**AML Algorithm #11 : Divisive hiererchical Clustering**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

class DivisiveHierarchicalClustering:

def \_\_init\_\_(self, data):

self.data = data

self.num\_samples, self.num\_features = data.shape

self.clusters = [[i] for i in range(self.num\_samples)]

def euclidean\_distance(self, a, b):

return np.linalg.norm(a - b)

def find\_closest\_clusters(self):

min\_distance = float('inf')

closest\_clusters = None

for i in range(len(self.clusters)):

for j in range(i + 1, len(self.clusters)):

distance = self.compute\_cluster\_distance(self.clusters[i], self.clusters[j])

if distance < min\_distance:

min\_distance = distance

closest\_clusters = (i, j)

return closest\_clusters

def compute\_cluster\_distance(self, cluster\_a, cluster\_b):

distance\_sum = 0

for i in cluster\_a:

for j in cluster\_b:

distance\_sum += self.euclidean\_distance(self.data[i], self.data[j])

return distance\_sum / (len(cluster\_a) \* len(cluster\_b))

def divisive\_clustering(self, k):

while len(self.clusters) < k:

i, j = self.find\_closest\_clusters()

self.merge\_clusters(i, j)

def merge\_clusters(self, i, j):

merged\_cluster = self.clusters[i] + self.clusters[j]

del self.clusters[j]

self.clusters[i] = merged\_cluster

def plot\_clusters(self):

for i, cluster in enumerate(self.clusters):

points = self.data[cluster]

plt.scatter(points[:, 0], points[:, 1], label=f'Cluster {i + 1}')

plt.title('Divisive Hierarchical Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

# Load the Iris dataset

iris = load\_iris()

data = iris.data[:, :2] # Use only the first two features for simplicity

# User input for the number of clusters

num\_clusters = int(input("Enter the number of clusters: "))

# Perform divisive clustering

divisive\_clusterer = DivisiveHierarchicalClustering(data)

divisive\_clusterer.divisive\_clustering(k=num\_clusters)

divisive\_clusterer.plot\_clusters()

**OUTPUT :**

