Fuzzy Clustering

Inclass Project 1 - MA4144

Name: Yapa Y.A.N.T.
Index : 200741B
This project contains 9 tasks/questions to be completed, some require written answers. Open markdow cell below the respective question that require written answers and provide (type) your answers. Questions that required written answers are given in blue fonts. Almost all written questions are open ended, they do not have a correct or wrong answer. You are free to give your opinions, but please provide related answers within the context.

After finishing project run the entire notebook once and save the notebook as a pdf (File menu -> Save and Export Notebook As -> PDF). You are required to upload this PDF on moodle .

Use this cell to use any include any imports

In [23]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Q1. In the following cell load the data (in the file SMDataRefined.csv) to a pandas dataframe, and display the data. Then get the data into a numpy matrix X, each row corresponding to a datapoint and each column to a feature, in our case whether a certain token exists (1) or not (0). Let us denote the number of rows of X by N and the number of columns by M.

About the dataset: Each row in the dataset refers to a social media post. The first column refers to the number of likes received by each post (standardized between to the interval $[0,1]$), the columns after that will denote the presence of a certain word (token) in the text of the post. For example, a 1 in the column "TextToken_life" denotes that the particular post contained the word "life" in it and 0 otherwise. These will be the features of our dataset. Our aim is to cluster these social media posts based on number of likes and words contained in it.

In [24]:

```
# Load the data into a pandas dataframe
file_path = '/kaggle/input/smdata/SMDataRefined.csv'
df = pd.read_csv(file_path)

# Display the data
print(df.head())

# Convert the dataframe to a numpy matrix
X = df.to_numpy()

# Get the number of rows and columns
N, M = X.shape

print(f"Number of rows (n): {N}")
print(f"Number of columns (d): {M}")


print(X)
```

```
      Likes  TextToken_life  TextToken_joy  TextToken_night  TextToken_heart  \
0  0.285714               0              0                0                0
1  0.000000               0              0                0                0
2  0.428571               0              0                0                0
```

```
3  0.071429                  0                0                0                0
4  0.214286                  0                0                0                0

   TextToken_laughter  TextToken_dreams  TextToken_feeling  TextToken_day  \
0                   0                 0                  0              1
1                   0                 0                  0              0
2                   0                 0                  0              0
3                   0                 0                  0              0
4                   0                 0                  0              0

   TextToken_like  TextToken_new
0               0              0
1               0              0
2               0              0
3               0              0
4               0              1
Number of rows (n): 732
Number of columns (d): 11
[[0.28571429 0.         0.         ... 1.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.42857143 0.         0.         ... 0.         0.         0.        ]
 ...
 [0.45714286 0.         1.         ... 0.         0.         0.        ]
 [0.47142857 0.         0.         ... 0.         0.         0.        ]
 [0.52857143 0.         0.         ... 0.         0.         0.        ]]
```

Q2. In the following cell create a new function named initMu, that takes in two parameters $\mathrm{npoints}$ and $\mathrm{nclusters}$ and outputs the membership matrix for $\mathrm{npoints}$ number of data points and $\mathrm{nclusters}$ number of clusters. Recall from the lecture that the membership matrix U of dimension $N \times c$ has the following properties.

1. $0 \leq u_{ij} \leq 1$
2. $\sum_{j=1}^{c} u_{ij} = 1$ **for all** $i = 1, 2, \cdots, N$

You can do a random initialization. Here $\mathrm{npoints} = N$ **and** $\mathrm{nclusters} = c$**.**

In [25]:

```python
def initMu(npoints, nclusters):
    # Randomly initialize a matrix of shape (npoints, nclusters)
    U = np.random.rand(npoints, nclusters)

    # Normalize each row so that the sum of elements in each row equals 1
    U = U / U.sum(axis=1, keepdims=True)

    return U
```

Q3. In the following cell create a function named $\mathrm{calculateCenters}$ that computes and returns centers v_j, $j = 1, 2, \cdots, c$ (as $c \times M$ matrix) given the data matrix X and the membership matrix U. Recall the following update formula for v_j from the lectures.

$$v_j = \frac{\sum_{i=1}^{N} \left(\frac{u_{ij}}{u_{max}}\right)^{m_1} x_i}{\sum_{i=1}^{N} \left(\frac{u_{ij}}{u_{max}}\right)^{m_1}}$$

Here $u_{max} = max\left(u_{1j}, u_{2j}, \cdots, u_{Nj}\right)$ and $m_1 \in [1, \infty)$ is a hyperparameter we discussed in class, and will be passed in as a parameter to the function.

The most efficient way to do this computation is by matrix multiplications. Try to find the appropriate matrix

multiplication. You may need to transpose certain matrices.

In [26]:

```
def calculateCenters(X, U, m):
    nclusters = U.shape[1]

    # Initialize the cluster centers matrix (c x M)
    centers = np.zeros((nclusters, X.shape[1]))

    for j in range(nclusters):
        u_max = np.max(U[:, j])
        weights = (U[:, j] / u_max) ** m
        numerator = np.dot(weights, X)
        denominator = np.sum(weights)

        # Compute the cluster center for cluster j
        centers[j, :] = numerator / denominator
    return centers
```

Q4. In the following cell create a function named updateMu that updates the membership matrix U given the data matrix X and the centers. Recall the following update formula for u_{ij} from the lectures.

$u_{ij} = \left[\sum_{l=1}^{c}\left(\frac{d_{ij}}{d_{il}}\right)^{\frac{2}{m_2-1}}\right]^{-1}$ **where d_{ij} is the distance between the ith data point and the jth cluster center. $m_2 \in (1,\infty)$ is another hyperparameter that is passed into the function.**

To compute distances sklearn.metrics.pairwise.pairwise_distances could be useful. For these computations, try to avoid for loops as much as possible and use the tools offered by numpy for matrix manipulations for example such as numpy.tile.

The function should finally return the updated U matrix.

In [27]:

```
from sklearn.metrics import pairwise_distances
def updateMu(X, centers, m):
    D = pairwise_distances(X, centers)
    exponent = 2.0 / (m - 1.0)

    # Initialize the membership matrix U
    U = np.zeros(D.shape)

    for j in range(centers.shape[0]):
        ratio = (D[:, j].reshape(-1, 1) / D) ** exponent
        U[:, j] = 1.0 / np.sum(ratio, axis=1)

    return U
```

Q5. Use following cell to create a function called fuzzyClustering that takes in a data matrix X the number of clusters nclusters, m_1, m_2 hyperparameters and then returns a membership matrix U and nclusters number of centers for each cluster as a matrix whose rows will correspond to the centers. Recall that the fuzzy clustering algorithm,

1. **Initialize the membership matrix U Repeat the following steps while max iterations (maxIter) reached or change in norm of U is greater than a specified tolerance (tol).**
2. **Compute cluster centers**
3. **Update membership matrix**

The function you create should be able to compute the norm between the U matrices from consecutive iterations and plot a graph depicting the variation of the change in norm of U against the number of iterations.

In [28]:

```
def fuzzyClustering(X, m1, m2, nclusters, maxiter=50, tol=1e-5):
    """Perform fuzzy clustering and return membership matrix U and cluster centers."""
    N, M = X.shape
    U = initMu(N, nclusters)
    norm_diffs = []

    for iteration in range(maxiter):
        centers = calculateCenters(X, U, m1)
        U_new = updateMu(X, centers, m2)
        norm_diff = np.linalg.norm(U_new - U)
        norm_diffs.append(norm_diff)

        if norm_diff < tol:
            #print(f"Converged after {iteration+1} iterations.")
            break
        U = U_new

    return norm_diffs, U, centers

def plot_convergence(norm_diffs, title):
    plt.plot(norm_diffs)
    plt.xlabel('Iteration')
    plt.ylabel('Change in Norm of U')
    plt.title(title)
    plt.grid(True)
    plt.show()
```

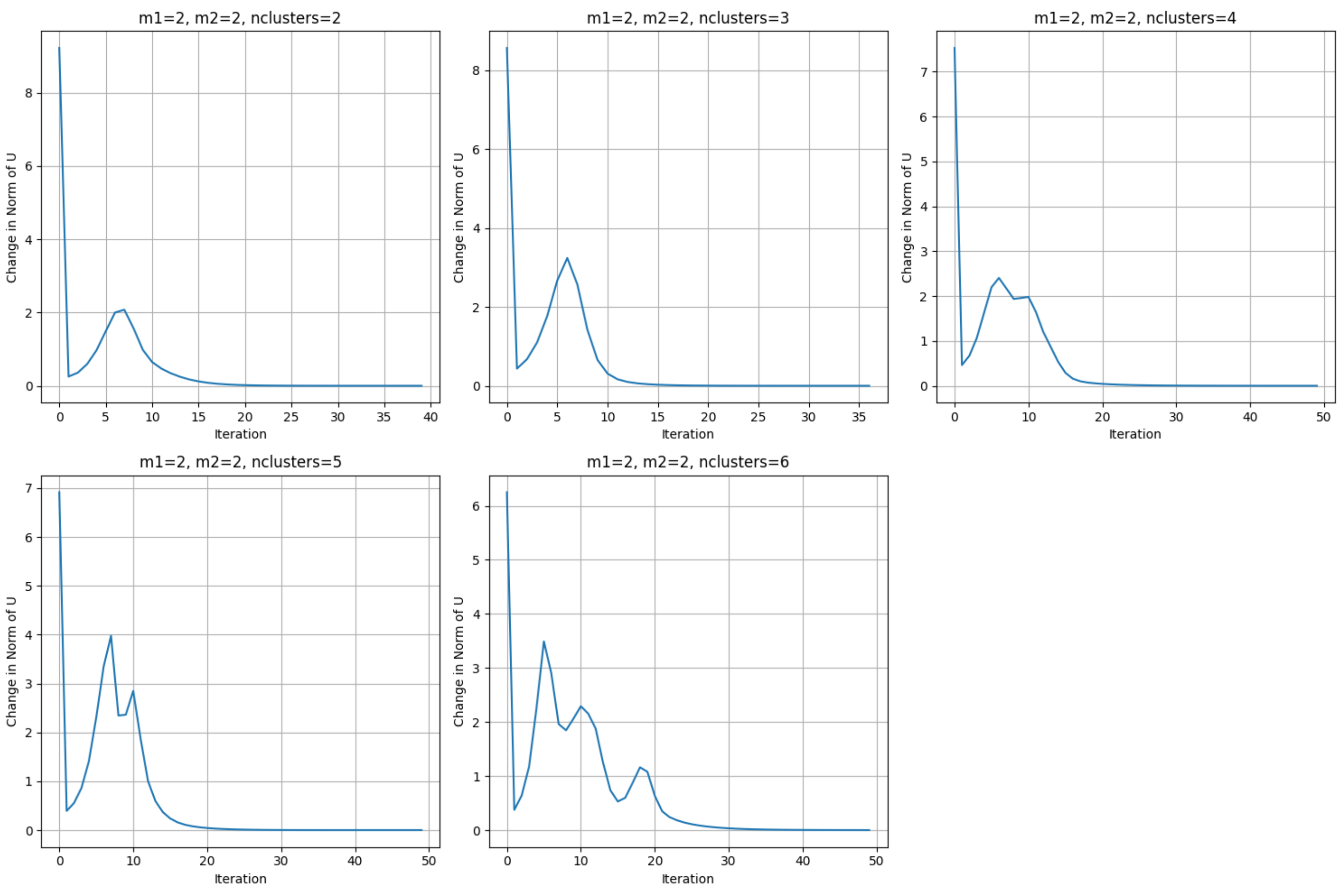
Q6. Run the fuzzyClasssification algorithm with different hyperparameters $m_1, m_2, \text{nclusters}$, note the plot of U-norm difference against the number of iterations. What can you say about it?

Now lets use the fuzzy membership to find some crisp cluster labelling yfuzzy to for each data point. For a given datapoint you'll assign the cluster labelling by looking at which cluster assigns the largest membership value. numpy.argmax function would be useful in this case. Again avoid for loops.

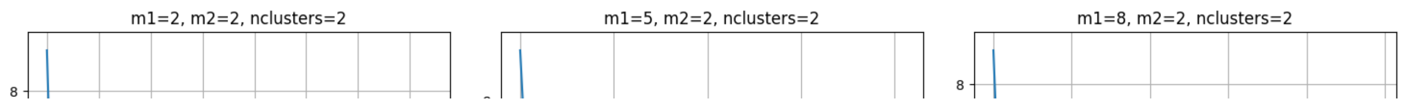
In [29]:

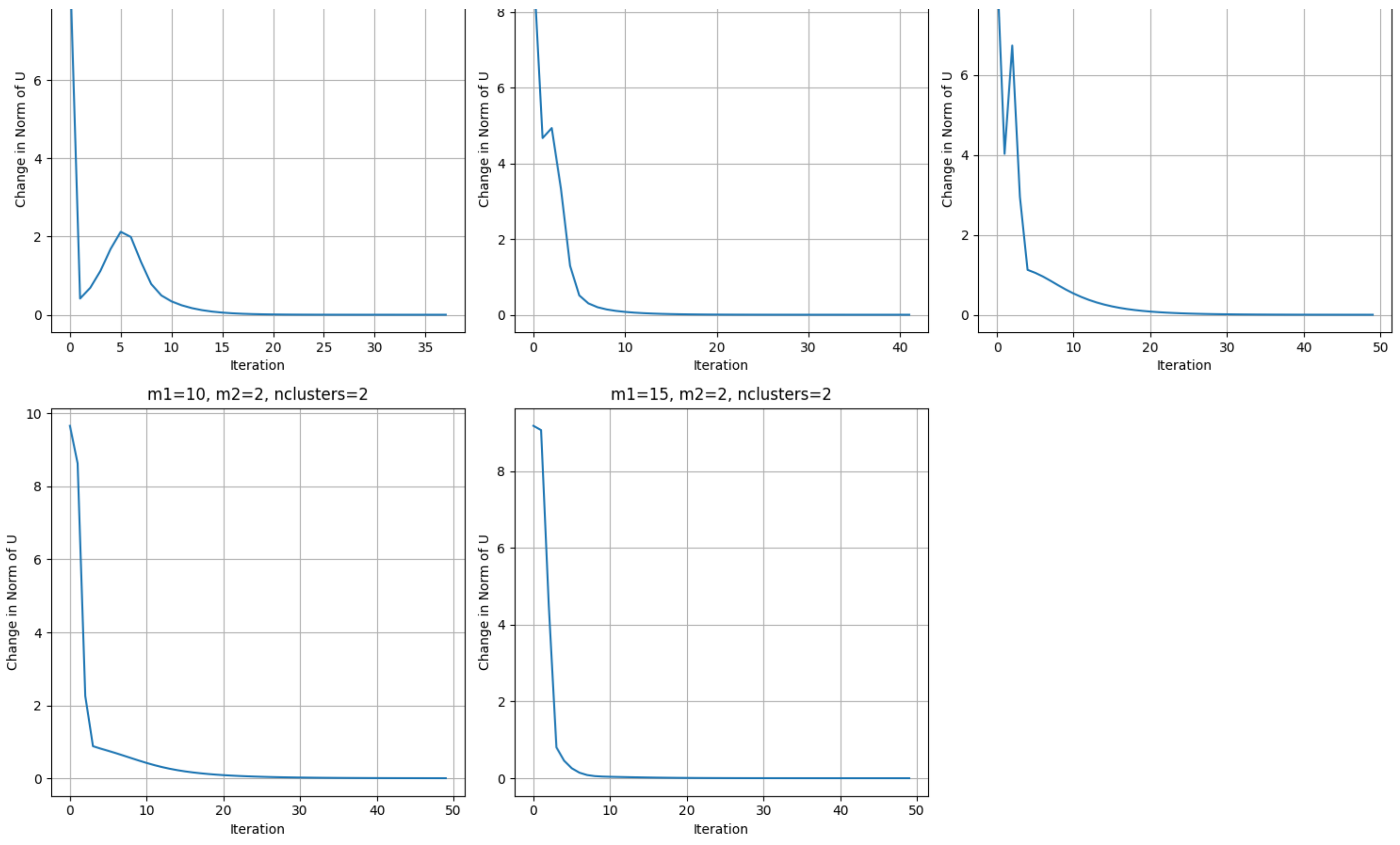
```
# Example parameters
m1_values = [2, 5, 8,10,15]  # Different m1 values
m2_values = [2, 5, 8,10,15]  # Different m2 values
nclusters_values = [2,3,4,5,6] # Different number of clusters

# Store the results for analysis
results = []
def DoCompare(m1_values, m2_values, nclusters_values):
    """Run fuzzy clustering with different parameters and plot the convergence results.""
"
    norm_diffs_dict = {}

    for m1 in m1_values:
        for m2 in m2_values:
            for nclusters in nclusters_values:
                print(f"Running fuzzyClustering with m1={m1}, m2={m2}, nclusters={nclust
ers}")
                norm_diffs, U, centers = fuzzyClustering(X, m1, m2, nclusters)
                key = f"m1={m1}, m2={m2}, nclusters={nclusters}"
                norm_diffs_dict[key] = norm_diffs
                results.append((m1, m2, nclusters, U, centers))


    # Plot the convergence results
    plot_convergence(norm_diffs_dict)

def plot_convergence(norm_diffs_dict):
    num_plots = len(norm_diffs_dict)
    rows = num_plots // 3 + (num_plots % 3 > 0)  # Calculate rows for the grid layout
    cols = min(num_plots, 3)  # Up to 3 columns for better readability

    fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(15, 5 * rows))
    axes = axes.flatten()  # Flatten axes array for easy iteration

    for ax, (title, norm_diffs) in zip(axes, norm_diffs_dict.items()):
        ax.plot(norm_diffs)
        ax.set_xlabel('Iteration')
        ax.set_ylabel('Change in Norm of U')
        ax.set_title(title)
        ax.grid(True)

    # Hide unused subplots
    for ax in axes[len(norm_diffs_dict):]:
        ax.axis('off')

    plt.tight_layout()
    plt.show()

DoCompare([2],[2],nclusters_values)
DoCompare(m1_values,[2],[2])
DoCompare([2],m2_values,[2])


def getCrispLabels(U):
    yfuzzy = np.argmax(U, axis=1)
    return yfuzzy

#for m1, m2, nclusters, U, centers in results:
   # yfuzzy = getCrispLabels(U)
   # print(f"Crisp labels for m1={m1}, m2={m2}, nclusters={nclusters}:")
   # print(yfuzzy)
```

```
Running fuzzyClustering with m1=2, m2=2, nclusters=2
Running fuzzyClustering with m1=2, m2=2, nclusters=3
Running fuzzyClustering with m1=2, m2=2, nclusters=4
Running fuzzyClustering with m1=2, m2=2, nclusters=5
Running fuzzyClustering with m1=2, m2=2, nclusters=6
```

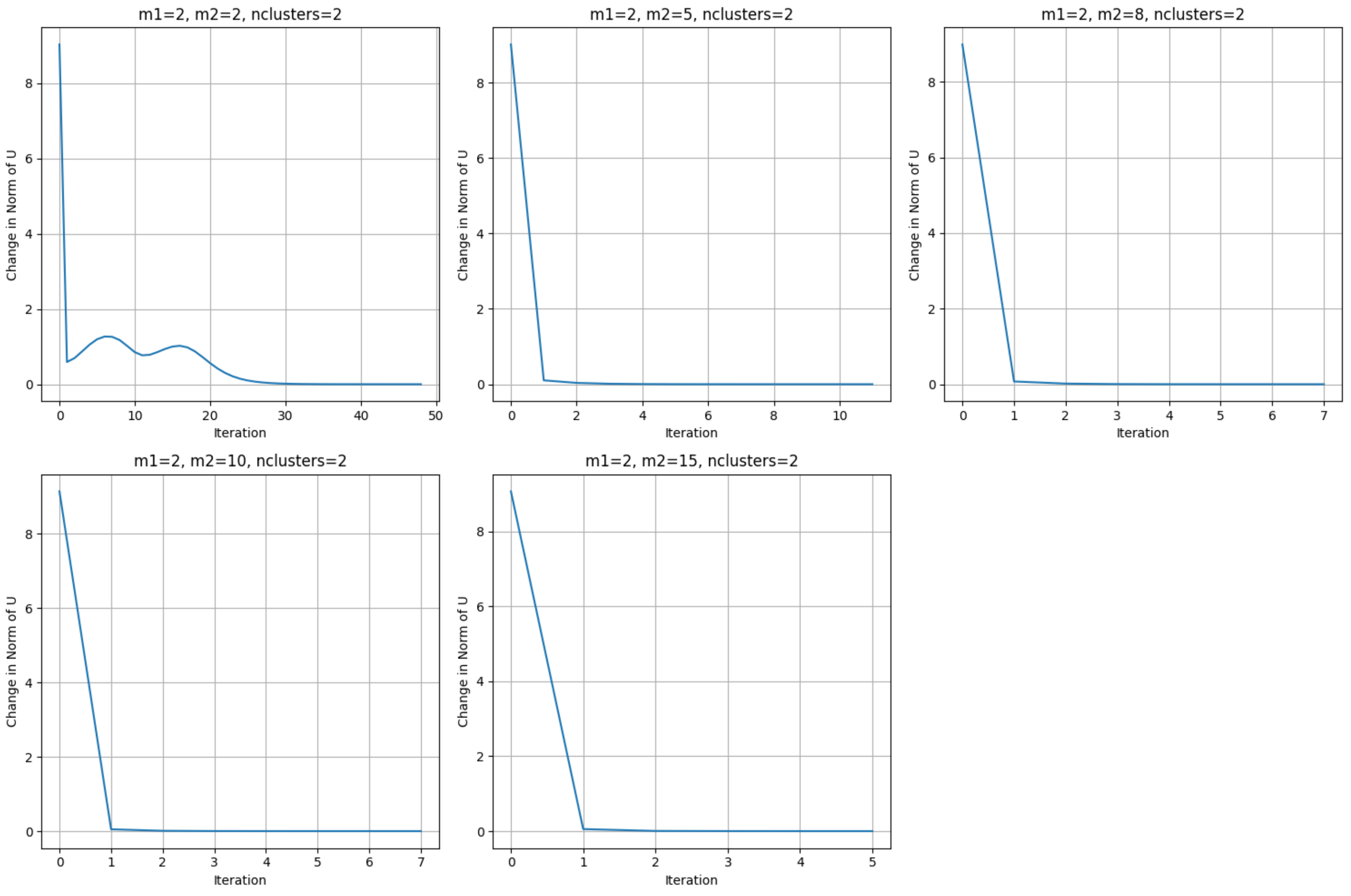


```
Running fuzzyClustering with m1=2, m2=2, nclusters=2
Running fuzzyClustering with m1=5, m2=2, nclusters=2
Running fuzzyClustering with m1=8, m2=2, nclusters=2
Running fuzzyClustering with m1=10, m2=2, nclusters=2
Running fuzzyClustering with m1=15, m2=2, nclusters=2
```





```
Running fuzzyClustering with m1=2, m2=2, nclusters=2
Running fuzzyClustering with m1=2, m2=5, nclusters=2
Running fuzzyClustering with m1=2, m2=8, nclusters=2
Running fuzzyClustering with m1=2, m2=10, nclusters=2
Running fuzzyClustering with m1=2, m2=15, nclusters=2
```



When m1 and m2 gets larger, clustering converges more quickly. When nclusters changes, convergence of the clustering also changes. But there is no propotionality between nclusters and number of clusters. It converges well when the nclusters matches the the dataset.

Q7. The Silhoutte score is a measure of how well the clustering has been done, the closer the score to 1.0 the better the clustering. Refer to sklearn.metrics.silhouette_score. Determine the best choice of hyperparameters $m_1, m_2, \text{nclusters}$ through a grid search based on the silhoutte score as the evaluation metric. Report the best combination of hyperparameters. For the grid search try $m_1 = 1.0, 1.2, \cdots, 3.0$, $m_2 = 1.2, 1.4, \cdots, 3.0$, $\text{nclusters} = 2, 3, \cdots, 15$.

In [30]:

```
from sklearn.metrics import silhouette_score

# Define the grid for m1, m2, and nclusters
m1_range = np.arange(1.0, 3.0, 0.2)
m2_range = np.arange(1.2, 3.0, 0.2)
nclusters_range = range(2,12,1)

best_silhouette = -1
best_hyperparams = None

# Perform grid search
for m1 in m1_range:
    for m2 in m2_range:
        for nclusters in nclusters_range:
            #print(f"Evaluating m1={m1}, m2={m2}, nclusters={nclusters}")

            # Run fuzzy clustering with current hyperparameters
            norm_diffs, U, centers = fuzzyClustering(X, m1, m2, nclusters)

            # Get the crisp labels from the fuzzy membership matrix
            yfuzzy = getCrispLabels(U)

            # Calculate the silhouette score for this clustering
            silhouette_avg = silhouette_score(X, yfuzzy)

            if silhouette_avg > best_silhouette:
                best_silhouette = silhouette_avg
                best_hyperparams = (m1, m2, nclusters)
                best_yfuzzy = getCrispLabels(U)
```

In [31]:

```
print(f"Best combination of hyperparameters: m1={best_hyperparams[0]}, m2={best_hyperpara
ms[1]}, nclusters={best_hyperparams[2]}")
print(f"Best Silhouette score: {best_silhouette}")
```

```
Best combination of hyperparameters: m1=1.7999999999999998, m2=1.4, nclusters=2
Best Silhouette score: 0.5463701278242009
```

In [32]:

```
print(best_yfuzzy)
```

```
[0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0
 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
```

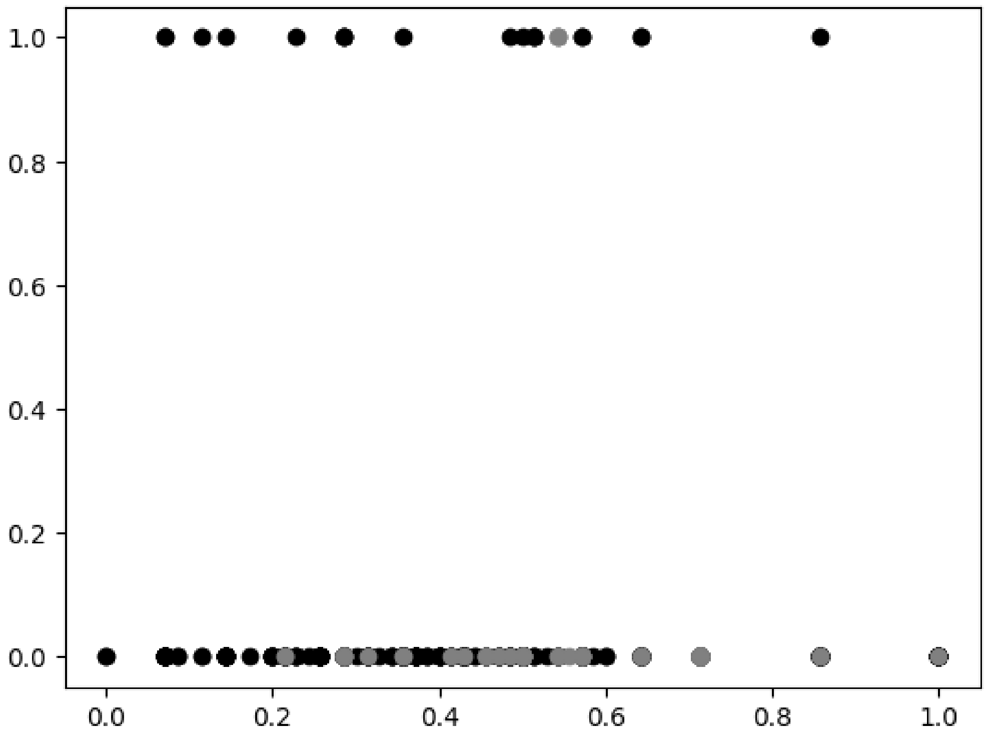
Q8. For the best combination of hyperparameters run the following visualization function with different values for $0 \leq f_1 < f_2 < \text{nclusters}$ **.** **What do you see? Explain the purpose of the code and what you would expect through this?**

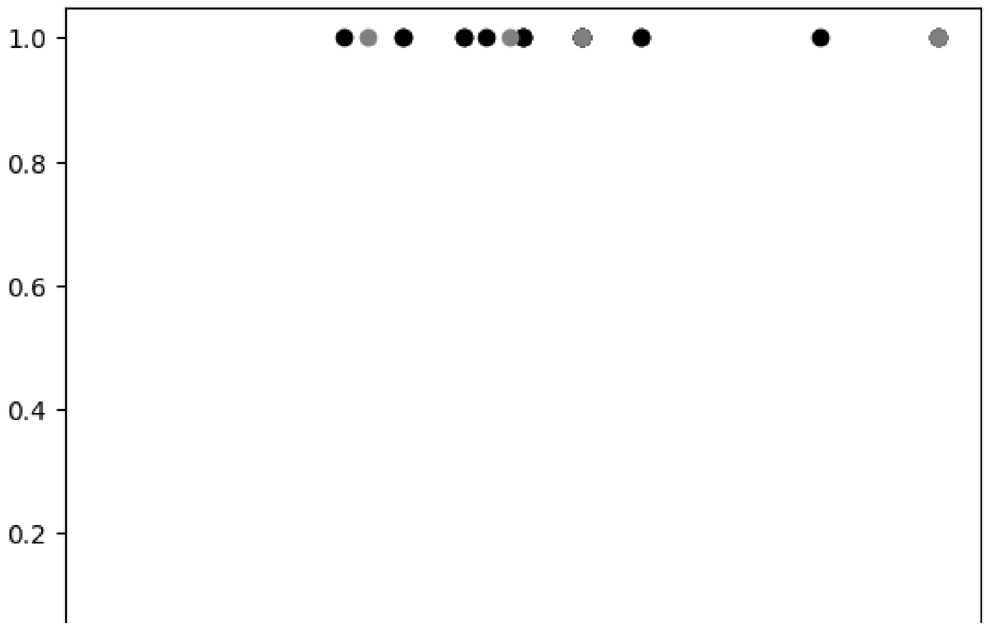
In [33]:

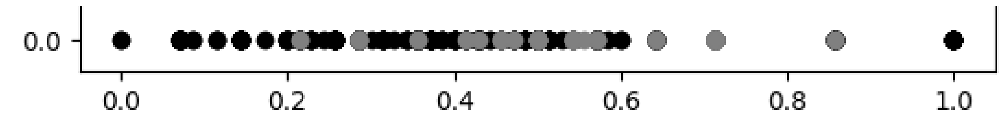
```
def visualizeClusters2D(X, y, f1, f2):
    colorlist = ['black', 'gray', 'red', 'sienna', 'green', 'blue', 'gold', 'darkorange'
, 'olive', 'lavendar', 'salmon', 'teal', 'pink', 'magenta', 'tan', 'wheat']


    for i in range(8):
        plt.scatter(X[y == i, f1], X[y == i, f2], color = colorlist[i])


    plt.show()
```

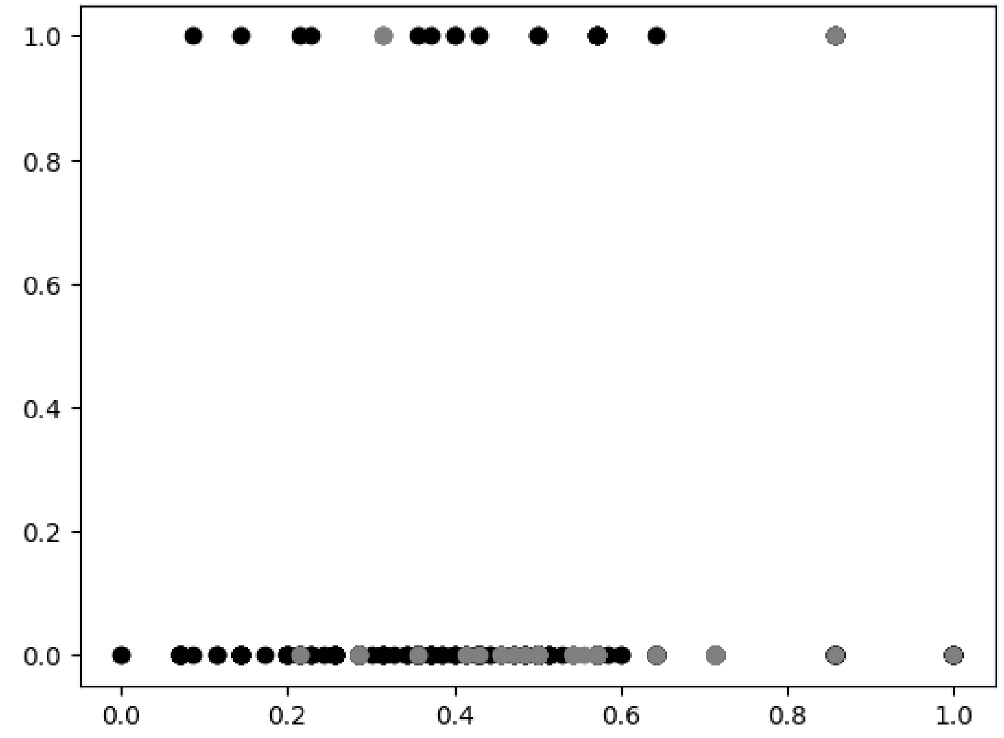
In [34]:

```
# Visualize clusters with valid feature indices
visualizeClusters2D(X, best_yfuzzy, 0, 1)
visualizeClusters2D(X, best_yfuzzy, 0, 2)
visualizeClusters2D(X, best_yfuzzy, 0, 4)
```









This visualization function visualize the clusters by color and feature1 (f1) in x axis and feature2 (f2) in y axis. this gives an idea about how clustering has occured according to the considering features.

Q9. Load (to a pandas dataframe) the actual dataset (SMData.csv) from which the previous dataset was cleaned out. Match each entry in this dataset with the labels you found out through clustering using the best combination of hyperparameters. Go through a few of those clustered entries and notice any patterns (or common sentiment) in the text (the "Text" column of this dataset) of each entry within clusters vs outside the clusters. The label of the ith entry in the dataset $= \mathrm{yfuzzy}[i]$. Explain any observations. Based on your observations, is the clustering successful?

In [37]:

```
import pandas as pd

# Load the original dataset
original_data = pd.read_csv('/kaggle/input/smdata/SMData.csv')
text_data = original_data.iloc[:, 2]  # Assuming text is in the 3rd column (index 2)

# Assuming cluster_labels is your best_yfuzzy variable
cluster_labels = best_yfuzzy

# Create a DataFrame with texts and their corresponding cluster labels
result_df = pd.DataFrame({
    'Text': text_data,
    'ClusterLabel': cluster_labels
})

# Print the first 30 rows
print(result_df.head(60))
```

```
                                                Text  ClusterLabel
0     Enjoying a beautiful day at the park!          ...             0
1     Traffic was terrible this morning.             ...             0
2     Just finished an amazing workout! □            ...             0
3     Excited about the upcoming weekend getaway!  ...             0
4     Trying out a new recipe for dinner tonight.  ...             1
5     Feeling grateful for the little things in lif...             0
6     Rainy days call for cozy blankets and hot coc...             0
7     The new movie release is a must-watch!         ...             1
```

```
8     Political discussions heating up on the timel...            0
9     Missing summer vibes and beach days.         ...            0
10    Just published a new blog post. Check it out!...            1
11    Feeling a bit under the weather today.       ...            0
12    Exploring the city's hidden gems.            ...            0
13    New year, new fitness goals! □               ...            1
14    Technology is changing the way we live.      ...            0
15    Reflecting on the past and looking ahead.    ...            0
16    Just adopted a cute furry friend! □          ...            0
17    Late-night gaming session with friends.      ...            0
18    Attending a virtual conference on AI.        ...            0
19    Winter blues got me feeling low.             ...            0
20    Sipping coffee and enjoying a good book.     ...            0
21    Exploring the world of virtual reality.      ...            0
22    Productive day ticking off my to-do list.    ...            0
23    Just finished a challenging workout routine. ...            0
24    Celebrating a milestone at work! □           ...            0
25    Sunday brunch with friends.                  ...            0
26    Learning a new language for personal growth. ...            1
27    Quiet evening with a good book.              ...            0
28    Reflecting on the importance of mental health...            0
29    New painting in progress! □                  ...            0
30    Weekend road trip to explore scenic views.   ...            0
31    Enjoying a cup of tea and watching the sunset...            0
32    Coding a new project with enthusiasm.        ...            1
33    Feeling inspired after attending a workshop. ...            0
34    Winter sports day at the local park.         ...            0
35    Quality time with family this weekend.       ...            0
36    Attending a live music concert tonight.      ...            0
37    Practicing mindfulness with meditation.      ...            0
38    Trying out a new dessert recipe.             ...            1
39    Excited about the upcoming gaming tournament....            0
40    Planning a garden makeover for spring.       ...            0
41    Celebrating a friend's birthday tonight! □   ...            0
42    Feeling accomplished after a productive day. ...            0
43    A cozy evening with a good movie.            ...            0
44    Exploring local art galleries this weekend.  ...            0
45    New book release from my favorite author!    ...            0
46    Attending a virtual reality meetup.          ...            0
47    Reflecting on the beauty of nature.          ...            0
48    Cooking a special dinner for loved ones.     ...            0
49    Feeling optimistic about the week ahead.     ...            0
50    Starting a new fitness challenge tomorrow! □ ...            1
51    Sunday bike ride through scenic trails.      ...            0
52    Can't believe the injustice happening in our ...            0
53    Feeling a sense of fear after watching a thri...            0
54    Heartbroken after hearing the news about a na...            0
55    The state of the world's environment is just ...            0
56    Pure happiness: celebrating a loved one's ach...            0
57    Laughter is the best medicine—enjoying a come...            0
58    Sharing love and positive vibes with everyone...            0
59    An amusing incident brightened up my day!    ...            0
```

In this clustering,what are the two clustering is not rally clear but two clusters are seem to be future plans vs others according to the above text and yfuzzy output. But this clustering is not successful since there are more suitable clustering like good and bad news