CSPC-54: INTRO TO AI AND ML

LAB - 8: REPORT

ASWAKTH BALAJI
106123017

AIM:

ALGORITHM:

CODE:

AGGLOMERATIVE CLUSTERING:

```
# ---- Agglomerative Clustering ----
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

# Step 1: Create a sample dataset
X, y = make_blobs(n_samples=150, centers=3, cluster_std=0.6, random_state=42)

# Step 2: Perform Agglomerative Clustering
agg = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = agg.fit_predict(X)

# Step 3: Plot cluster result
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title("Agglomerative Clustering Result")
plt.show()

# Step 4: Dendrogram visualization (optional)
Z = linkage(X, method='ward')
plt.figure(figsize=(8, 4))
dendrogram(Z)
plt.title("Dendrogram - Hierarchical Clustering")
plt.show()
```

FUZZY C-MEANS CLUSTERING:

```
# ---- Fuzzy C-Means Clustering ----
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import skfuzzy as fuzz

# Step 1: Create the dataset
X, y = make_blobs(n_samples=150, centers=3, cluster_std=0.6, random_state=42)

# Step 2: Perform Fuzzy C-Means Clustering
# Transpose X to match the expected input shape (features x samples)
cntr, u, _, _, _, _, _ = fuzz.cluster.cmeans(
    X.T, c=3, m=2, error=0.005, maxiter=1000, init=None)
```

```
# Step 3: Find hard clusters (maximum membership)
labels = np.argmax(u, axis=0)

# Step 4: Plot cluster result
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title("Fuzzy C-Means Clustering Result")
plt.show()

# Optional: print membership of first 5 samples
print("Membership values (first 5 points):")
print(u[:, :5])
```

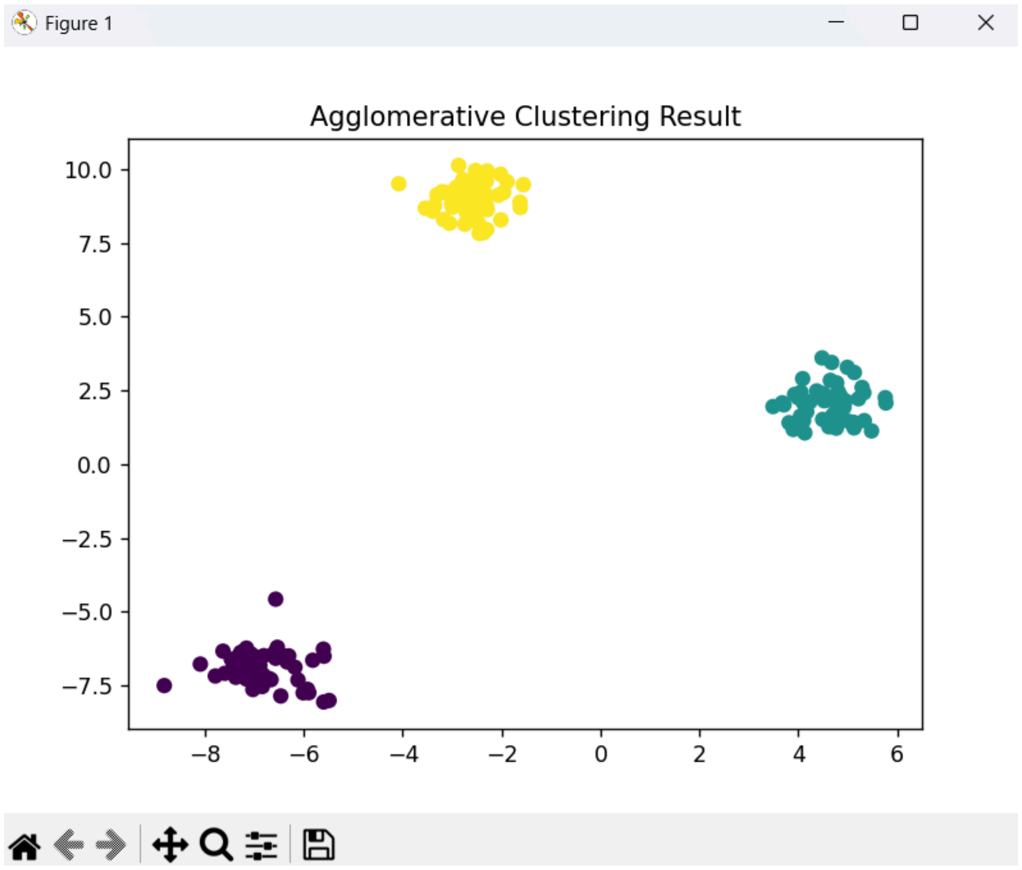
DIVISIVE CLUSTERING:

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

def divisive_clustering(X, max_depth=1, labels=None, current_label=0):
    if labels is None:
        labels = np.zeros(X.shape[0], dtype=int)
    if max_depth == 0 or len(X) < 2:
        return labels

    km = KMeans(n_clusters=2, random_state=42)
    sub_labels = km.fit_predict(X)

    max_label = labels.max()
    new_labels = labels.copy()

    new_labels[sub_labels == 1] = max_label + 1

    idx0 = np.where(new_labels == max_label)[0]
    idx1 = np.where(new_labels == max_label + 1)[0]

    if len(idx0) > 1:
        new_labels[idx0] = divisive_clustering(X[idx0], max_depth - 1,
new_labels[idx0], max_label)
    if len(idx1) > 1:
        new_labels[idx1] = divisive_clustering(X[idx1], max_depth - 1,
new_labels[idx1], max_label + 1)

    return new_labels

# Generate synthetic data with 4 clusters
X, _ = make_blobs(n_samples=100, centers=4, cluster_std=1.0, random_state=42)
```

```
labels = divisive_clustering(X, max_depth=2)

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50)
plt.title('Divisive Clustering with make_blobs data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

OUTPUT:

1)



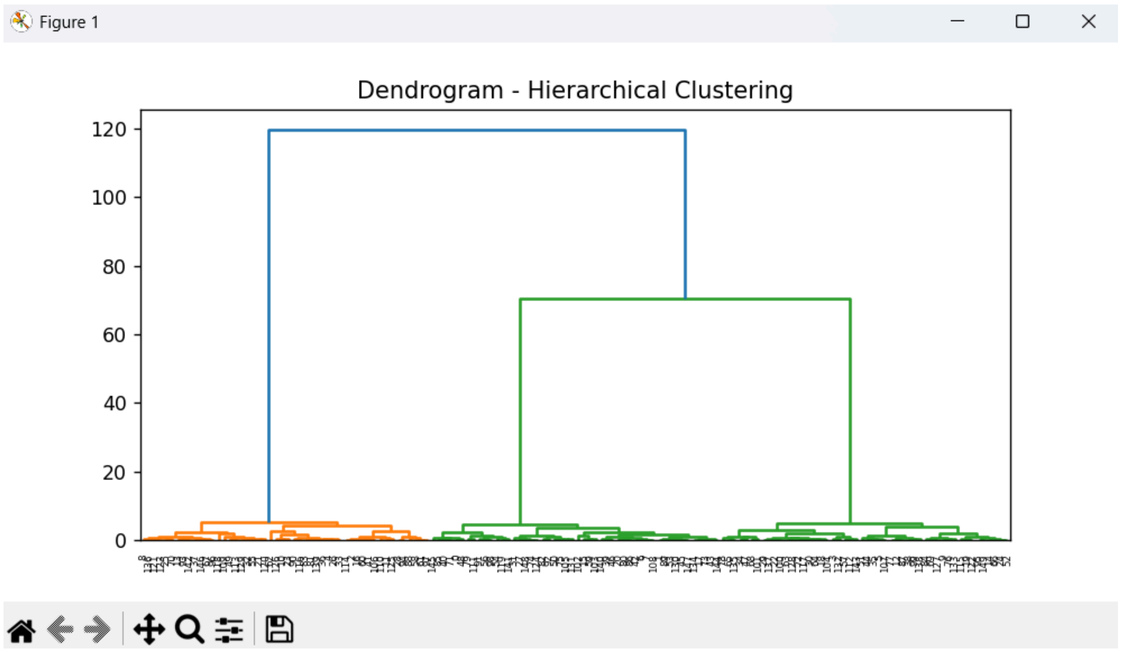


Figure 1
Dendrogram - Hierarchical Clustering
120
100
80
60
40
20
0

2)

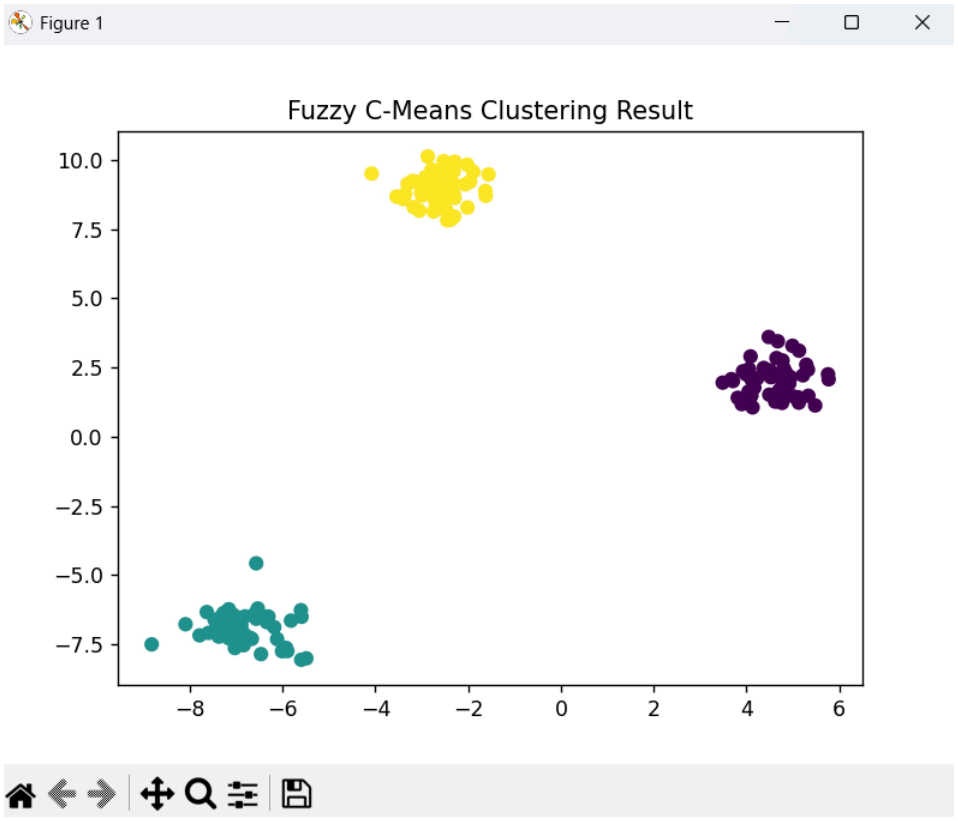


Figure 1
Fuzzy C-Means Clustering Result
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−8
−6
−4
−2
0
2
4
6

3)

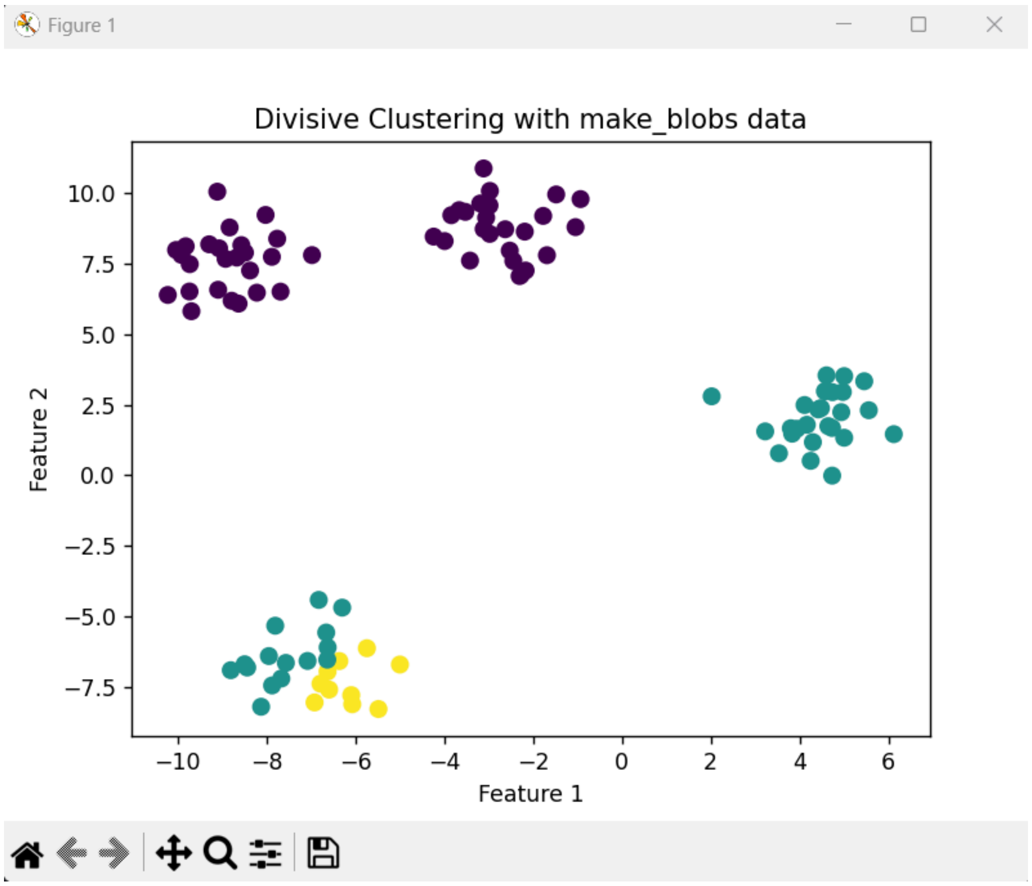


Figure 1
Divisive Clustering with make_blobs data
Feature 2
Feature 1
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10
−8
−6
−4
−2
0
2
4
6