**Lab 7: Clustering**

1. **Use K means algorithm to cluster the given datapoints where K=3:**

**A1(2, 10), A2(2,5), A3(8, 4),**

**B1(5, 8), B2(7,5), B3(6, 4),**

**C1(1, 2), C2(4, 9)**

import numpy as np

from sklearn.cluster import KMeans

# Define the data points

data\_points = np.array([

    [2, 10], [2, 5], [8, 4],

    [5, 8], [7, 5], [6, 4],

    [1, 2], [4, 9]

])

# Apply KMeans clustering with K=3

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(data\_points)

# Get the labels (which cluster each point belongs to) and centroids

labels = kmeans.labels\_

centroids = kmeans.cluster\_centers\_

# Print the cluster labels for each data point

print("Cluster Labels:", labels)

# Print the centroids of the clusters

print("Centroids of the clusters:", centroids)

# Initialize a dictionary to store the clusters

clusters = {0: [], 1: [], 2: []}

# Organize data points into clusters

for i, label in enumerate(labels):

    clusters[label].append(tuple(data\_points[i]))

# Print the clusters in the requested format

for i in range(3):

    print(f"Cluster {i+1}: {', '.join(map(str, clusters[i]))}")

#Code for viaualization

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))

# Plot the data points with different colors for each cluster

for i in range(3):

    plt.scatter([data\_points[j][0] for j in range(len(data\_points)) if labels[j] == i],

                [data\_points[j][1] for j in range(len(data\_points)) if labels[j] == i],

                label=f"Cluster {i+1}")

# Plot the centroids

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x', s=200, label="Centroids")

# Label the clusters

plt.title('K-Means Clustering (K=3) with Centroids')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.legend()

1. **Use DBSCAN algorithm to cluster the given datapoints where minpoints=3 and Epsilon =3:**

**A1(2, 10), A2(2,5), A3(8, 4), B1(5, 8), B2(7,5), B3(6, 4), C1(1, 2), C2(4, 9)**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

# Define the datapoints

points = {

    "A1": (2, 10),

    "A2": (2, 5),

    "A3": (8, 4),

    "B1": (5, 8),

    "B2": (7, 5),

    "B3": (6, 4),

    "C1": (1, 2),

    "C2": (4, 9)

}

# Extract coordinates and labels

labels = list(points.keys())

X = np.array(list(points.values()))

# Apply DBSCAN

dbscan = DBSCAN(eps=3, min\_samples=3)

clusters = dbscan.fit\_predict(X)

# Plotting the results

plt.figure(figsize=(8, 6))

unique\_clusters = set(clusters)

colors = ['red', 'green', 'blue', 'orange', 'purple']

for cluster\_id in unique\_clusters:

    cluster\_points = X[clusters == cluster\_id]

    plt.scatter(cluster\_points[:, 0], cluster\_points[:, 1],

                label=f'Cluster {cluster\_id}' if cluster\_id != -1 else 'Noise',

                color=colors[cluster\_id % len(colors)] if cluster\_id != -1 else 'black')

# Annotate the points with their labels

for i, label in enumerate(labels):

    plt.annotate(label, (X[i, 0]+0.1, X[i, 1]+0.1))

plt.title("DBSCAN Clustering (eps=3, min\_samples=3)")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.legend()

plt.grid(True)

plt.show()