COMPUTE 2

NAME : ARJUN RAMAVATH

SECTION: CS-A-01

ROLL NO : 12112010

Go through the details given under (URL), related to a DataSet 'GTZAN'

Conduct the following task on the dataset & record your observations:
1st task: perform data cleaning, if any, in the dataset.

As the data is already cleaned so there is no requirement of data cleaning

```
// Importing the modules and frameworks



import numpy as np

import pandas as pd

import os



df = pd.read_csv("dataset.csv")



df.dropna()

print(df)
```

2nd task: perform ***K-means Clustering for K=3,5,7*** and also ***Fuzzy C means***. Capture the Clusters generated with Both K Means & C means.

```
// Performance of K means and Fuzzy C Means

import pandas as pd

from sklearn.cluster import KMeans



from sklearn.preprocessing import StandardScaler
```

```
from sklearn.decomposition import PCA

from fcmeans import FCM


import matplotlib.pyplot as plt


df = pd.read_csv('dataset.csv') // dataset file


X = df.iloc[:, 1:59].values


scaler = StandardScaler()

X = scaler.fit_transform(X)


pca = PCA(n_components=2)

X_pca = pca.fit_transform(X)


k_values = [3, 5, 7]

fuzzy_cmeans_c = [3, 5, 7]


// K means code

for k in k_values:

    kkmeans = KMeans(n_clusters=k, random_state=42)

    y_kkmeans = kkmeans.fit_predict(X_pca)
```

```
    plt.figure(figsize=(6, 4))

    plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_kkmeans, cmap='viridis')

    plt.title('K-means clustering (K = ' + str(k) + ')')

    plt.xlabel('Principal Component 1')

    plt.ylabel('Principal Component 2')

    plt.show()

 //Fuzzy means code

for c in fuzzy_cmeans_c:

    fcm = FCM(n_clusters=10, m=c)

    fcm.fit(X_pca)

    y_fcm = fcm.predict(X_pca)


    plt.figure(figsize=(6, 4))

    plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_fcm, cmap='viridis')

    plt.title('Fuzzy C means clustering (c = ' + str(c) + ')')

    plt.xlabel('Principal Component 1')

    plt.ylabel('Principal Component 2')

    plt.show()
```

Output :

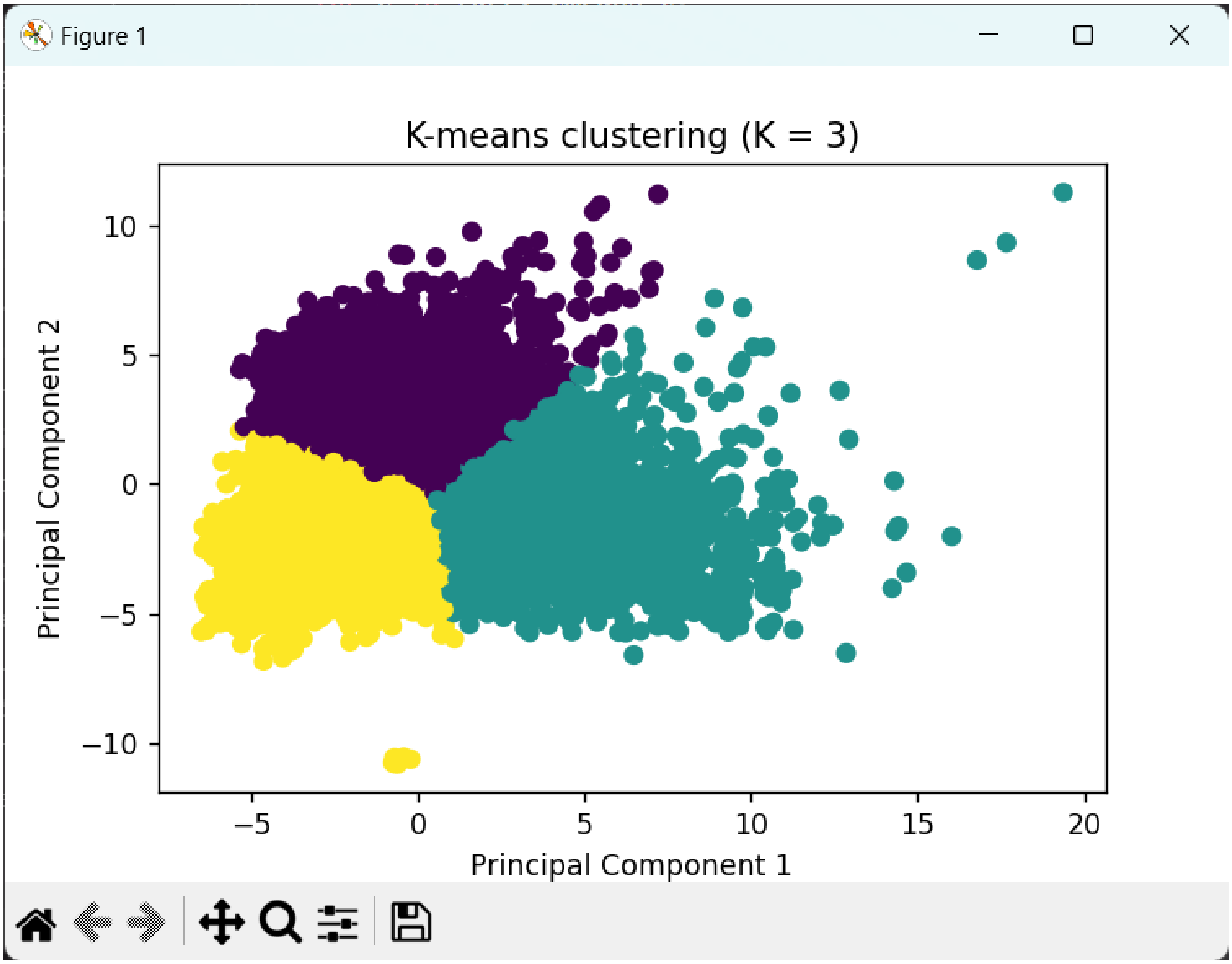


Figure 1
K-means clustering (K = 3)
Principal Component 2
Principal Component 1
10
5
0
−5
−10
−5
0
5
10
15
20

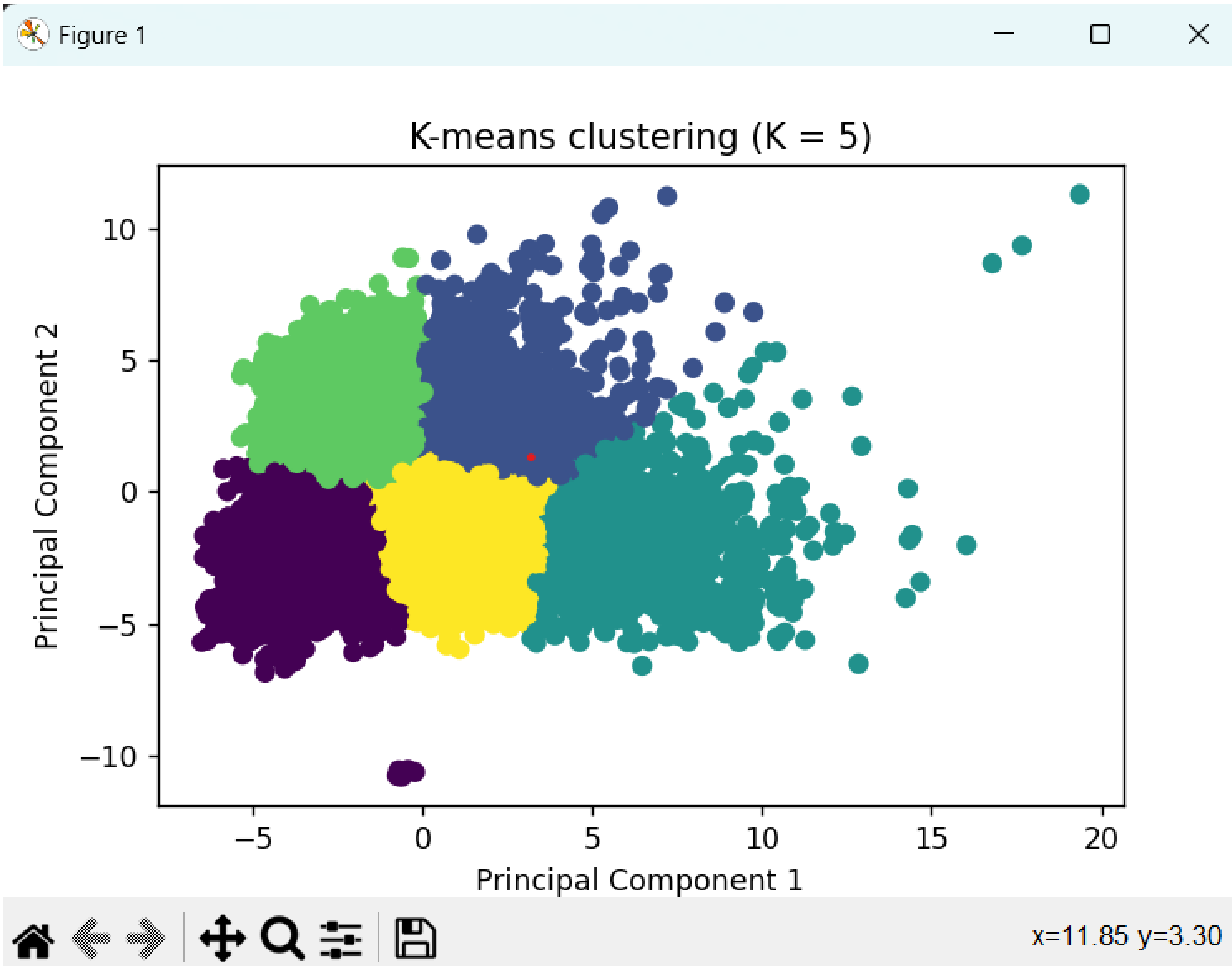
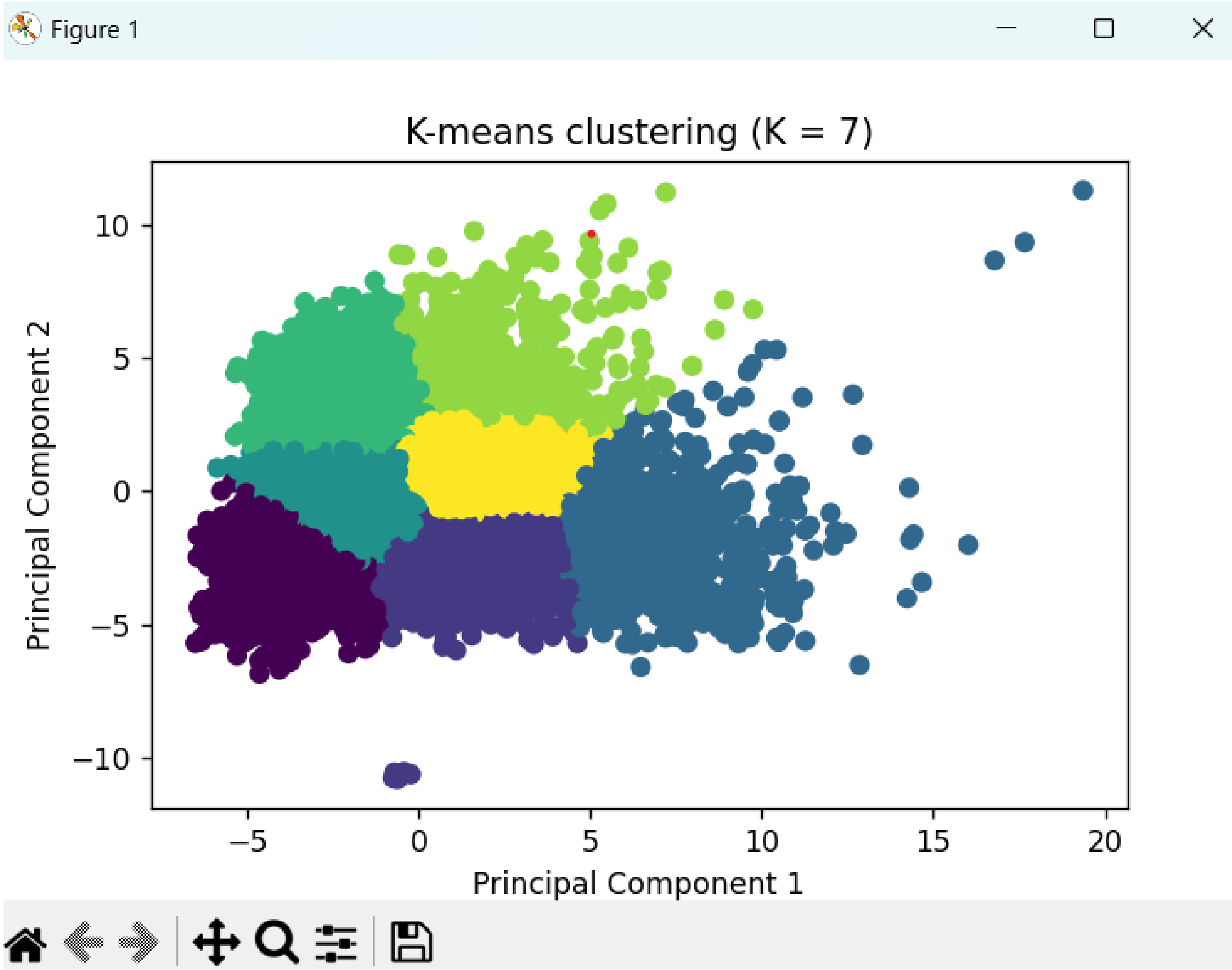


Figure 1
K-means clustering (K = 5)
Principal Component 2
Principal Component 1
10
5
0
−5
−10
−5
0
5
10
15
20
x=11.85 y=3.30



FUZZY C MEANS

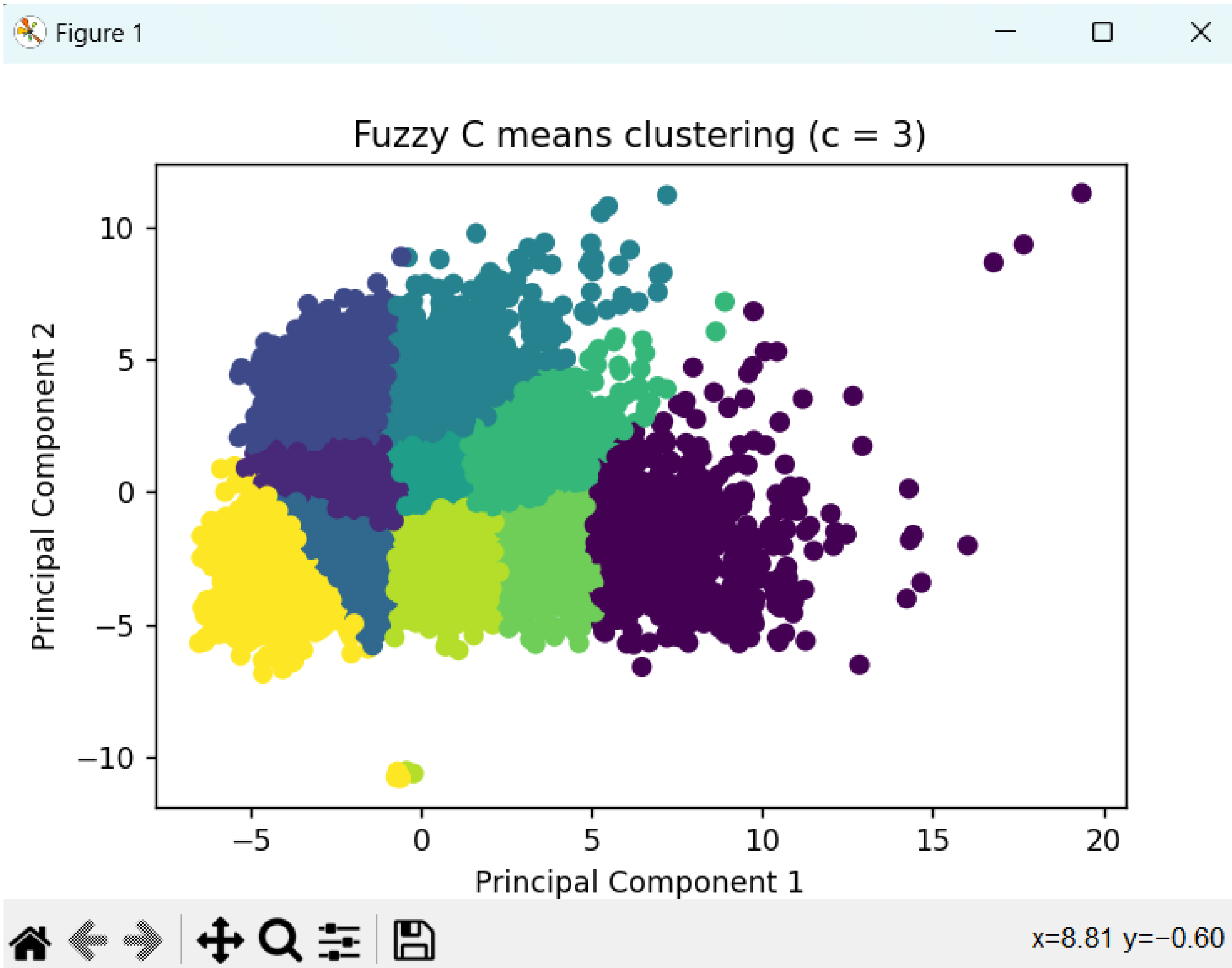


Figure 1
Fuzzy C means clustering (c = 3)
Principal Component 2
Principal Component 1
10
5
0
−5
−10
−5
0
5
10
15
20
x=8.81 y=−0.60

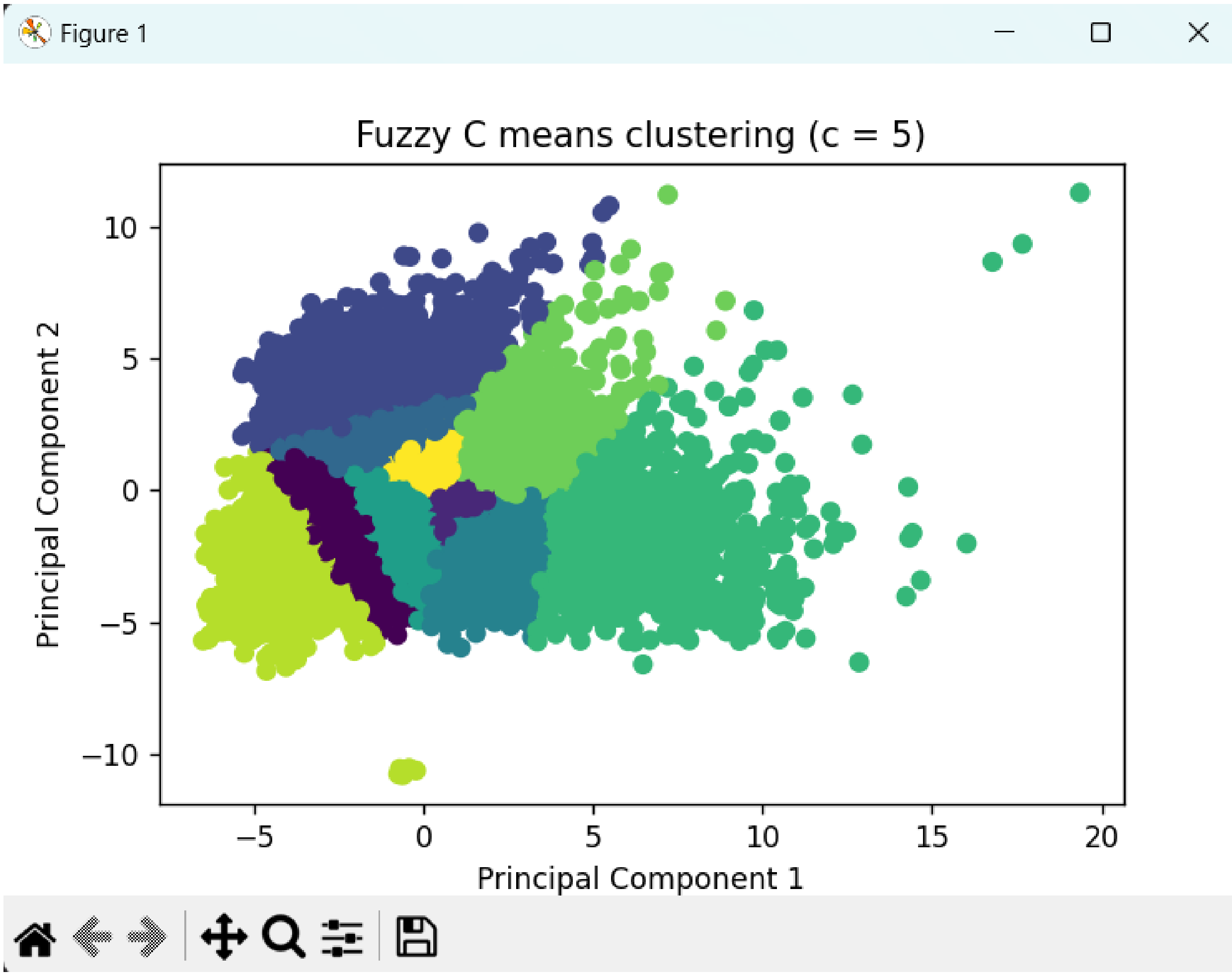
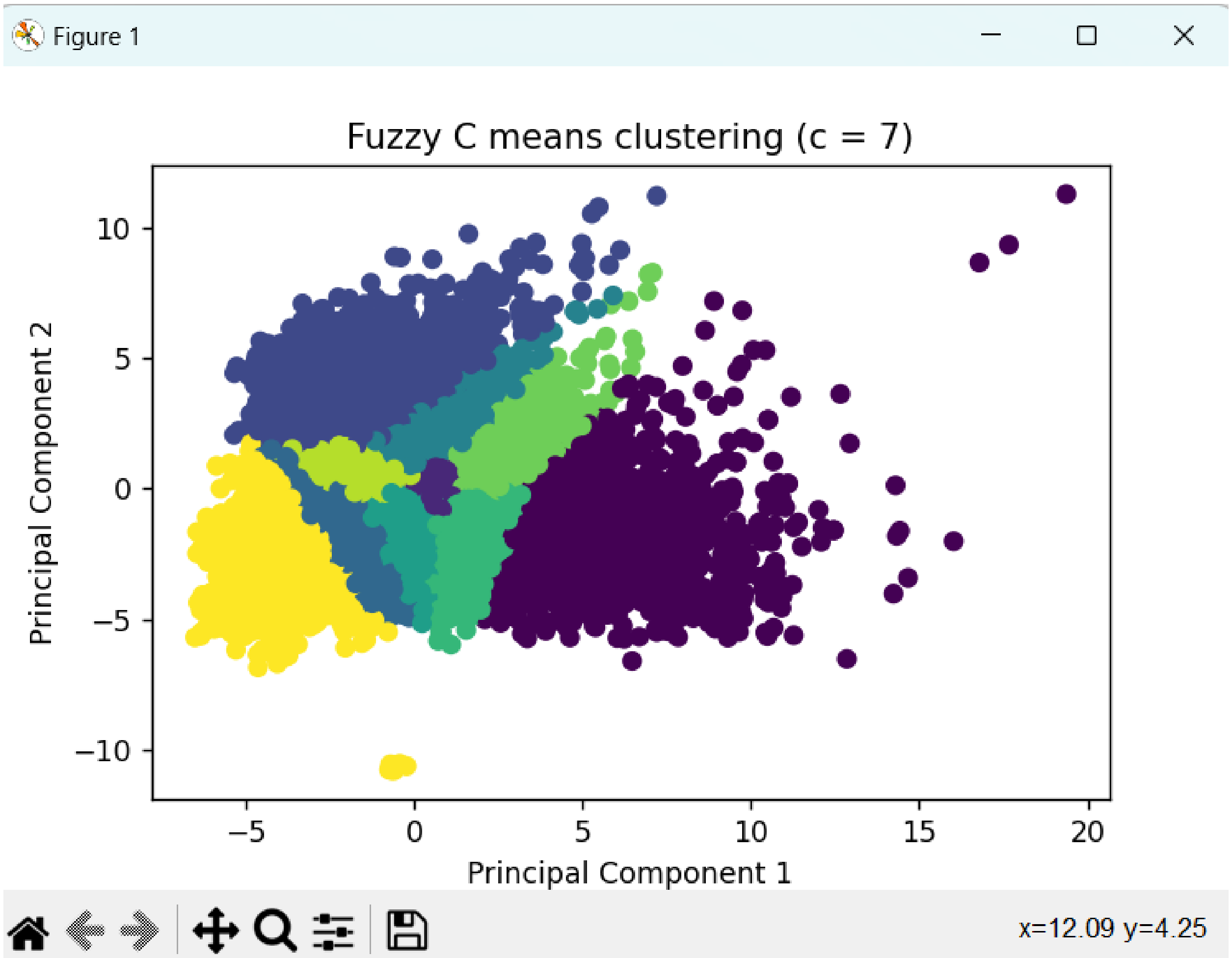


Figure 1
Fuzzy C means clustering (c = 5)
Principal Component 2
Principal Component 1
10
5
0
−5
−10
−5
0
5
10
15
20



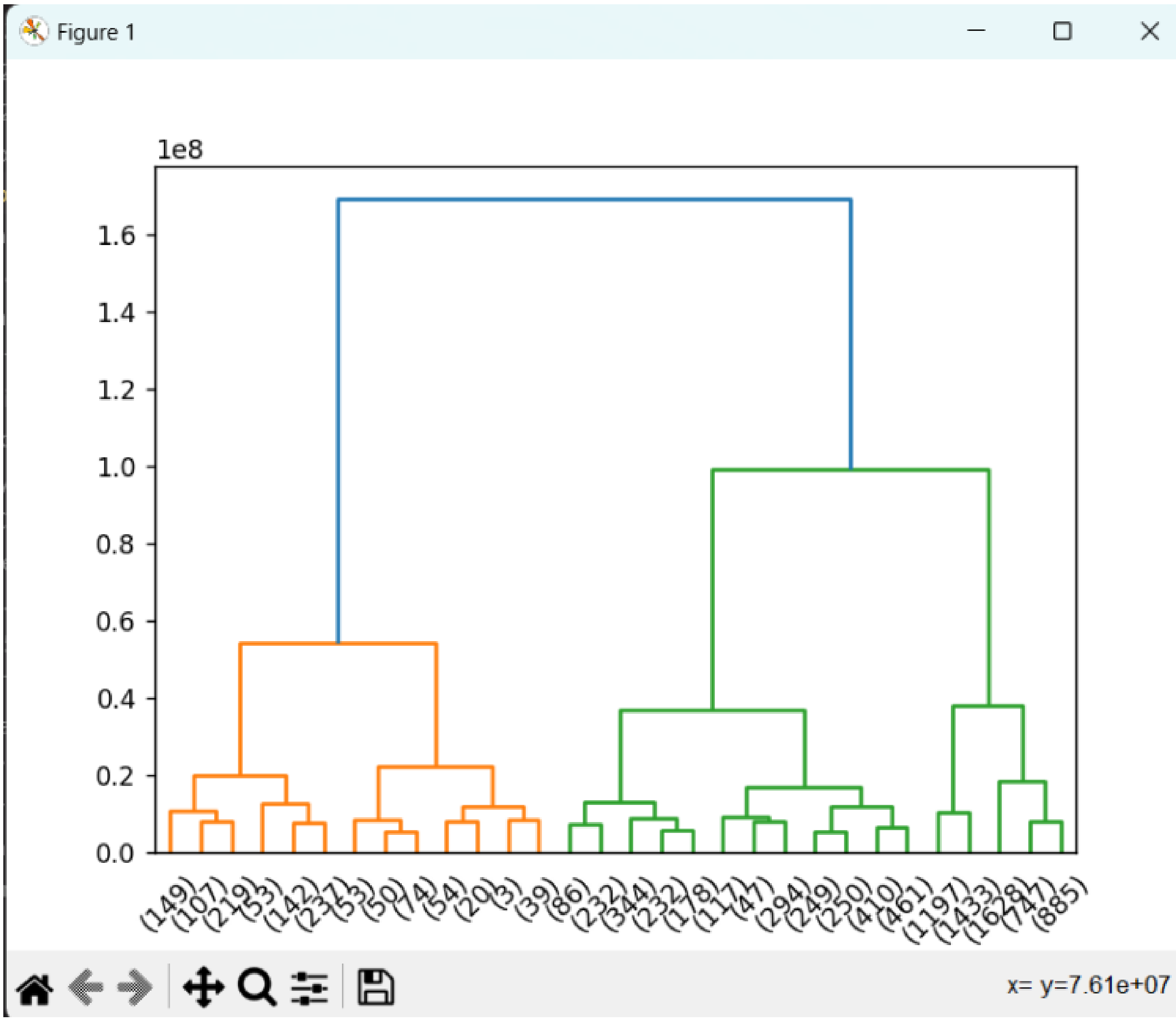
3rd task: perform ***Bottom-up Clustering (Agglomerative clustering)***. Capture the Clusters generated at a different level, and also prepare dendrograms.

```
// importing modules and frameworks

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering
```

```
from scipy.cluster.hierarchy import dendrogram, linkage


data = pd.read_csv('dataset.csv')


mfccs = np.array(data.iloc[:, 1:59])


agg_clustering = AgglomerativeClustering(n_clusters=None, linkage='ward', distance_threshold=0)
clusters = agg_clustering.fit_predict(mfccs)


linked = linkage(mfccs, method='ward')
dendrogram(linked, truncate_mode='lastp', p=30, orientation='top')
plt.show()


for i in range(2, 12):
    clustering = AgglomerativeClustering(n_clusters=i, linkage='ward')
    clustering.fit(mfccs)
    print(f'Clusters at level {i}: {clustering.labels_}')

// Observations
# Observations:


# The dendrogram shows the hierarchy of clusters formed by the agglomerative clustering algorithm.
# We can see that the clusters start merging from the bottom level and go up to the top level.
# At the top level, we can see that all the data points belong to a single cluster.
```

By looking at the dendrogram, we can choose the appropriate level to get the desired number of clusters.

We can also see that at each level, the clustering algorithm forms a different set of clusters based on the distance threshold and linkage criterion used.



4rth task: perform ***density-based (DBSCAN) Clustering***,
5th Task: prepare a brief Comparative summary of clusters generated using the above clustering techniques.

```
// Importing modules and frameworks


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

from sklearn.decomposition import PCA


data = pd.read_csv('dataset.csv')


mfccs = np.array(data.iloc[:, 1:-1])


dbscan_clustering = DBSCAN(eps=20, min_samples=5)

clusters = dbscan_clustering.fit_predict(mfccs)


pca = PCA(n_components=2)

reduced_features = pca.fit_transform(mfccs)


plt.scatter(reduced_features[:, 0], reduced_features[:, 1], c=clusters, cmap='viridis')

plt.show()


# Observations:


# The DBSCAN clustering algorithm forms clusters based on the density of the data points.
```

```
# We can see that the resulting clusters are not well
```