实验4：基于WEKA的DBSCAN聚类算法

**1 实验目的**

1. 掌握WEKA平台的DBSCAN聚类算法。

**2 实验平台与工具**

1. Windows、Linux操作系统
2. 数据分析与挖掘系统WEKA

**3 实验内容**

1. （必做）自主设定数据挖掘的DBSCAN聚类应用，采用WEKA平台完成整个挖掘过程，具体要求如下：
   1. 解释说明该挖掘应用的背景、动机和需求；
   2. 自主选择相应的数据集，数据规模不需要太大；
   3. 对于挖掘所得的聚类结果具有可解释性，请解释所得簇的含义；
2. （选做，加分题）基于WEKA平台DBSCAN聚类算法的性能测试与分析
   1. 随数据规模变化后的DBSCAN算法性能变化分析；
   2. 相同数据规模，不同eps、minpts变化后的挖掘算法性能变化分析；
   3. 相同eps、minpts，数据规模变化后的挖掘算法性能变化分析；

**4. 规则与要求**

（1）独立完成，严禁相互抄袭（如有发现抄袭和被抄袭均判为0分），以及从网络上直接摘抄别人的观点和总结（该行为将影响报告成绩）。

（2）实验报告符合学术写作的排版要求，请参考群文件中的“报告模板.docx”和“参考文献格式.docx”的排版格式。

（3）实验报告内容详实，采用图文混合的方式叙述安装和配置过程。

Tip：Win+Shift+S 在Windows中可以快速截屏。

（4）报告文件见附件，提交报告时请以附件形式插入到超星作业中。

实验报告

报告标题：基于WEKA的DBSCAN聚类算法

学号：21190630

姓名：黄艺杰

日期：5月23日

# 一、实验环境

1. 操作系统：Windows10

2. 软件（含版本号）：weka-3-8-6-azul-zulu-windows

# 二、实验内容及其完成情况

（1）挖掘应用的背景、动机和需求

背景：DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是一种基于密度的聚类算法，用于发现具有相似特征的数据点并将其分组为簇。DBSCAN算法会将数据点分为三类：核心点、边界点和噪声点。根据聚类结果，可以解释得到的簇的含义：

核心点：核心点是指在eps邻域内至少有minPts个数据点的点。它们属于高密度区域的中心，代表了簇的核心。

边界点：边界点是指在eps邻域内不满足minPts条件，但位于核心点的邻域内的点。它们可以被视为与核心点相邻的点，但是它们自身的邻域密度较低。

噪声点：噪声点是指既不是核心点也不是边界点的点。它们位于稀疏区域或者远离任何簇的位置。

动机：DBSCAN算法作为一种基于密度的聚类算法，被广泛应用于数据挖掘领域。使用DBSCAN算法对Glass数据集进行聚类分析的动机包括以下几点：

1）可以帮助提取不同类型玻璃样本的特征信息，包括折射率、氧化物含量等，这有助于进一步研究和理解玻璃样本的特性和特征。

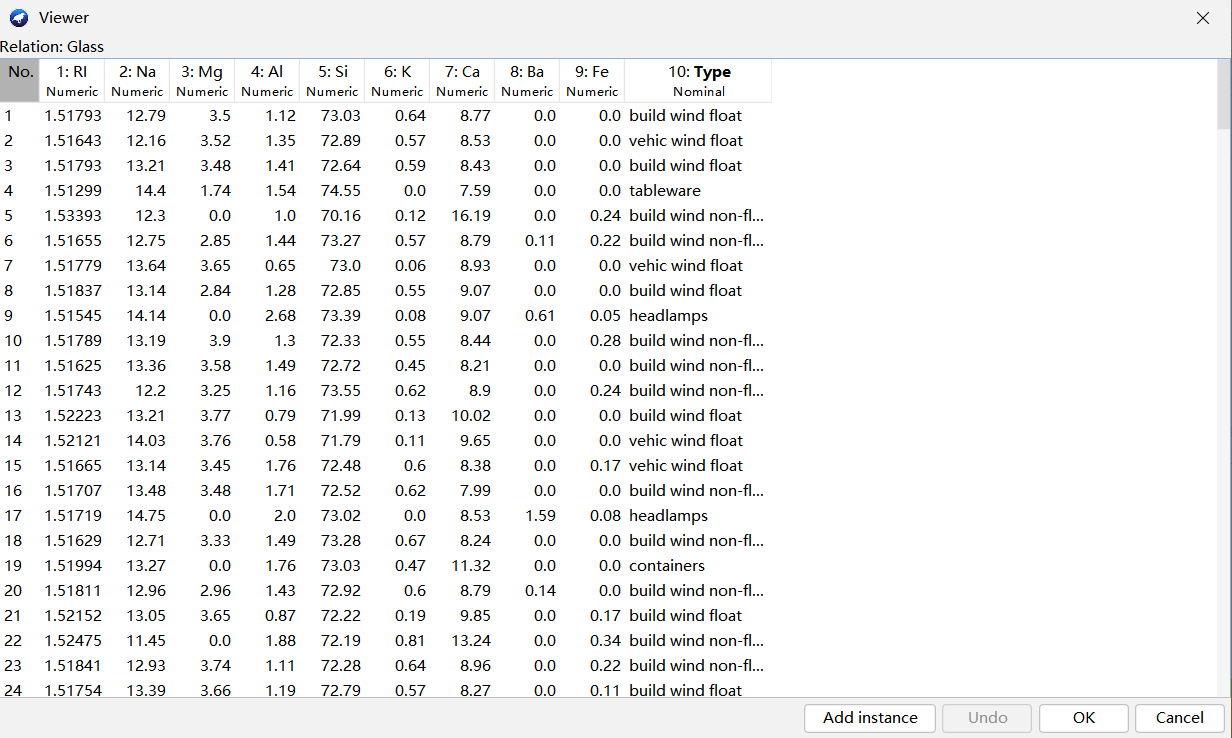
2）可以将具有相似特征的玻璃样本划分为不同的簇，并为每个簇分配一个标签。这有助于玻璃分类和识别任务，帮助区分不同类型的玻璃样本

3）可以识别与其他样本差异较大的异常样本。这有助于在玻璃制造过程中进行异常检测和质量控制，及时发现和处理不合格的产品。

需求：DBSCAN算法的聚类结果受到eps（邻域半径）和minPts（最小邻域点数）等参数的影响。合理选择这些参数是为了获得准确的聚类结果。因此，需求包括确定合适的参数选择策略，以获得具有较好性能的聚类结果。

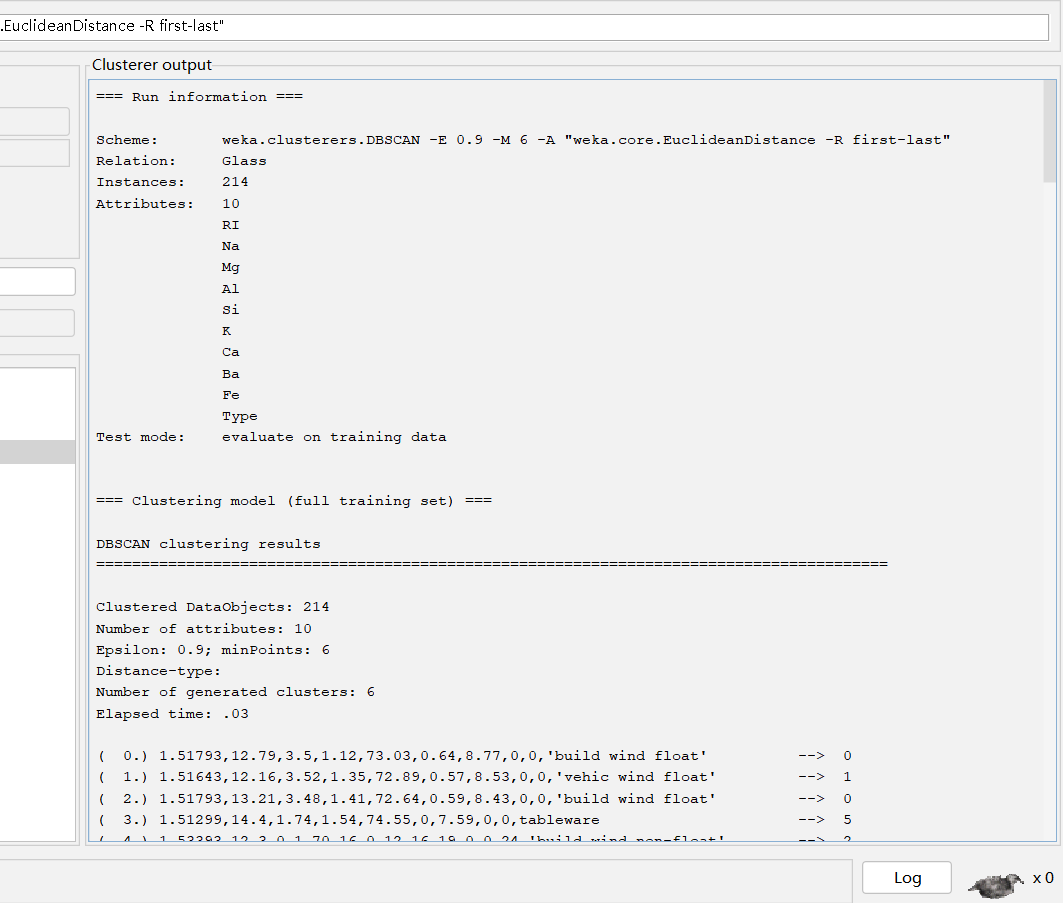
（2）数据集介绍

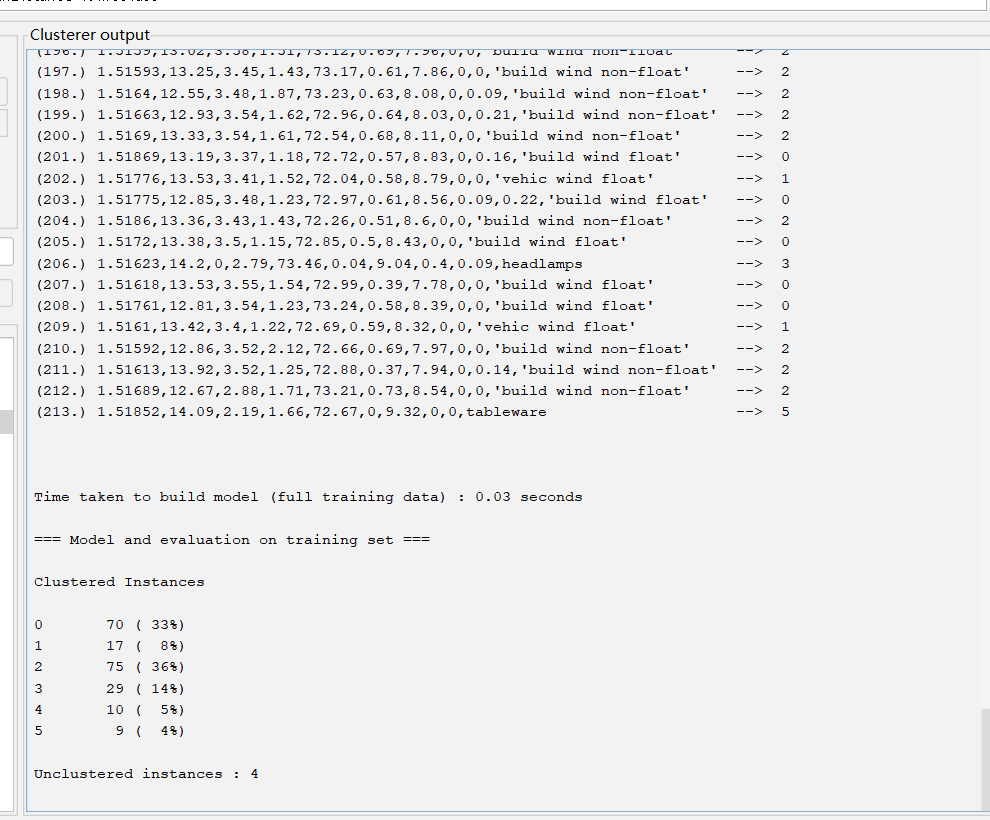
Glass数据集是一个经典的数据集，包含了不同类型的玻璃样本，每个样本都具有一系列特征，如折射率、氧化物含量等。该数据集常用于玻璃分类和质量分析的研究领域。文档自带的Glass数据集包括241条数据，每条数据有9个特征属性，共有6个种类。部分数据如下图所示



（3）实验结果的解释

使用weka平台上的DBSCAN算法对glass.arff数据集进行聚类的结果如下图：





聚类模型的参数设置为：Epsilon (邻域半径)为0.9，minPoints (邻域内最小点数)为6，数据集中共有214个数据对象，每个数据对象具有10个属性，使用欧几里德距离作为距离度量，构建模型的时间为0.03秒。

实验结果显示，共生成了6个簇。每个数据对象都被分配到一个簇中，并通过簇的编号进行标识。除此之外，还有4条数据被标记为噪声（NOISE），它们既不是核心点也不是边界点的点。

簇 0: 该簇包含70个数据对象。该簇代表"build wind float"类型。

簇 1: 该簇包含17个数据对象，该簇代表"vehic wind float"类型。

簇 2: 该簇包含75个数据对象，该簇代表"build wind non-float"类型。

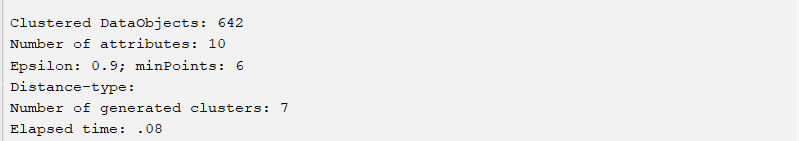
簇 3: 该簇包含29个数据对象，该簇代表"headlamps"类型。

簇 4: 该簇包含10个数据对象，该簇代表"containers"类型。

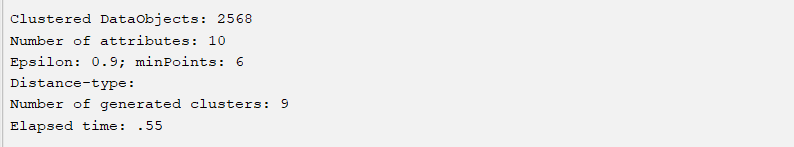
簇 5: 该簇包含9个数据对象，该簇代表"tableware"类型。

1. 基于WEKA平台DBSCAN聚类算法的性能测试与分析
2. 随数据规模变化后的DBSCAN算法性能变化分析；

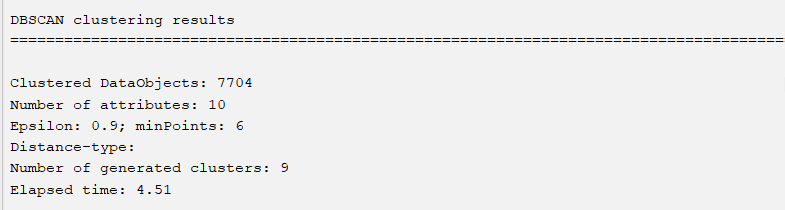
通过逐渐增加数据集的规模来观察算法的性能变化。我分别使用642条数据、2568条、7704条数据进行实验，并保持其他参数不变，结果它们所消耗的时间分别为0.08s、0.55s和4.51s，可以明显看出DBSCAN算法的执行时间会随着数据集规模的增加而增加。这是因为随着数据对象的数量增加，计算每个对象之间的距离和邻域关系的复杂度也会增加。



数据量=642



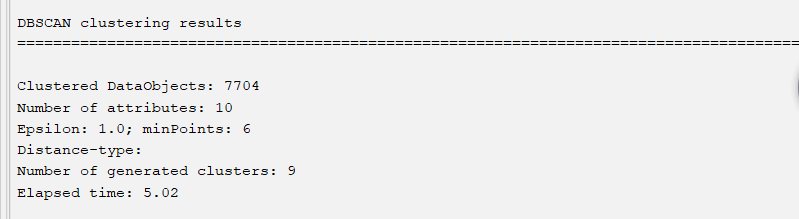
数据量=2568



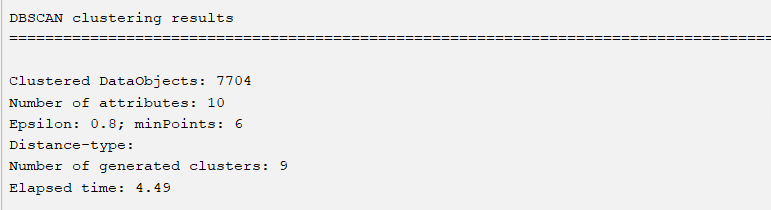
数据量=7704

1. 相同数据规模，不同eps、minpts变化后的挖掘算法性能变化分析；

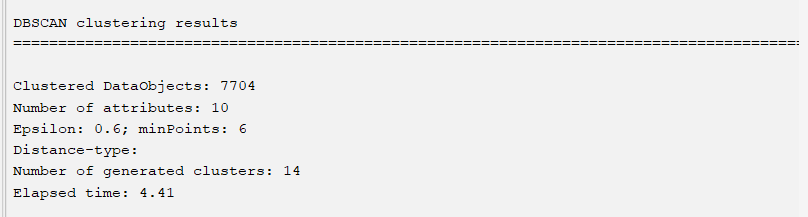
eps参数定义了邻域半径的大小，而minpts参数定义了最小邻域内数据对象的数量。我保持数据量不变，设定minpts为6，分别设定eps的值为1.0、0.8、0.6、0.4，得到的运行时间为5.02s、4.49s、4.41s、4.15s。



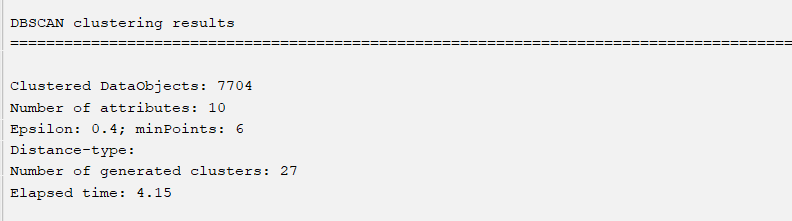
eps=1.0



esp=0.8

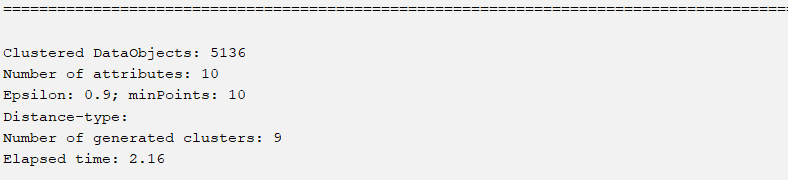


esp=0.6

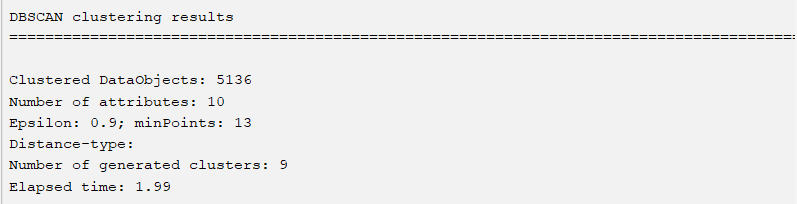


esp=0.4

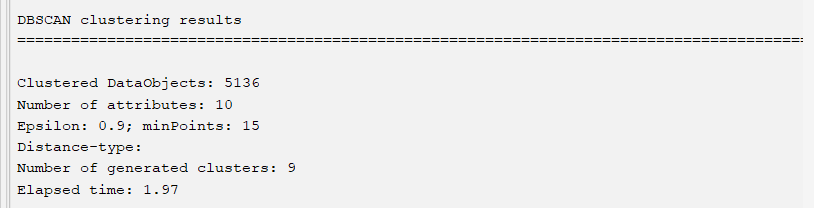
保持esp的值不变，我分别将minpts设置为10、13、15、20、22，得到的运行时间为：2.16s、1.99s、1.97s、1.96s、1.95s



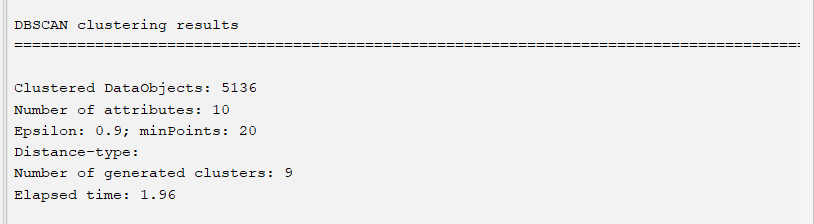
minpts=10



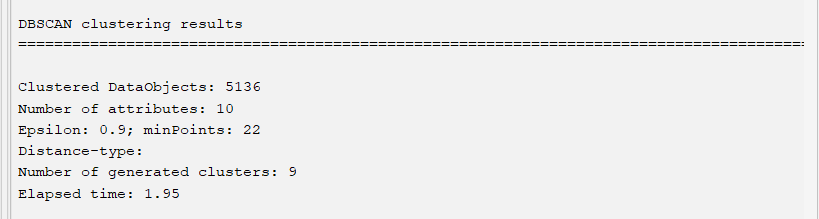
minpts=13



minpts=15



minpts=20



minpts=22

通过上述实验比较，可以得出以下结论：

增大

增大minPts会使得算法运行时间减少，这是因为增大minPts时，聚类的点必须更加密集，从而减少了噪声点的数量并且减少了需要被搜索的点的数量。因此，算法所需的计算次数也会减少。同时，因为增大minPts使得聚类的点更加密集，这也意味着可以更快地检测到一些簇并将它们合并。本实验中的效果并不明显，可能因为本数据集非常稠密且存在噪声点。

1. 相同eps、minpts，数据规模变化后的挖掘算法性能变化分析；

与a）情况一致，在相同的eps和minpts参数下，通过改变数据集的规模来分析DBSCAN算法的性能变化。通过分析数据规模变化对DBSCAN算法性能的影响，可以明显看到执行时间增加，因为较大的数据集需要更多的计算操作来计算对象之间的距离和邻域关系。

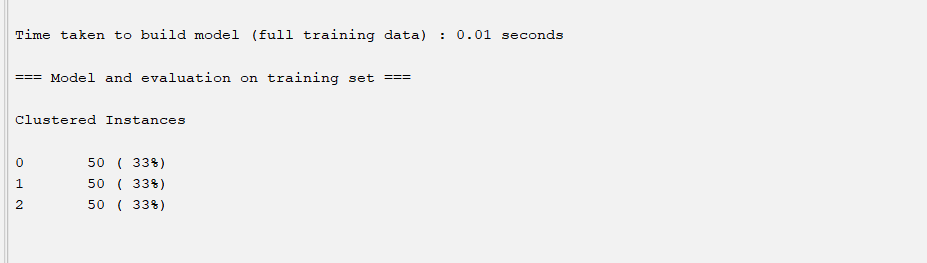
# 三、实验总结

（可以总结实验中出现的问题以及解决的思路，也可以列出没有解决的问题）

问题1：

1. 问题描述

刚开始的时候，我数据集选择的是鸢尾花，但经过实验发现鸢尾花数据的噪声太少，实验结果全部都正确聚类划分，如下图所示



1. 问题分析（可能的原因、难点、挑战）

实验中使用的鸢尾花数据集的数据规模小，数据噪声小且相互区分大，不同簇之间密度较高，没有很多稀疏的点，即既不是核心点也不是边界点的点

1. 解决方案

更换数据集到Glass数据集，重新运行