**聚类算法在不同数据集实验结果的整理和分析**

2024年10月14日-10月28日 Report

10月30日修改

**1 实验内容回顾**

接上次总结，我在10月14日前完成了对所有数据集的清洗工作，采用目前综合表现相对优秀的Raha-Baran算法对6个数据集[[1]](#footnote-1)进行了错误检测和错误修复。

10月14日-10月27日逐步对每个数据集的11种不同错误率（从0%-70%不等）[[2]](#footnote-2)分别运行了6种不同的聚类算法，具体实验步骤如下：

**STEP 1：数据预处理。**包括读取CSV文件、删除ID列、随机选择目标列进行聚类分析。对分类数据进行频率编码，并对数据进行标准化处理，以确保聚类的效果。

**STEP 2：参数调优。**对聚类算法的核心参数（如DBSCAN的eps和min\_samples，OPTICS的xi和min\_cluster\_size）进行了广泛调优[[3]](#footnote-3)：

这里详细地总结了不同的聚类算法的特征和参数的选择过程，以便于后期分析实验结果与算法的关系：

**(i) 基于原型的聚类（Prototype-based Clustering）**

这些算法将聚类中心或模型作为原型，通过优化目标函数将数据点分配给不同的原型。

**K-means**是经典的基于原型的聚类算法，假设簇是球状的，并通过最小化簇内平方误差（SSE）来分配点到簇。适合处理球状且大小均匀的簇。

实验中参数选择：

最大簇数的设定：使用数据集大小的平方根乘以2来确定max\_clusters，为不同数据集提供合理的初始簇数范围。

**Gaussian Mixture Model (GMM)** 将数据视为由多个高斯分布组成的混合模型，通过最大似然估计来确定簇的中心和协方差矩阵。适合处理椭圆形簇且簇大小不均的情况。

实验中参数选择：

最大簇数的设定：使用数据集大小的平方根乘以2来确定max\_clusters，为不同数据集提供合理的初始簇数范围。

高斯混合模型的主要参数：

n\_components（高斯混合模型中的成分数量）：范围设定为2到max\_clusters，表示可能的簇数。

covariance\_type（协方差类型）：包括full、tied、diag和spherical四种类型，用于控制协方差矩阵的计算方式。

**(ii) 基于层次的聚类（Hierarchical Clustering）**

这些算法通过构建树状的层次结构来进行聚类，簇之间具有嵌套关系。

**Hierarchical Clustering (HC)** 层次聚类方法，通过自下而上（凝聚式）或自上而下（分裂式）策略，逐步合并或分割簇，直到形成最终的聚类。适合需要分析不同尺度上的聚类结构的场景。

实验中参数选择：

最大簇数的设定：使用数据集大小的平方根乘以2来确定max\_clusters，为不同数据集提供合理的初始簇数范围。

**(iii) 基于密度的聚类（Density-based Clustering）（跑10次取最好的结果）**

这些算法通过找到密度较高的数据区域来形成簇，能有效应对噪声和形状复杂的簇。

**DBSCAN**通过设定半径和最小样本点数来形成密度簇，并标记低密度区域中的点为噪声。适合处理任意形状的簇和具有噪声的数据。

实验中参数选择：

eps（半径距离）：用于定义样本点在同一个簇内的最大距离。参数范围设置为0.1到1.5，以步长0.01进行遍历。

min\_samples（最小样本数）：用于定义一个核心点需要邻居点的数量，范围设置为3到50，以3为步长进行调整。

**OPTICS**是DBSCAN的改进版本，适合处理簇间密度差异较大的场景。通过生成 Reachability Plot 来展示簇的层次结构，能够识别不同密度的簇。

实验中参数选择：

min\_samples（最小样本数）：用于定义一个样本点被视为核心点所需的最少邻居数。其范围设定为[5, 10, 20, 30]，分别进行测试。

xi（集群密度的变化率）：用于确定一个簇的边界。值越小，要求簇的边界越清晰。其范围设定为[0.01, 0.05, 0.1]。

min\_cluster\_size（最小簇大小）：定义一个簇中至少包含的样本比例。设置为[0.01, 0.02, 0.05]，以保证簇的大小适应数据集的规模。

**(iv) 基于图的聚类（Graph-based Clustering）**

这些算法通过构建节点和边的图结构来进行聚类。

**Affinity Propagation (AP)** 通过信息传递的方式寻找簇的代表点（中心），不需要预先定义簇的个数。适合在不确定簇数的情况下进行聚类，但对于大规模数据，计算代价较高。

实验中参数选择：

damping（阻尼系数）：用于控制簇中心更新的速度，范围设置为0.5到0.9，步长为0.05。

preference（偏好）：用于影响点被选择为簇中心的倾向，范围设置为-500到-100，步长为50。

**2 聚类实验结果整理总结**

对于每次运行，我首先绘制出了Davies-Bouldin分数和Silhouette 分数随聚类簇数变化的关系图像[[4]](#footnote-4)，作为第一个中间结果。以下是最合适簇数的寻找过程：

**STEP 1：利用肘部法则（Elbow Method）。**对于Davies-Bouldin分数，通常可以观察到当簇数增加时，DB分数会逐渐减少，直到某一拐点之后减少的速度变慢或者接近收敛。在拐点处是理想的簇数选择，因为在此之后增加簇数并不会显著提高结果。通过图像可以观察DB分数随簇数减少的情况，在这条线的拐点（肘部）就是一个较好的聚类簇数。

**STEP 2：轮廓系数 (Silhouette Score)**。轮廓系数表示聚类的质量，数值越高表示聚类效果越好，通常介于0到1之间。当簇数增加时，如果轮廓系数达到一个峰值，说明这是一个较好的聚类选择。我通过观察轮廓系数随簇数的变化，寻找局部最大值的点。

**STEP 3：结合观察DB与轮廓系数。**在我所绘制的图像中，Davies-Bouldin分数越低越好，而轮廓系数越高越好，因此我会寻找在两个指标都表现较优时的簇数位置。例如，在某些簇数点，DB分数较低且轮廓系数相对较高时，可能是最合适的聚类簇数。

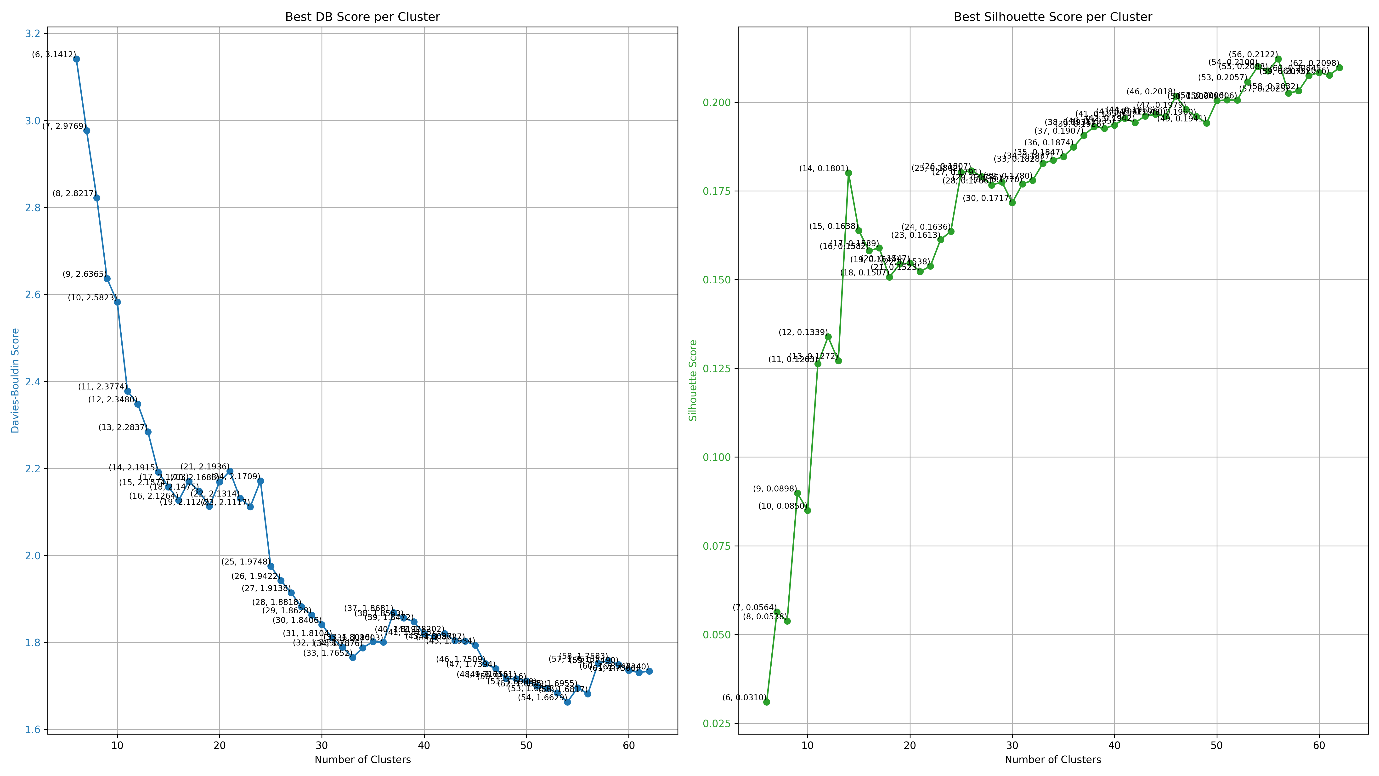


图2.1 *hospital*数据集在21.65%错误率下进行清洗后运行HC算法的结果，可选33作为最佳簇数

在选出最佳聚类簇数后，我将该运行图像下最佳簇数对应的Davies-Bouldin分数和Silhouette 分数分别计入表格。接下来，我绘制出了Raha-Baran算法修复下不同数据集在不同聚类算法下评价指标随错误率变化的关系图像[[5]](#footnote-5)，作为下一个整理结果。

图表, 折线图

描述已自动生成

图2.2 flights数据集进行清洗后运行HC算法，得到的检测指标随错误率变化的图像

不过这个图像并没有明显呈现不同聚类算法之间的对比，所以我做了适当的整合，绘制出了Raha-Baran算法修复下不同数据集进行聚类算法的最终结果（这里以*beers*数据集为例）。

图表, 折线图

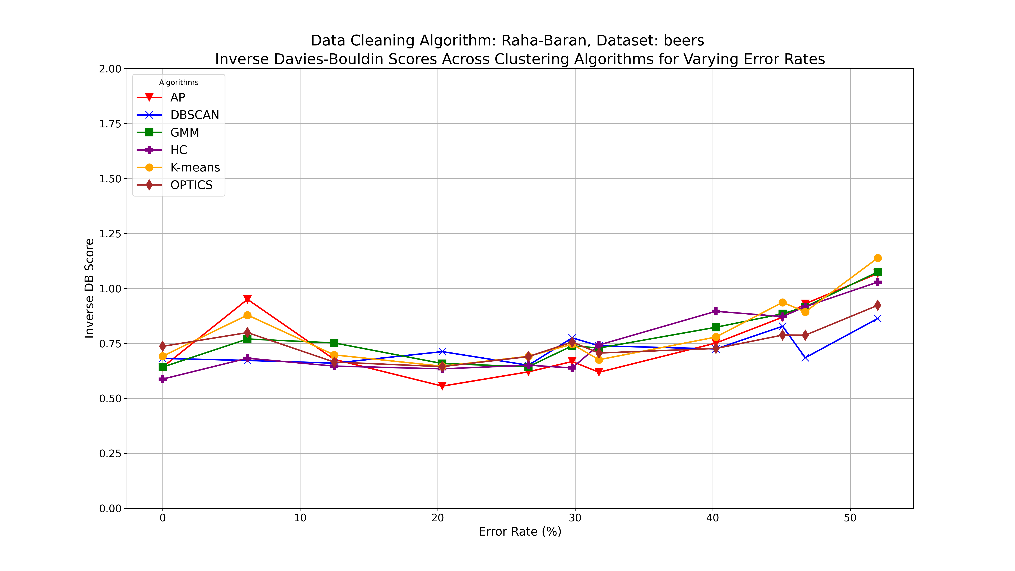
描述已自动生成

图2.3 使用Raha-Baran清洗系统后运行不同聚类算法得到的DB分数倒数在*beers*数据集上随错误率变化的图像

**3 实验结果的进一步分析与总结**

**3.1 实验结果呈现**

**图表, 折线图

描述已自动生成**首先整理所有数据集上运行不同聚类算法后的评估指标随错误率变化的图像如下：（原图见github链接，空间有限没有对图片进行编号，可以通过图片标题获知图片内容：<https://github.com/tianchanghrbcn/data_experiments/tree/master/results/final_results/dataset_algorithms_all%20cluster%20methods>）

**图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成**

**图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成**

**图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成3.2 聚类算法评估指标的综合分析**

3.2.1 Davies-Bouldin 分数分析：

(i) 总体趋势：

· 层次聚类 (HC) 和 K-means 在多个数据集上表现出较好的 DB 分数，尤其在错误率较高时，DB 分数有所下降，聚类质量提升。

* DBSCAN 和 OPTICS 在某些数据集上 DB 分数较高，聚类效果不如预期。

(ii) 数据集差异

* 在 *beers、flights* 和 *movies* 数据集上，随着错误率增加，HC 和 K-means 的 DB 分数普遍下降，聚类质量提升。

· *hospital* 数据集上，所有算法的 DB 分数都随着错误率增加而上升（变差），但 HC 增加幅度较小。

3.2.2 轮廓系数分析

(i) 总体趋势

· HC 和 K-means 在多数数据集和错误率下都取得了较高的轮廓系数，聚类效果较好。

· DBSCAN 和 OPTICS 的轮廓系数在某些数据集和错误率下为负值，表示聚类效果较差。

(ii) 数据集差异

· 在 *rayyan* 和 *restaurants* 数据集上，HC 的轮廓系数最高，表明其对文本和混合型数据有较好的适应性。

· *movies* 数据集上，K-means 在高错误率下轮廓系数有所提升，聚类效果改善。

**3.3 不同聚类算法在各数据集上的表现**

**Affinity Propagation (AP)**

· 对错误率较为敏感，聚类效果在高错误率下显著下降。

· 适用于中小规模数据集，如 *beers* 和 *rayyan*，在错误率较低时表现较好。

**DBSCAN**

· 对参数敏感，在高维度或噪声较多的数据集上效果不佳。

· 在 *beers* 和 *flights* 数据集上，随着错误率增加，聚类效果未明显提升。

**GMM**

· 对错误率的适应性一般，在某些数据集（如 *beers* 和 *restaurants*）上表现较好。

· 在 *hospital* 数据集上，随着错误率增加，聚类质量下降。

**HC**

· 整体表现最佳，对错误率有较强的鲁棒性。

· 在所有数据集上，HC 都能在高错误率下保持较好的聚类质量。

**K-means**

· 对错误率有一定的适应性，特别是在 *beers、flights* 和 *movies* 数据集上。

· 在高错误率下，K-means 的聚类质量有时反而提升，可能与数据结构的简化有关。

**OPTICS**

· 对参数和噪声较为敏感，聚类效果不稳定。

· 在多数数据集上，轮廓系数较低，聚类效果欠佳。

**3.4 数据集特征与算法性能的关系**[[6]](#footnote-6)

(i) 数据规模

· 小规模数据集（如 *rayyan*）上，AP 和 HC 表现较好。

· 大规模数据集（如 *restaurants*）上，HC 和 GMM 更具优势。

(ii) 数据类型

· 混合型数据（数值和分类）上，HC 和 K-means 有较好的适应性。

· 文本数据或高维稀疏数据上，HC 的表现优于其他算法。

(iii) 错误率影响

· 高错误率下，数据清洗后的数据结构可能简化，使得 HC 和 K-means 能更容易识别聚类结构[[7]](#footnote-7)。

· 一些算法（如 DBSCAN 和 OPTICS）在高错误率下对噪声敏感，聚类效果下降。

**4 实验中遇到的问题与思考**

**4.1 大数据聚类的性能问题。**聚类算法在大数据集上的运行速度较慢，特别是在DBSCAN和OPTICS等基于密度的算法中，由于它们需要计算大量的距离矩阵，性能瓶颈显著。

**4.2 小数据集聚类效果误差较大。**在使用聚类算法时，小数据集往往会表现出不同于大数据集的行为。具体来说，小数据集在部分算法中会导致聚类簇数的合适范围较窄，从而影响聚类效果，进而导致更高的误差。此外，个别样本对簇中心或聚类边界的影响更大，这种高方差会使得算法对数据的细微变化过于敏感。

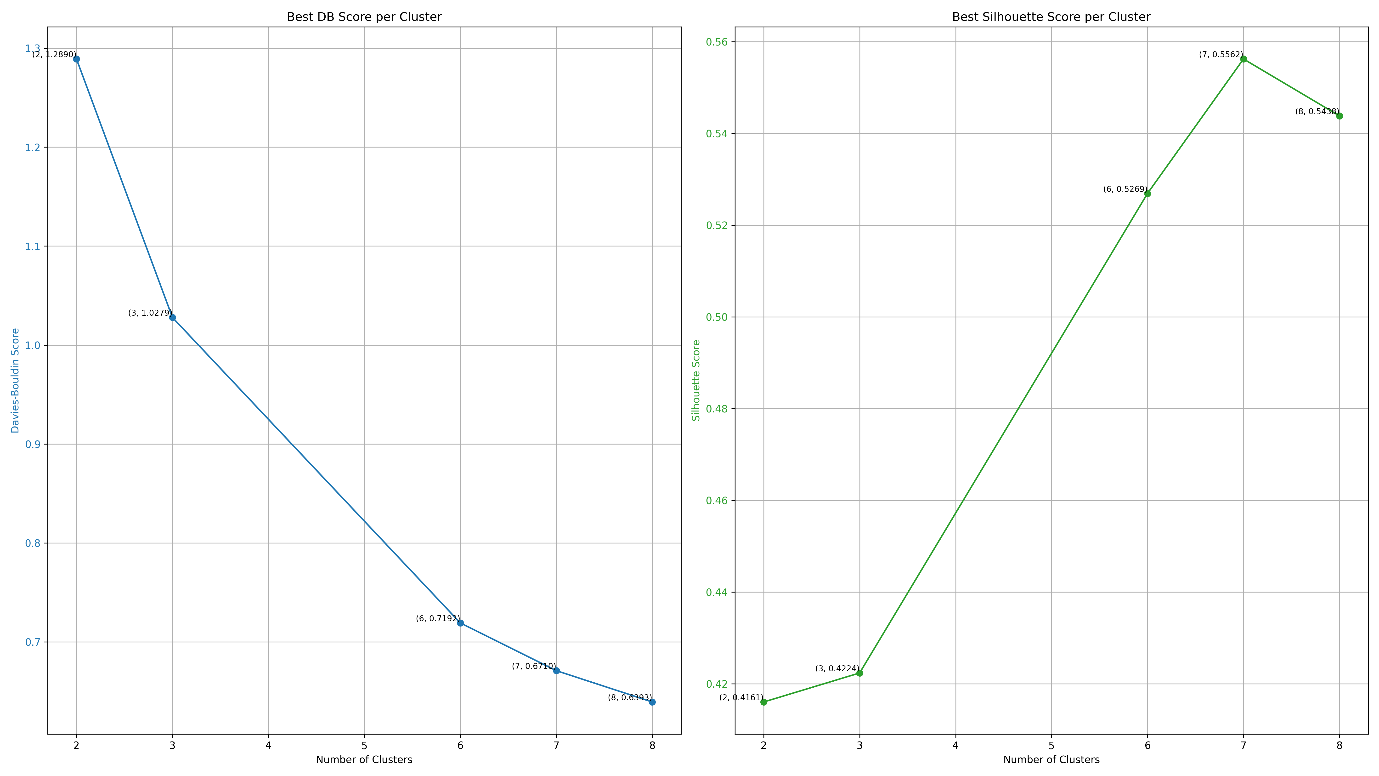


图4.1 *rayyan*数据集在AP算法下仅找到了5个有效簇

**4.3 合适参数的选择。**DBSCAN和OPTICS的参数对聚类结果有很大影响，不同的数据集需要手动调整参数，以获得最佳效果。某些聚类算法的参数分布是不均匀的，尚未找到最合适的参数搜索方案，所以仅采用循环遍历的方式，在大数据集规模下效率较低。

**4.4 评估指标的合理性。**在评估指标上，发现DBSCAN和OPTICS的轮廓系数有时为负值，影响了对聚类结果的判断[[8]](#footnote-8)。在这种情况下，考虑使用轮廓系数的绝对值进行可视化以保持图像的直观性，但也面临意义不明确、失去真实评估的风险。而且这些都是真实搜集到的实验结果，难以判断是数据集本身，清洗系统，聚类算法，参数选择、评价指标、簇数选择这些因素中哪一个不合适造成的影响。

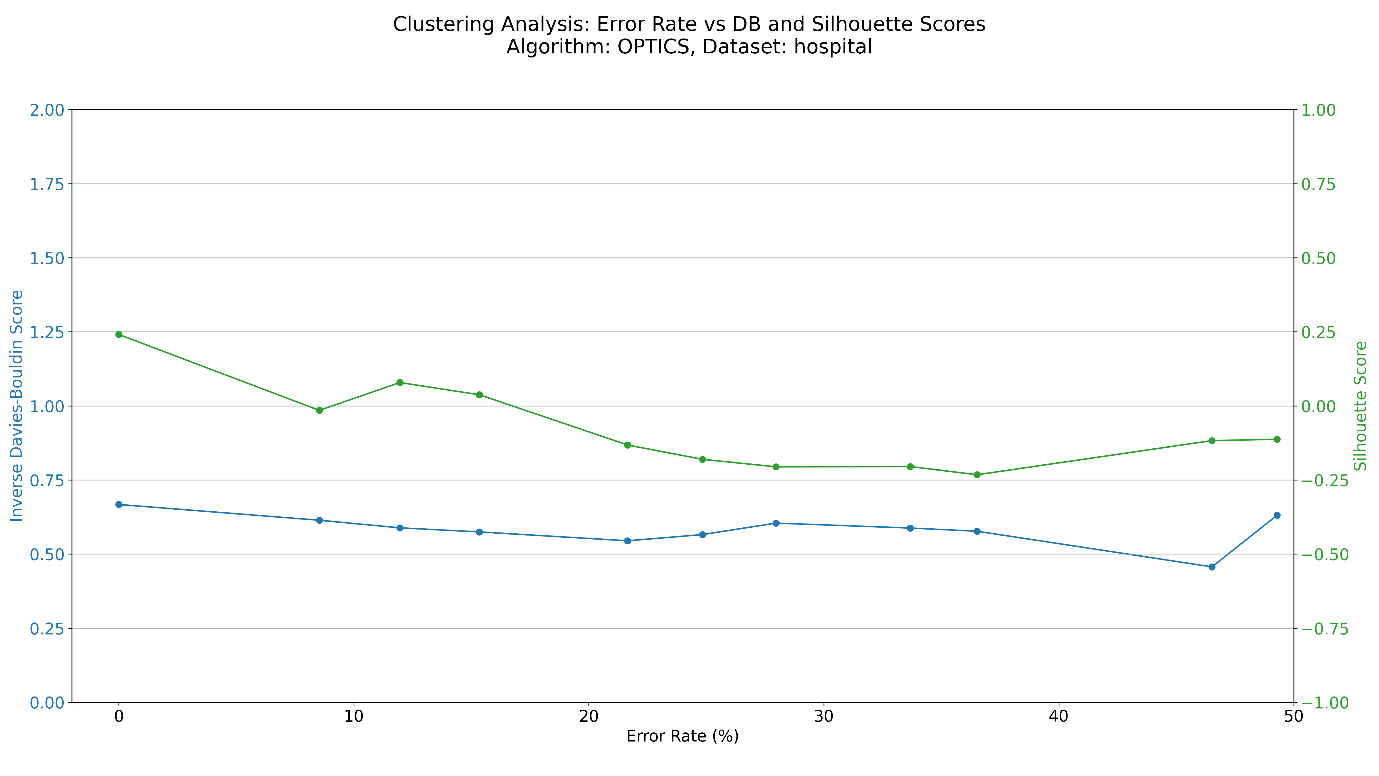


图4.2 *hospital*数据集进行清洗后运行OPTICS算法，得到的轮廓系数有部分为负数

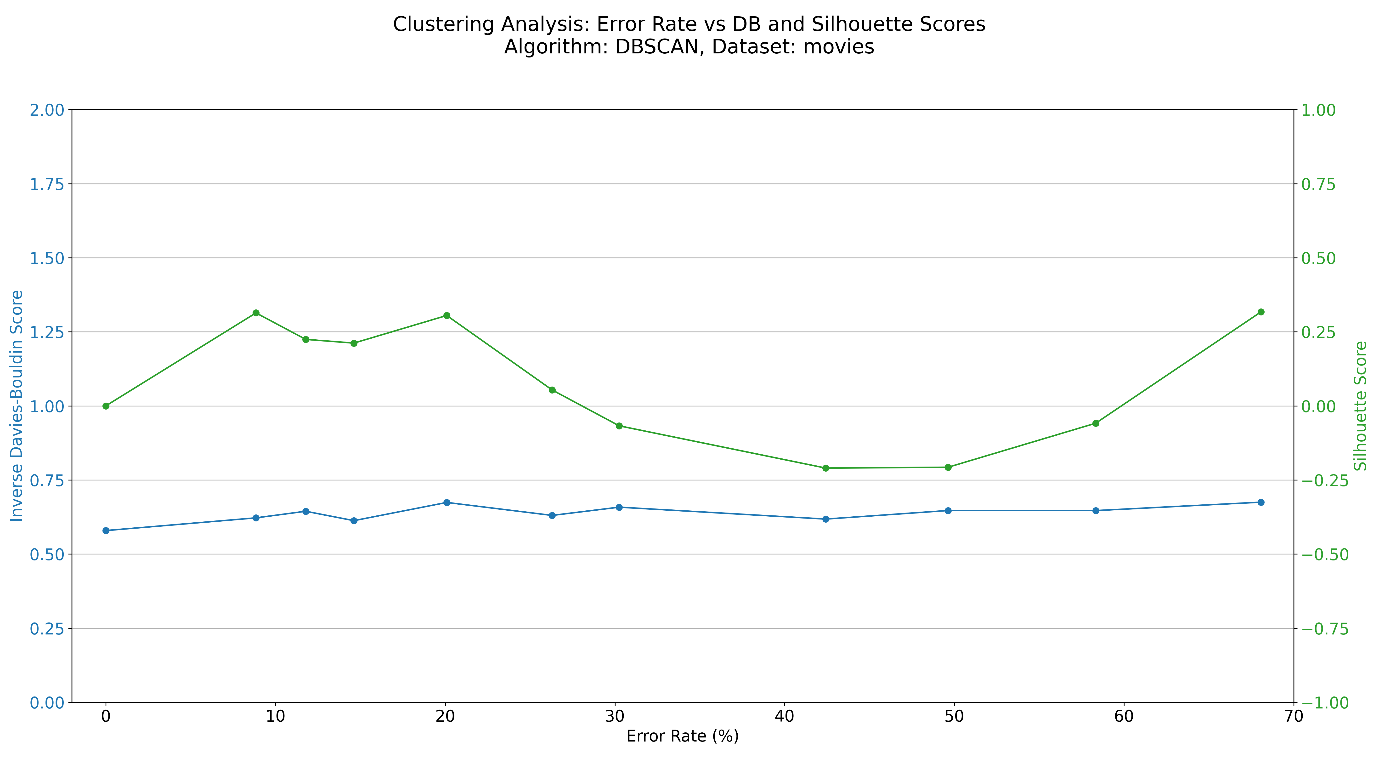


图4.3 *movies*数据集进行清洗后运行DBSCAN算法，得到的轮廓系数有部分为负数

**5 实验的可能调整和下一步的方向[[9]](#footnote-9)**

**5.1 多数据清洗算法的比较研究**[[10]](#footnote-10)

评估不同数据清洗算法对聚类性能的影响，找出在不同数据集和错误率下最有效的清洗方法。在现有的 Raha-Baran 算法基础上，加入其他主流的数据清洗算法，进行对比实验。不仅要关注清洗后的数据质量指标（如准确率、召回率），还要关注清洗对下游聚类任务的影响。

**5.2 针对不同规模数据集调整聚类策略**[[11]](#footnote-11)

针对大数据集上的性能问题，研究基于深度采样、近似算法或分布式计算的方法，提高聚类算法的可扩展性。对于小数据集，考虑使用聚类集成方法或数据增强技术，提升聚类结果的稳定性和可靠性。

**5.3 聚类算法参数的自动调优**[[12]](#footnote-12)

针对 DBSCAN 和 OPTICS，尝试改进算法或采用其变体，以提高在高维度和高噪声数据上的性能。利用自动机器学习（AutoML）技术，自动搜索最佳参数组合，例如使用贝叶斯优化、遗传算法等。

**5.4 数据清洗与聚类算法的协同优化（组合方案）**

设计一个联合优化框架，同时考虑数据清洗和聚类过程，可能通过迭代的方法逐步提升聚类效果。或者尝试将多种聚类算法结合，如先用 HC 进行初始聚类，再用 K-means 细化结果，寻找最佳组合以提高聚类性能。

**5.5 不同数据集特征对算法性能的影响（内部结构）**

深入分析数据集的特性（如维度、规模、噪声类型、数据类型），理解其对数据清洗和聚类算法性能的影响。对各数据集进行统计分析，了解其分布、相关性、稀疏性等特征。将数据集按照特征进行分类，分别研究数据清洗和聚类算法在不同类型数据集上的表现。

**5.6 深入研究评价指标的适用性（多指标、可视化分析）**

评价指标在不同情况下可能无法准确反映聚类效果，需要探讨更适合的指标或改进现有指标[[13]](#footnote-13)。同时，采用主成分分析（PCA）、t-SNE 等降维方法，降低数据维度，提高聚类算法的效率和效果。

**5.7\* 研究深度学习在数据清洗和聚类中的应用（结合最新模型）**

利用深度学习的方法，提升数据清洗和聚类的效果，尤其是在处理高维和非结构化数据时。探索使用深度神经网络进行数据错误检测和修复，例如基于自编码器的异常检测，或者尝试使用深度聚类模型，如深度嵌入聚类（DEC）、变分自编码器（VAE）等。

**5.8\* 开发一个综合性的数据清洗与聚类框架（可扩展性、可移植性）**

将数据清洗、参数优化、聚类、评价指标等步骤模块化，方便替换和扩展不同的算法，为不同特征的数据集提供合理的数据清洗和聚类的组合方案，同时设计一个可重复、可扩展的实验平台。最后将框架开源，促进社区交流和合作，获取更多的反馈和改进建议。

1. 包含4个小规模的数据集和2个中等规模的数据集，大数据集遇到的问题在本篇末尾有描述。 [↑](#footnote-ref-1)
2. 错误率是随机注入和选择的，并没有专门实现均等的错误率分布。 [↑](#footnote-ref-2)
3. 3 由于本次是初步实验，在聚类算法运行时参考了scikit-learn的模型，采用迭代的方式遍历聚类参数。 [↑](#footnote-ref-3)
4. 若在同一个簇数下出现检测指标的多个结果，则取效果最优的指标结果。 [↑](#footnote-ref-4)
5. 这里使用了Davies-Bouldin分数的倒数，使得数值越大结果越好 [↑](#footnote-ref-5)
6. 本部分为初步推测的结论，尚未进行定量或对比性研究。 [↑](#footnote-ref-6)
7. 这里需要进一步探讨高错误率数据经过修复后对聚类算法运行的影响。 [↑](#footnote-ref-7)
8. 当使用允许噪声的聚类算法计算轮廓系数时，噪声点可能会对结果产生很大的影响。因为这些点没有被分配到任何簇，它们在计算轮廓系数时与邻近的其他簇的距离会显得非常接近，从而导致负的轮廓系数。 [↑](#footnote-ref-8)
9. 最后两部分标注了星号，表示目前暂时有较大的完成距离。黄色标注是我认为需要首先实现的部分，旨在改进和优化实验结果。 [↑](#footnote-ref-9)
10. 此部分因为mlnclean算法没有修改完成所以暂时搁置了，会尽快补充，而其他清洗系统的效果不佳。 [↑](#footnote-ref-10)
11. 此部分需要结合其他领域的技术并参考相关文献，寻找大数据集和小数据集的对应解决方案。 [↑](#footnote-ref-11)
12. 此部分可以作为本研究的子问题，也需要查阅相关参考文献，目前已经找好，以改进基于密度聚类算法的处理质量。部分文献可获知各参数对聚类效果的影响程度以及参数与数据特征之间的关系模型。 [↑](#footnote-ref-12)
13. 如调整兰德指数（Adjusted Rand Index）、互信息（Mutual Information）、Calinski-Harabasz 指数等，提供更全面的聚类质量评估。 [↑](#footnote-ref-13)