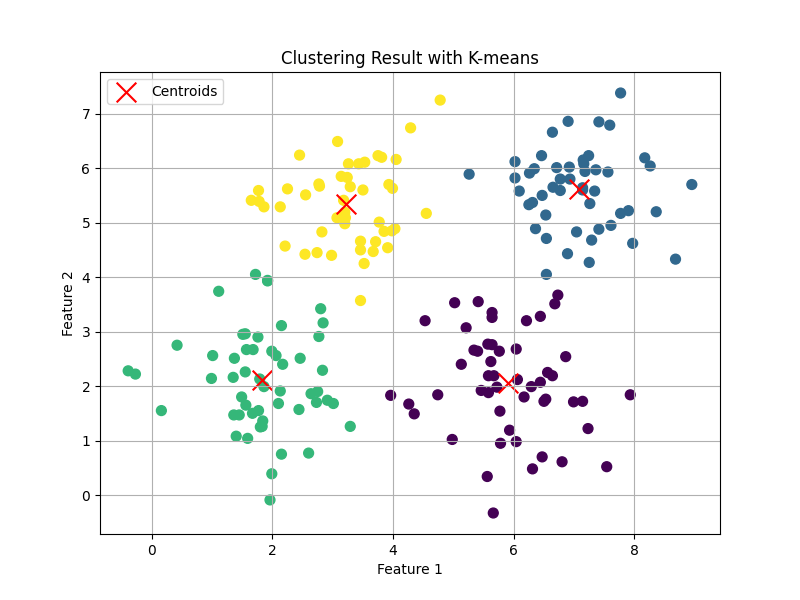
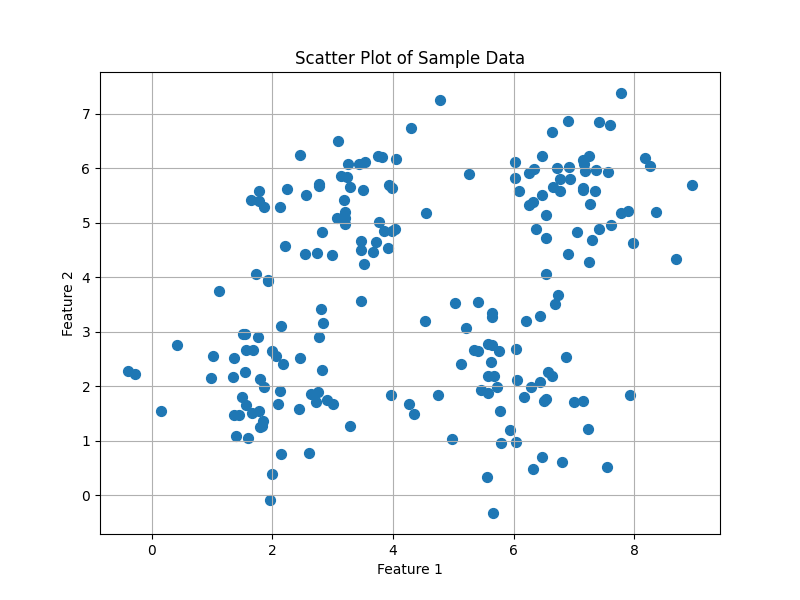
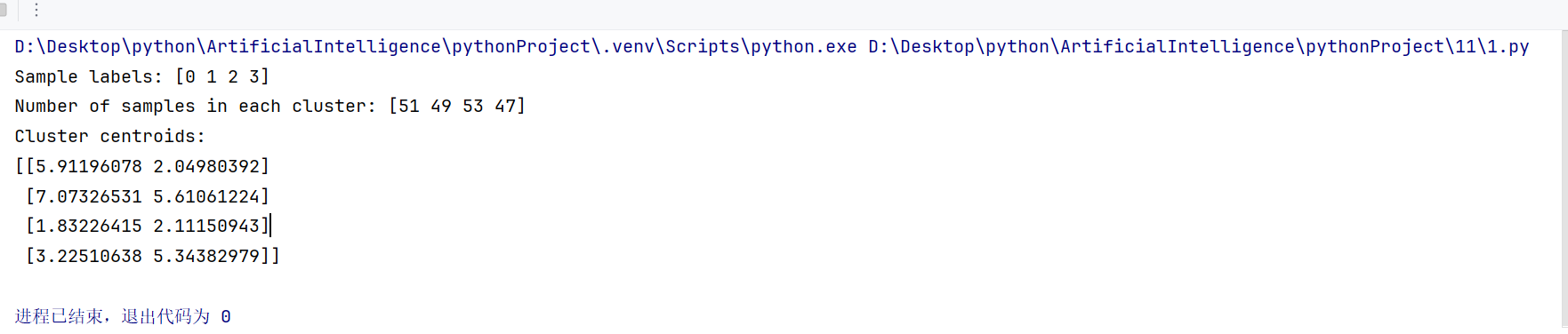
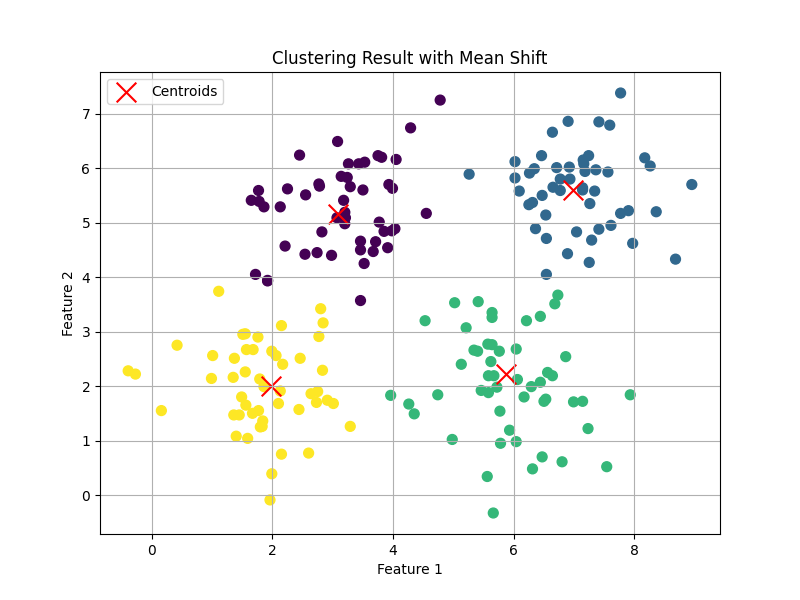
1. 读取data\_multivar\_cluster.txt数据，使用scatter（）将样本数据可视化，根据画图确定簇的个数，使用K-means聚类算法训练样本数据，输出样本label及label的个数，聚类后的中心点，并将中心点画在原来的样本图中。

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
  
# 读取数据  
data = np.loadtxt('data\_multivar\_cluster.txt',delimiter=',')  
  
# 将数据可视化  
plt.figure(figsize=(8, 6))  
plt.scatter(data[:, 0], data[:, 1], s=50)  
plt.title('Scatter Plot of Sample Data')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.grid(True)  
plt.show()  
# 使用K-means算法聚类数据  
k = 4 # 簇的个数  
kmeans = KMeans(n\_clusters=k)  
kmeans.fit(data)  
  
# 输出样本label及label的个数  
labels = kmeans.labels\_  
unique\_labels, label\_counts = np.unique(labels, return\_counts=True)  
print("Sample labels:", unique\_labels)  
print("Number of samples in each cluster:", label\_counts)  
  
# 输出聚类后的中心点  
centroids = kmeans.cluster\_centers\_  
print("Cluster centroids:")  
print(centroids)  
  
# 可视化聚类结果  
plt.figure(figsize=(8, 6))  
plt.scatter(data[:, 0], data[:, 1], c=labels, s=50, cmap='viridis')  
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200, color='red', label='Centroids')  
plt.title('Clustering Result with K-means')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.legend()  
plt.grid(True)  
plt.show()



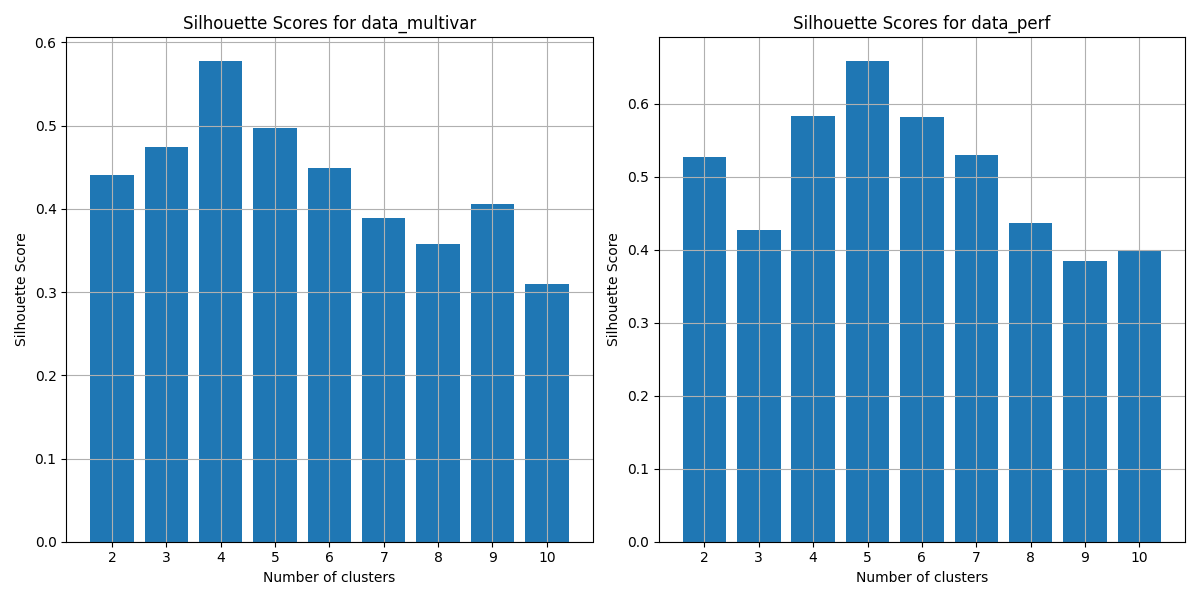
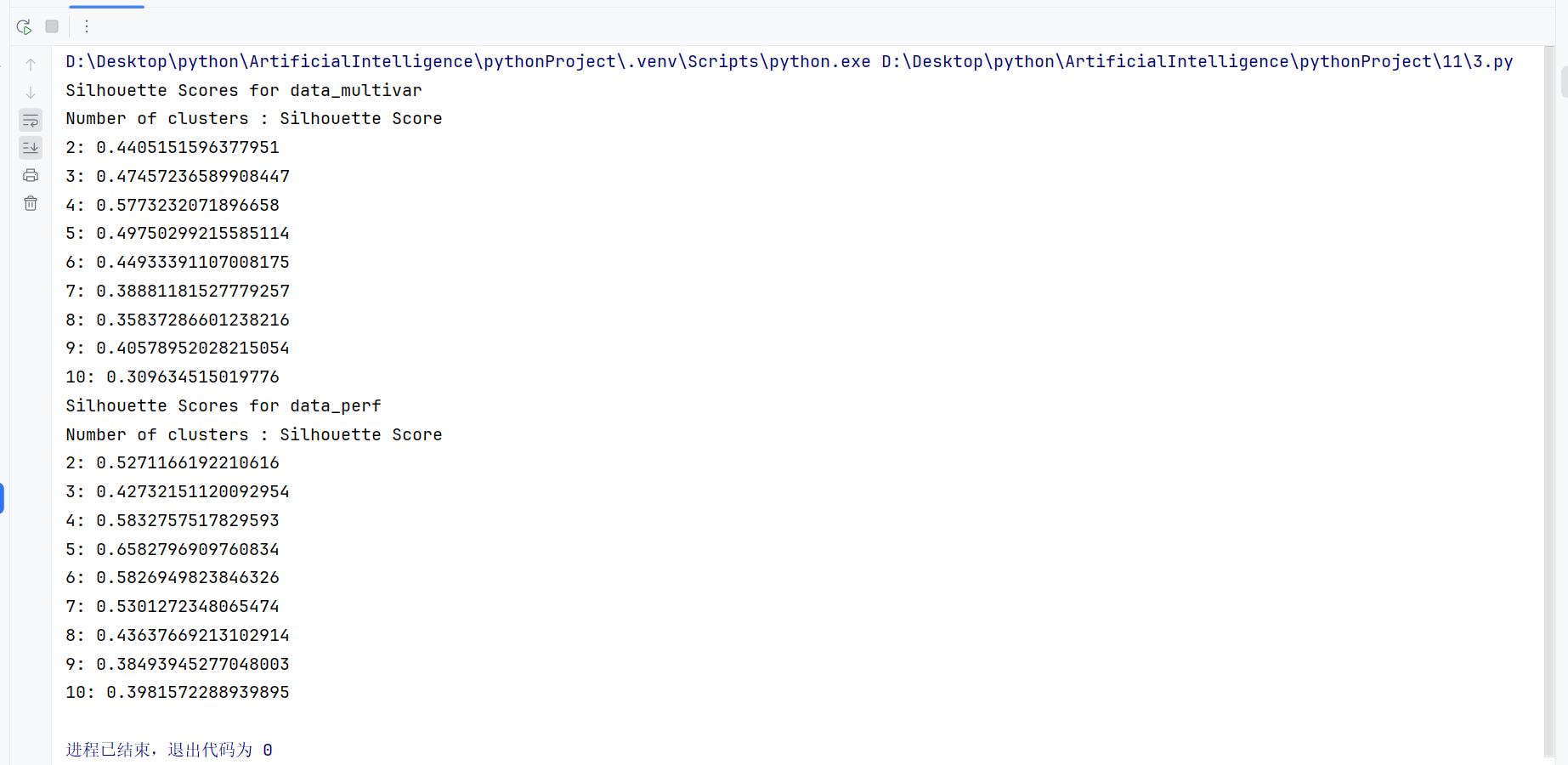
1. 使用mean-shift聚类算法训练data\_multivar.txt数据，输出样本label及label的个数，聚类后的中心点，一并画出样本数据及中心点

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
  
# 读取数据  
data = np.loadtxt('data\_multivar\_cluster.txt',delimiter=',')  
  
# 估计带宽（bandwidth）  
bandwidth = estimate\_bandwidth(data, quantile=0.2, n\_samples=500)  
  
# 使用mean-shift算法聚类数据  
ms = MeanShift(bandwidth=bandwidth, bin\_seeding=True)  
ms.fit(data)  
  
# 输出样本label及label的个数  
labels = ms.labels\_  
unique\_labels, label\_counts = np.unique(labels, return\_counts=True)  
print("Sample labels:", unique\_labels)  
print("Number of samples in each cluster:", label\_counts)  
  
# 输出聚类后的中心点  
centroids = ms.cluster\_centers\_  
print("Cluster centroids:")  
print(centroids)  
  
# 可视化聚类结果  
plt.figure(figsize=(8, 6))  
plt.scatter(data[:, 0], data[:, 1], c=labels, s=50, cmap='viridis')  
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=200, color='red', label='Centroids')  
plt.title('Clustering Result with Mean Shift')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.legend()  
plt.grid(True)  
plt.show()



1. 分别读取data\_multivar.txt和data\_perf.txt数据，使用轮廓系数得分评估k-means ，输出在簇个数分别为2-10上的得分，使用条形图显示得分结果

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
  
# 读取数据  
data\_multivar = np.loadtxt('data\_multivar\_cluster.txt', delimiter=',')  
data\_perf = np.loadtxt('data\_perf.txt', delimiter=',')  
  
  
def calculate\_silhouette\_scores(data, min\_clusters=2, max\_clusters=10):  
 silhouette\_scores = []  
 for n\_clusters in range(min\_clusters, max\_clusters + 1):  
 kmeans = KMeans(n\_clusters=n\_clusters)  
 labels = kmeans.fit\_predict(data)  
 silhouette\_avg = silhouette\_score(data, labels)  
 silhouette\_scores.append(silhouette\_avg)  
 return silhouette\_scores  
  
  
min\_clusters = 2  
max\_clusters = 10  
  
silhouette\_scores\_multivar = calculate\_silhouette\_scores(data\_multivar, min\_clusters, max\_clusters)  
silhouette\_scores\_perf = calculate\_silhouette\_scores(data\_perf, min\_clusters, max\_clusters)  
plt.figure(figsize=(12, 6))  
  
plt.subplot(1, 2, 1)  
plt.bar(range(min\_clusters, max\_clusters + 1), silhouette\_scores\_multivar)  
plt.title('Silhouette Scores for data\_multivar')  
plt.xlabel('Number of clusters')  
plt.ylabel('Silhouette Score')  
print('Silhouette Scores for data\_multivar')  
print('Number of clusters : Silhouette Score')  
for i in range(min\_clusters, max\_clusters + 1):  
 print(f'{i}: {silhouette\_scores\_multivar[i-min\_clusters]}')  
plt.xticks(range(min\_clusters, max\_clusters + 1))  
plt.grid(True)  
  
plt.subplot(1, 2, 2)  
plt.bar(range(min\_clusters, max\_clusters + 1), silhouette\_scores\_perf)  
plt.title('Silhouette Scores for data\_perf')  
plt.xlabel('Number of clusters')  
plt.ylabel('Silhouette Score')  
print('Silhouette Scores for data\_perf')  
print('Number of clusters : Silhouette Score')  
for i in range(min\_clusters, max\_clusters + 1):  
 print(f'{i}: {silhouette\_scores\_perf[i-min\_clusters]}')  
plt.xticks(range(min\_clusters, max\_clusters + 1))  
plt.grid(True)  
  
plt.tight\_layout()  
plt.show()



4、使用DBSCAN模型对数据data\_perf.txt数据聚类，

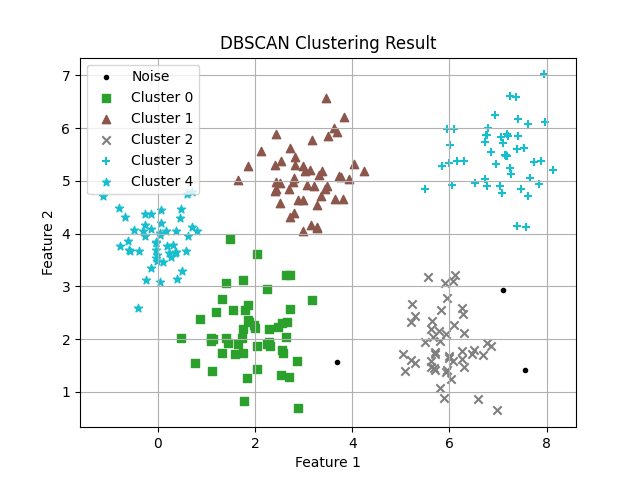
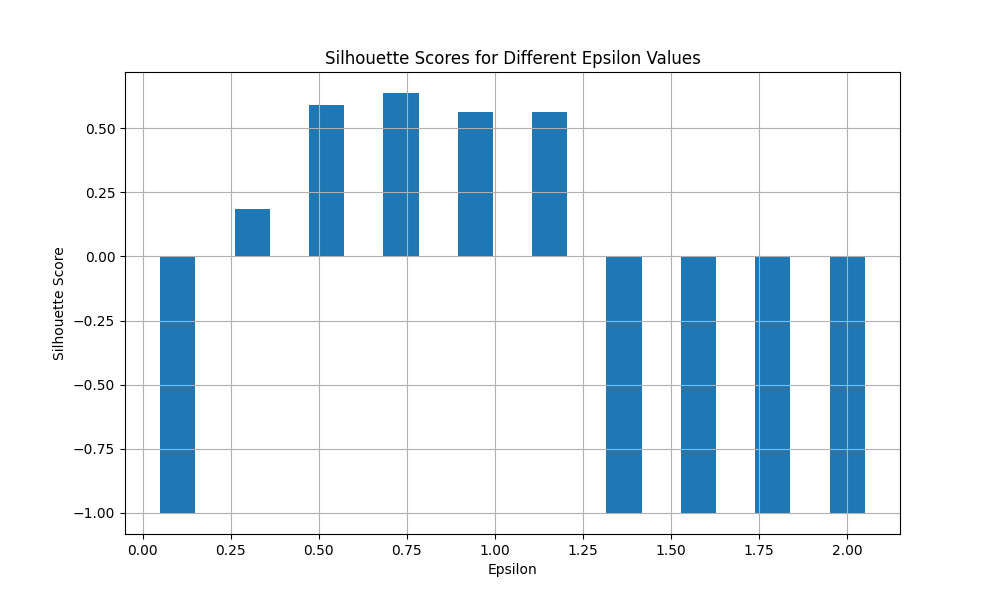
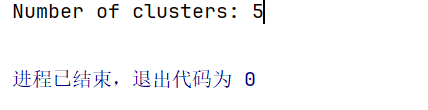
（1）分别使用10个距离参数统计轮廓系数得分（将10个轮廓系数得分画条形图）

（2）返回轮廓系数得分最大的距离参数值，model，labels，

（3）查找labels中是否有未聚类的点（labels为-1）,输出最终聚类的个数

（4）分别不用不同的marker（根据类别个数）画出样本数据，未聚类的使用标记’.‘

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import DBSCAN  
from sklearn.metrics import silhouette\_score  
  
# 读取数据  
data\_perf = np.loadtxt('data\_perf.txt', delimiter=',')  
  
def calculate\_silhouette\_scores(data, epsilon\_values):  
 """  
 计算不同距离参数值下的轮廓系数得分  
 """  
 silhouette\_scores = []  
 for epsilon in epsilon\_values:  
 dbscan = DBSCAN(eps=epsilon)  
 labels = dbscan.fit\_predict(data)  
 if len(np.unique(labels)) > 1: # 至少有两个类别才能计算轮廓系数  
 silhouette\_avg = silhouette\_score(data, labels)  
 else:  
 silhouette\_avg = -1 # 如果只有一个类别，则轮廓系数为-1  
 silhouette\_scores.append(silhouette\_avg)  
 return silhouette\_scores  
  
def find\_best\_epsilon(data, epsilon\_values):  
 """  
 找到轮廓系数得分最大的距离参数值  
 """  
 best\_epsilon = None  
 best\_score = -1  
 best\_model = None  
 best\_labels = None  
 for epsilon in epsilon\_values:  
 dbscan = DBSCAN(eps=epsilon)  
 labels = dbscan.fit\_predict(data)  
 if len(np.unique(labels)) > 1: # 至少有两个类别才能计算轮廓系数  
 silhouette\_avg = silhouette\_score(data, labels)  
 if silhouette\_avg > best\_score:  
 best\_score = silhouette\_avg  
 best\_epsilon = epsilon  
 best\_model = dbscan  
 best\_labels = labels  
 return best\_epsilon, best\_model, best\_labels  
  
def check\_unclustered\_points(labels):  
 """  
 检查标签中是否有未聚类的点，并输出最终聚类的个数  
 """  
 unique\_labels = np.unique(labels)  
 if -1 in unique\_labels:  
 num\_clusters = len(unique\_labels) - 1  
 else:  
 num\_clusters = len(unique\_labels)  
 return num\_clusters  
  
  
def plot\_clusters(data, labels):  
 """  
 绘制聚类结果  
 """  
 unique\_labels = np.unique(labels)  
 num\_clusters = len(unique\_labels) - 1 if -1 in unique\_labels else len(unique\_labels)  
 markers = ['o', 's', '^', 'x', '+', '\*', 'p', 'h', 'd', '|']  
 colors = plt.cm.get\_cmap('tab10', num\_clusters)  
 for i, label in enumerate(unique\_labels):  
 if label == -1:  
 plt.scatter(data[labels == label, 0], data[labels == label, 1], marker='.', color='k', label='Noise')  
 else:  
 plt.scatter(data[labels == label, 0], data[labels == label, 1], marker=markers[i], color=colors(i),  
 label=f'Cluster {label}')  
 plt.xlabel('Feature 1')  
 plt.ylabel('Feature 2')  
 plt.title('DBSCAN Clustering Result')  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
# 使用10个距离参数统计轮廓系数得分  
epsilon\_values = np.linspace(0.1, 2.0, 10)  
silhouette\_scores = calculate\_silhouette\_scores(data\_perf, epsilon\_values)  
  
# 绘制轮廓系数得分条形图  
plt.figure(figsize=(10, 6))  
plt.bar(epsilon\_values, silhouette\_scores, width=0.1)  
plt.xlabel('Epsilon')  
plt.ylabel('Silhouette Score')  
plt.title('Silhouette Scores for Different Epsilon Values')  
plt.grid(True)  
plt.show()  
  
# 返回轮廓系数得分最大的距离参数值，model，labels  
best\_epsilon, best\_model, best\_labels = find\_best\_epsilon(data\_perf, epsilon\_values)  
# print(f"Best epsilon value: {best\_epsilon}")  
  
# 查找labels中是否有未聚类的点，输出最终聚类的个数  
num\_clusters = check\_unclustered\_points(best\_labels)  
print(f"Number of clusters: {num\_clusters}")  
  
# 分别不用不同的marker（根据类别个数）画出样本数据，未聚类的使用标记’.‘  
plot\_clusters(data\_perf, best\_labels)



5使用AGNES模型训练分别聚类三种生成的数据，参数connectivitys使用两种方式任选其一，画出聚类后的数据样本图，并分别计算其轮廓系数得分输出.

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import silhouette\_score  
  
  
def train\_and\_evaluate(data, connectivity=None):  
 """  
 训练 AGNES 模型并计算轮廓系数得分  
 """  
 # 训练 AGNES 模型  
 AgglomerativeClustering()  
 agnes = AgglomerativeClustering(n\_clusters=5, connectivity=connectivity)  
 labels = agnes.fit\_predict(data)  
  
 # 计算轮廓系数得分  
 silhouette\_avg = silhouette\_score(data, labels)  
  
 # 绘制聚类结果  
 plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis')  
 plt.title('Agglomerative Clustering Result')  
 plt.xlabel('Feature 1')  
 plt.ylabel('Feature 2')  
 plt.colorbar(label='Cluster')  
 plt.grid(True)  
 plt.show()  
  
 return silhouette\_avg  
  
  
# 生成三种不同形状的数据  
from sklearn.cluster import AgglomerativeClustering  
  
def add\_noise(x, y, amplitude):  
 X = np.concatenate((x, y))  
 X += amplitude \* np.random.randn(2, X.shape[1])  
 return X.T  
def get\_spiral(t, noise\_amplitude=0.5):  
 r = t  
 x = r \* np.cos(t)  
 y = r \* np.sin(t)  
 return add\_noise(x, y, noise\_amplitude)  
def get\_rose(t, noise\_amplitude=0.02):  
 # Equation for "rose" (or rhodonea curve); if k is odd, then  
 # the curve will have k petals, else it will have 2k petals  
 k = 5  
 r = np.cos(k\*t) + 0.25  
 x = r \* np.cos(t)  
 y = r \* np.sin(t)  
 return add\_noise(x, y, noise\_amplitude)  
def get\_hypotrochoid(t, noise\_amplitude=0):  
 a, b, h = 10.0, 2.0, 4.0  
 x = (a - b) \* np.cos(t) + h \* np.cos((a - b) / b \* t)  
 y = (a - b) \* np.sin(t) - h \* np.sin((a - b) / b \* t)  
 return add\_noise(x, y, 0)  
  
  
n\_samples=500  
np.random.seed(2)  
t = 2.5 \* np.pi \* (1 + 2 \* np.random.rand(1, n\_samples))  
# X = get\_spiral(t)  
X = get\_rose(t)  
# X = get\_hypotrochoid(t)  
  
silhouette\_score = train\_and\_evaluate(X)  
  
# 输出轮廓系数得分  
print("Silhouette Score for Data:", silhouette\_score)

