1、聚类方法应用-用户群体划分

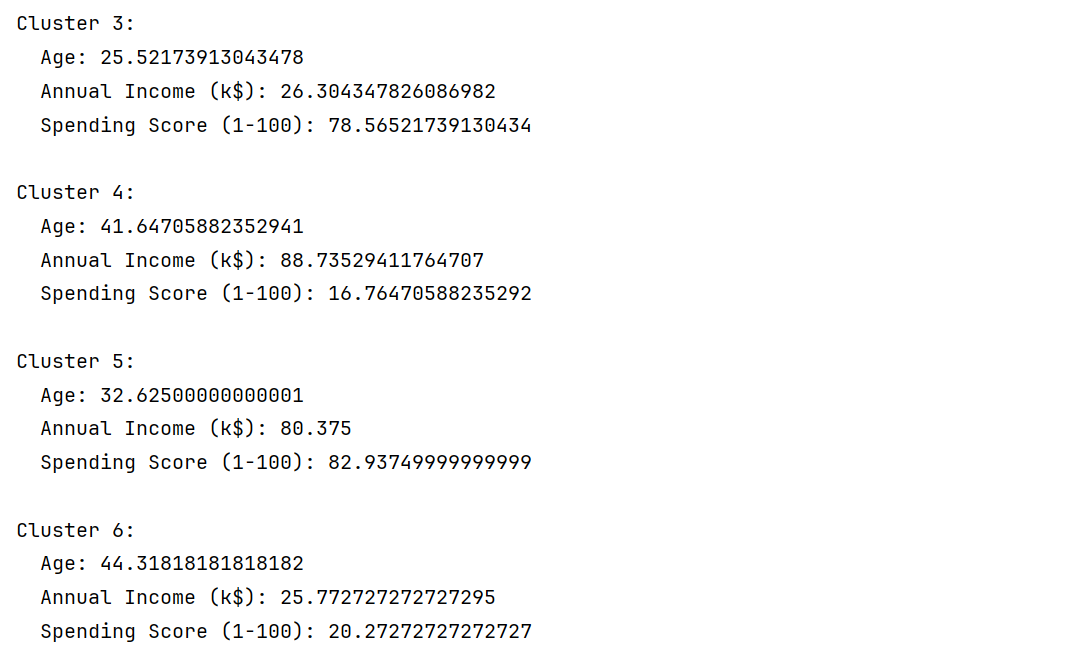
（1）读取Mall\_Customers.csv数据，对数据进行可视化；

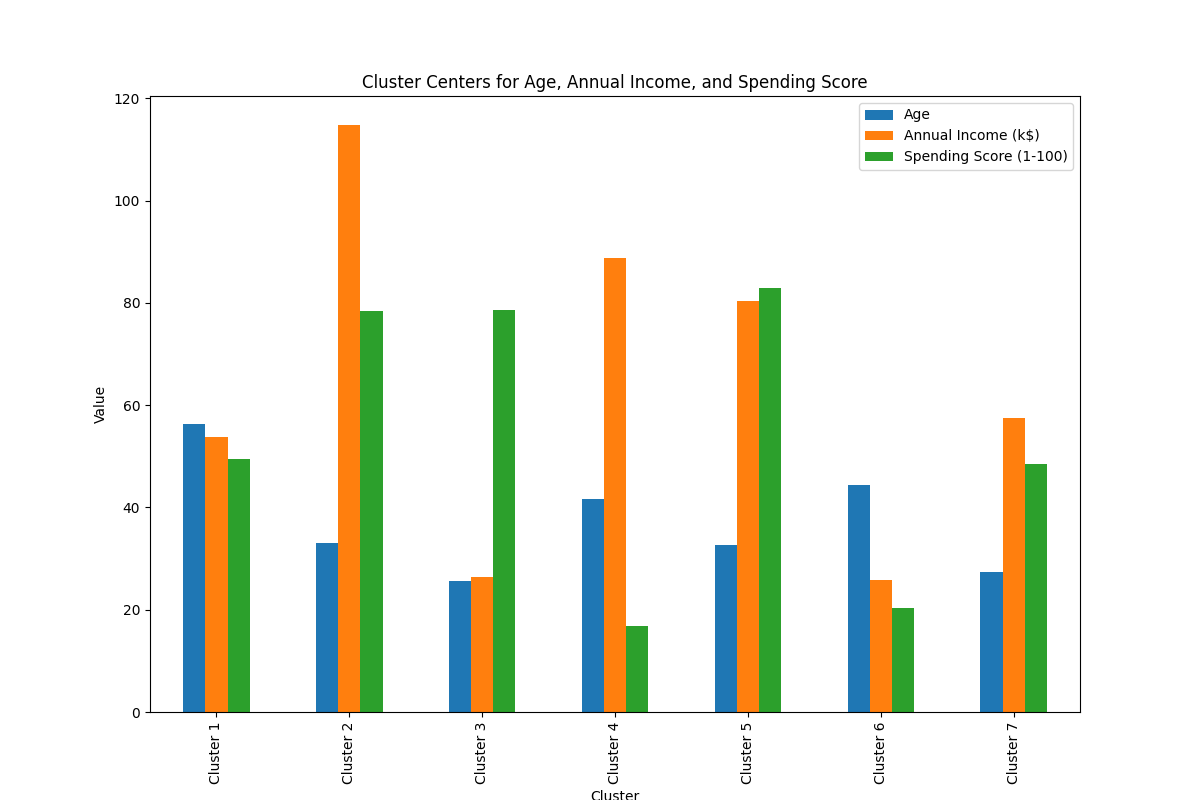
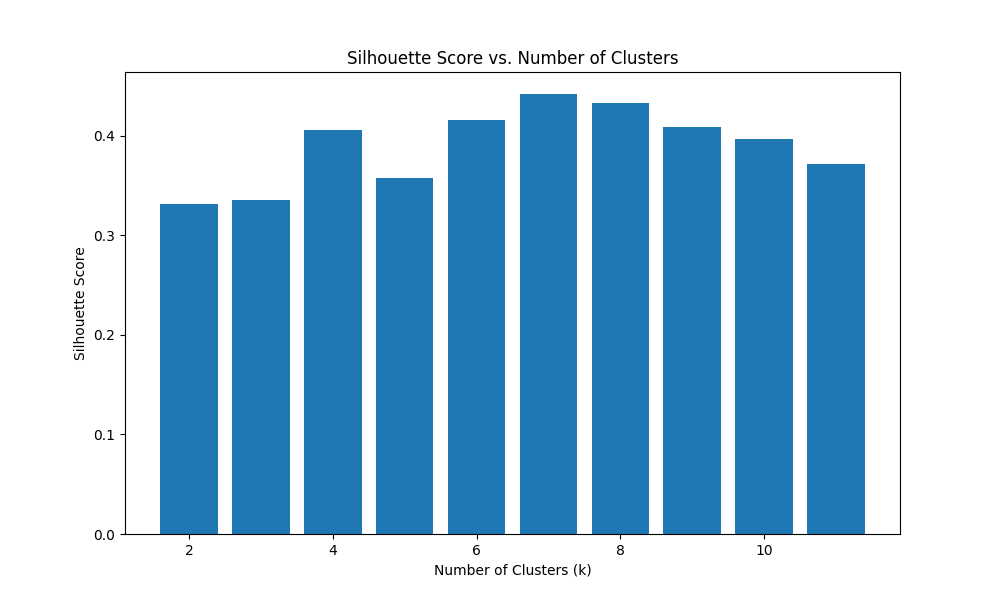
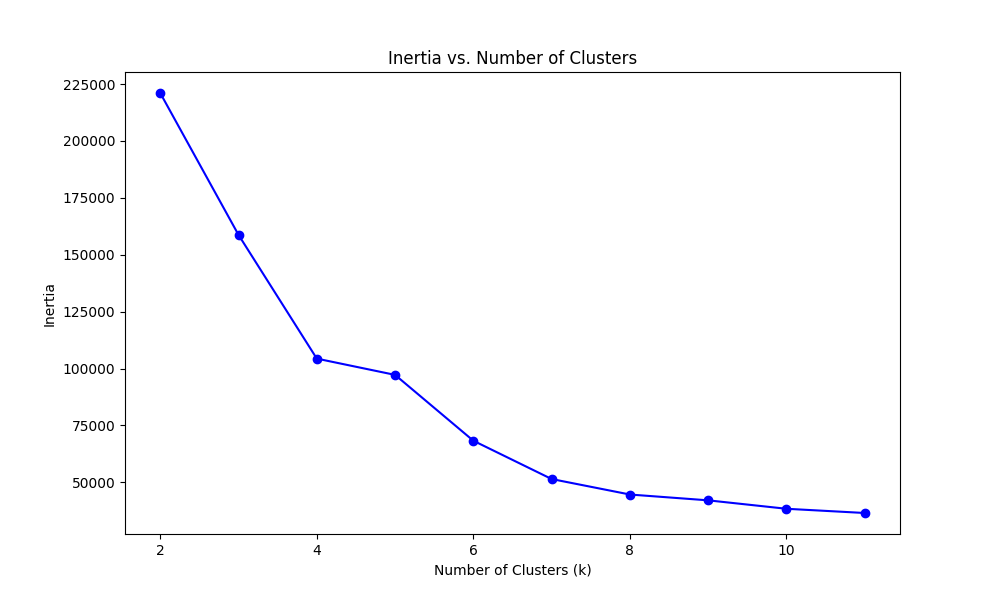
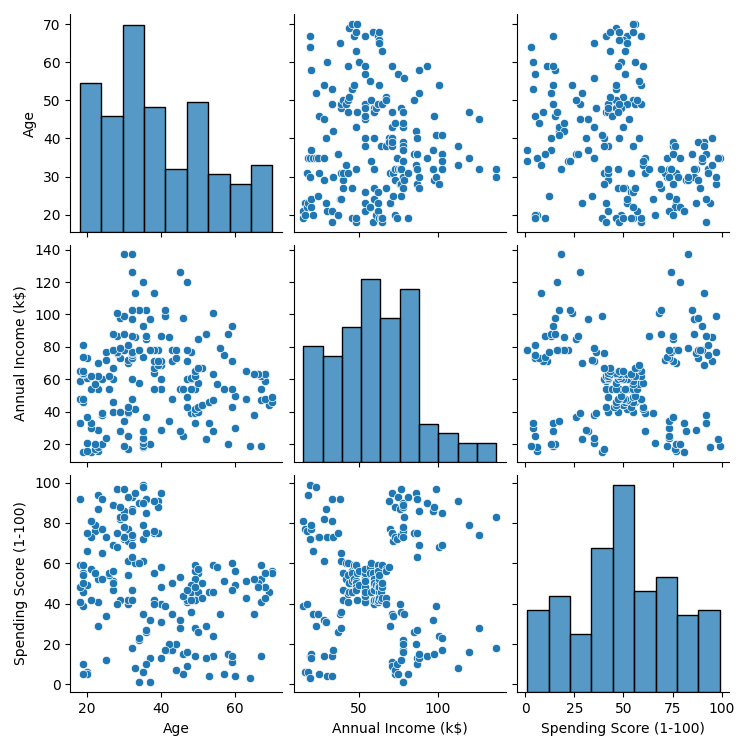
（2）选取后三列的数据作为训练数据，使用Kmean模型测试k值范围在2-11属性值inertia\_(总距离平方和)和轮廓系数得分，在不同K值条件下使用散点图画总距离平方和，使用条形图画轮廓系数得分，选择最优K值

（3）使用最优K值训练数据，输出每个簇的中心点

（4）使用平铺条形图的方式将三个特征年龄、年收入和消费的k个簇的中心点数据可视化，并简单分析结果

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
import numpy as np  
  
# 读取数据  
data = pd.read\_csv('Mall\_Customers.csv')  
  
# 查看数据的前几行  
print(data.head())  
  
# 数据可视化  
sns.pairplot(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']])  
plt.show()  
  
# 选取后三列数据作为训练数据  
X = data.iloc[:, 2:5]  
  
# 初始化列表存储inertia和轮廓系数得分  
inertia = []  
silhouette\_scores = []  
  
# 测试k值范围在2到11  
k\_range = range(2, 12)  
  
for k in k\_range:  
 kmeans = KMeans(n\_clusters=k, random\_state=42)  
 kmeans.fit(X)  
 inertia.append(kmeans.inertia\_)  
 silhouette\_scores.append(silhouette\_score(X, kmeans.labels\_))  
  
# 绘制总距离平方和(inertia)的散点图  
plt.figure(figsize=(10, 6))  
plt.plot(k\_range, inertia, 'bo-')  
plt.xlabel('Number of Clusters (k)')  
plt.ylabel('Inertia')  
plt.title('Inertia vs. Number of Clusters')  
plt.show()  
  
# 绘制轮廓系数得分的条形图  
plt.figure(figsize=(10, 6))  
plt.bar(k\_range, silhouette\_scores)  
plt.xlabel('Number of Clusters (k)')  
plt.ylabel('Silhouette Score')  
plt.title('Silhouette Score vs. Number of Clusters')  
plt.show()  
  
# 根据图选择最优K值，假设最优K值为7  
optimal\_k = 7  
  
# 使用最优K值训练数据  
kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)  
kmeans.fit(X)  
  
# 输出每个簇的中心点  
cluster\_centers = kmeans.cluster\_centers\_  
print("Cluster Centers:\n", cluster\_centers)  
  
# 将中心点数据转换为DataFrame  
cluster\_centers\_df = pd.DataFrame(cluster\_centers, columns=['Age', 'Annual Income (k$)', 'Spending Score (1-100)'])  
  
# 可视化簇的中心点  
cluster\_centers\_df.plot(kind='bar', figsize=(12, 8))  
plt.xticks(ticks=np.arange(optimal\_k), labels=[f'Cluster {i+1}' for i in range(optimal\_k)])  
plt.title('Cluster Centers for Age, Annual Income, and Spending Score')  
plt.xlabel('Cluster')  
plt.ylabel('Value')  
plt.show()  
  
# 简单分析结果  
for i, row in cluster\_centers\_df.iterrows():  
 print(f"Cluster {i+1}:")  
 print(f" Age: {row['Age']}")  
 print(f" Annual Income (k$): {row['Annual Income (k$)']}")  
 print(f" Spending Score (1-100): {row['Spending Score (1-100)']}\n")





**Cluster 1:**

**年龄、年收入和消费评分较为均衡，都在40到60之间。**

**属于中等年龄、中等收入和中等消费水平的人群。**

**Cluster 2:**

**年收入显著高于其他特征，接近120k$。**

**年龄和消费评分也较高，分别在60左右。**

**这一簇代表高收入、高消费且年龄较大的群体。**

**Cluster 3:**

**年收入和消费评分较高，分别在80k$和60左右。**

**年龄相对较低，在20到30之间。**

**主要是年轻、高收入和高消费的群体。**

**Cluster 4:**

**年收入最高，接近120k$。**

**年龄较低，消费评分较低。**

**这一簇可能代表年轻、高收入但消费意愿较低的人群。**

**Cluster 5:**

**年龄较高，年收入和消费评分都在80左右。**

**这一簇主要是年长、高收入且高消费的人群。**

**Cluster 6:**

**年龄、年收入和消费评分均较低，分别在30到40之间。**

**属于年轻、低收入和低消费的人群。**

**Cluster 7:**

**年收入和消费评分较为均衡，都在60到70之间。**

**年龄较高，在40到50之间。**

**主要是年长、中等收入和中等消费的人群。**

**消费意愿强的人主要在20-40岁。**

2、聚类方法应用-图像分割

（1）读取一个jpg图像文件，转换图像并显示

（2）对图像进行预处理，分别使用K-means聚类算法和均值漂移聚类算法对像素进行聚类，将每个像素点的值替换为其所属的聚类中心的值，并将numpy数组重新变成图像并显示

import numpy as np  
import matplotlib.pyplot as plt  
from skimage import io  
from sklearn.cluster import KMeans, MeanShift, estimate\_bandwidth  
  
# 读取图像文件  
image = io.imread('cat.jpg')  
plt.figure(figsize=(10, 10))  
plt.imshow(image)  
plt.title('Original Image')  
plt.axis('off')  
plt.show()  
  
# 将图像从RGB转换为二维数组  
image\_2d = image.reshape((-1, 3))  
  
# 使用K-means聚类算法  
kmeans = KMeans(n\_clusters=4, random\_state=42)  
kmeans.fit(image\_2d)  
kmeans\_labels = kmeans.predict(image\_2d)  
kmeans\_centers = kmeans.cluster\_centers\_  
# print(kmeans\_centers.shape)  
# print(type(kmeans\_centers))  
# print(kmeans\_labels.shape)  
# print(kmeans\_labels[:10])  
# print(kmeans\_centers[kmeans\_labels][:10])  
# 替换像素点的值为其所属的聚类中心的值  
kmeans\_segmented\_image = kmeans\_centers[kmeans\_labels].reshape(image.shape).astype(np.uint8)  
  
# 显示K-means分割后的图像  
plt.figure(figsize=(10, 10))  
plt.imshow(kmeans\_segmented\_image)  
plt.title('K-means Segmented Image')  
plt.axis('off')  
plt.show()  
  
# 使用均值漂移聚类算法  
bandwidth = estimate\_bandwidth(image\_2d, quantile=0.1, n\_samples=100)  
meanshift = MeanShift(bandwidth=bandwidth, bin\_seeding=True)  
meanshift.fit(image\_2d)  
meanshift\_labels = meanshift.predict(image\_2d)  
meanshift\_centers = meanshift.cluster\_centers\_  
  
# 替换像素点的值为其所属的聚类中心的值  
meanshift\_segmented\_image = meanshift\_centers[meanshift\_labels].reshape(image.shape).astype(np.uint8)  
  
# 显示均值漂移分割后的图像  
plt.figure(figsize=(10, 10))  
plt.imshow(meanshift\_segmented\_image)  
plt.title('Mean Shift Segmented Image')  
plt.axis('off')  
plt.show()  
# n\_clusters\_meanshift = len(np.unique(meanshift\_labels))  
# print(f"Mean Shift分割后的簇数: {n\_clusters\_meanshift}")



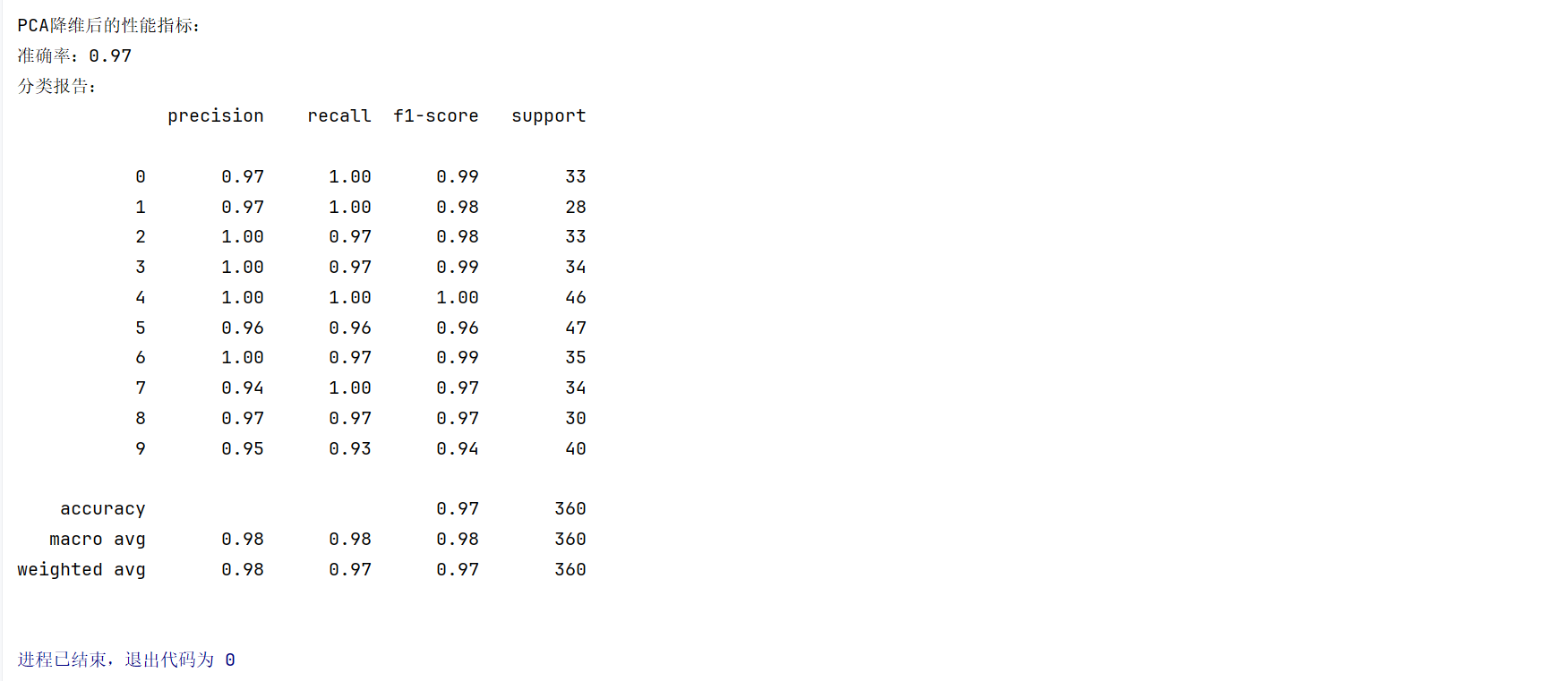
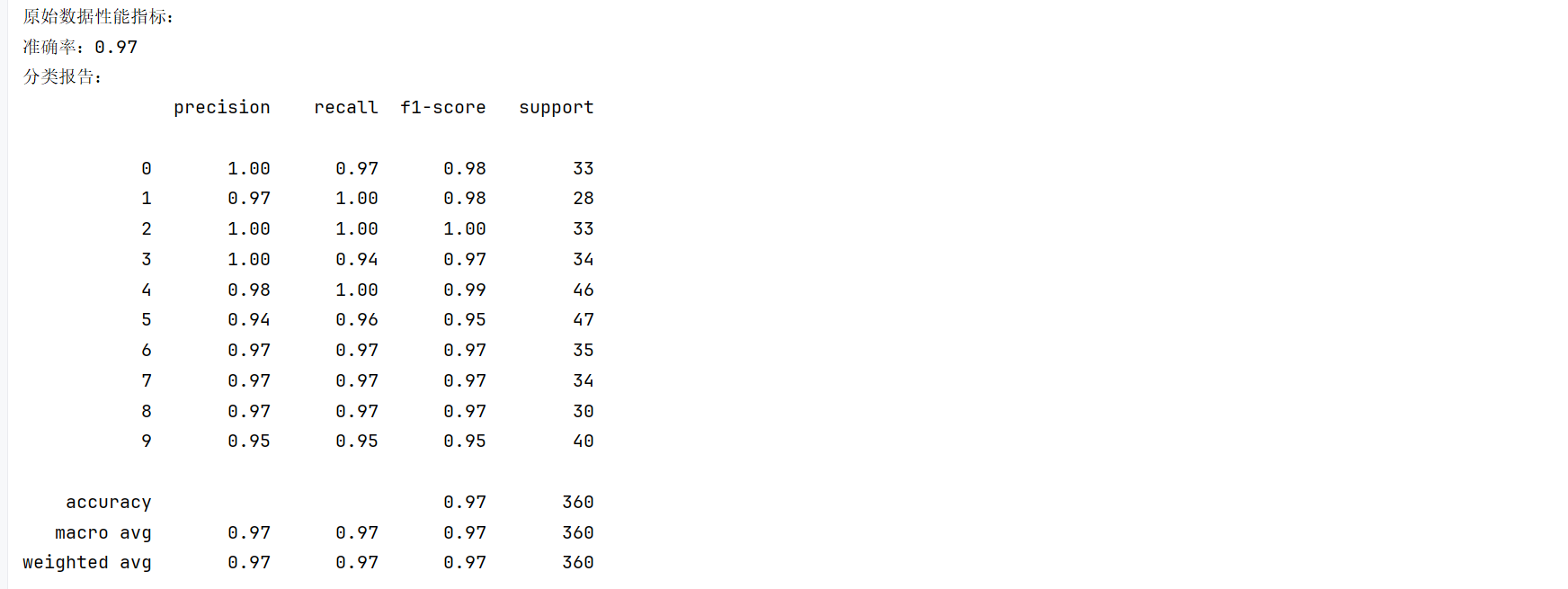
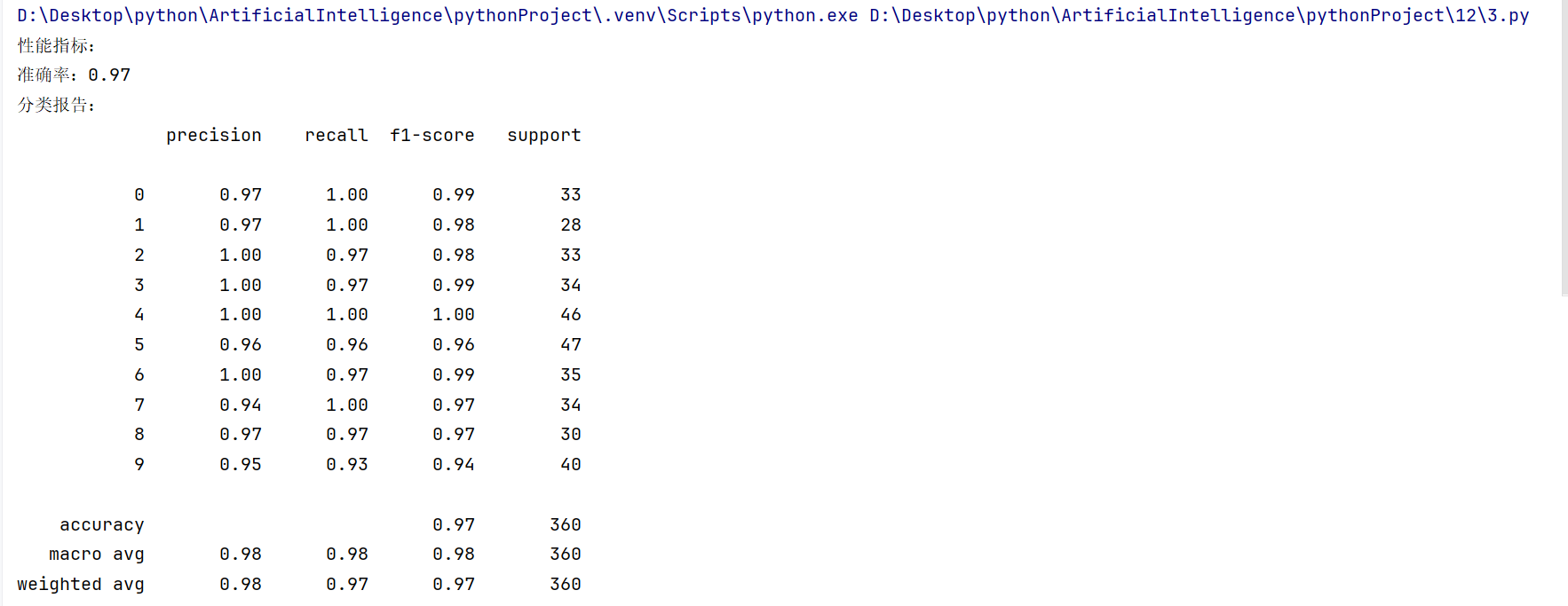
3、PCA降维应用

（1）使用PCA对digits数据进行降维（选取几个参数），使用分类模型对降维后的数据集进行分类，输出性能指标

（2）对比降维前和降维后的分类性能指标

（3）基于digits数据对比特征选择和降维的分类性能，并说明特征选择和降维的区别

from sklearn.datasets import load\_digits  
from sklearn.model\_selection import train\_test\_split  
from sklearn.decomposition import PCA  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import accuracy\_score, classification\_report  
  
# 加载digits数据集  
digits = load\_digits()  
X, y = digits.data, digits.target  
# print(digits.data.shape)  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# 使用PCA进行降维  
n\_components = 20 # 设置PCA的主成分数量  
pca = PCA(n\_components=n\_components, random\_state=42)  
X\_train\_pca = pca.fit\_transform(X\_train)  
X\_test\_pca = pca.transform(X\_test)  
  
# 使用分类模型（这里使用随机森林分类器）  
rf\_classifier = RandomForestClassifier(random\_state=42)  
rf\_classifier.fit(X\_train\_pca, y\_train)  
  
# 在测试集上进行预测  
y\_pred = rf\_classifier.predict(X\_test\_pca)  
  
# 输出性能指标  
accuracy = accuracy\_score(y\_test, y\_pred)  
report = classification\_report(y\_test, y\_pred)  
  
print("性能指标：")  
print(f"准确率：{accuracy:.2f}")  
print("分类报告：")  
print(report)  
  
# 使用原始数据进行分类  
rf\_classifier\_raw = RandomForestClassifier(random\_state=42)  
rf\_classifier\_raw.fit(X\_train, y\_train)  
y\_pred\_raw = rf\_classifier\_raw.predict(X\_test)  
accuracy\_raw = accuracy\_score(y\_test, y\_pred\_raw)  
report\_raw = classification\_report(y\_test, y\_pred\_raw)  
  
# 输出原始数据的性能指标  
print("原始数据性能指标：")  
print(f"准确率：{accuracy\_raw:.2f}")  
print("分类报告：")  
print(report\_raw)  
  
# 输出PCA降维后的性能指标  
print("\nPCA降维后的性能指标：")  
print(f"准确率：{accuracy:.2f}")  
print("分类报告：")  
print(report)



**特征选择：**

**特征选择是从原始特征集中选择最重要的特征子集，而不对特征进行任何变换。它可以基于统计方法、基于模型的方法或其他启发式方法来选择特征。**

**特征选择保留了原始特征的原始含义，不对数据进行变换，因此能够更容易地解释模型。**

**通过选择最重要的特征子集，特征选择可以提高模型的性能、减少过拟合，并提高模型的训练和预测效率。**

**降维：**

**降维是通过线性或非线性变换将原始特征映射到一个低维空间，从而保留数据中的主要信息。主成分分析（PCA）是降维的一种常见方法。**

**降维通过减少特征数量来降低数据的复杂性，并提高模型的性能和效率。它可以帮助解决维度灾难问题，并提高模型的泛化能力。**

**降维通常会导致原始特征的信息损失，因为它将多个特征合并成少数几个主成分，这可能会导致某些信息丢失。**

4、PCA与聚类的综合应用

（1）读取wholesale.csv数据集，进行归一化(标准化）处理

（2）维度分析，查看相关系数

（3）使用PCA降维

（4）对降维后的数据选择一种聚类方法进行聚类，对聚类后的结果进行数据可视化

import pandas as pd  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA  
from sklearn.cluster import KMeans  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# 读取数据集  
data = pd.read\_csv('wholesale.csv')  
  
# 归一化处理  
scaler = StandardScaler()  
normalized\_data = scaler.fit\_transform(data)  
normalized\_df = pd.DataFrame(normalized\_data, columns=data.columns)  
  
# 查看相关系数  
correlation\_matrix = normalized\_df.corr()  
  
# 使用热力图可视化相关系数矩阵  
plt.figure(figsize=(10, 8))  
sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)  
plt.title('Correlation Matrix')  
plt.show()  
# 使用PCA进行降维  
pca = PCA()  
pca.fit(normalized\_df)  
explained\_variance\_ratio = pca.explained\_variance\_ratio\_  
cumulative\_explained\_variance\_ratio = explained\_variance\_ratio.cumsum()  
n\_components = 2 # 这里选择2个主成分进行演示  
pca = PCA(n\_components=n\_components)  
pca.fit(normalized\_df)  
pca\_transformed = pca.transform(normalized\_df)  
  
# 使用K-means对降维后的数据进行聚类  
kmeans = KMeans(n\_clusters=3, random\_state=42)  
kmeans.fit(pca\_transformed)  
cluster\_labels = kmeans.labels\_  
  
# 可视化聚类结果  
plt.scatter(pca\_transformed[:, 0], pca\_transformed[:, 1], c=cluster\_labels, cmap='viridis')  
plt.xlabel('Principal Component 1')  
plt.ylabel('Principal Component 2')  
plt.title('Clustered Data Visualization')  
plt.colorbar(label='Cluster Label')  
plt.show()

