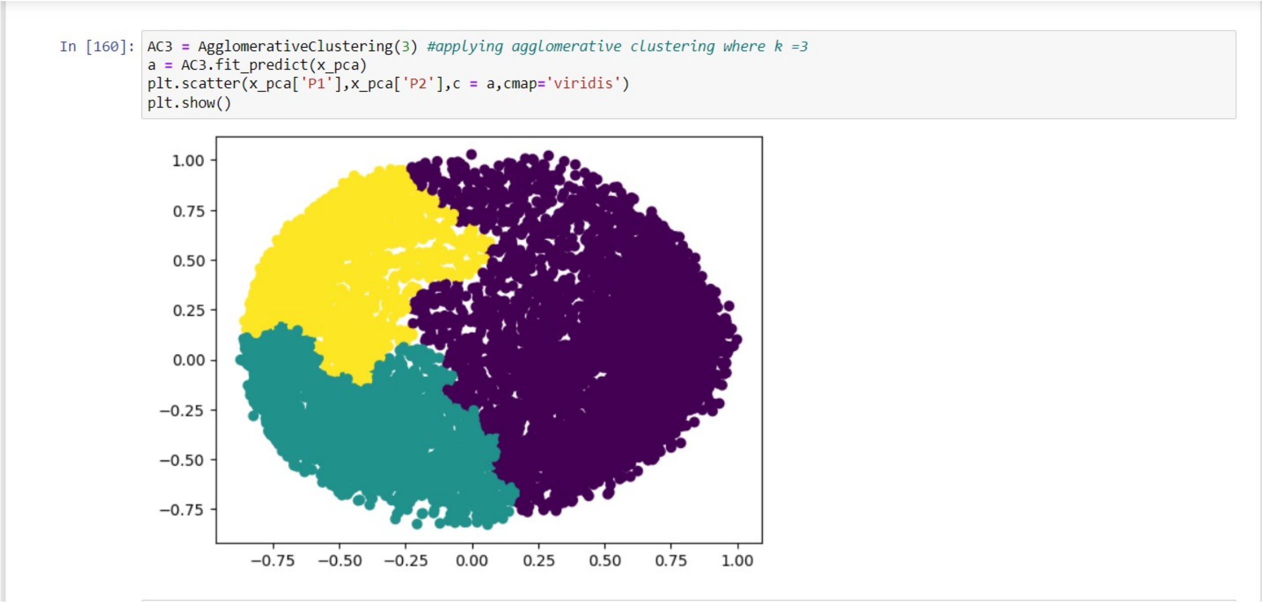
For Question 2(c), I have implemented PCA where I taken k = 2. So, the dataset x_norm has been transformed into array. I have again transform the array into panda dataframe which has 2 column named ‘P1’,’P2’ and the name of the dataset is x_pca. It is displayed in the screenshot.



In [158]:
```
#Question 2(d)
from sklearn.cluster import AgglomerativeClustering
AC2 = AgglomerativeClustering(2) #applying agglomerative clustering where k =2
a = AC2.fit_predict(x_pca)
```

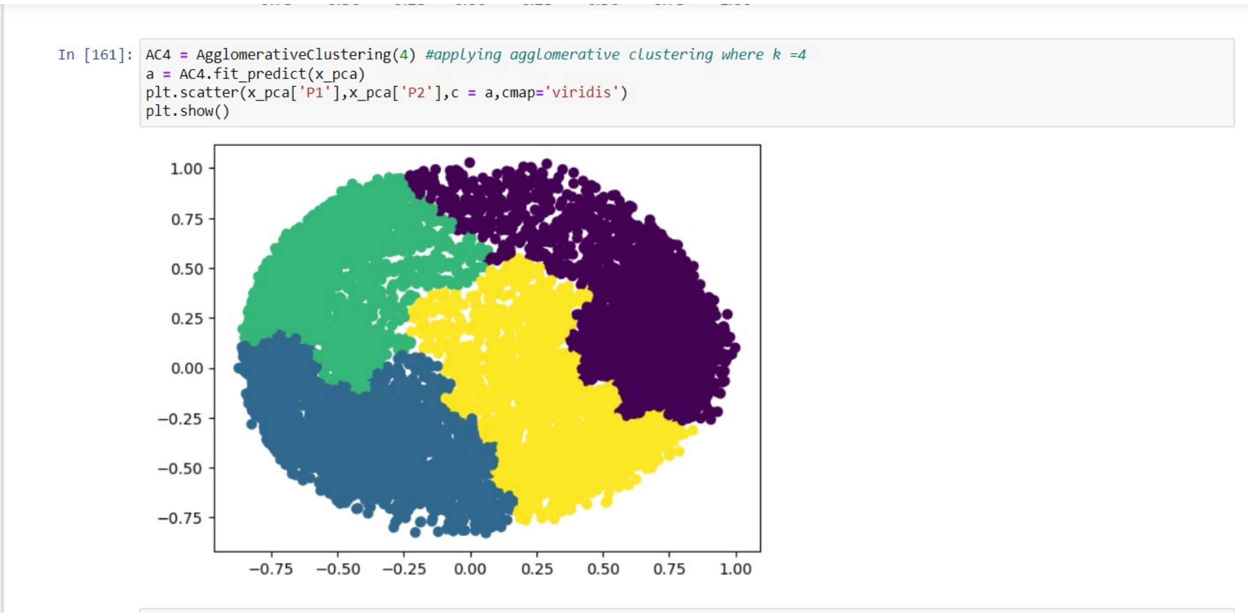
In [159]:
```
plt.scatter(x_pca['P1'],x_pca['P2'],c = a,cmap='viridis')
plt.show()
```

For question 2(d), I have implemented agglomerative clustering using sklearn library. Where the number of clusters is 2. Also, the output has been displayed using the scatterplot. 2 different cluster has been displayed using two different colors.



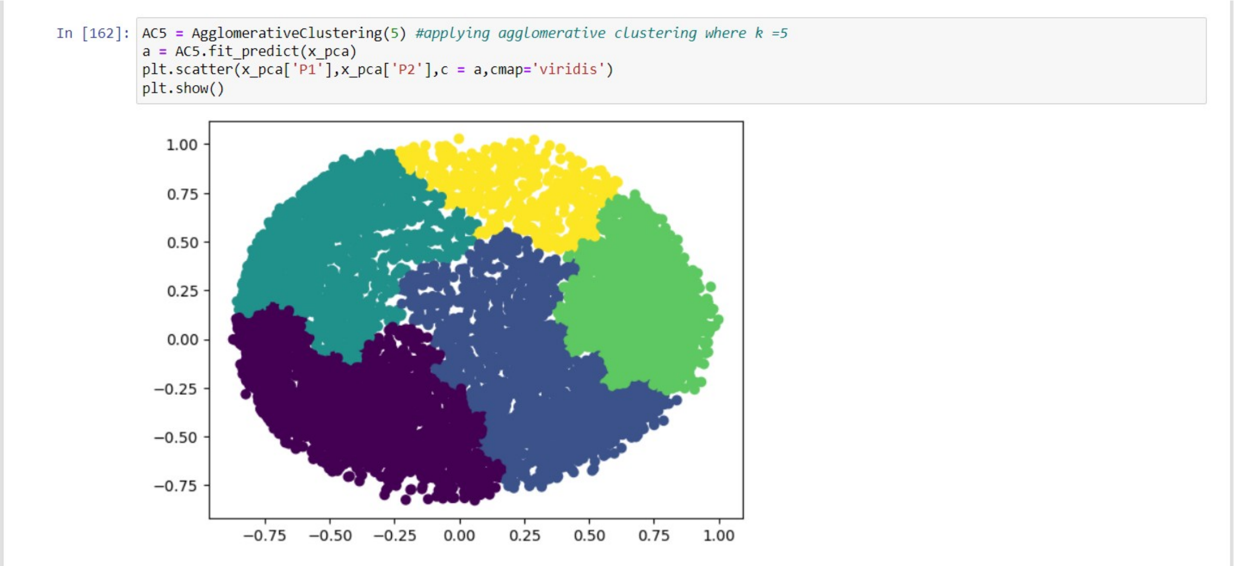
```
In [160]: AC3 = AgglomerativeClustering(3) #applying agglomerative clustering where k =3
          a = AC3.fit_predict(x_pca)
          plt.scatter(x_pca['P1'],x_pca['P2'],c = a,cmap='viridis')
          plt.show()
```

Above I have implemented the agglomerative cluster where number of clusters is 3. Three different colors represent three clusters.



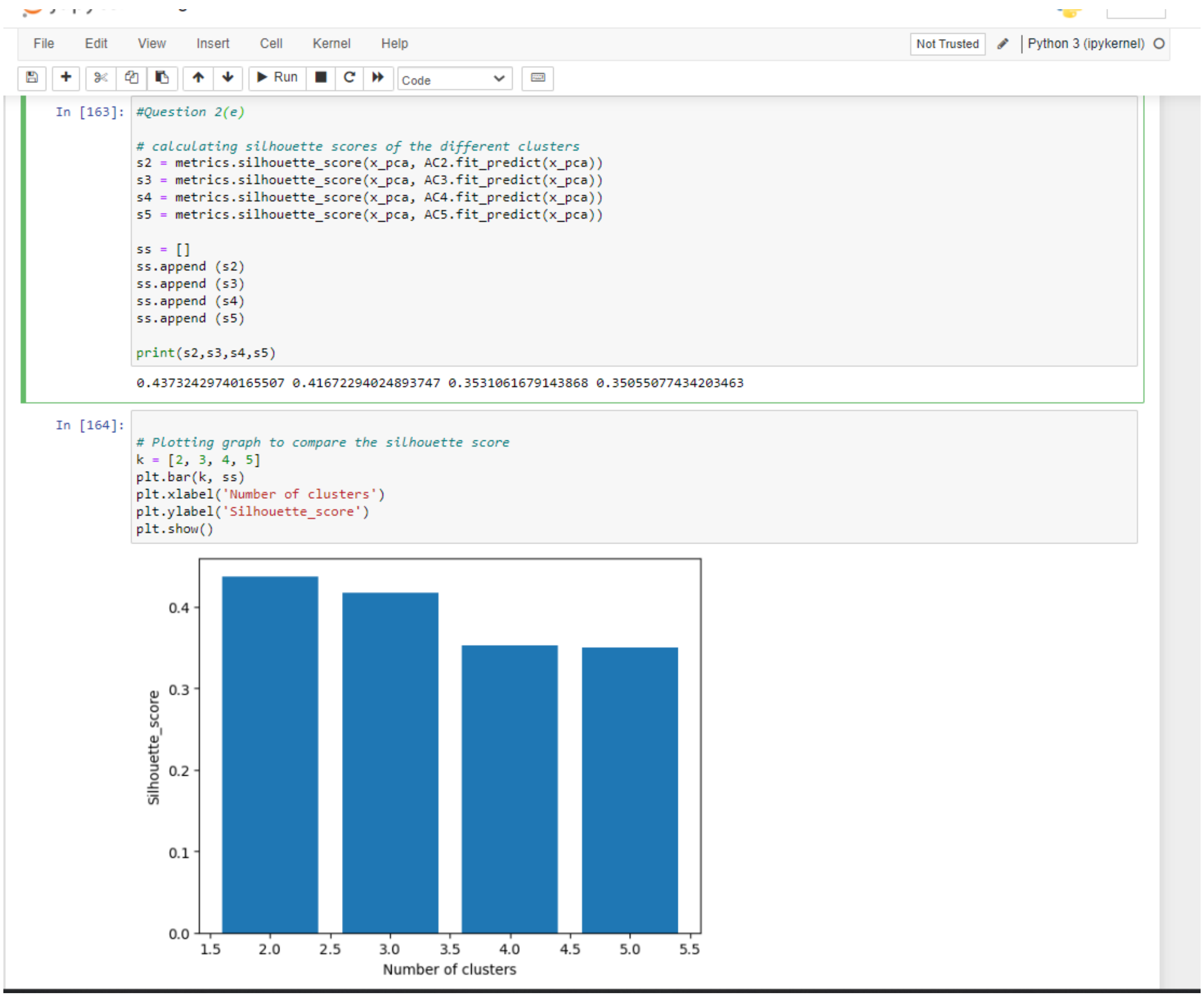
```
In [161]: AC4 = AgglomerativeClustering(4) #applying agglomerative clustering where k =4
          a = AC4.fit_predict(x_pca)
          plt.scatter(x_pca['P1'],x_pca['P2'],c = a,cmap='viridis')
          plt.show()
```

Here is the implementation of the agglomerative cluster with number of the cluster 4.



```
In [162]: AC5 = AgglomerativeClustering(5) #applying agglomerative clustering where k =5
          a = AC5.fit_predict(x_pca)
          plt.scatter(x_pca['P1'],x_pca['P2'],c = a,cmap='viridis')
          plt.show()
```

Above is the implementation of agglomerative cluster where number of cluster is 5.



```
#Question 2(e)

# calculating silhouette scores of the different clusters
s2 = metrics.silhouette_score(x_pca, AC2.fit_predict(x_pca))
s3 = metrics.silhouette_score(x_pca, AC3.fit_predict(x_pca))
s4 = metrics.silhouette_score(x_pca, AC4.fit_predict(x_pca))
s5 = metrics.silhouette_score(x_pca, AC5.fit_predict(x_pca))

ss = []
ss.append (s2)
ss.append (s3)
ss.append (s4)
ss.append (s5)

print(s2,s3,s4,s5)
```

```
0.43732429740165507 0.41672294024893747 0.3531061679143868 0.35055077434203463
```

```
# Plotting graph to compare the silhouette score
k = [2, 3, 4, 5]
plt.bar(k, ss)
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette_score')
plt.show()
```

For question 2(e), first I have calculated the silhouette score for all clusters model named “S2,S3,S4,S5” and added to the list named “ss”.

I have used the bar graph to represent the silhouette score of each model. In bar graph y-axis represent the silhouette score and x-axis represent cluster models.

Appendix:

Video Link: https://drive.google.com/drive/folders/1PamTY5iQkH8F-eKRbP199xw1w9zDCH5N?usp=sharing

GitHub:https://github.com/pallavi234/ML_Assignment6/tree/main