華中科技大学

课 程 实 验 报 告

课程名称： 大数据分析

专业班级： **CS2202**

学　　号： **U202215378**

姓　　名： 冯瑞琦

指导教师： 崔金华

报告日期： **2024 年 6 月 14 日**

计算机科学与技术学院

目录

实验四 kmeans 算法及其实现

4.1 实验目的

1、加深对聚类算法的理解,进一步认识聚类算法的实现；
2、分析 kmeans 流程,探究聚类算法院里；
3、掌握 kmeans 算法核心要点；
4、将 kmeans 算法运用于实际，并掌握其度量好坏方式。

4.2 实验内容

提供动漫得分数据集（anime.csv）,包含用户对动漫评分(Score 2~Score 10)、动漫的欢迎程度(Popularity)等数据。

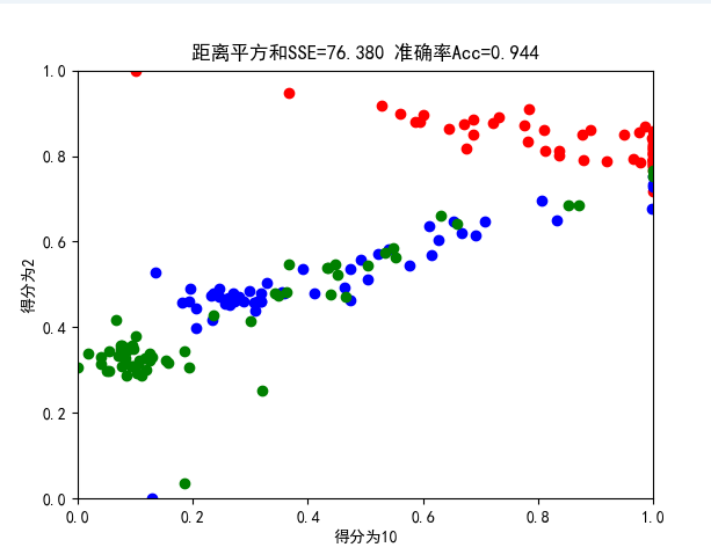
在对数据集进行处理时，按照 Popularity 列进行降序排序，在其中选择 K 类（eg. 选择 Popularity 高、中、低三类），每类选择一定数量的数据（eg. 每类选择 60 个数据），将选出的 K 类数据的 K 作为标签与 Popularity 和 Score2~Score10 组合成一个 11 维的数据，对除 K 以外的数据进行归一化处理。

编写 kmeans 算法，算法的输入是归一化后的数据集，动漫数据集一共 11 维数据，代表着动漫的 11 维特征，请在欧式距离下对动漫的所有数据进行聚类，聚类的数量为 K。

以处理后的 anime.csv 作为输入文件。

在本次实验中，最终评价 kmeans 算法的精准度有两种，第一是处理后的动漫数据集已经给出的 K 个聚类，和自己运行的 K 个聚类做准确度判断。第二个是计算所有数据点到各自质心距离的平方和。请各位同学在实验中计算出这两个值。

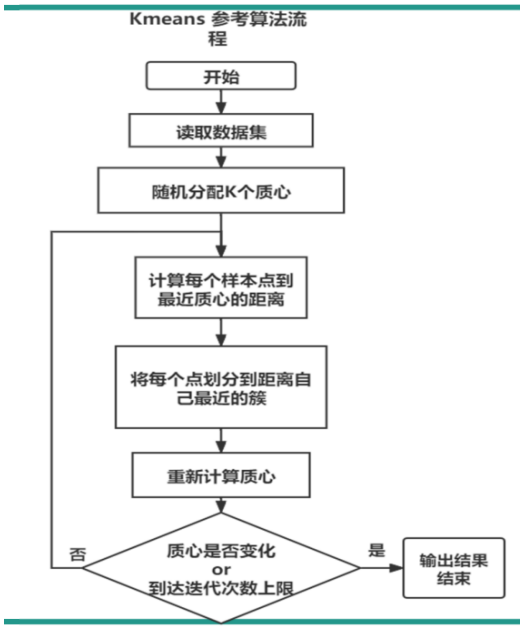
进阶任务：在聚类之后，任选两个维度（为了效果良好建议选择 Score 10 和 Score 2 列数据进行展示），以 K 种不同的颜色对自己聚类的结果进行标注，最终以二维平面中点图的形式来展示所有的样本点。效果展示图可如图所示。



4.3 实验过程

4.3.1 编程思路

聚类算法是一种无监督学习方法，用于将数据集中的数据点分成若干组（簇），使得同一簇内的数据点彼此相似，不同簇间的数据点差异显著。K-means 是常见的聚类算法之一，其基本步骤包括随机选择初始质心、迭代更新质心和分配数据点标签，直到质心不再变化或达到最大迭代次数。K-means 通过最小化簇内数据点到质心的距离平方和，实现数据的有效分组和模式识别。Kmeans 算法流程大致如下图所示。



具体编程思路如下：

1) 数据预处理

首先，读取动漫数据集并选择需要的特征列。将含有‘Unknown’的值转换为 NaN，并删除包含 NaN 值的行。数据按流行度（Popularity）排序并分为三类：高流行度、中流行度和低流行度，每类各 60 条记录。为每个类别添加标签，并将数据合并，归一化处理特征值。

```
# 将 'Unknown' 转换为 NaN，并删除包含 NaN 的行
for feature in features_columns:
    anime_data[feature] = pd.to_numeric(anime_data[feature], errors='coerce')
anime_data = anime_data.dropna(subset=features_columns)
```

2) 数据划分与归一化：

将数据按流行度（Popularity）降序排序并分为三类：高流行度、中流行度和低流行度，每类各 60 条记录。其中高流行度选择第 60 到 120 个数据，中流行度选择中间值到中间值+60 个数据，低流行度选择最后六十个数据。为每个类别添加标签并合并数据。使用 MinMaxScaler 对特征值进行归一化处理，使不同特征的数据范围归一化，从而提高算法的收敛速度和精度。

```
# 假设Popularity列的值已经离散化为高、中、低三类
anime_data = anime_data.sort_values(by='Popularity', ascending=False)
mid = anime_data.shape[0] // 2
high = anime_data.iloc[num_per_class: 2 * num_per_class].copy()  # 高流行度
middle = anime_data.iloc[mid: mid + num_per_class].copy()        # 中流行度
low = anime_data.tail(num_per_class).copy()                      # 低流行度
```

3) K-means 聚类

先初始化一个随机种子，以确保算法的可重复性。接着随机选择数据集中 n_clusters 个数据点作为初始质心。random_state.choice 函数从数据集中选择不重复的随机索引，并用这些索引来初始化质心。通过最大迭代次数 max_iter 进行循环迭代。在每次迭代中，计算每个数据点到所有质心的欧氏距离，并根据最近的质心分配标签。然后检查标签是否变化，如果没有变化则退出循环，表明算法已收敛。接下来，更新质心为当前簇中所有数据点的均值。重复上述过程直到达到最大迭代次数或质心不再变化。

调用 kmeans 函数对归一化后的数据进行聚类，返回质心和标签。使用 predict 函数对数据进行预测，分配每个数据点到最近的质心，并将结果存储在 anime_data_selected['Cluster']中。为了使聚类结果更具解释性，我们对每个簇内流行度（Popularity）的平均值进行排序。根据排序结果，将原始簇标签重新映射为新的标签，使得标签与流行度高低对应。具体做法是计算每个簇内流行度的平均值，并按降序排序，然后创建一个映射，将旧标签转换为

新标签，重新分配给数据点。

4) 准确率和 SSE

计算聚类结果的准确度和 SSE（误差平方和）距离。准确度是通过比较原始标签和重新分配的簇标签来计算的，表示聚类结果与预期标签的一致性。SSE 距离是每个数据点到其质心的距离平方和，用于衡量簇内数据点的紧密程度。通过这两个指标，我们可以评估聚类的效果和质量。

5) 结果可视化

对 K-means 聚类结果进行可视化。创建一个 10x6 大小的图表，并定义三种颜色（红、绿、蓝）分别表示三个簇。然后，使用 plt.scatter 函数绘制二维散点图，横轴为 Score-2，纵轴为 Score-10，根据簇标签将数据点分配不同颜色。设置图表标题，包含聚类准确度和 SSE 距离，添加横轴和纵轴标签，并显示图例和图表，直观展示聚类结果的分布情况。

4.3.2 遇到的问题及解决方式

1) NaN 值处理

问题描述：在最初的数据预处理中，没有处理包含 NaN 值的行，导致在后续的归一化和 K-means 算法中出现错误。

解决方式：通过将‘Unknown’值转换为 NaN 并删除包含 NaN 值的行，确保数据的完整性和准确性。

2) 流行度数据选择

问题描述：在选择高、中、低流行度数据时，发现高流行度选择降序排列的前六十个时结果准确率很低。

解决方式：高流行度改为选择第 60-120 个数据，经过这一调整发现结果准确率大幅提升。推测原因可能是前六十个数据不够具有代表性，和裁判打分时去掉最高分一样，选择相对前面，但不是最头部的数据更能够保证结果的准确性。

3) 质心选择

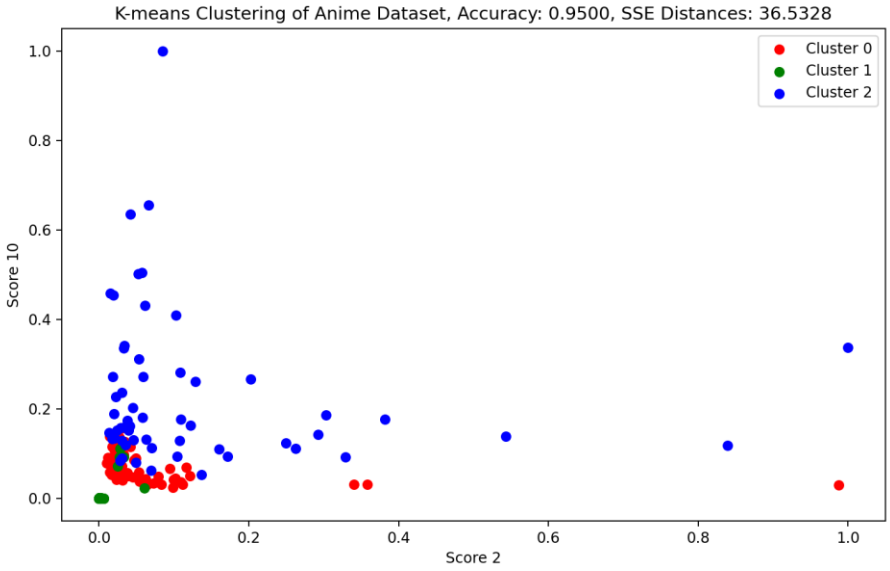
问题描述：在实现 K-means 聚类时，最初仅考虑了随机选择初始质心并迭代更新质心。然而，发现初始质心的选择对最终结果影响较大，有时会陷入局部最优，导致聚类效果不佳。

解决方式：引入多次运行 K-means 算法的机制，并选取最优结果。具体操作是通过多次运行 K-means 算法，记录每次的 SSE 距离，选取 SSE 距离最小的结果作为最终聚类结果。这种方法有效地避免了因初始质心选择不当而导致的局部最优问题，显著提高了聚类的稳定性和准确性。

4.3.3 实验测试与结果分析

实验测试结果如图所示，通过 K-means 聚类算法对动漫数据集进行了聚类分析，达到了 95%的准确率，SSE 距离为 36.5328。这表明算法在将不同流行度的动漫分成三类时效果较好。图中不同颜色的点表示不同的簇，每个簇包含相似特征的动漫。高准确度说明大部分数据点被正确分类，较低的 SSE 距离表明簇内数据点与质心的距离较小，聚类效果较好。

```
hon.exe d:/大二下/大数据实验/task4/kmeans.py
Accuracy: 0.95
SSE Distances: 36.532828739217344
```



4.4 实验总结

通过本次实验，我深入理解了 K-means 聚类算法及其在实际数据集上的应用。在数据预处理阶段，处理包含 NaN 值的行确保了数据的完整性和准确性。在实现 K-means 算法过程中，初始质心的选择对最终结果影响较大，通过多次运行 K-means 并选取最优结果的方法，显著提高了算法的稳定性和准确性。实验结果表明，合理的数据预处理和算法优化对聚类效果至关重要。尤其是在处理大规模数据时，数据的清洗和规范化是成功的关键因素。

此外，通过可视化展示聚类结果，使得我们可以直观地观察到不同簇的分布和聚类效果。这不仅有助于验证算法的正确性，还提供了有价值的洞察，帮助我们更好地理解数据的结构和特征。

本次实验不仅增强了我对聚类算法的理解，也提高了我处理实际数据问题的能力。通过实际编码和调试，我认识到算法优化和数据结构选择在大规模数据处理中的关键作用。此次实验为我未来深入研究和应用机器学习技术打下了坚实的基础，并激发了我进一步探索数据挖掘和机器学习领域的兴趣。