实验报告（7）

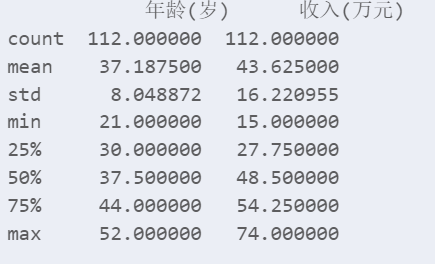
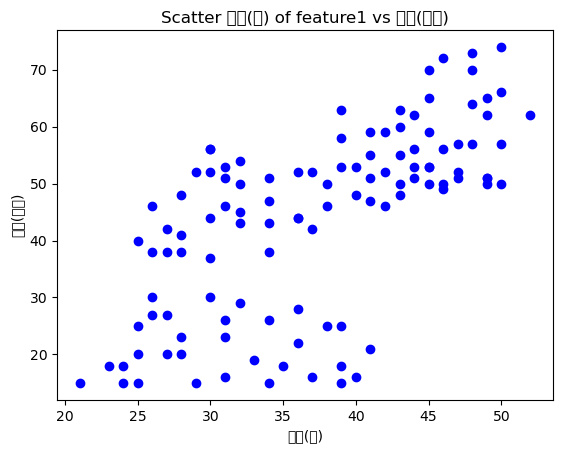
学号：202202206341 姓名：王子俊

1. **实验名称： 聚类**
2. **实验目的**
3. 理解聚类
4. 理解KMeans算法，高斯混合聚类等方法
5. 了解sklearn库
6. 了解Matplotlib库
7. 了解pandas库
8. **实验内容**

本次实验通过实现两种常用的聚类算法——K-Means 和 高斯混合模型（GMM），对一组数据进行了聚类分析。实验中包括数据预处理、聚类模型的训练、聚类结果的可视化、预测新数据的标签以及评估聚类效果的过程。以下是实验的详细总结

1. **实验结果与分析**

**1. 数据预处理**

* 首先，使用 **Pandas** 库读取并加载了包含客户信息的 Excel 数据集（例如，年龄(岁) 和 收入(万元) 特征）。
* 数据被用作聚类分析的输入特征。
* 
* 

**2. K-Means 聚类**

* + **K-Means** 是一种基于距离的聚类方法，将数据点分成 k 个簇。
  + 在实验中，首先通过 **肘部法则**（Elbow Method）选择了合适的聚类数 k。
  + 聚类的步骤：
    - * 使用 **KMeans** 类进行训练并指定聚类数 k=3。
      * 通过 **fit\_predict()** 方法获取每个数据点的聚类标签。
      * 使用 **matplotlib** 绘制散点图，按照聚类标签为每个点着色，直观展示聚类结果。

**代码片段：**

from sklearn.cluster import KMeans

import numpy as np

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False  # 避免负号显示为方块

# 假设我们要聚类的数据是 df 中的某些特征

X = df[['年龄(岁)', '收入(万元)']]  # 根据实际特征选择

# 寻找最优k值

inertia = []  # 存储不同k值的聚类结果

k\_range = range(1, 11)  # 假设我们实验k从1到10

for k in k\_range:

    kmeans = KMeans(n\_clusters=k, random\_state=42)

    kmeans.fit(X)

    inertia.append(kmeans.inertia\_)  # 记录聚类的总误差（即“惯性”）

# 绘制k值与惯性值的折线图

plt.plot(k\_range, inertia, marker='o')

plt.title('Elbow Method for Optimal k')

plt.xlabel('Number of Clusters (k)')

plt.ylabel('Inertia')

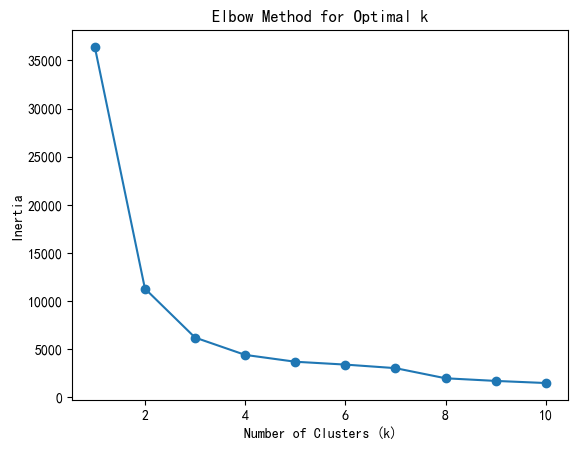
plt.show()

# 选择最佳k值进行聚类（可以基于惯性曲线的“肘部”选择）

optimal\_k = 3  # 假设最佳k值为3

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

df['cluster'] = kmeans.fit\_predict(X)  # 聚类标签添加到数据中

****

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=3, random\_state=42)

df['kmeans\_cluster'] = kmeans.fit\_predict(X)

# 可视化K-Means聚类结果

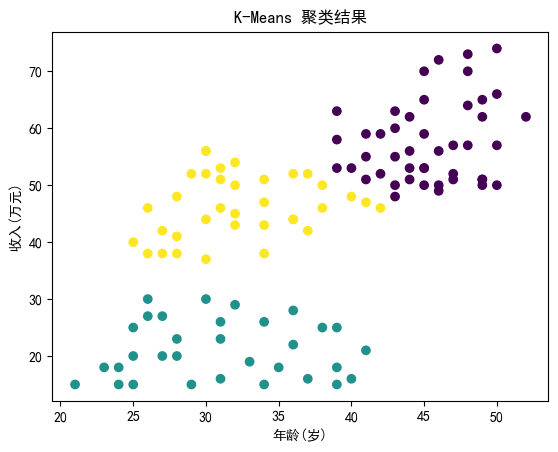
plt.scatter(df['年龄(岁)'], df['收入(万元)'], c=df['kmeans\_cluster'], cmap='viridis', marker='o')

plt.title('K-Means 聚类结果')

plt.xlabel('年龄(岁)')

plt.ylabel('收入(万元)')

plt.show()



**3. 高斯混合模型（GMM）聚类**

* **高斯混合模型（GMM）** 是基于概率模型的聚类方法，假设数据点来自多个高斯分布的混合模型。与 K-Means 相比，GMM 更适合处理形状复杂、密度不均的聚类问题。
* 使用 **GaussianMixture** 类进行训练，设定聚类数 n\_components=3。
* 聚类的步骤：
  + 使用 **fit()** 方法训练 GMM 模型。
  + 使用 **predict()** 方法获取每个数据点的聚类标签。
  + 同样，通过散点图展示 GMM 聚类结果。

**代码片段：**

from sklearn.mixture import GaussianMixture

from sklearn.metrics import adjusted\_rand\_score  # 用于计算ARI

import matplotlib.pyplot as plt

import numpy as np

# 假设你已经加载了数据

X = df[['年龄(岁)', '收入(万元)']]  # 替换为你的特征列

# 使用 GMM 聚类

gmm = GaussianMixture(n\_components=3, random\_state=42)  # 设定聚类数为3（可以根据实际需求调整）

gmm.fit(X)

# 获取聚类标签

df['gmm\_cluster'] = gmm.predict(X)

# 可视化聚类结果

plt.scatter(df['年龄(岁)'], df['收入(万元)'], c=df['gmm\_cluster'], cmap='viridis', marker='o')

plt.title('高斯混合聚类结果', fontsize=14)  # 中文标题

plt.xlabel('年龄(岁)', fontsize=12)  # 中文X轴标签

plt.ylabel('收入(万元)', fontsize=12)  # 中文Y轴标签

plt.show()

# 预测新数据点

new\_data = np.array([[30, 10], [45, 20], [60, 15]])  # 假设有新的数据点

new\_predictions = gmm.predict(new\_data)

print(f'新数据点的聚类预测结果: {new\_predictions}')

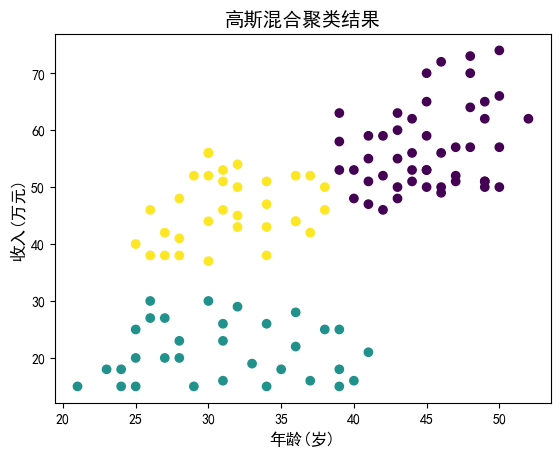
# 计算准确率（如果有真实标签）

# 假设你有真实的标签（真实标签列为true\_labels）

true\_labels = df['真实标签']  # 替换为实际的真实标签列

ari = adjusted\_rand\_score(true\_labels, df['gmm\_cluster'])

print(f'聚类结果的调整兰德指数 (ARI): {ari}')

****

**4. 预测新数据**

* 通过训练好的模型，我们可以对新的数据点进行聚类预测。
* 使用 **predict()** 方法预测新数据点的聚类标签，并输出结果。

**代码片段：**

new\_data = np.array([[30, 10], [45, 20], [60, 15]]) # 新数据

new\_predictions = gmm.predict(new\_data) # 使用GMM进行预测

print(f'新数据点的聚类预测结果: {new\_predictions}')

**5. 聚类评估**

* 聚类任务中，使用 **调整兰德指数（ARI）** 来评估聚类结果的质量。ARI 衡量了聚类标签与真实标签的一致性，值越接近1说明聚类效果越好。
* 在实验中，假设已经拥有真实标签并计算 **ARI**，以此来评估聚类算法的效果。

**代码片段：**

from sklearn.metrics import adjusted\_rand\_score

# 假设我们有真实标签

true\_labels = df['真实标签'] # 真实标签列

ari = adjusted\_rand\_score(true\_labels, df['gmm\_cluster'])

print(f'聚类结果的调整兰德指数 (ARI): {ari}')

**6. 结果分析与比较**

* **K-Means** 聚类适合于具有类似球形分布的聚类。由于它是基于距离的算法，它对噪声和离群点较为敏感。
* **GMM** 聚类适合于更复杂的聚类任务，它能够识别具有不同形状和密度的聚类，且比 K-Means 更为灵活。
* 对于没有真实标签的数据，无法直接使用 **准确率** 来评估聚类效果，因此可以使用 **ARI** 或 **轮廓系数** 来评估聚类的质量。

**7. 实验结论**

* **K-Means** 和 **GMM** 都能有效地对数据进行聚类，但根据数据的性质（如聚类的形状、密度分布等），GMM 可能会提供更好的聚类效果。
* 在实际应用中，选择合适的聚类算法取决于数据的特性以及对聚类效果的需求。

本次实验通过 **K-Means** 和 **高斯混合模型（GMM）** 聚类对数据集进行了聚类分析，并通过可视化方法展示了聚类效果。同时，我们还进行了新数据的预测和聚类评估，提供了对聚类质量的定量评估。这两种算法在实际应用中具有重要的参考价值，K-Means 更适合球形分布的聚类问题，而 GMM 更适合复杂和不规则的聚类任务。

1. **实验总结**

本次实验让我深入了解了聚类算法的基本原理和应用，特别是 K-Means 和 高斯混合模型（GMM） 的具体使用。通过实验，我学到了如何使用这两种算法对数据进行无监督学习，掌握了如何通过模型的聚类结果进行数据分析和可视化。K-Means 算法利用欧氏距离来确定聚类中心，对于数据的分布有一定要求，适用于簇形较为规则的情况。而 GMM 作为基于概率模型的聚类方法，能够处理更为复杂的数据分布，适用于形状和密度不均匀的聚类任务，能够提供比 K-Means 更加灵活的聚类结果。通过实验，我还学习了如何评估聚类效果，尤其是如何通过调整兰德指数（ARI）等指标对聚类结果进行定量评价。

在未来，聚类算法有很大的发展潜力，尤其是在大数据和复杂数据的分析中。随着数据的多样化和复杂性增加，如何根据数据特性选择合适的聚类方法，将是一个重要的研究方向。除了 K-Means 和 GMM，还可以深入探索其他高级聚类方法，如层次聚类、密度聚类（如 DBSCAN）等，进一步提高聚类算法的适用性和准确性。另外，聚类算法的优化和扩展也是未来值得研究的方向，如通过结合深度学习方法，发展自监督学习或强化学习等技术，进一步提升模型在高维数据和大规模数据上的表现。