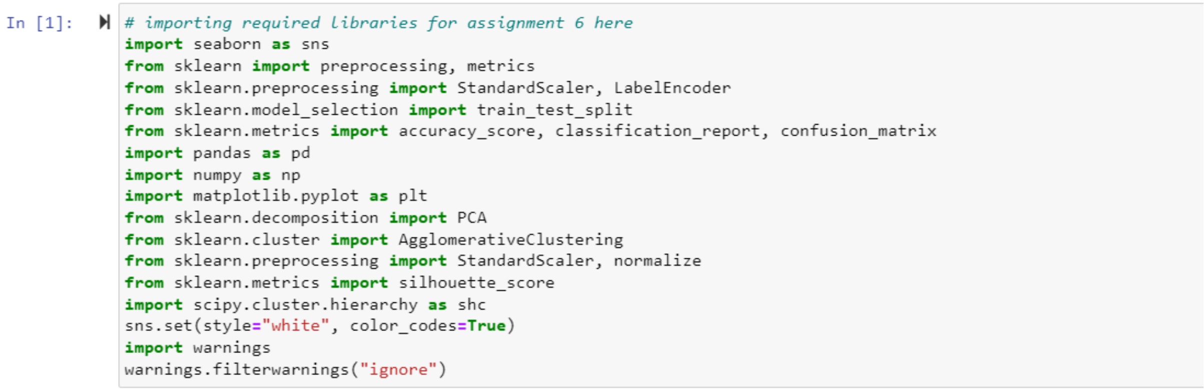
Machine Learning_Assignment6

Git Hub: https://github.com/AishaFar/ML_Assignmnet6_Farhana_700735341
Name: Ayesha Farhana
Id: 700735341



In [1]:

```
# importing required libraries for assignment 6 here
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")
```

importing required libraries for assignment 6 here

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)

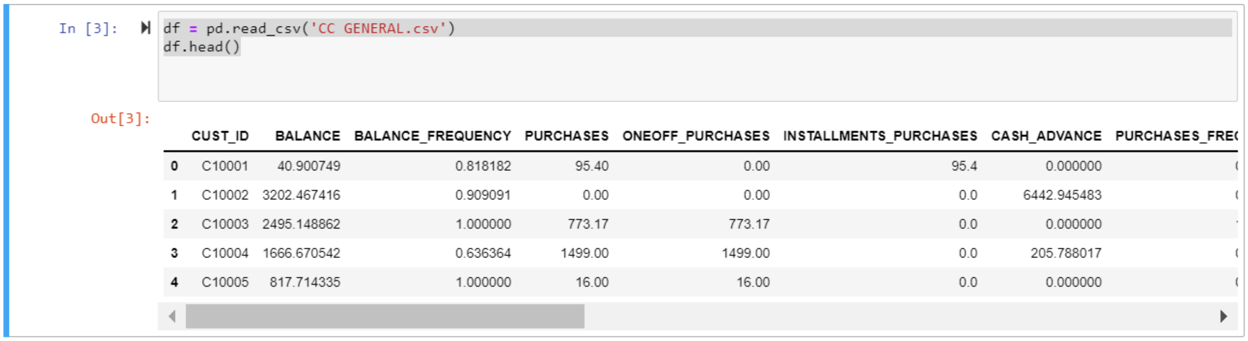
import warnings

warnings.filterwarnings("ignore")

---

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

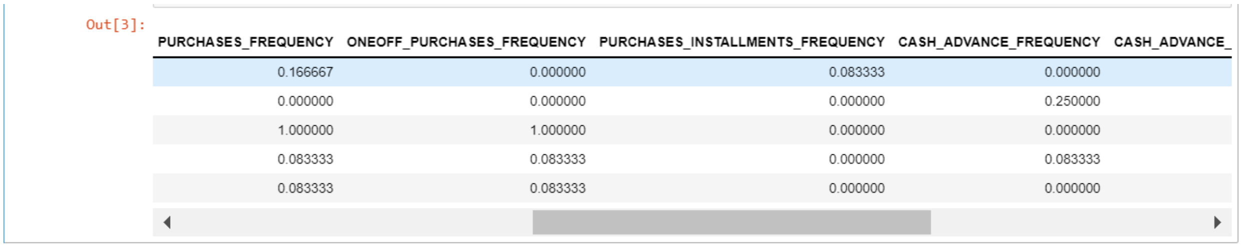
df.head()



```
In [3]:  df = pd.read_csv('CC GENERAL.csv')
         df.head()
```

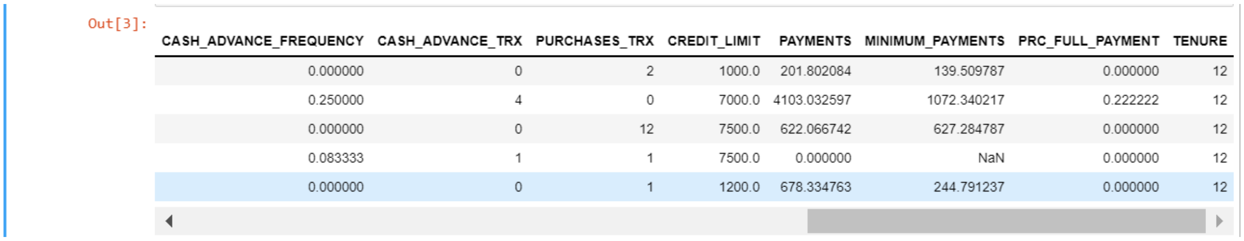
Out[3]:

	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	



Out[3]:

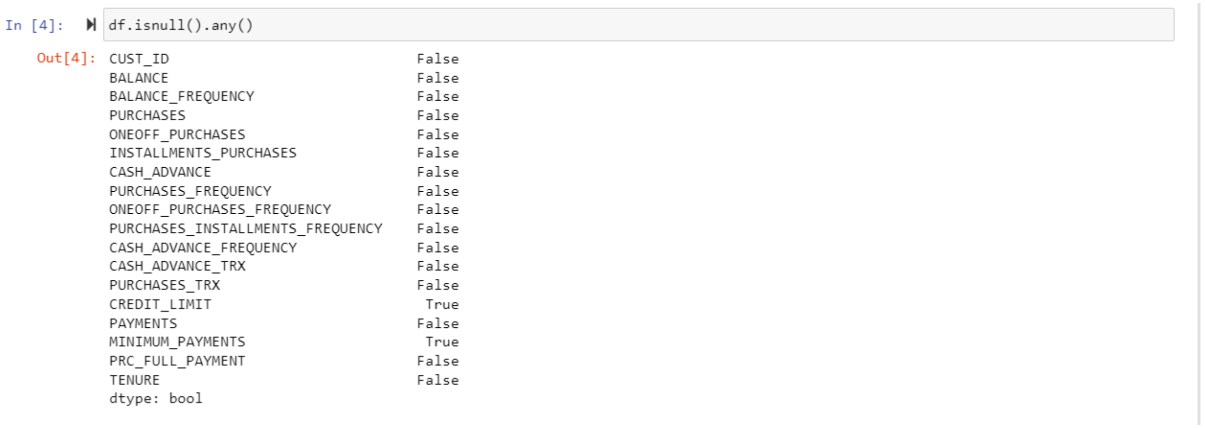
PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_
0.166667	0.000000	0.083333	0.000000	
0.000000	0.000000	0.000000	0.250000	
1.000000	1.000000	0.000000	0.000000	
0.083333	0.083333	0.000000	0.083333	
0.083333	0.083333	0.000000	0.000000	



Out[3]:

CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
0.000000	0	2	1000.0	201.802084	139.509787	0.000000	12
0.250000	4	0	7000.0	4103.032597	1072.340217	0.222222	12
0.000000	0	12	7500.0	622.066742	627.284787	0.000000	12
0.083333	1	1	7500.0	0.000000	NaN	0.000000	12
0.000000	0	1	1200.0	678.334763	244.791237	0.000000	12

reads the csv file then the head() takes first 5 records from the file cc general
The output is resulted in 5 records with all coulmns and fields



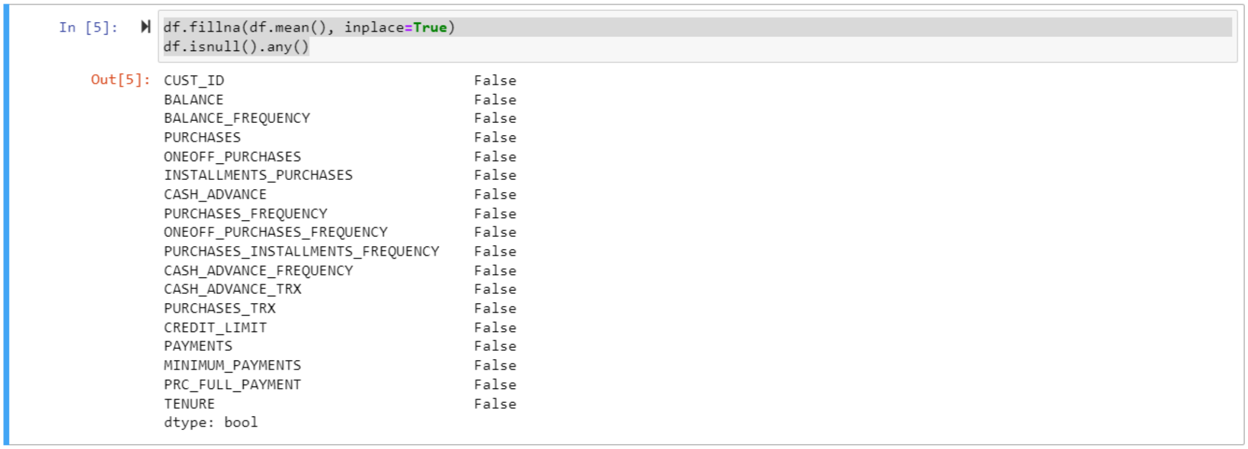
```
In [4]:  df.isnull().any()

Out[4]: 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
```

df.isnull().any()

If you make it df.isnull().any() , you can **find just the columns that have NaN values**: 0 False 1 True 2 False 3 True 4 False 5 True dtype: bool. One more .any() will tell you if any of the above are True > df.isnull().

df.fillna(df.mean(), inplace=True)

df.isnull().any()

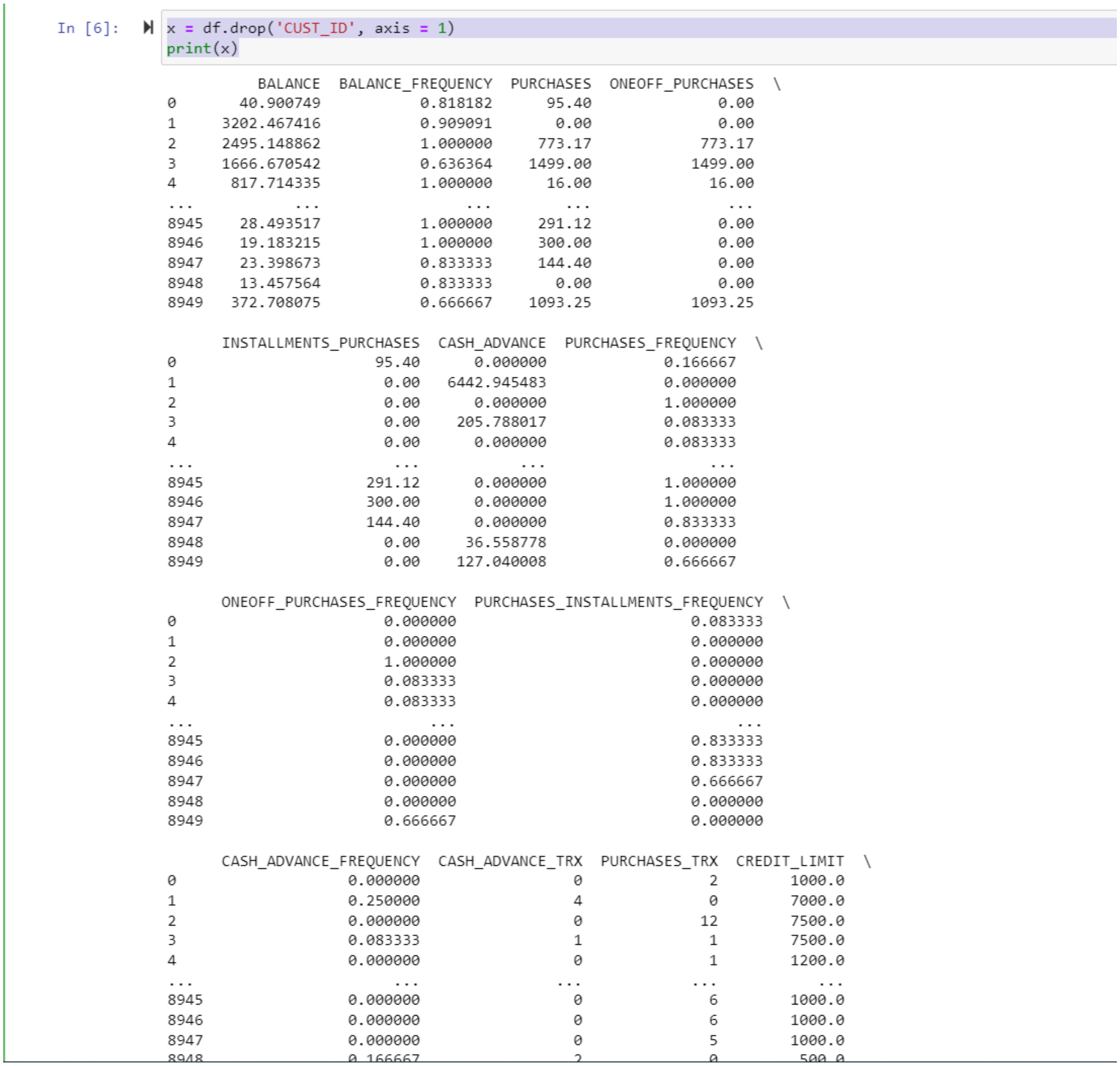


```
In [5]:  df.fillna(df.mean(), inplace=True)
         df.isnull().any()

Out[5]: 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                        False
        PAYMENTS                            False
        MINIMUM_PAYMENTS                    False
        PRC_FULL_PAYMENT                    False
        TENURE                              False
        dtype: bool
```

The fillna() method is used to replace the ‘NaN’ in the dataframe.

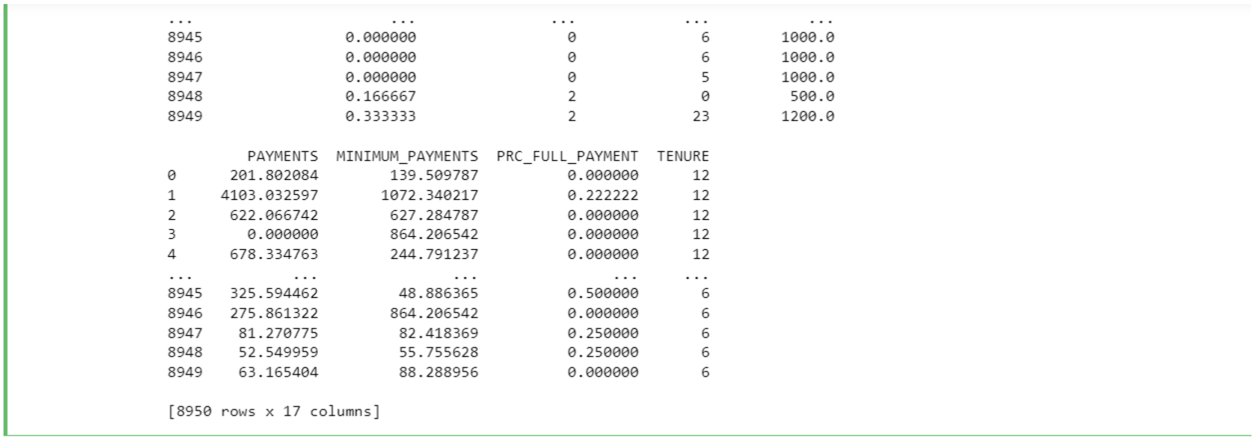
When inplace = True , the data is modified in place, which means it will return nothing and the dataframe is now updated. When inplace = False , which is the default, then the operation is performed and it returns a copy of the object. You then need to save it to something.

x = df.drop('CUST_ID', axis = 1)
print(x)



```
In [6]:  x = df.drop('CUST_ID', axis = 1)
         print(x)
```

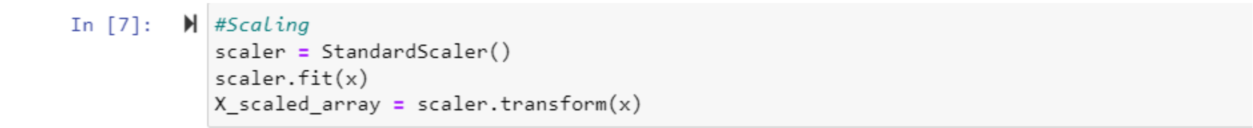
```
          BALANCE  BALANCE_FREQUENCY  PURCHASES  ONEOFF_PURCHASES  \
0       40.900749           0.818182      95.40              0.00
1     3202.467416           0.909091       0.00              0.00
2     2495.148862           1.000000     773.17            773.17
3     1666.670542           0.636364    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
```



```
...                   ...                 ...            ...          ...
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]
```

The drop() function is used to drop specified labels from rows or columns.

Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names. When using a multi-index, labels on different levels can be removed by specifying the level.



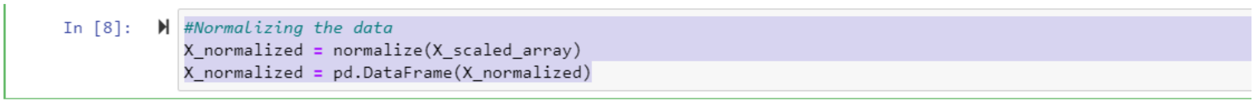
```
In [7]:  #Scaling
         scaler = StandardScaler()
         scaler.fit(x)
         X_scaled_array = scaler.transform(x)
```

#Scaling

scaler = StandardScaler()

scaler.fit(x)

X_scaled_array = scaler.transform(x)

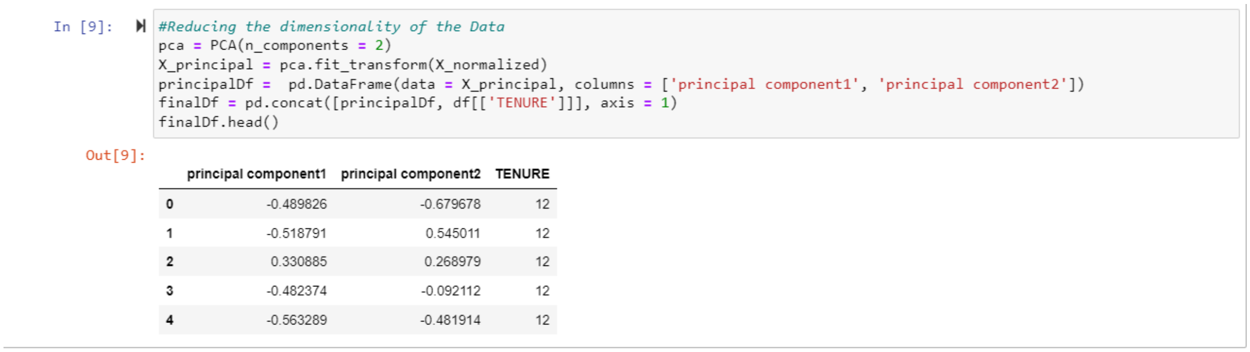


```
In [8]:  #Normalizing the data
         X_normalized = normalize(X_scaled_array)
         X_normalized = pd.DataFrame(X_normalized)
```

#Normalizing the data

X_normalized = normalize(X_scaled_array)

X_normalized = pd.DataFrame(X_normalized)



In [9]:

```
#Reducing the dimensionality of the Data
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(X_normalized)
principalDf =  pd.DataFrame(data = X_principal, columns = ['principal component1', 'principal component2'])
finalDf = pd.concat([principalDf, df[['TENURE']]], axis = 1)
finalDf.head()
```

Out[9]:

	principal component1	principal component2	TENURE
0	-0.489826	-0.679678	12
1	-0.518791	0.545011	12
2	0.330885	0.268979	12
3	-0.482374	-0.092112	12
4	-0.563289	-0.481914	12

#Reducing the dimensionality of the Data

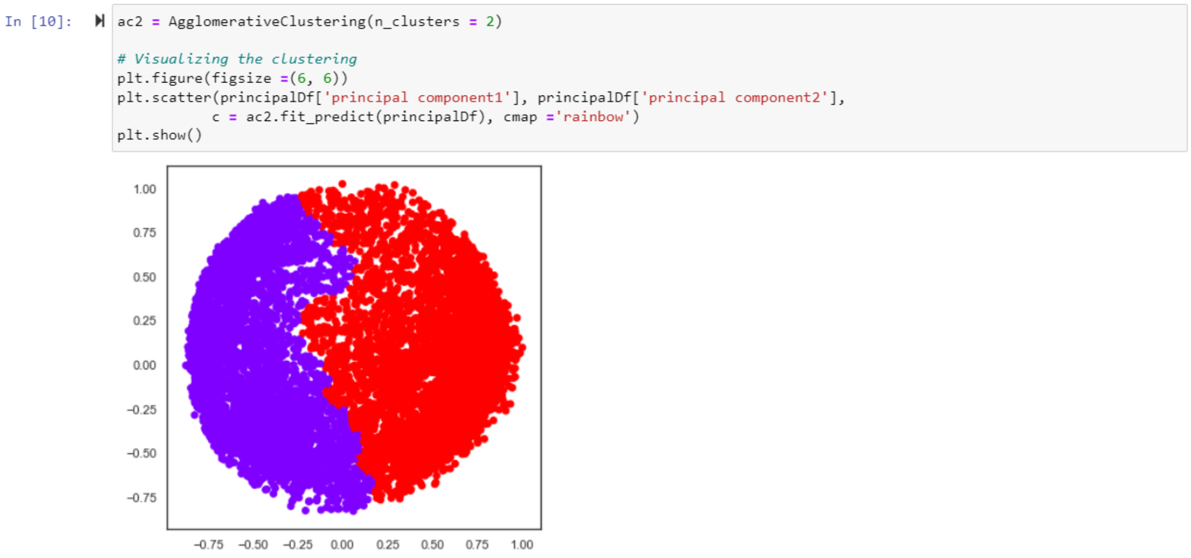
pca = PCA(n_components = 2)

X_principal = pca.fit_transform(X_normalized)

principalDf = pd.DataFrame(data = X_principal, columns = ['principal component1', 'principal component2'])

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

finalDf.head()

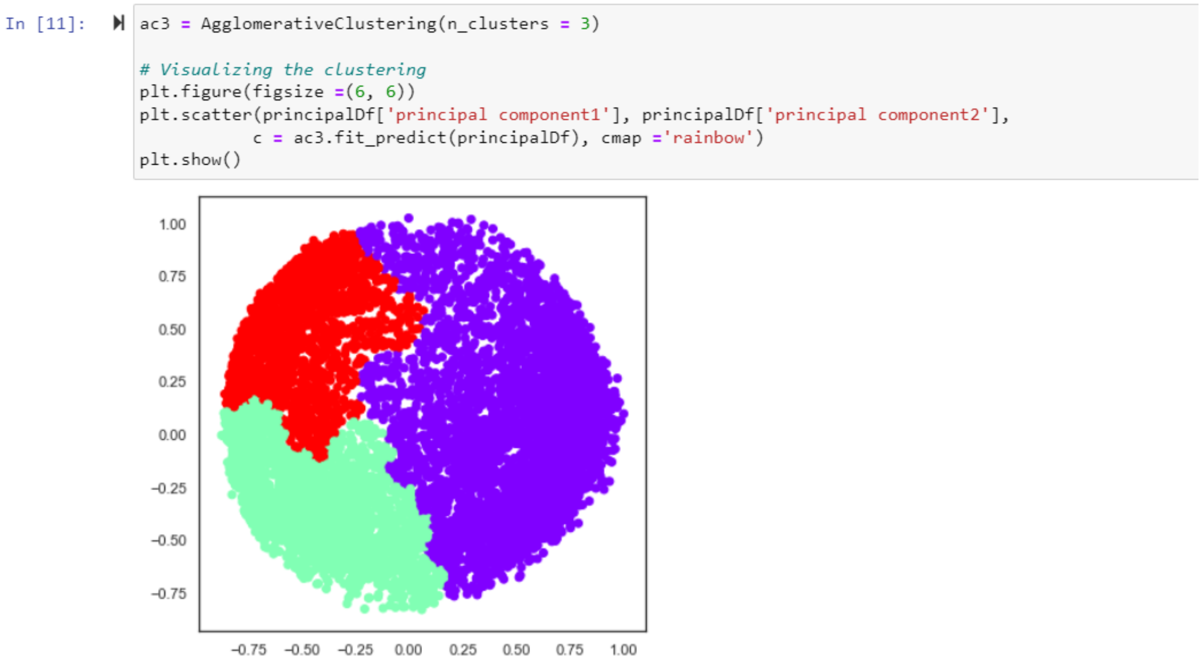


In [10]:

```
ac2 = AgglomerativeClustering(n_clusters = 2)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(principalDf['principal component1'], principalDf['principal component2'],
           c = ac2.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```

ac2 = AgglomerativeClustering(n_clusters = 2)

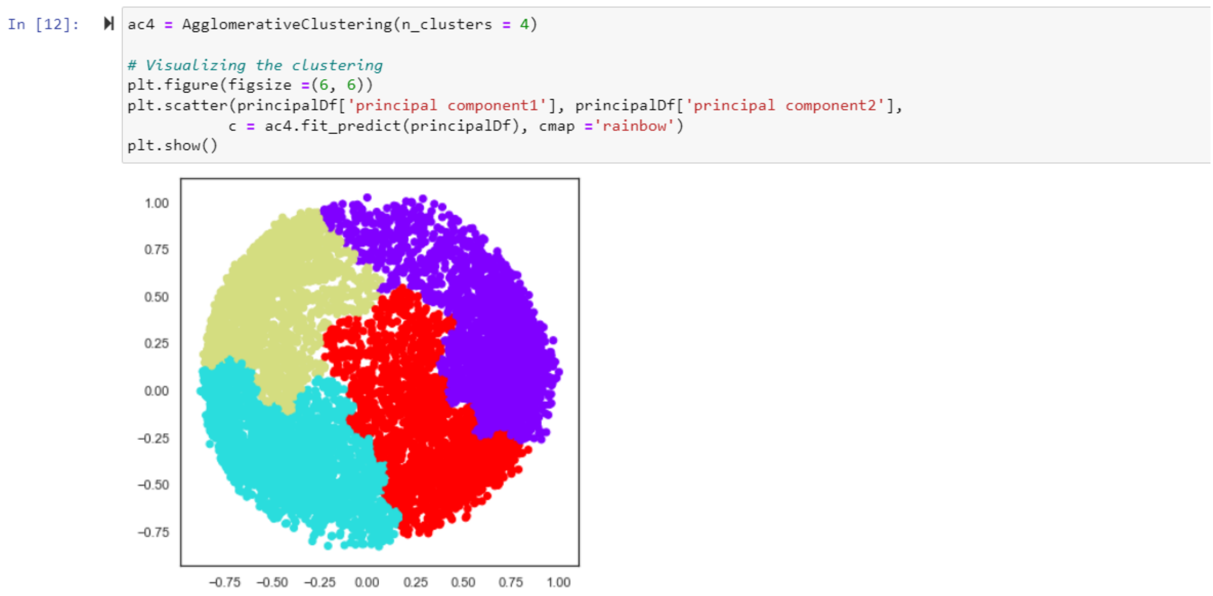
```
# Visualizing the clustering

plt.figure(figsize =(6, 6))

plt.scatter(principalDf['principal component1'], principalDf['principal component2'],

           c = ac2.fit_predict(principalDf), cmap ='rainbow')

plt.show()
```



```
In [11]:  ac3 = AgglomerativeClustering(n_clusters = 3)

          # Visualizing the clustering
          plt.figure(figsize =(6, 6))
          plt.scatter(principalDf['principal component1'], principalDf['principal component2'],
                     c = ac3.fit_predict(principalDf), cmap ='rainbow')
          plt.show()
```

```
ac3 = AgglomerativeClustering(n_clusters = 3)


# Visualizing the clustering

plt.figure(figsize =(6, 6))

plt.scatter(principalDf['principal component1'], principalDf['principal component2'],

        c = ac3.fit_predict(principalDf), cmap ='rainbow')
```

plt.show()



In [12]:

```
ac4 = AgglomerativeClustering(n_clusters = 4)

# Visualizing the clustering
plt.figure(figsize =(6, 6))
plt.scatter(principalDf['principal component1'], principalDf['principal component2'],
           c = ac4.fit_predict(principalDf), cmap ='rainbow')
plt.show()
```

ac4 = AgglomerativeClustering(n_clusters = 4)

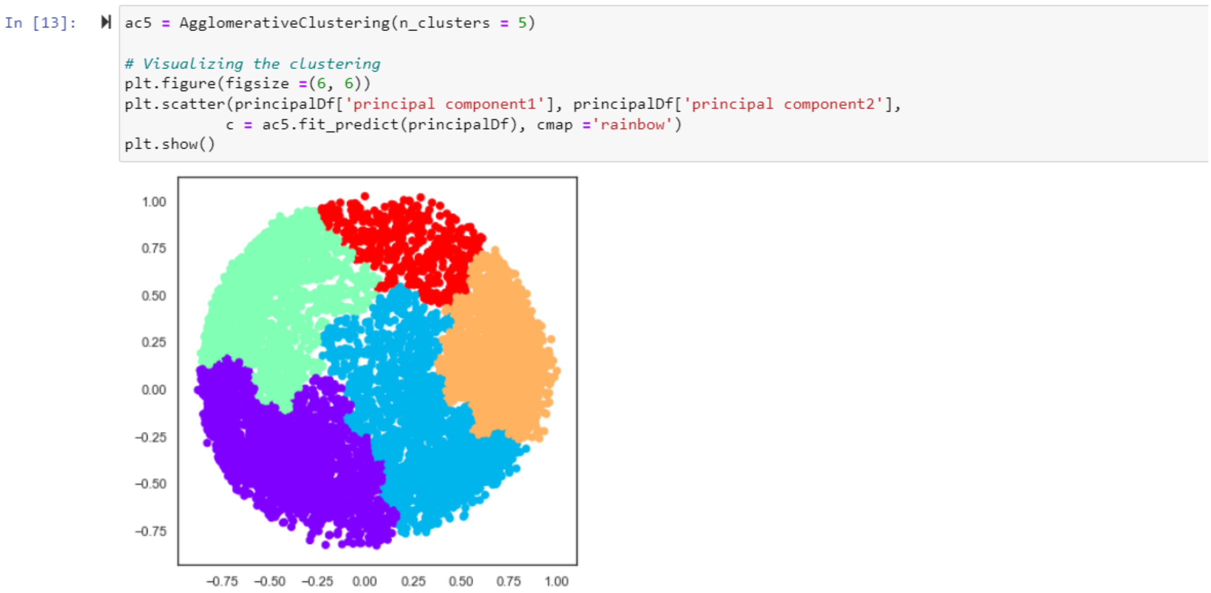
Visualizing the clustering

plt.figure(figsize =(6, 6))

plt.scatter(principalDf['principal component1'], principalDf['principal component2'],

    c = ac4.fit_predict(principalDf), cmap ='rainbow')

plt.show()



```
In [13]:  ac5 = AgglomerativeClustering(n_clusters = 5)

          # Visualizing the clustering
          plt.figure(figsize =(6, 6))
          plt.scatter(principalDf['principal component1'], principalDf['principal component2'],
                     c = ac5.fit_predict(principalDf), cmap ='rainbow')
          plt.show()
```

ac5 = AgglomerativeClustering(n_clusters = 5)

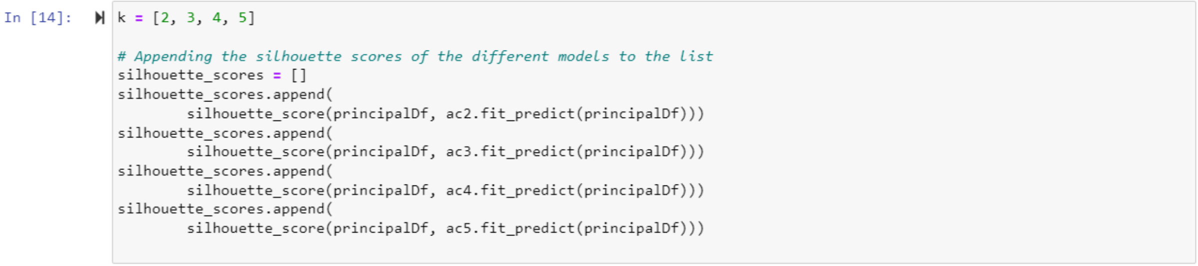
Visualizing the clustering

plt.figure(figsize =(6, 6))

plt.scatter(principalDf['principal component1'], principalDf['principal component2'],

c = ac5.fit_predict(principalDf), cmap ='rainbow')

plt.show()



```
In [14]:  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)))
```

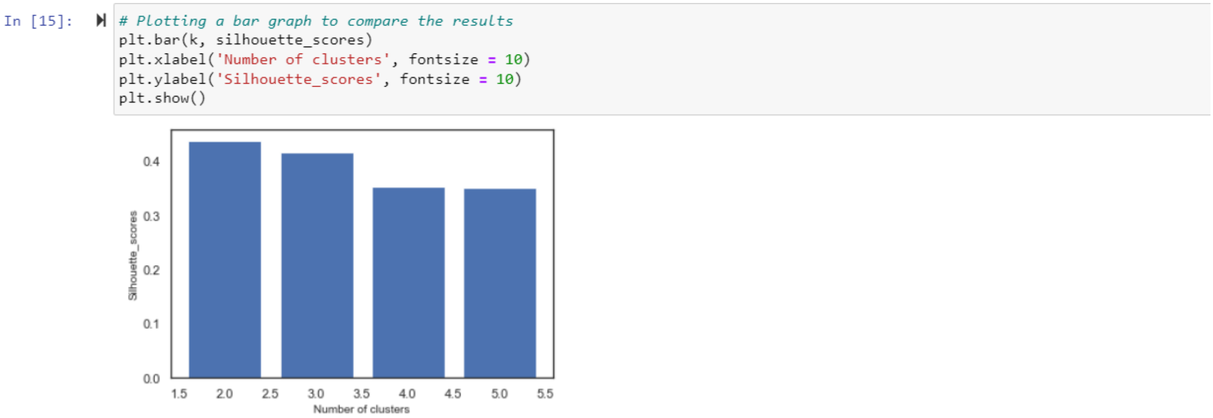
k = [2, 3, 4, 5]

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)))
```



In [15]:

```
# Plotting a bar graph to compare the results
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 10)
plt.ylabel('Silhouette_scores', fontsize = 10)
plt.show()
```

```
# Plotting a bar graph to compare the results
plt.bar(k, silhouette_scores)
plt.xlabel('Number of clusters', fontsize = 10)
plt.ylabel('Silhouette_scores', fontsize = 10)
plt.show()
```