**一、**简述**Kmeans及DBSCAN聚类算法**的原理、关键操作点（包括intertia、n\_clusters、algorithm、eps、min\_samples等参数的调参等，不要超过**300字**，逻辑要通，**要有小标题**，要排版清晰，请不要网络大幅摘抄）

**1.K均值聚类算法**1.1原理：K均值聚类通过迭代将样本分配到K个簇中，使得每个样本到其所属簇的中心点的距离最小化，从而实现簇内样本的紧密度最大化。关键操作点：1.2 inertia：簇内误差平方和，衡量簇内样本的紧密度。1.3 n\_clusters：指定簇的数量。1.4 algorithm：用于计算的算法，如"auto"、"full"或"elkan"。1.5参数调参：通过调整n\_clusters选择最优簇数，以最小化inertia。

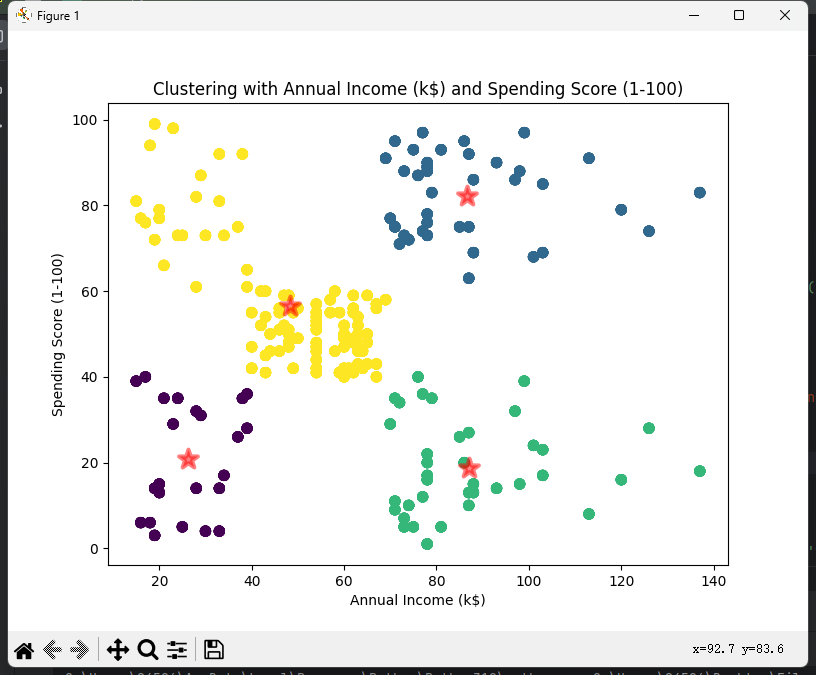
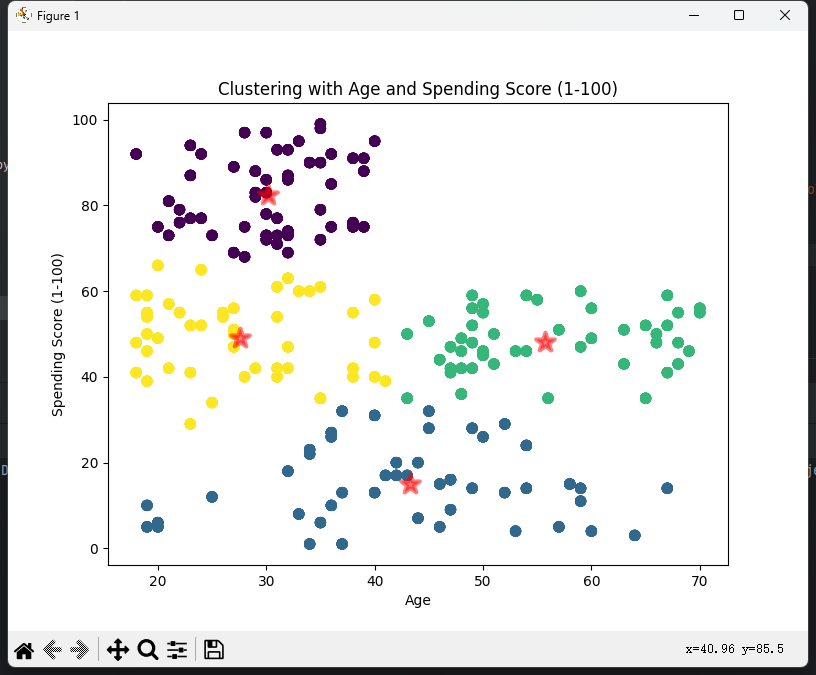
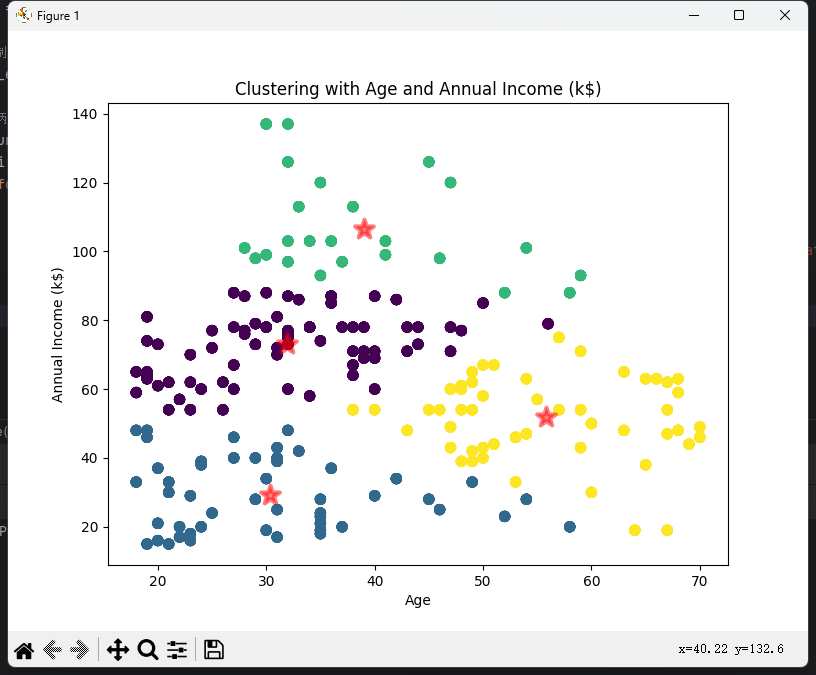
**2.DBSCAN聚类算法**2.1原理：DBSCAN根据密度将样本分配到不同的簇中，能够发现任意形状的簇，对噪声数据具有鲁棒性。关键操作点：2.2 eps：指定邻域半径，影响簇的形成。2.3 min\_samples：邻域内最小样本数，影响簇的紧密度。2.4参数调参：通过调整eps和min\_samples来适应不同密度的数据，以发现合适的簇结构

**二、利用聚类算法解决问题。**

# 1.书本中P164,利用Kmeans算法对于“mall\_customers.csb数据集”进行聚类分析。1）将原有200条数据拟合出2000条；2）除CustomerID特征外的其他4个特征2组合、3组合、4组合不同情形下的聚类情况分析。

import pandas as pd  
import numpy as np  
from sklearn.cluster import KMeans  
import matplotlib.pyplot as plt  
data = pd.read\_csv('Mall\_Customers1.csv')  
  
# 复制数据以生成2000条数据  
data\_extended = pd.concat([data] \* 10, ignore\_index=True)  
  
# 两两特征组合的聚类分析  
features\_2 = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']  
for i in range(len(features\_2)):  
 for j in range(i+1, len(features\_2)):  
 features = [features\_2[i], features\_2[j]]  
 x = data\_extended[features].values  
 algorithm = KMeans(n\_clusters=4, init='k-means++', n\_init=10, max\_iter=300, tol=0.0001, random\_state=111,  
 algorithm='elkan')  
 algorithm.fit(x)  
 labels = algorithm.labels\_  
 centroids = algorithm.cluster\_centers\_  
 # 绘制聚类结果  
 plt.figure(figsize=(8, 6))  
 plt.scatter(x[:, 0], x[:, 1], c=labels, s=50, cmap='viridis')  
 plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, alpha=0.5, marker='\*', linewidth=3)  
 plt.xlabel(features[0])  
 plt.ylabel(features[1])  
 plt.title(f'Clustering with {features[0]} and {features[1]}')  
 plt.show()  
  
# 三个特征组合的聚类分析  
for i in range(len(features\_2)):  
 for j in range(i+1, len(features\_2)):  
 for k in range(j+1, len(features\_2)):  
 features = [features\_2[i], features\_2[j], features\_2[k]]  
 x = data\_extended[features].values  
 algorithm = KMeans(n\_clusters=4, init='k-means++', n\_init=10, max\_iter=300, tol=0.0001, random\_state=111,  
 algorithm='elkan')  
 algorithm.fit(x)  
 labels = algorithm.labels\_  
 # 展示聚类结果的数量  
 unique\_labels = len(np.unique(labels))  
 print(f'Number of clusters with {features}: {unique\_labels}')  
  
# 四个特征组合的聚类分析  
x = data\_extended[features\_2].values  
algorithm = KMeans(n\_clusters=4, init='k-means++', n\_init=10, max\_iter=300, tol=0.0001, random\_state=111,  
 algorithm='elkan')  
algorithm.fit(x)  
labels = algorithm.labels\_  
  
# 展示四个特征组合的聚类结果的数量  
unique\_labels = len(np.unique(labels))  
print({unique\_labels})

**结果：**

****

**2.**大学校园网的日志数据，学生的校园网使用情况数据，包括学号、性别、开始上网时间、停止上网时间及上网时长等。

如下某个学生的上网时间，先构造500条相关数据，然后用DBSCAN进行聚类分析。

|  |  |
| --- | --- |
| 学号 | 2020012345 |
| 性别 | Femail |
| 开始上网 | 2020-08-29 20:30:10 |
| 停止上网 | 2020-08-29 22:32:14 |
| 上网时长 | 7324 |

实验目的：通过**DBSCAN聚类**，分析学生上网时间和上网时长的模式。

要求：

1）针对当前数据、模型，**详叙数据预处理、模型参数调整过程及效果比对**；

2）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

3）运行结果要截图，结果要文字说明；

4）注意排版

import pandas as pd  
import numpy as np  
from datetime import datetime, timedelta  
  
# 生成模拟数据  
np.random.seed(0)  
# 生成500个学生的学号  
student\_ids = np.random.randint(2020000000, 2020999999, size=500)  
# 生成性别数据  
genders = np.random.choice(['Male', 'Female'], size=500)  
# 生成开始上网时间、停止上网时间和上网时长  
start\_times = []  
stop\_times = []  
durations = []  
# 生成随机的上网时长  
for \_ in range(500):  
 start\_time = datetime(2020, 8, 1) + timedelta(days=np.random.randint(0, 30),  
 hours=np.random.randint(0, 23),  
 minutes=np.random.randint(0, 59),  
 seconds=np.random.randint(0, 59))  
 duration = np.random.randint(600, 3600) # 上网时长在10分钟到1小时之间  
 stop\_time = start\_time + timedelta(seconds=duration)  
 start\_times.append(start\_time)  
 stop\_times.append(stop\_time)  
 durations.append(duration)  
# 构造DataFrame  
data = pd.DataFrame({  
 '学号': student\_ids,  
 '性别': genders,  
 '开始上网时间': start\_times,  
 '停止上网时间': stop\_times,  
 '上网时长': durations  
})  
print(data.head())  
# 数据预处理  
# 将时间特征转换为数值特征  
data['开始上网时间'] = data['开始上网时间'].astype(np.int64) // 10\*\*9 # 将时间戳转换为秒  
data['停止上网时间'] = data['停止上网时间'].astype(np.int64) // 10\*\*9 # 将时间戳转换为秒  
print(data.head())  
from sklearn.cluster import DBSCAN  
from sklearn.preprocessing import StandardScaler  
# 特征标准化  
scaler = StandardScaler()  
X = scaler.fit\_transform(data[['开始上网时间', '停止上网时间', '上网时长']])  
  
# 使用DBSCAN进行聚类分析  
dbscan = DBSCAN(eps=0.5, min\_samples=5)  
data['cluster'] = dbscan.fit\_predict(X)  
print(data['cluster'].value\_counts())  
  
cluster\_means = data.groupby('cluster')[['开始上网时间', '停止上网时间', '上网时长']].mean()  
print(cluster\_means)  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10, 6))  
for cluster in data['cluster'].unique():  
 cluster\_data = data[data['cluster'] == cluster]  
 plt.scatter(cluster\_data['开始上网时间'], cluster\_data['上网时长'], label=f'Cluster {cluster}')  
  
from pylab import \*  
mpl.rcParams['font.sans-serif']=['SimHei']  
mpl.rcParams['axes.unicode\_minus']=False  
plt.xlabel('开始上网时间')  
plt.ylabel('上网时长')  
plt.title('学生上网时间与上网时长的模式')  
plt.legend()  
plt.show()

**结果：**

图表, 散点图

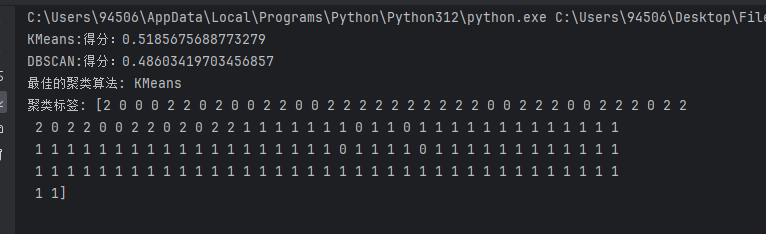
描述已自动生成

可以看到，大部分点集中在开始上网时间较早的区域，并且上网时长相对较短。反映了学生在晚上较早开始上网，且上网时间不会过长的模式。另外，也可以观察到在开始上网时间较晚的区域，上网时长较长的点也存在，这可能反映了另一部分学生在晚上较晚开始上网，并且上网时长较长的行为模式。

**3.**对鸢尾花数据（已删除类别标签）进行聚类，聚类结果与原始类别进行比对，找出最佳的聚类算法及最优参数组合。

from sklearn.datasets import load\_iris  
from sklearn.cluster import KMeans,DBSCAN  
from sklearn.metrics import silhouette\_score  
import numpy as np  
iris = load\_iris()  
X = iris.data  
# 定义要尝试的聚类算法和参数  
clustering\_algorithms = [  
 ('KMeans', KMeans(n\_clusters=3)),  
 ('DBSCAN', DBSCAN(eps=0.5, min\_samples=5))  
]  
  
# 尝试不同的聚类算法  
best\_score = -1  
best\_algorithm = None  
best\_labels = None  
  
for name, algorithm in clustering\_algorithms:  
 algorithm.fit(X)  
 if hasattr(algorithm, 'labels\_'):  
 labels = algorithm.labels\_  
 else:  
 labels = algorithm.predict(X)  
 # 计算轮廓系数评分  
 score = silhouette\_score(X, labels)  
 print(f'{name}:得分：{score}')  
  
 # 更新最佳结果  
 if score > best\_score:  
 best\_score = score  
 best\_algorithm = name  
 best\_labels = labels  
# 输出最佳聚类结果及算法  
print(f'最佳的聚类算法: {best\_algorithm}')  
print(f'聚类标签: {best\_labels}')  
  
import matplotlib.pyplot as plt  
# 提取前两个特征维度  
X2 = X[:, :2]  
# 绘制散点图  
plt.figure(figsize=(10, 6))  
for label in np.unique(best\_labels):  
 plt.scatter(X2[best\_labels == label, 0], X2[best\_labels == label, 1], label=f'Cluster {label}')  
plt.title('Clustering Result')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.legend()  
plt.show()

**结果：**



**K均值算法结果比DBSCAN更优**

