**1.实验目的**

完成 Kmeans 聚类算法以及二分 kmeans 的实现，并利用该算法对一些数据集进行预测，以此掌握基本的 Kmeans 算法以及二分 kmeans 算法，并期望能够利用该模型对数据集做出合理的预测。

首先完成了 Kmeans 基本聚类算法的实现，在此基础上针对 Kmeans 初始化 k 个中心点时可能出现由于位置太极端导致某些点在迭代过程中消失的问题，实现了二分kmeans 算法。同时，意识到二分 Kmeans 因为采取每次二分的方式，因此迭代次数会比较大。因此，针对这个问题，再次改进了算法，实现了 Kmeans++算法，同样是初始化 K个数据点，但 Kmeans++的基本原则是使得初始化中心点间的距离尽可能远，从而避免出现 Kmeans 问题。 最后，针对以上 Kmeans 算法以及改进算法中心点的个数 K 是手动选择可能不是最优解，给出了两种可视化方法：聚类散点图，以及手肘法效果图，以判断不同的 K 的聚类效果的好坏。

**2.普通K-means**

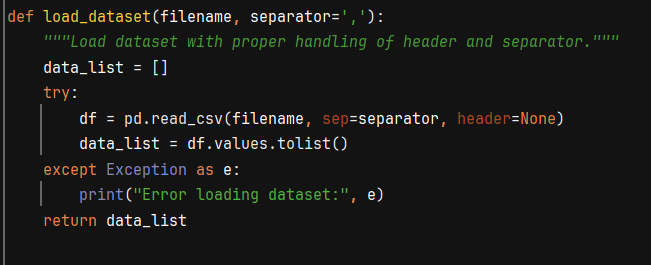
Kmeans 聚类算法：聚类算法属于无监督学习，将一个庞大的数据集中具有相似特征的 数据汇聚在一类，称为一个簇。簇内的数据点距离越近越好，簇外的数据点相聚越远越好。 Kmeans 就是 K 均值，即构造 K 个不同的簇，且每一个簇的中心点均由该簇内点的平均值构成。

Kmeans 分类核心是欧氏距离，公式为：𝑑 = √∑𝑛 𝑖=1 (𝑥𝑖 − 𝑦𝑖) 2。Kmeans 通过计算数据点之间的欧氏距离来判断数据点的远近，并以此判断该数据点属于哪一个类簇。Kmeans 首先随机地初始化 k 个数据点，然后分别计算每个样本点到中心点的距离，并将该样本点归到最近的中心点的簇中。其次，在所有簇划分完毕后，计算每一个簇中样本点的平均值，以此形成新的中心点并替代原来该簇中的中心点。重复以上过程，直到每一个样本点的归类均不再改变为止。

2.1 函数设计

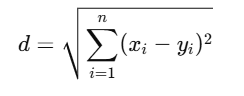
load\_dataset

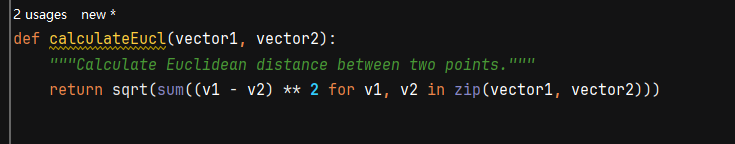
加载数据集，并将其转换为二维列表。使用 pandas 读取文件，避免分隔符或数据格式问题。返回一个列表形式的数据集，供后续处理。



calculateEucl

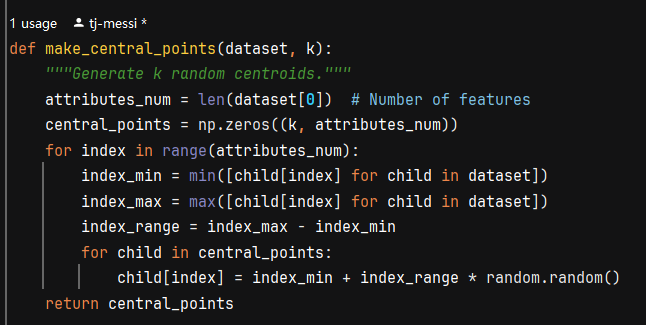
计算两点之间的欧氏距离。对应维度的差值平方累加。对累加和取平方根，返回欧氏距离。





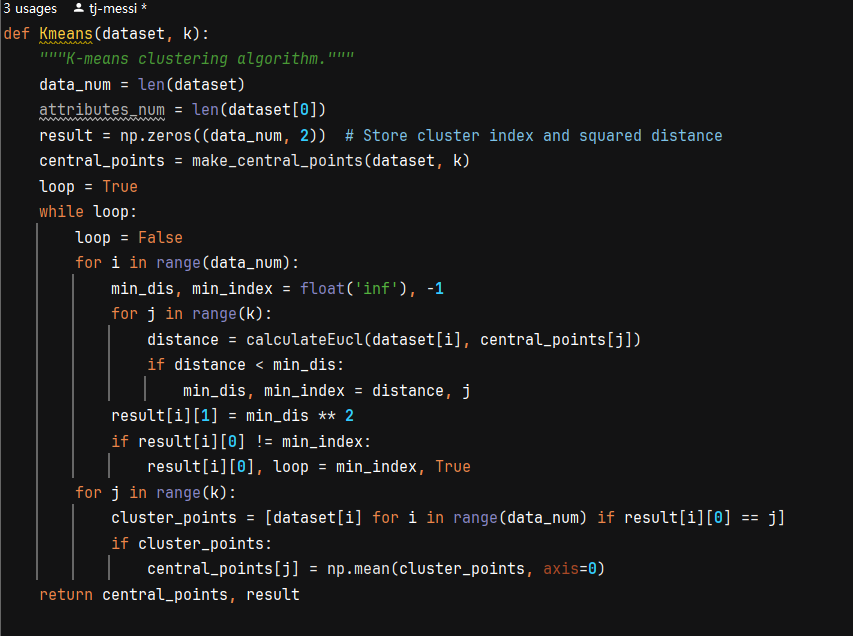
make\_central\_points

生成 k 个随机初始质心。遍历数据的每个维度，找到最小值和最大值。在每个维度的范围内随机生成 k 个点，作为初始质心。

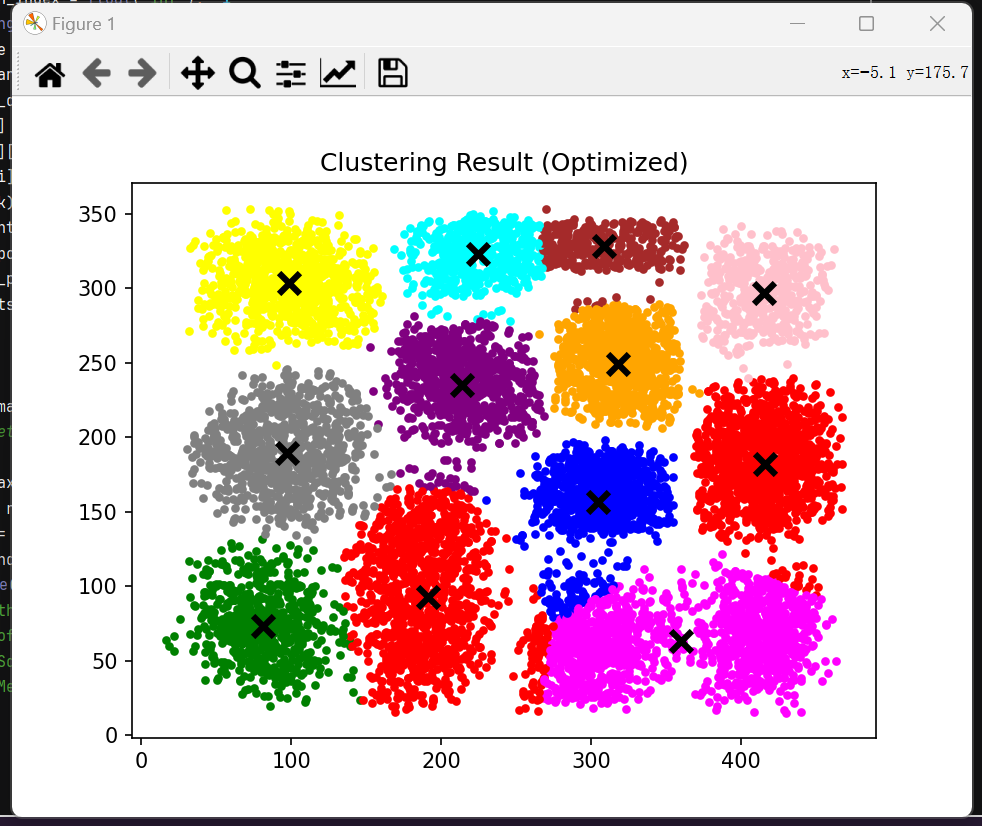


Kmeans

实现标准的 K-means 聚类算法。初始化 k 个质心。**重复迭代**直到质心不再移动：计算每个点到质心的距离，分配到最近的簇。计算每个簇的均值，更新质心位置。 返回最终的质心和每个点的簇分配结果。



2.2效果



**3.改进的二分K-means**

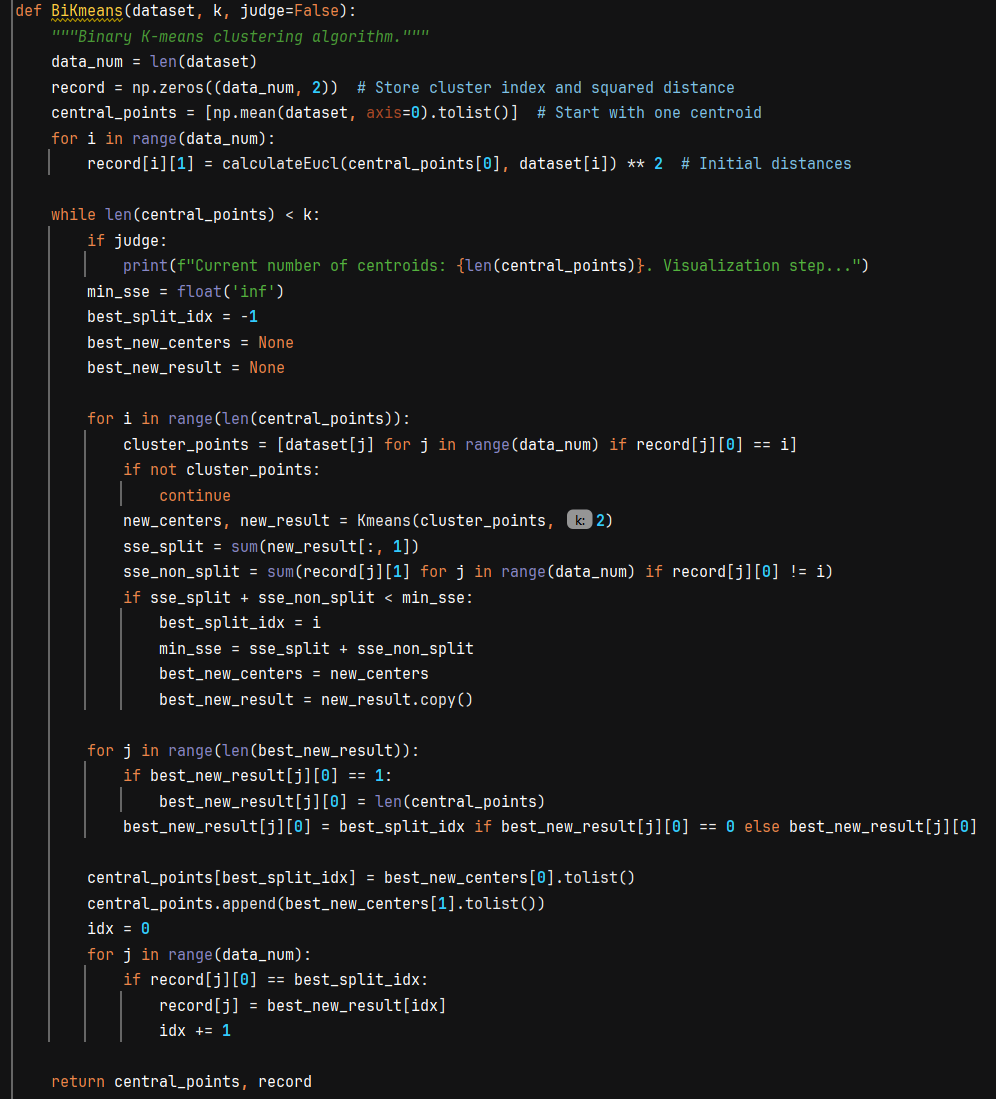
将所有点视为一个簇，以均值为初始质心。

**循环**直到簇数达到 k：选择一个簇进行二分，使用标准 K-means 分成两个簇。更新所有点的簇分配和距离记录。返回最终的质心和簇分配结果。

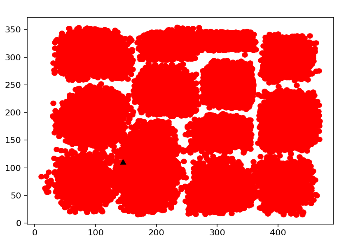
3.1功能函数

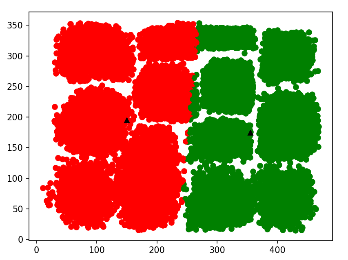
BiKmeans

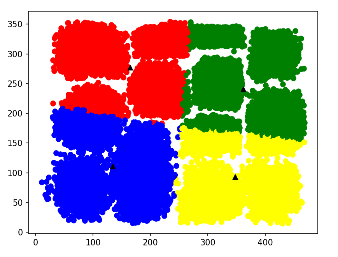
选择一个簇进行二分，使用标准 K-means 分成两个簇。在第一个点用mean平均选择好后，选择这个簇，调用二分mean

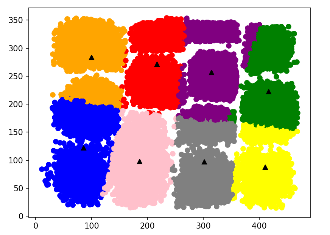


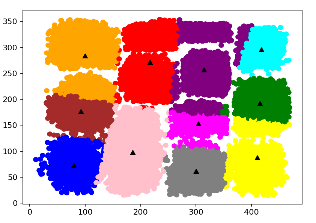
3.2效果（二分的流程）











**4.可视化工具**

show\_cluster\_optimized

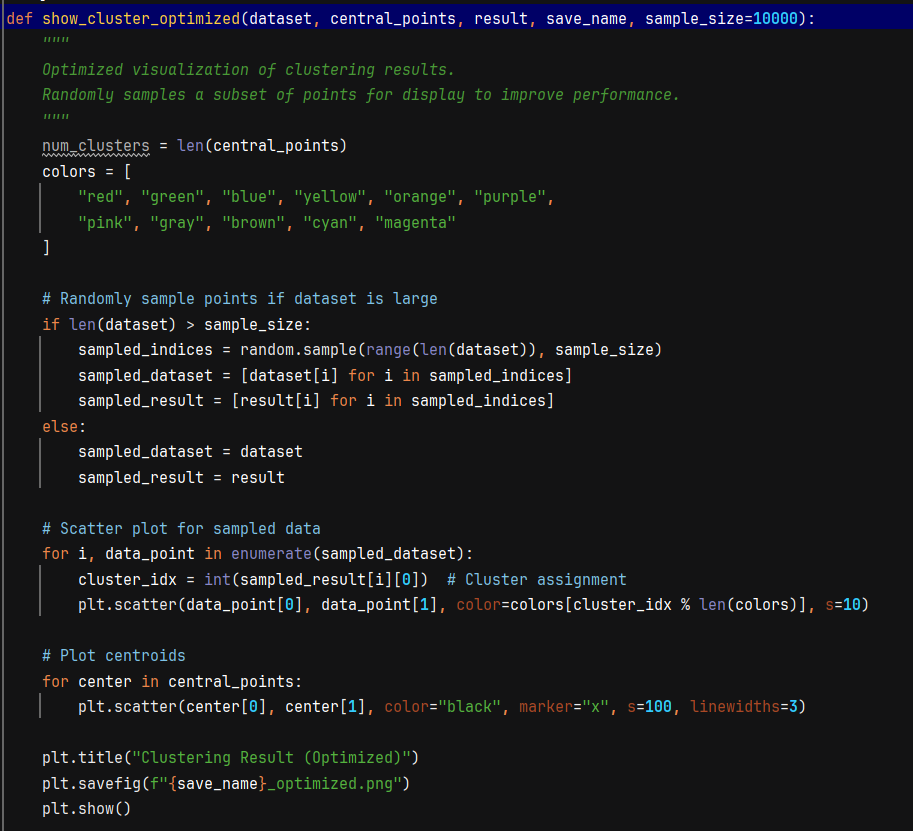
如果数据点过多，随机抽样部分数据点进行可视化。

根据每个点的簇分配，绘制不同颜色的散点图。

绘制质心，使用黑色叉标记。

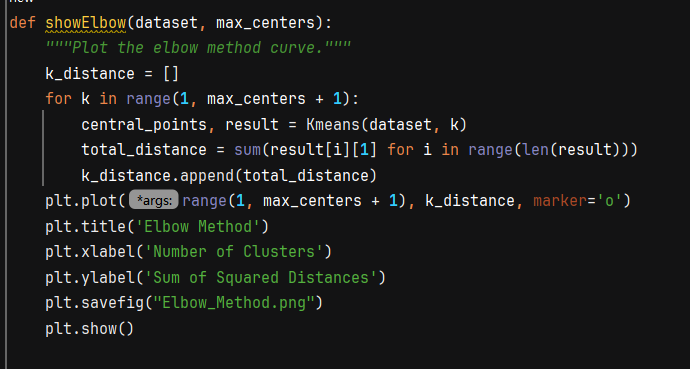
实现样本分布的散点图。设置六种颜色，每一个类簇对应一种颜色。采用

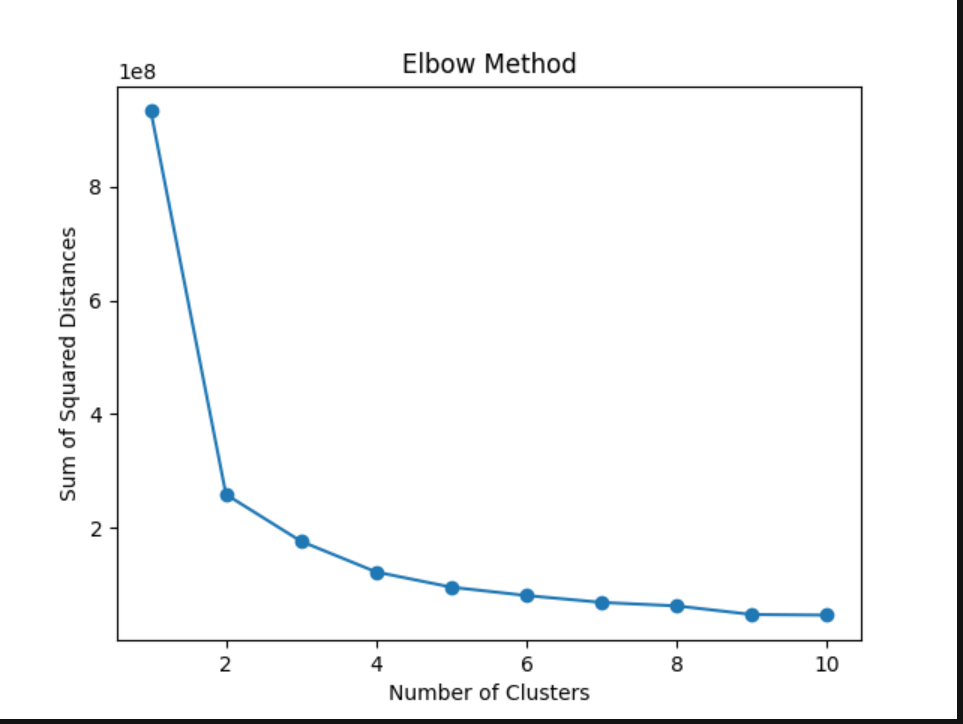
matplotlib 库中的 pyplot 进行绘制。



showElbow

因为认为设定的 K 值未必一定是最佳聚类(虽然一般来说聚类的类簇数量应该由人为控制,这里只用于研究哪一种 K 值是最佳的)，所以我们采用手肘法将每一个 K 与其欧式距离之和的函数关系表现出来。





1. **数据集**

在 DataCastle 中的 Clustering Exercise 数据集中选 box3.csv 用作测试

数据，该测试数据有 21644 条数据。希望通过该数据集进一步验证模型的准确

性。

1. **心得体会**

Kmeans 实现的大致思路是通过随机初始化中心点确定初始的类簇，随后再通过

类簇中样本点欧氏距离的平均值修改中心点坐标，以及修改相应的类簇中的数据点，直到每一个类簇中的样本点均不再改变终止循环。Kmeans 可能出现的聚类效果不好的问题主要在于其初始化中心点的过程，其 K个中心点完全是随机的，故如果中心点选取的不好，很有可能导致聚类效果不好、更新次数过多甚至无法收敛等问题。因此，在此基础上，我们有两个改进，都是针对中心点的选取的算法的改进。

一.二分 Kmeans：改进中心点选取的方案一。只初始化一个中心点，再逐步对中

心点进行二分，确保聚类能够收敛，但有可能迭代次数过多，消耗较大。

二.Kmeans++：改进中心点选取的方案二。与 Kmeans 的初始化类似，都是要选择K 个中心点，但是 Kmeans++的中心点并不是随机的，而是使得中心点之间的距离尽可能远，从而对数据有整体的把握，因而中心点选取的效果较好，聚类的效果也较好。