

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#==Install Pacage==#
#!pip install scikit-fuzzy
import skfuzzy as fuzz
```

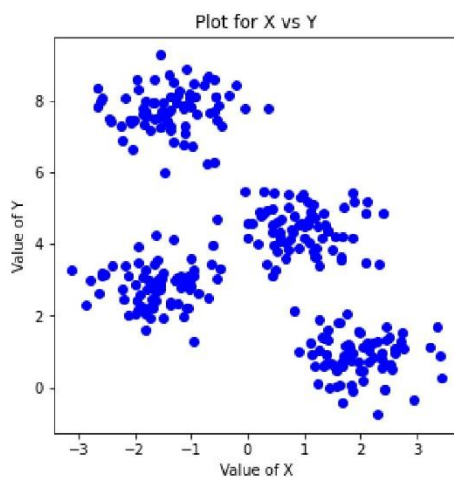
```
In [2]: from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import make_moons
from sklearn.datasets import make_blobs

# Generate moons dataset
X, y = make_moons(n_samples=300, noise=0.1,
                  # random_state=42)

# Generate synthetic data
X, y_true = make_blobs(n_samples=300, centers=4,
                       cluster_std=0.60, random_state=0)
```

```
In [3]: D=pd.DataFrame(X)
D.columns=['X', 'Y']
```

```
In [4]: plt.figure(figsize=(5, 5))
plt.plot(D['X'],D['Y'],'o',color='b')
plt.xlabel("Value of X")
plt.ylabel("Value of Y")
plt.title('Plot for X vs Y')
plt.savefig('D:/Fuzzy C-Means_Scatter plot for Data.jpg',dpi=1000)
plt.show()
```



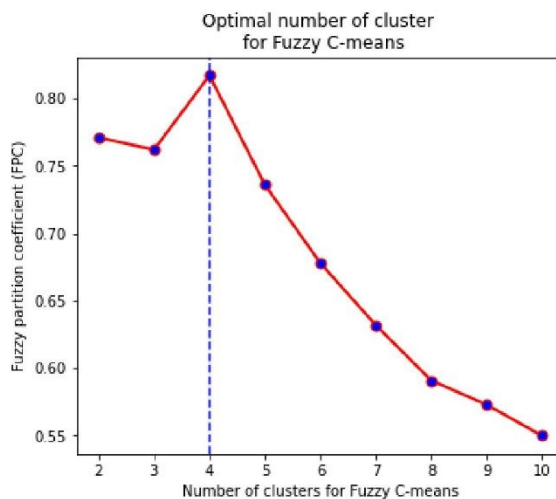
```
In [6]: np.random.seed(104729)
FPC = []
K=[]
for k in range(2, 11):
    # Perform fuzzy c-means clustering
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
        D.T, k, m=2, error=0.0005, maxiter=1000, init=None)
    FPC.append(fpc)
    K.append(k)

K
FPC
```

```
In [8]: k=pd.DataFrame(FPC).idxmax()+2
k
k[0]
```

Out[8]: 4

```
In [9]: plt.figure(figsize = (6, 5))
plt.plot(range(2, 11), np.round(FPC,3), ls='-',
        linewidth=2, color='red',marker='o',
        markerfacecolor='blue', markersize=7)
plt.axvline(x=k[0], color='b',ls='--',linewidth=1.5)
plt.title('Optimal number of cluster \n for Fuzzy C-means')
plt.xlabel('Number of clusters for Fuzzy C-means')
plt.ylabel('Fuzzy partition coefficient (FPC)')
plt.savefig('D:/Fuzzy C-Means_Optimal number of Cluster.jpg',dpi=700)
plt.show()
```



```
In [10]: # Perform fuzzy c-means clustering
np.random.seed(199999)
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
    D.T, k[0], m=2, error=0.0005, maxiter=1000, init=None)
```

```
In [11]: # Print results
print("Cluster centers:\n", cntr)
print("\nWithin-cluster sum of squares:\n", jm)
print("\nFinal partition matrix:\n", np.round(u,3))
print("\nNumber of iterations:\n", p)
print("\nThe fuzzy partition coefficient (FPC):\n", fpc)
```

Cluster centers:

```
[[-1.39414161  7.78950882]
 [-1.57973953  2.81334601]
 [ 1.98497358  0.85507323]
 [ 0.92114413  4.43037045]]
```

Within-cluster sum of squares:

```
[894.72610656 696.36791952 673.85921402 604.14025834 463.49898089
 345.46936153 282.79395782 209.09360959 175.14037931 171.48453836
 171.23319314 171.21564553 171.21435508 171.21425655 171.21424887]
```

Final partition matrix:

```
[[0.038 0.981 0.033 ... 0.016 0.695 0.002]
 [0.223 0.007 0.036 ... 0.032 0.076 0.991]
 [0.474 0.003 0.024 ... 0.014 0.043 0.003]
 [0.266 0.01 0.907 ... 0.937 0.186 0.005]]
```

Number of iterations:

15

The fuzzy partition coefficient (FPC):

0.8173107841333914

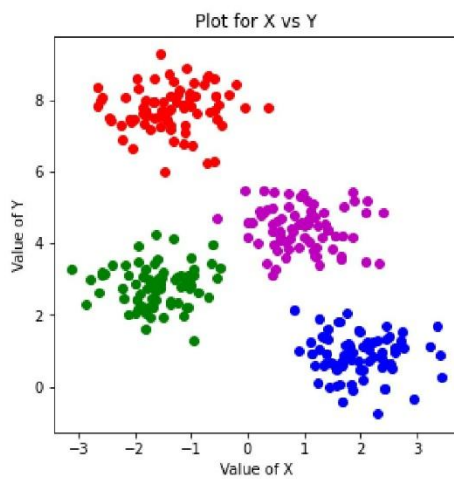
```
In [10]: # Predict cluster membership for each data point
cluster_membership = np.argmax(u, axis=0)
cluster_membership
# Print the cluster membership for each data point
print('Cluster Membership:', cluster_membership)
```

```
Cluster Membership: [2 0 3 0 2 2 1 3 0 0 1 0 3 0 2 3 3 2 1 1 2 2 3 1 1 2 2 3 1 1 3 2 3 1 3 0 0 3 0 0
0 0
0 1 2 3 1 3 3 1 1 0 1 0 2 1 2 0 2 2 1 0 1 0 2 0 3 0 1 1 1 0 2 0 1 3 1 0 1
1 0 1 3 2 0 2 3 2 2 0 3 2 3 0 0 3 2 0 1 1 3 2 2 3 1 0 2 0 2 3 2 2 3 0 3 1
1 2 0 2 3 0 2 2 3 1 2 1 2 2 2 2 1 2 1 0 1 1 2 0 1 1 0 3 0 0 1 3 1 3 1 0 3
0 0 0 3 0 3 2 1 0 1 2 3 0 3 3 2 3 1 1 3 2 3 3 0 2 3 1 0 2 2 3 1 2 3 1 1 3
3 3 3 2 0 3 1 3 3 1 1 1 3 1 0 3 1 2 1 3 0 1 0 3 0 3 1 3 3 0 1 1 2 2 3 0 2
2 1 2 1 3 0 0 3 3 0 3 2 1 3 2 1 0 1 2 3 2 0 0 0 0 1 1 0 3 1 2 3 1 1 1 2 2
0 3 3 1 2 0 1 3 0 3 2 2 1 1 3 2 2 2 3 0 0 2 2 3 2 2 2 0 1 0 3 2 2 0 0 0 2
2 3 0 1]
```

```
In [11]: # Create a list of colors for each cluster
colors = ['r', 'g', 'b', 'm', 'c', 'y', 'k']

# Create a figure with specified size
plt.figure(figsize=(5, 5))
# Plot the data points with their respective cluster colors
for i in range(len(D)):
    plt.plot(D.iloc[i, 0], D.iloc[i, 1], 'o',
             color=colors[cluster_membership[i]])

# Set the x-axis and y-axis labels
plt.xlabel("Value of X")
plt.ylabel("Value of Y")
plt.title('Plot for X vs Y')
plt.savefig('D:/Fuzzy C-Means_Scatter plot for Solution.jpg',dpi=1000)
plt.show()
```



In []: