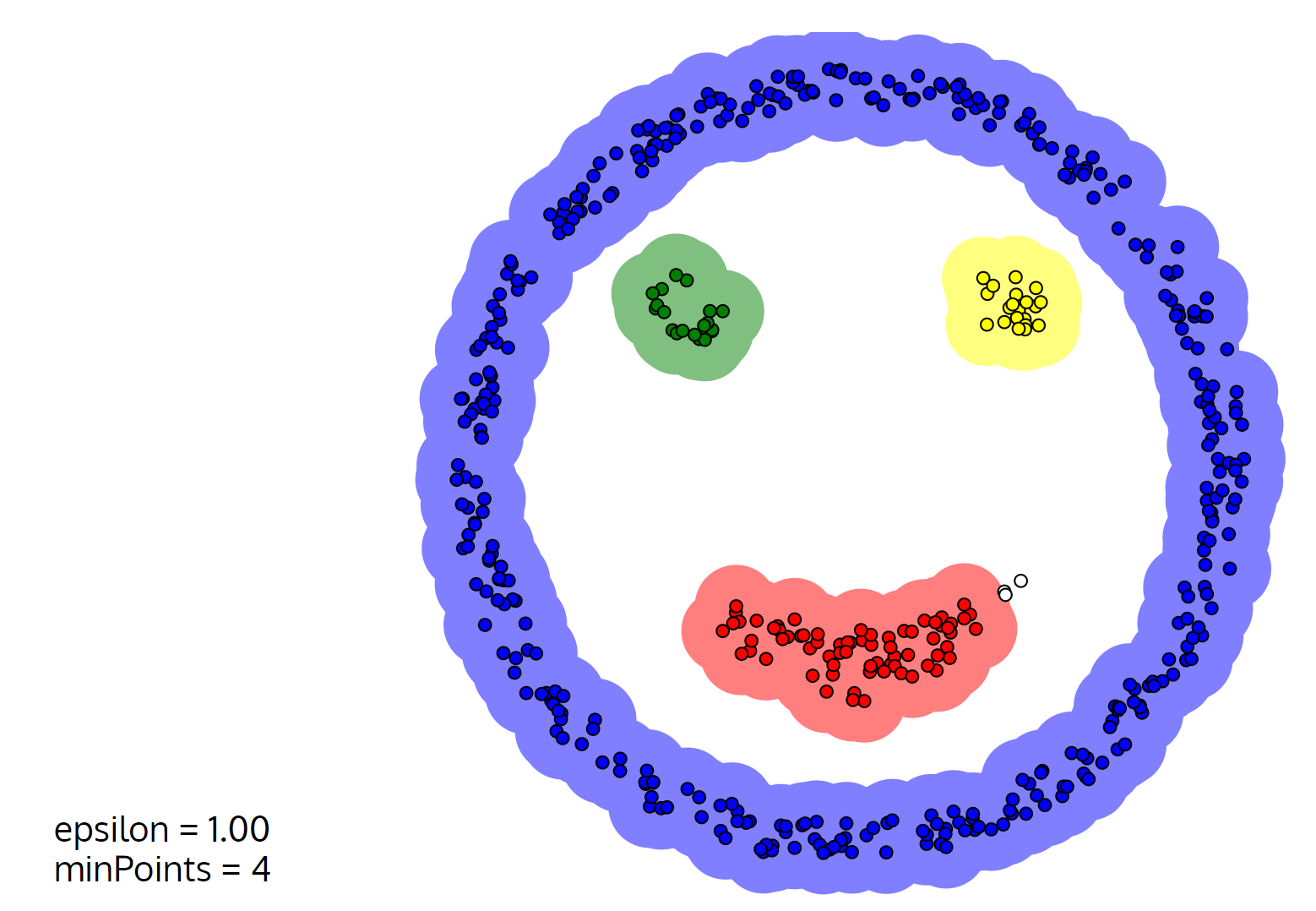
**基于密度的噪声应用空间聚类（DBSCAN）**

**一、简介**

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) 是一种基于密度的聚类算法，由Martin Ester, Hans-Peter Kriegel, Jörg Sander和Xiaowei Xu于1996年提出。它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。



**基本概念：**

ε-邻域：对于样本集中的xi, 它的ε-邻域为样本集中与它距离小于ε的样本所构成的集合。

在DBSCAN算法中，数据点被分为以下三类：

1.核心点：若样本xi的ε-邻域内至少包含了MinPts个点，则为核心点。

2.边界点：若样本xi的ε-邻域内包含的点的数量小于MinPts，但它在其他核心点的ε-邻域内，则为边界点。

3.噪声点：既非核心点也非边界点则为噪声点。

密度直达：若q处于p的 ε-邻域内，且p为核心点，则称q由p密度直达。

密度可达：若有一个点的序列q0、q1、....、qk，若对任意的qi->qi-1是密度直达的，则称从q0到qk密度可达。

密度相连：若从某核心点p出发，点q和点k都是密度可达的，则称点q和点k密度相连。

**算法需要定义参数：**

1、阈值minPts

2、半径ε

**二、用代码实现DBSCAN算法**

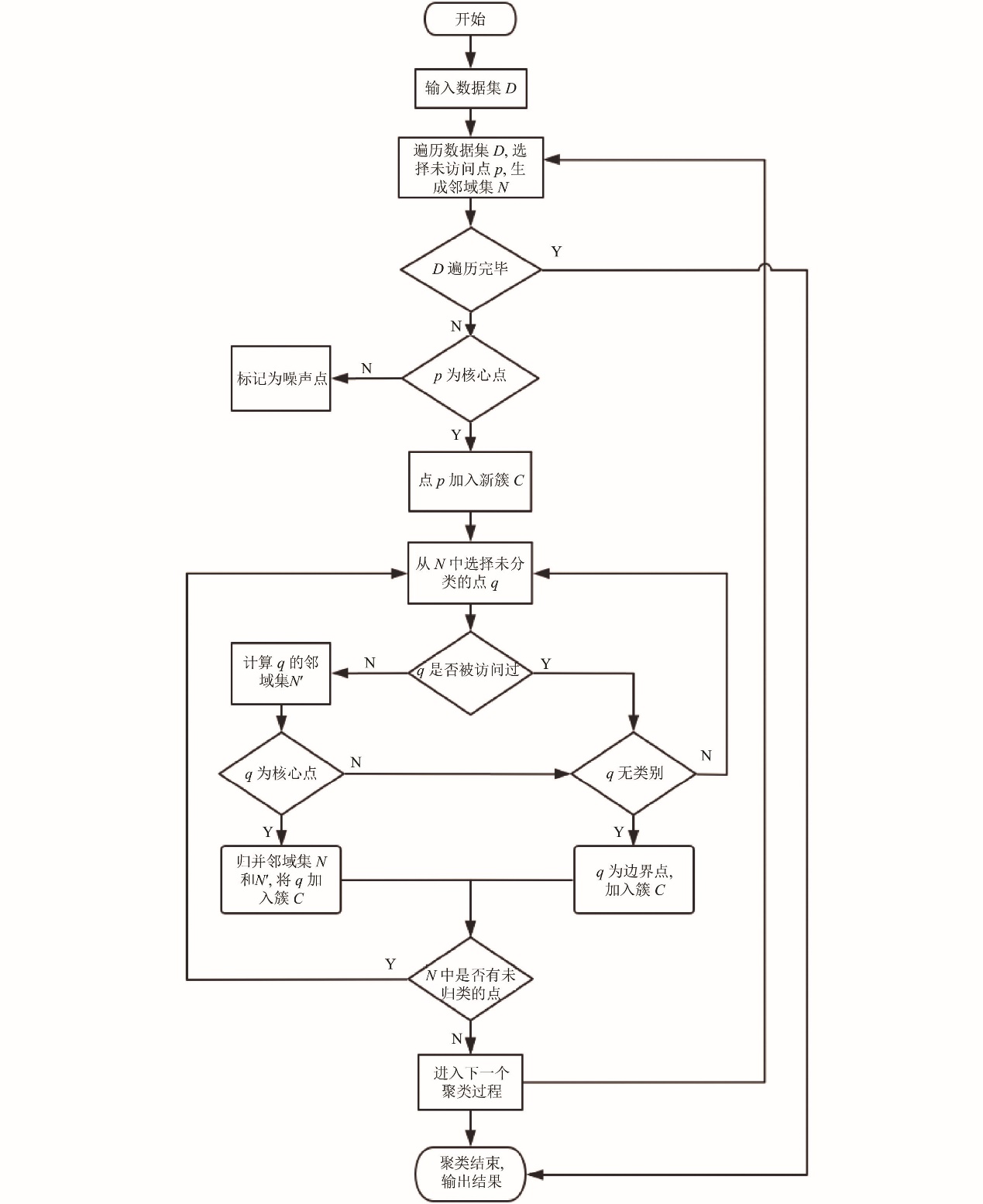
**DBSCAN算法的基本思路：**

1、选择核心点：如果一个点的ε-邻域内点数超过minPts，将其标记为核心点。

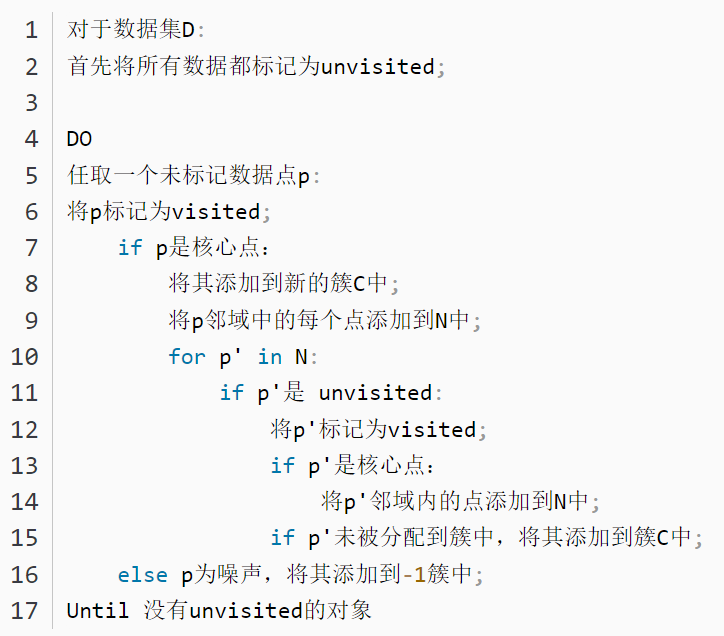
2、构建邻域链：对每个核心点，将它的ε-邻域内所有点（包括其他核心点）连接起来，形成一个聚类。

3、边界点的归属：将边界点分配给与之相连的核心点的聚类。

4、标记噪声：最后，未被归入任何聚类的点被标记为噪声。



**伪代码：**



**代码：**



**执行步骤：**

1、初始化：

labels数组，用于存储每个点的聚类标签，初始值为-1表示未分类。

visited数组，用于标记点是否被访问过，以避免重复处理。

2、邻域查询函数 (region\_query)：

对于每个点，计算与其他所有点的欧式距离。

return距离小于或等于eps的所有点的索引。

3、扩展聚类函数 (expand\_cluster)：

从一个核心点开始，通过递归地探索其邻域中的所有点来扩展聚类。

使用队列来管理待探索的点，以实现广度优先搜索。

对于队列中的每个点，如果它是未访问的，则检查它是否能成为新的核心点，并继续扩展。

4、迭代处理：

遍历所有点，对于每个未访问的点，执行邻域查询。

如果邻域内的点数满足minPts，则从该点开始扩展聚类。

每成功扩展一个聚类，聚类ID递增。

**时间复杂度优化：**

采用了sklearn库中的euclidean\_distances函数来计算点与所有其他点之间的距离，这是一种向量化操作，可以比逐个计算距离更快地完成。然后进行了区域查询的优化，即通过先计算所有点的距离，然后筛选出在ε范围内的点，减少了重复的距离计算。虽然这种方法在最坏情况下的时间复杂度仍然为O(n^2)，但在实际应用中，由于距离矩阵的预计算和内存中的高效访问，性能得到了实质提升。如需要后续提升，可使用空间索引（如k-d树、R树）优化ε距离内的点的访问以降低复杂性。

**空间复杂度优化：**

该算法主要空间开销来自于存储每个点的标签labels和访问状态visited，这两者都是O(n)。额外的空间开销来自于存储距离计算结果和队列。通过使用简单的布尔数组来标记访问过的点，并利用双端队列（deque）来处理聚类扩展。队列只有在极端情况下可能需要存储接近n个元素，这样做既高效又节省空间。

**三、参数调优**

**数据集选择：**

为了测试DBSCAN算法，我使用了UCI机器学习库中的Iris数据集。

该数据集包含了150个样本和4个特征，每个样本描述了鸢尾花的花萼和花瓣的长度和宽度。Iris数据集的规模适中，且特征维度较低，这不仅有助于展示DBSCAN算法处理多维数据的能力，而且便于进行计算和可视化。此外，这个数据集包含三个不同种类的鸢尾花，其在特征空间中呈现出明显的聚类结构，这使得它成为测试聚类算法特别是DBSCAN性能的理想选择。由于数据集已经包含了真实的类标签，它还允许我们使用监督学习的评价指标来评估聚类的效果，例如调整兰德指数。但是，Iris数据集中并不包含噪声点或异常值，不足以展示DBSCAN在处理噪声方面的能力，所以可以人为添加一些随机分布的数据点。

**参数影响：**

DBSCAN算法的效果在很大程度上取决于ε和minPts这两个参数的选择。参数的不同取值可能会导致聚类结果的显著变化。

较小的ε值将导致大多数数据点被视为噪声，因为不够多的邻近点满足minPts条件，造成大量的单点聚类或者完全没有聚类，从而导致较低的ARI值；较大的ε值将本应分开的多个聚类合并为一个聚类，因为不同聚类之间的边界点可以互相到达，导致过度聚类，同样降低ARI值。

较小的minPts值使得较少的邻近点就可以形成核心点，导致更多的点被包含在聚类中，增加了噪声点被错误归类的风险；而较大的minPts值需要更多的邻近点才能形成核心点，这可以帮助算法识别出真正的密集区域，减少噪声点的干扰，但同时可能会导致一些边缘但有效的聚类点被视为噪声。

**参数选择：**

对于选择邻域半径ε，除了遍历枚举，有一个常见方法是使用k-距离图。

k-距离：给定数据集P={p(i)；i=0，1，2，......n}，计算点P(i)到集合D的子集S中所有点之间的距离，距离按照从小到大的顺序排序，d(k)就被称为k-距离。简单来说，对于数据集中的每一个点，计算它与最近的k个点之间的距离，并绘制这些距离的图。通常，这个图会在合适的ε值处出现一个拐点，在后面会具体说明。

对于选择minPts则较多凭经验。minPts定义了一个点的邻域中需要有多少个点才能将其视为核心点。minPts的选择与数据的维度、密度和噪声水平密切相关。一般来说，更高的维度和噪声水平需要更大的minPts值。

**实验设定：**

我将在Iris数据集上，尝试不同的ε和MinPts参数组合，以此来比较不同的组合对于结果的影响，尝试找到最好的参数组合。

由于轮廓系数不考虑噪声点，而DBSCAN可能识别出大量的噪声点，这可能会扭曲聚类效果的评价。所以采用调整兰德指数（Adjusted Rand Index, ARI）衡量聚类结果。而这就需要已经包含真实的类标签的数据集，所以Iris数据集被纳入考量。同时，之前提到过我会在数据集中人为添加一些随机分布的数据点，这些点在特征空间中与其他数据点明显不同。



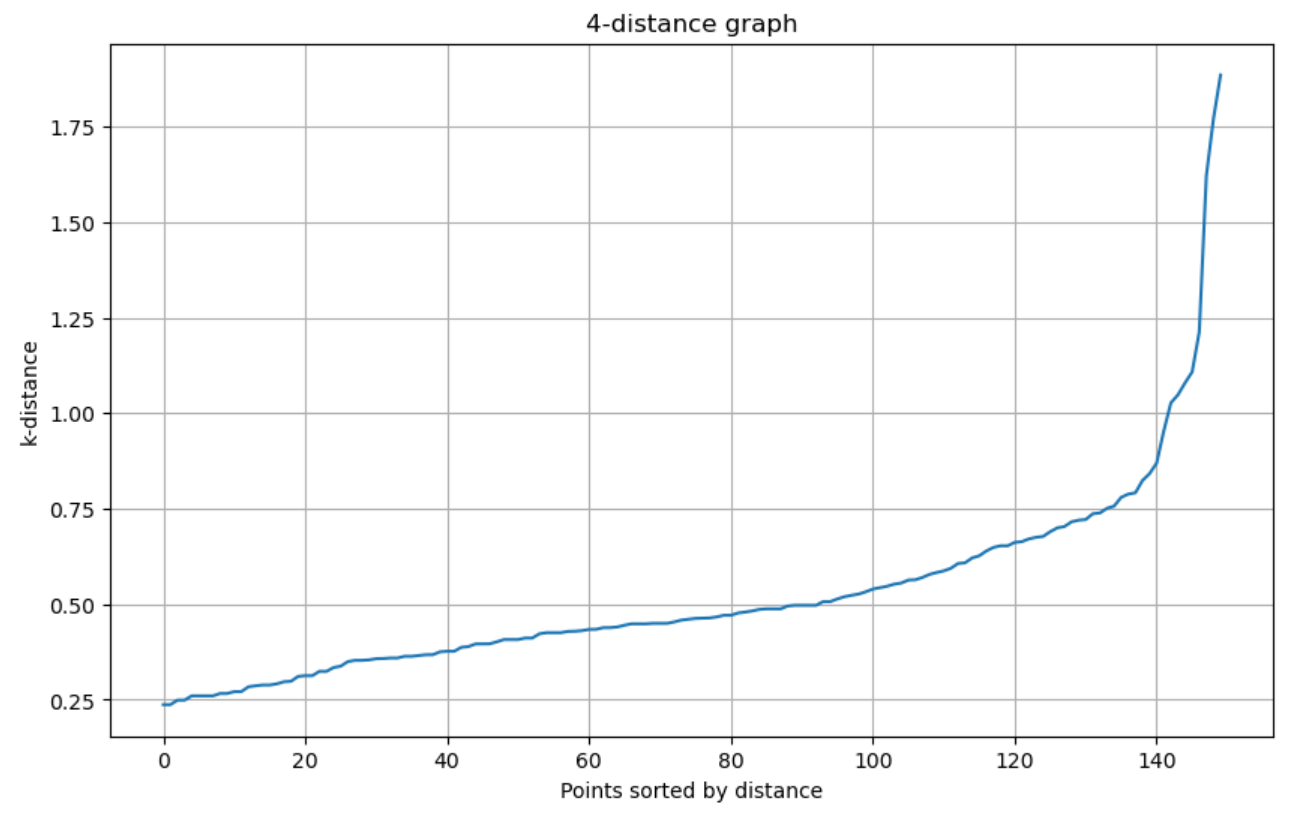
运行得到最佳参数组合：



然后我尝试绘制**k-距离图**来确定最佳参数组合。

DBSCAN的核心思想是基于密度的聚类。在k-距离图中，如果某个点的k-距离突然增大，这通常意味着从该点到其第k近邻的距离比其他点要大得多。这样的变化暗示了数据点之间密度的显著变化，即从密集区到相对稀疏区的过渡。因此，这个拐点附近的距离可以作为eps的一个理想选择，用来分隔密集的聚类与稀疏区或噪声。此外，当选择一个较小的eps值时，DBSCAN可能只能识别出非常密集的聚类，而忽视较少点的区域，这些区域实际上也可以构成有效的聚类。相反，如果eps过大，则可能将本应分开的不同聚类合并成一个。k-距离图中的拐点提供了一个平衡点，能有效区分不同聚类，同时识别出噪声。

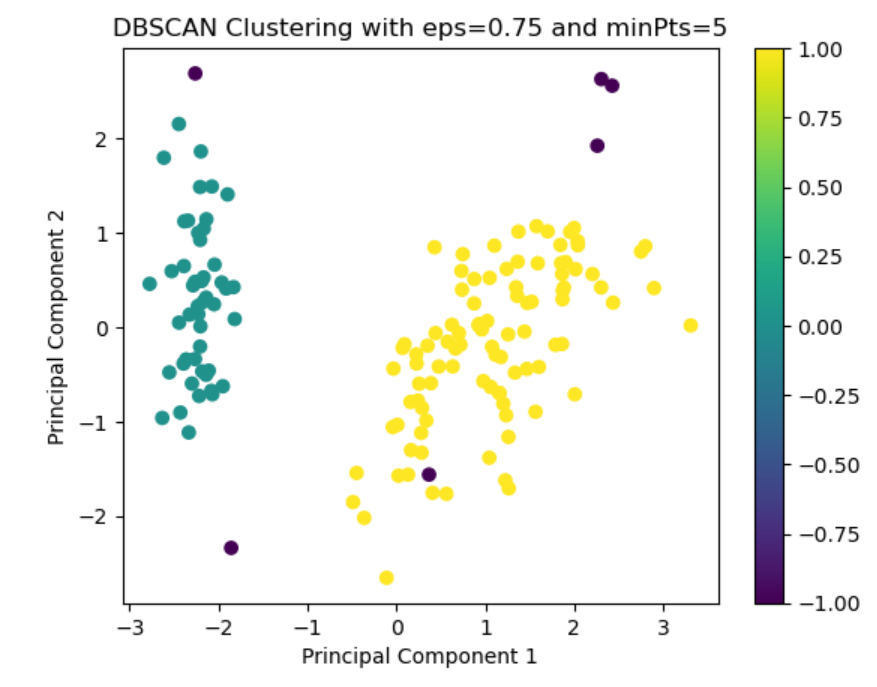




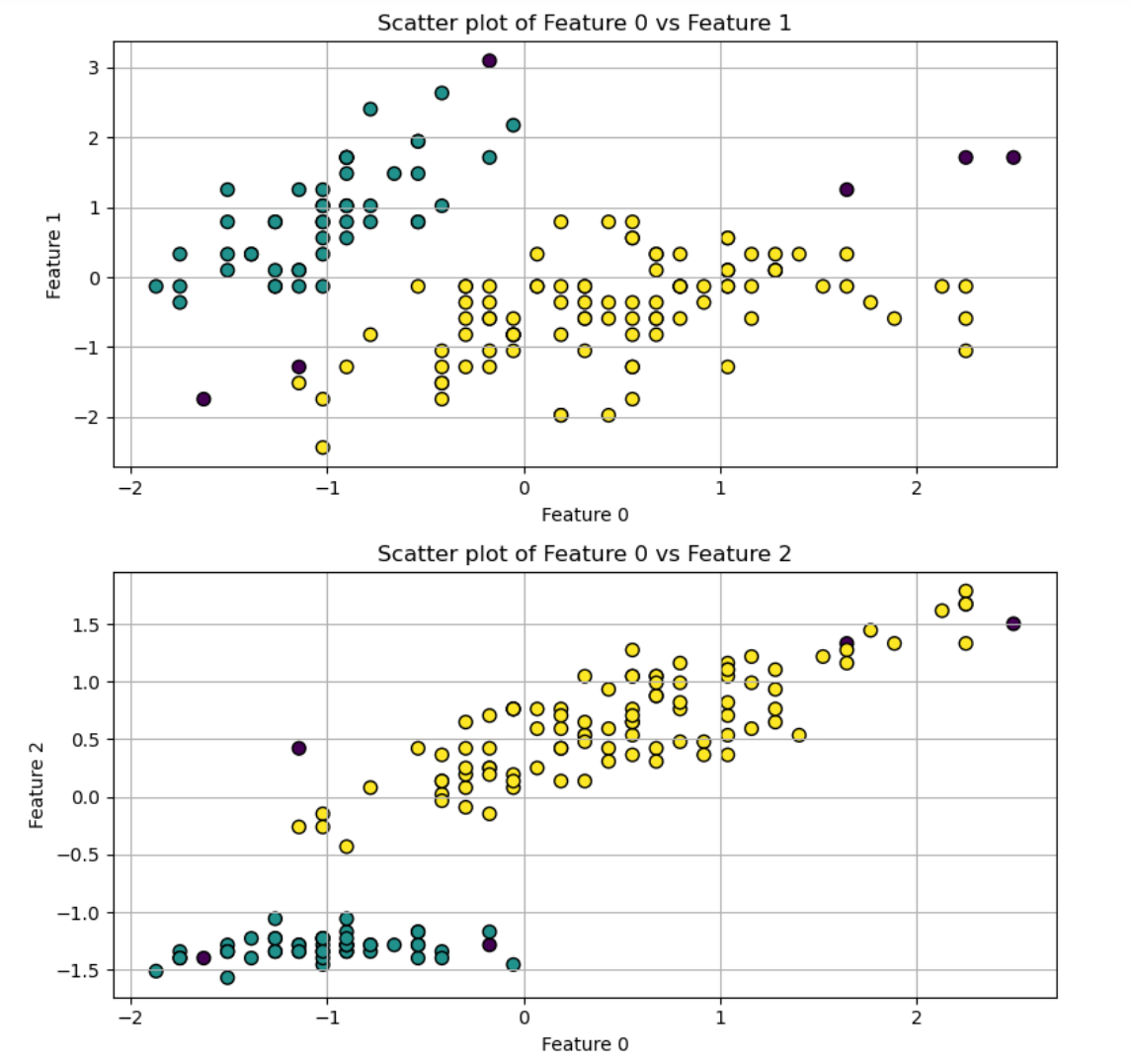
可观察到当纵坐标大于0.75时，曲线陡然上升，可取eps=0.75。

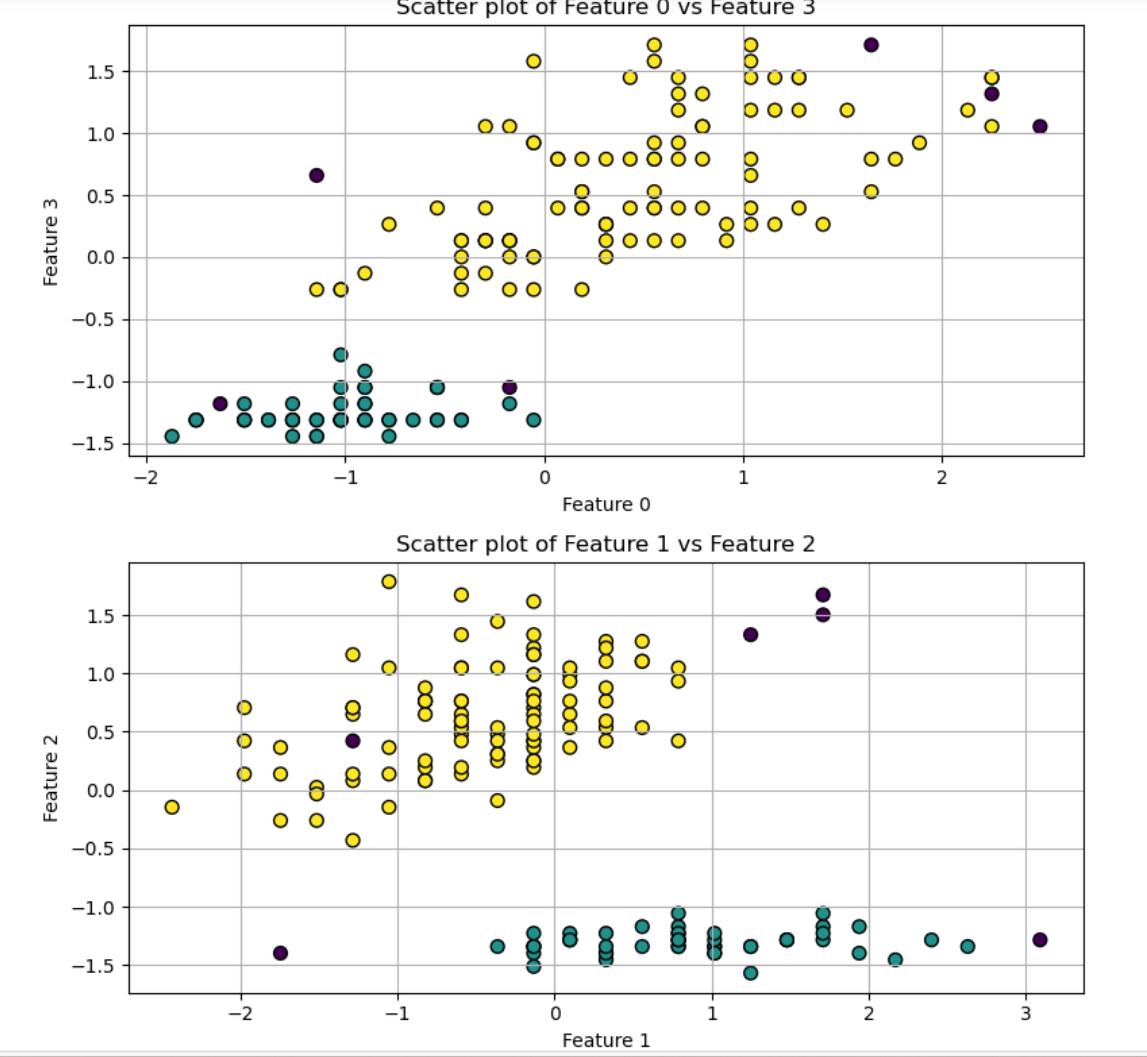
**聚类结果展示：**

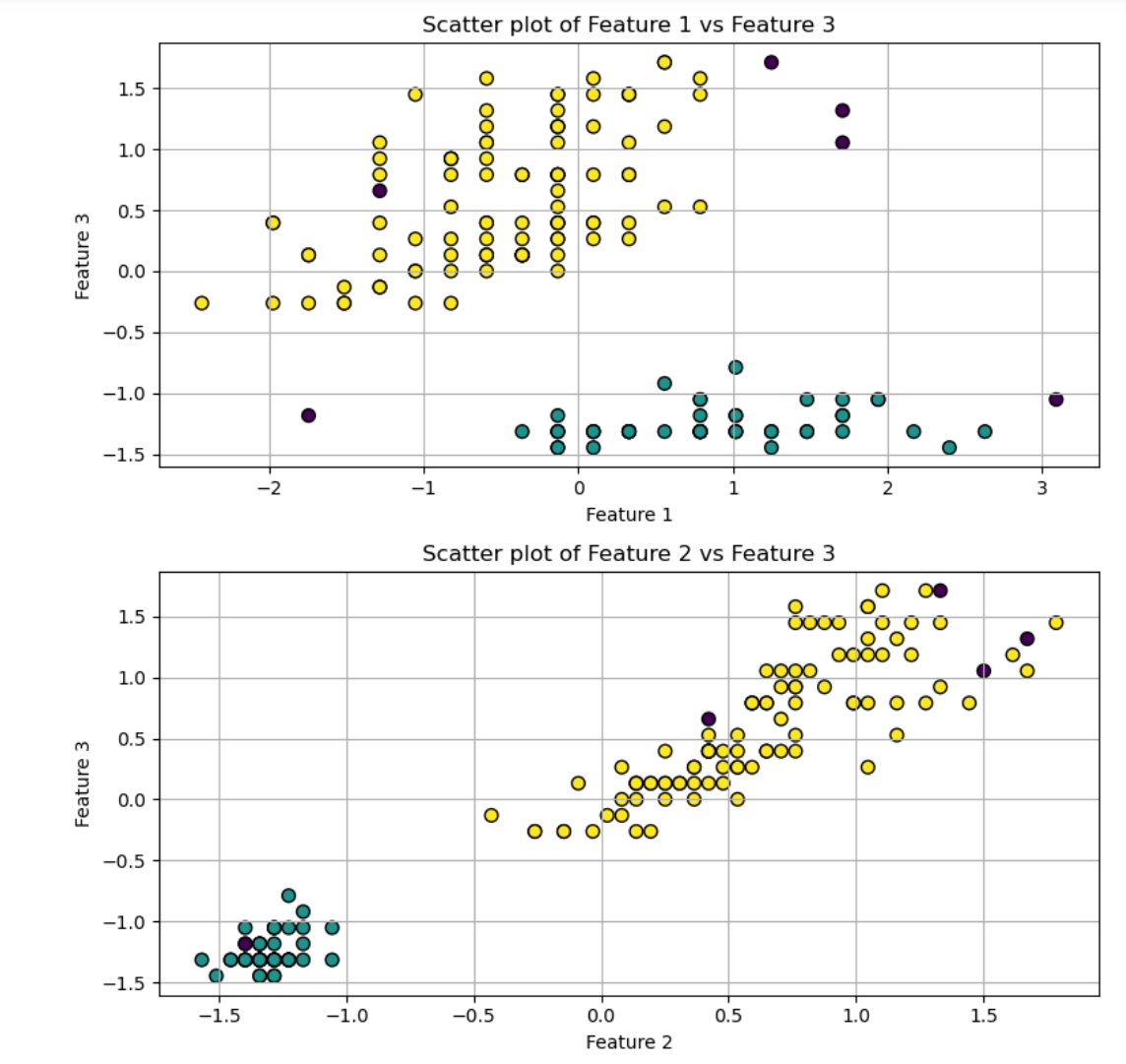
1. 通过PCA降维，绘制二维可视化图。



2、通过绘制特征对的散点图矩阵，这样可以观察不同特征组合下的聚类表现。







**四、DBSCAN算法的优缺点**

优势：

（1）不需要指定簇个数

（2）可以发现任意形状的簇

（3）擅长找到离群点

（4）对于数据库中样本的顺序不敏感

（5）只需要两个参数就足够

劣势：

（1）不能很好反映高维数据（可以做降维）

（2）参数难以选择（参数对结果的影响非常大）

（3）如果样本集的密度不均匀、[聚类](https://baike.baidu.com/item/%E8%81%9A%E7%B1%BB/593695?fromModule=lemma_inlink)间距差相差很大时，聚类质量较差