K-Means算法实验报告

范潇

2023年12月17日

1 实验内容

在本次实验中，我使用Python语言实现二分K-Means，并使用matplotlib库对聚类结果进行可视化。

实验过程为：

1. 选择数据集
2. 数据集预处理
3. 实现二分 K-Means 算法
4. 对算法进行评价
5. 可视化

2 函数说明

在实现该算法的过程中，使用的主要数据结果为DataFrame。数据集的原始数据存储在 data变量中，并且最后一列'origin_row'存储的是各个行的行标；经过 t-SNE，降维后的数据存储在 reduced变量中，并且最后两列分别为 'origin_row'和用于存储所属类簇信息的'color'列。

2.1 求类簇SSE

```
def get_sse(df):
    sse = ((df.iloc[:, :-1] - df.iloc[:, :-1].mean(axis=0, numeric_only=True)) ** 2).sum
        ().sum()
    # 减去平均值后求平方，然后对矩阵中的所有元素进行求和
    return sse
```

2.2 类簇二分

```
def split(df):
    # 随机挑选两个初始点
    init1 = random.randint(0, df.shape[0] - 1)
    init2 = random.randint(0, df.shape[0] - 1)
    while init2 == init1:
        init2 = random.randint(0, df.shape[0] - 1)
    df1 = df[((df - df.iloc[init1, :-1]) ** 2).sum(axis=1) <= ((df - df.iloc[init2,
        :-1]) ** 2).sum(axis=1)]
    df2 = df[((df - df.iloc[init2, :-1]) ** 2).sum(axis=1) < ((df - df.iloc[init1, :-1])
         ** 2).sum(axis=1)]
    # 返回分裂所得的两个数据表，以及它们的和方差之和
    return df1, df2, get_sse(df1) + get_sse(df2)
```

本函数利用DataFrame的布尔过滤功能来实现分裂

2.3 求轮廓系数

```
def make_silhouette(clusters):
    # 求各类簇的中心
    center = []
    for cluster in clusters:
        center.append(cluster.iloc[:, :-1].mean())
    center = pd.DataFrame(center)
    # 用于存储由各类簇中的数据的轮廓系数所组成的list
    alphas = []
    for i, cluster in enumerate(clusters):
        alpha = []
        for j, data in enumerate(cluster.values):
            # 求到簇内其他数据的平均距离
            a = (((cluster.iloc[:, :-1] - data[:-1]) ** 2).sum(axis=1) ** 0.5).mean(axis
                =0)
            # 求最近的其他类簇中心
            nearest = (((center - data[:-1]) ** 2).sum(axis=1).drop(index=i) ** 0.5).
                idxmin()
            # 求到最近类簇内的数据的平均距离
            b = (((clusters[nearest].iloc[:, :-1] - data[:-1]) ** 2).sum(axis=1) ** 0.5).
                mean(axis=0)
            # 计算轮廓系数
            alpha.append((b - a) / max(a, b))
        alphas.append(alpha)
    return alphas
```

3 K-Means算法主体

```
# 将数据集添加进类簇列表中
cluster.append(data)
# 将数据集自身的和方差添加至和方差列表中
sse.append(get_sse(data))
# 初始化目前类簇数
cur_num = 1
while 1:
    # 作散点图
    make_scatter(fig, grid, cmap, cur_num, cluster)
    pos = -1
    max_loss = 0
    # 求二分后使得降低最多的类簇SSE
    for i, cur_cluster in enumerate(cluster):
        cur_sse = get_sse(cur_cluster)
        cluster1, cluster2, pos_sse = split(cur_cluster)
        if cur_sse - pos_sse > max_loss:
            pos_cluster1 = cluster1
            pos_cluster2 = cluster2
            max_loss = cur_sse - pos_sse
            pos = i
    # 没有找到
    if pos == -1:
        break
    # 找到
    # 更新类簇列表
    cluster.pop(pos)
    cluster.append(pos_cluster1)
    cluster.append(pos_cluster2)
    # 更新原类簇中数据的所属类簇
    reduced.loc[list(pos_cluster2['origin_row']), 'color'] = cur_num
    # 更新和方差总和
    new_sse = 0
    for cur_cluster in cluster:
        new_sse += get_sse(cur_cluster)
    sse.append(new_sse)
    # 更新轮廓系数列表
    score.append(pd.DataFrame(make_silhouette(cluster)).sum(axis=0).sum(axis=0) /
        reduced.shape[0])
    cur_num += 1
    # 达到预设类簇数，停止循环
    if cur_num == cluster_num:
        make_scatter(fig, grid, cmap, cur_num, cluster)
        break
```

4 测试说明

4.1 测试集说明

本次实验中所采用的数据集来自Kaggle平台[1]。

该数据集记录的是多个品种的企鹅的外貌信息，总共包括五列：'culmen_length_mm'、'culmen_depth_mm'、'flipper_length_mm'、'body_mass_g:'、'sex'。其中前4列为数值型数据，最后一列为标称型数据，其中并没有提供标签。原数据集中共有344条数据，有少量数据缺失。

4.2 预处理说明

在本次实验中，我使用Jupyter Notebook进行预处理。

首先我将该数据集中含有部分缺失的数据删去。然后我将'sex'列进行独热编码，将'MALE'和'FEMALE'分别编码为0和1。最后我将前4列数据进行归一化，转换为均值为0，方差为1的数据。

4.3 测试结果及可视化说明

由于所选数据集中并没有提供标签，所以采用内部指标——轮廓系数，对聚类结果进行评价。

本次实验中，我将算法进行分步骤可视化，将每次“二分”后所得到的新类簇用散点图的形式进行可视化。同时，当类簇数达到预设值时，绘制不同类簇数对应的 SSE值和轮廓系数所对应的折线图。

下面给出的是三次评价对应的图表。

评价结果1中，从肘部图中可以看出，类簇数为3时，肘部图的曲率最大。当类簇数为4时，肘部图的曲率也较大。在这两处所对应的轮廓系数都达到了局部的最大值。

评价结果2中，从肘部图中可以看出，类簇数为5时，肘部图的曲率最大，轮廓系数也达到了局部的最大值。

评价结果3中，从肘部图中可以看出，类簇数为3时，肘部图的曲率最大。当类簇数为6时，肘部图的曲率也较大。在这两处所对应的轮廓系数都达到了局部的最大值。

总的来看，根据肘部图确定了类簇数后，对应的聚类结果对应的轮廓系数均超过了0.5。

从分步骤的散点图中可以看出，导致评价结果有波动的原因是：二分 K-Means算法在进行“二分”的过程中，仍然采用的是随机算选取初始点的策略。

[1]https://www.kaggle.com/datasets/youssefaboelwafa/clustering-penguins-species/data

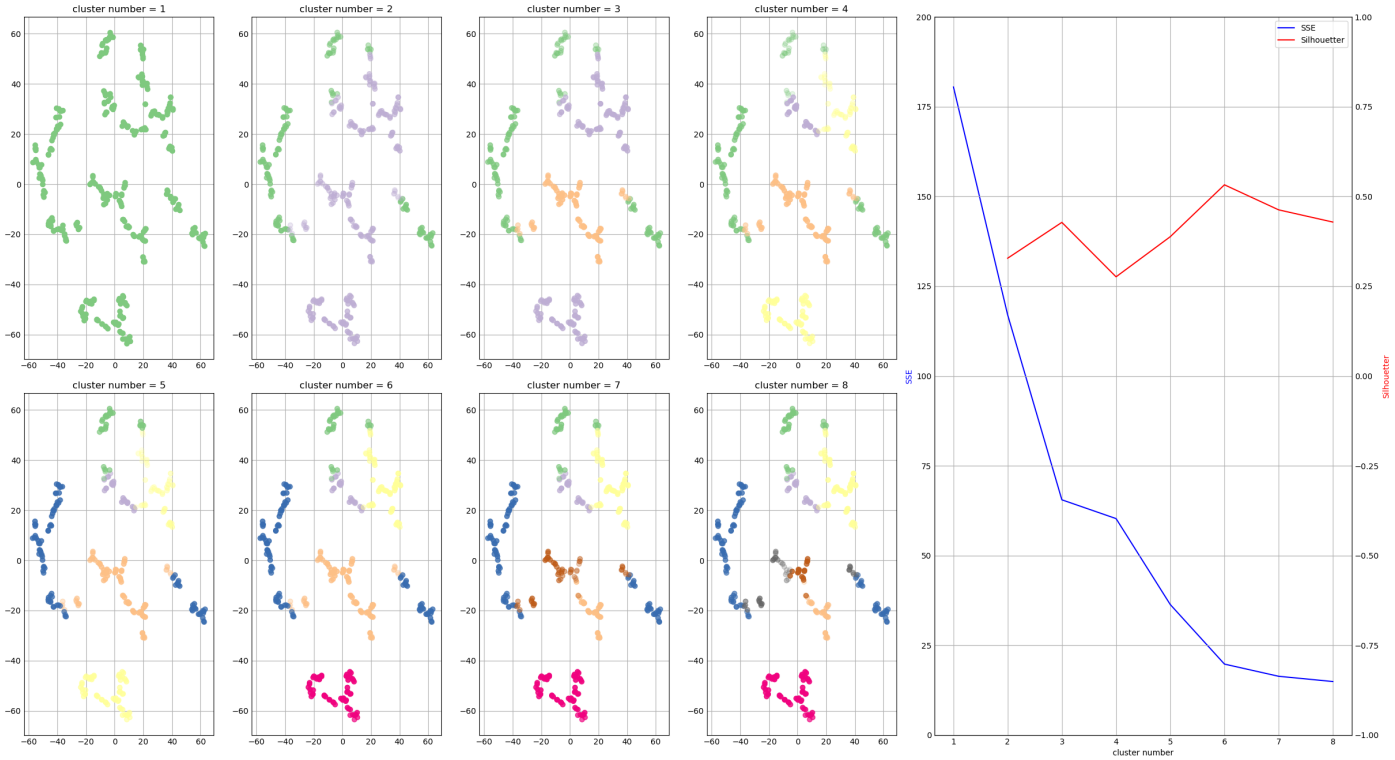

图 1: 评价结果1

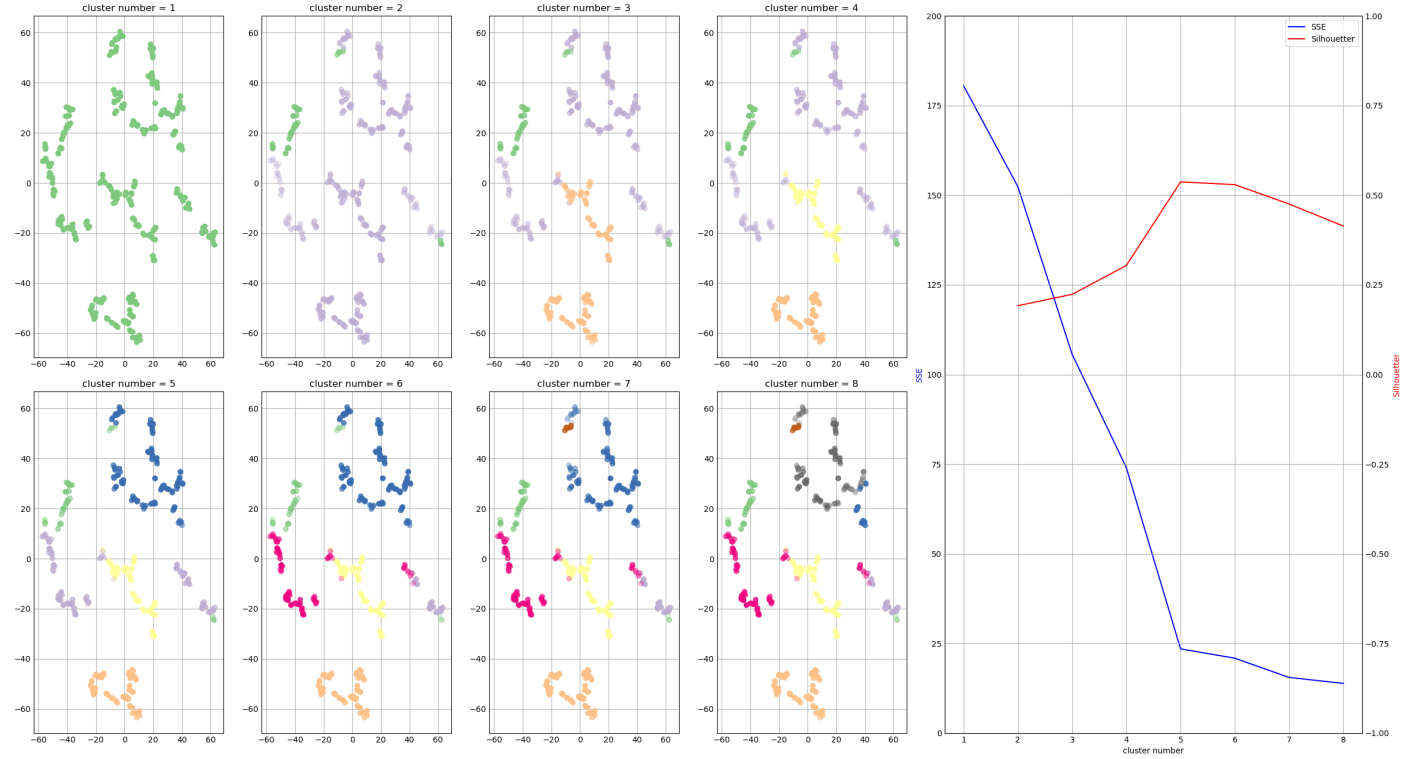

图 2: 评价结果2

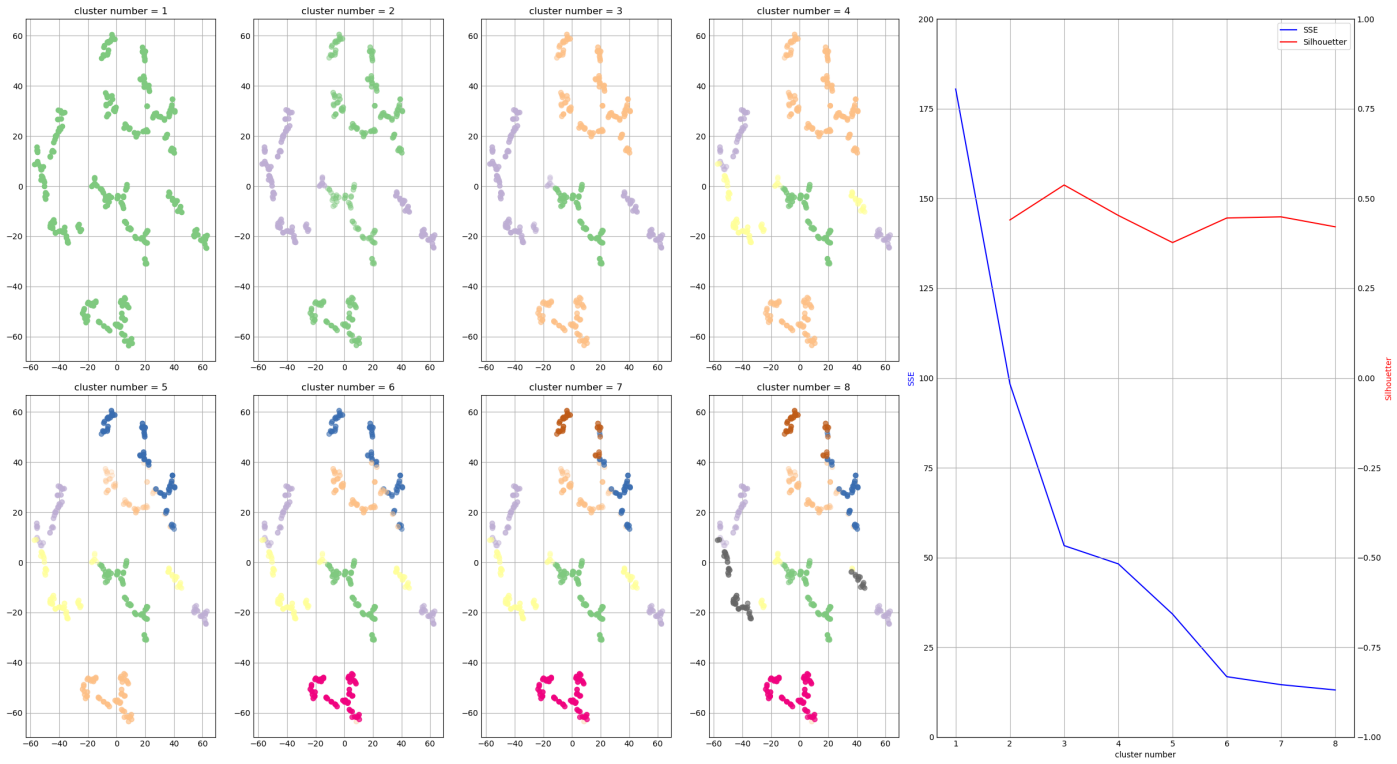

图 3: 评价结果3

5 改进之处

本次实验中，我对常规的聚类可视化方法——散点图进行了改进。

许多数据集的维度都较高，为了能够利用散点图进行可视化，通常需要使用PCA或t-SNE等方法对数据进行降维。可是降维过后的数据往往丢失了原来的相对位置关系，导致散点图中点之间的距离并不能反映出原来高维空间中的距离，从而难以直接根据散点图判断出数据点分类是否合理。因此，我将各个数据点的轮廓系数映射到散点图对应点的透明度上，使得轮廓系数高的数据在散点图上显示清晰，表明该数据点的分类较为合理，而轮廓系数低的数据在散点图上的醒目程度较低，表明对该数据点的分类并不可信。通过这一改进，散点图既可以通过颜色的不同来展现出高维数据的聚类结果，同时还可以通过颜色的深浅来直观地反映出单个数据分类结果的好坏以及单个类簇的总体分类结果如何。

6 心得体会

本次实验过程中，Pandas以及Matplotlib库的使用大大简化了我的代码。以函数get_sse为例，在使用DataFrame后，只需要一行代码便可以实现求均方和的功能，这和上次使用C++实现ID3算法的体验有着极大差别，使我深刻体会到了掌握合适工具的重要性。

在算法评价的过程中，我也深刻地认识到了初始点的选取策略对于 K-Means算法的重要程度。通过优化初始点的选取策略， K-Means算法还有很大的改进空间。

同时，在对于聚类结果进行可视化的过程中，我也感受到了可视化的重要性。如果没有进行可视化，聚类的结果只是一张张表格，从中很难直接获取有用的信息，尤其是当数据量略大的时候。通过可视化，不仅仅能够将结果清晰直观地展现出来，还能够帮助人们从数据中发掘出更多有用的信息。