

**课 程 实 验 报 告**

**课程名称： 大数据分析大作业**

**专业班级： 大数据2101班**

**学 号： U202115578**

**姓 名： 罗理恒**

**指导教师： 王蔚**

**报告日期： 2023年5月13日**

**计算机科学与技术学院**

**目录**

[大作业 推荐系统实验 5](#_Toc1271859757)

[1.1实验目的 5](#_Toc94713661)

[1.2 实验内容 5](#_Toc567118000)

[1.3 实验过程 7](#_Toc1019800614)

[1.3.1 系统整体框架 7](#_Toc721932791)

[1.3.3 实验测试 14](#_Toc241812787)

[1.3.4 对比实验及分析 16](#_Toc1108450985)

[1.4 实验总结 18](#_Toc315067170)

[1.4.1 实验中遇到的问题及解决 18](#_Toc1786736335)

[1.4.2 实验总结 18](#_Toc1413746344)

[大作业推荐系统实验 参考文献 19](#_Toc1075733200)

# 

# 大作业 推荐系统实验

## **1.1实验目的**

1. 了解推荐系统的多种推荐算法并理解其原理。
2. 实现**User-User的协同过滤算法**并对用户进行推荐。
3. 实现**基于内容的推荐算法**并对用户进行推荐。
4. 对两个算法进行电影预测评分对比
5. 在学有余力的情况下，**加入minihash算法**对效用矩阵**进行降维处理**

## **1.2 实验内容**

给定MovieLens数据集，包含电影评分，电影标签等文件，其中电影评分文件分为训练集train\_set和测试集test\_set两部分

基础版必做一：**基于用户的协同过滤推荐算法**

对训练集中的评分数据构造用户-电影效用矩阵，使用**pearson相似度计算方法计算用户之间的相似度**，也即相似度矩阵。**对单个用户进行推荐时，找到与其最相似的k个用户，用这k个用户的评分情况对当前用户的所有未评分电影进行评分预测，选取评分最高的n个电影进行推荐。预测评分按照以下方式计算：**

在测试集中包含100条用户-电影评分记录，用于计算推荐算法中预测评分的准确性，对测试集中的每个用户-电影需要**计算其预测评分**，**再和真实评分进行对比，误差计算使用SSE误差平方和**。

选做部分提示：此算法的进阶版采用minihash算法对效用矩阵进行降维处理，从而得到相似度矩阵，注意minihash采用jarcard方法计算相似度，需要对效用矩阵进行01处理，也即**将0.5-2.5的评分置为0，3.0-5.0的评分置为1**。

基础版必做二：**基于内容的推荐算法**

将数据集movies.csv中的电影类别作为特征值，**计算这些特征值的tf-idf值**，得到关于电影与特征值的n（电影个数）\*m（特征值个数）的**tf-idf特征矩阵**。根据得到的tf-idf特征矩阵，**用余弦相似度的计算方法，得到电影之间的相似度矩阵**。

对某个用户-电影进行预测评分时，获取当前用户的已经完成的所有电影的打分，通过电影相似度矩阵获得已打分电影与当前预测电影的相似度，按照下列方式进行打分计算：

**选取相似度大于零的值进行计算，如果已打分电影与当前预测用户-电影相似度大于零，加入计算集合，否则丢弃**。（相似度为负数的，强制设置为0，表示无相关）假设计算集合中一共有n个电影，score为我们预测的计算结果，score’(i)为计算集合中第i个电影的分数，sim(i)为第i个电影与当前用户-电影的相似度。如果n为零，则score为该用户所有已打分电影的平均值。

**要求能够对指定的userID用户进行电影推荐，推荐电影为预测评分排名前k的电影。userID与k值可以根据需求做更改。**

推荐算法准确值的判断：**对给出的测试集中对应的用户-电影进行预测评分，输出每一条预测评分，并与真实评分进行对比，误差计算使用SSE误差平方和**。

选做部分提示：进阶版采用minihash算法对特征矩阵进行降维处理，从而得到相似度矩阵，注意minihash采用jarcard方法计算相似度，特征矩阵应为01矩阵。因此进阶版的特征矩阵选取采用方式为，**如果该电影存在某特征值，则特征值为1，不存在则为0，从而得到01特征矩阵**。

**进阶部分：**

本次大作业的进阶部分是在基础版本完成的基础上大家可以尝试做的部分。进阶部分的主要内容是使用**迷你哈希（MiniHash）算法对协同过滤算法和基于内容推荐算法的相似度计算进行降维**。同学可以把迷你哈希的模块作为一种近似度的计算方式。

协同过滤算法和基于内容推荐算法都会涉及到相似度的计算，迷你哈希算法在牺牲一定准确度的情况下对相似度进行计算，其能够有效的降低维数，尤其是对大规模稀疏01矩阵。同学们可以**使用哈希函数或者随机数映射来计算哈希签名**。哈希签名可以计算物品之间的相似度。

最终降维后的维数等于我们定义映射函数的数量，我们设置的映射函数越少，整体计算量就越少，但是准确率就越低。**大家可以分析不同映射函数数量下，最终结果的准确率有什么差别**。

**对基于用户的协同过滤推荐算法和基于内容的推荐算法进行推荐效果对比和分析，选做的完成后再进行一次对比分析。**

## **1.3 实验****过程**

## **1.3.1 系统整体框架**

推荐算法（Recommend System）是一种广泛应用的算法，本次实验聚焦两个推荐算法，分别是基于用户的协同过滤推荐算法（User-based Collaborative Filtering）和基于内容的推荐算法（Content-Based Filtering）。两种算法的示意图如图1-1所示。[1]

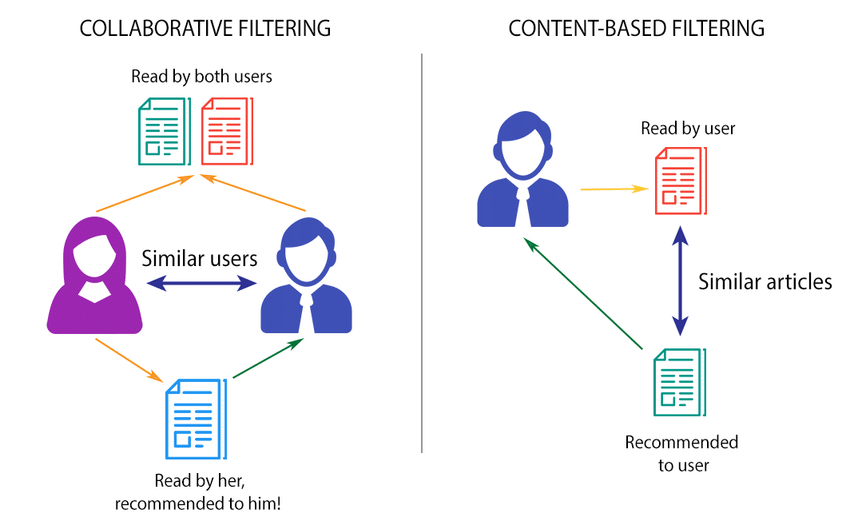


图1-1 两种推荐算法示意图

在原理上，基于用户的协同过滤推荐算法（User-based Collaborative Filtering）基于用户之间的相似性，假设有相似喜好的用户在未来会有相似的行动；基于内容的推荐算法（Content-Based Filtering）基于用户喜好的物品特征，假设用户喜欢与其过去喜欢的物品相似的物品。[2]

**1.3.2 编程思路**

**1.3.2.1基于用户的协同过滤推荐算法（User-based Collaborative Filtering）**

1. **算法描述**

首先我简单描述算法的运行流程，算法伪代码如图1-2所示：

1. 根据数据集中用户对电影的评分生成效用矩阵。
2. 计算用户之间的相关度，找到与目标用户k个最相似的其他用户。
3. 根据相似用户的评分对目标用户未评分对电影进行评分。
4. 根据评分向目标用户推荐电影。

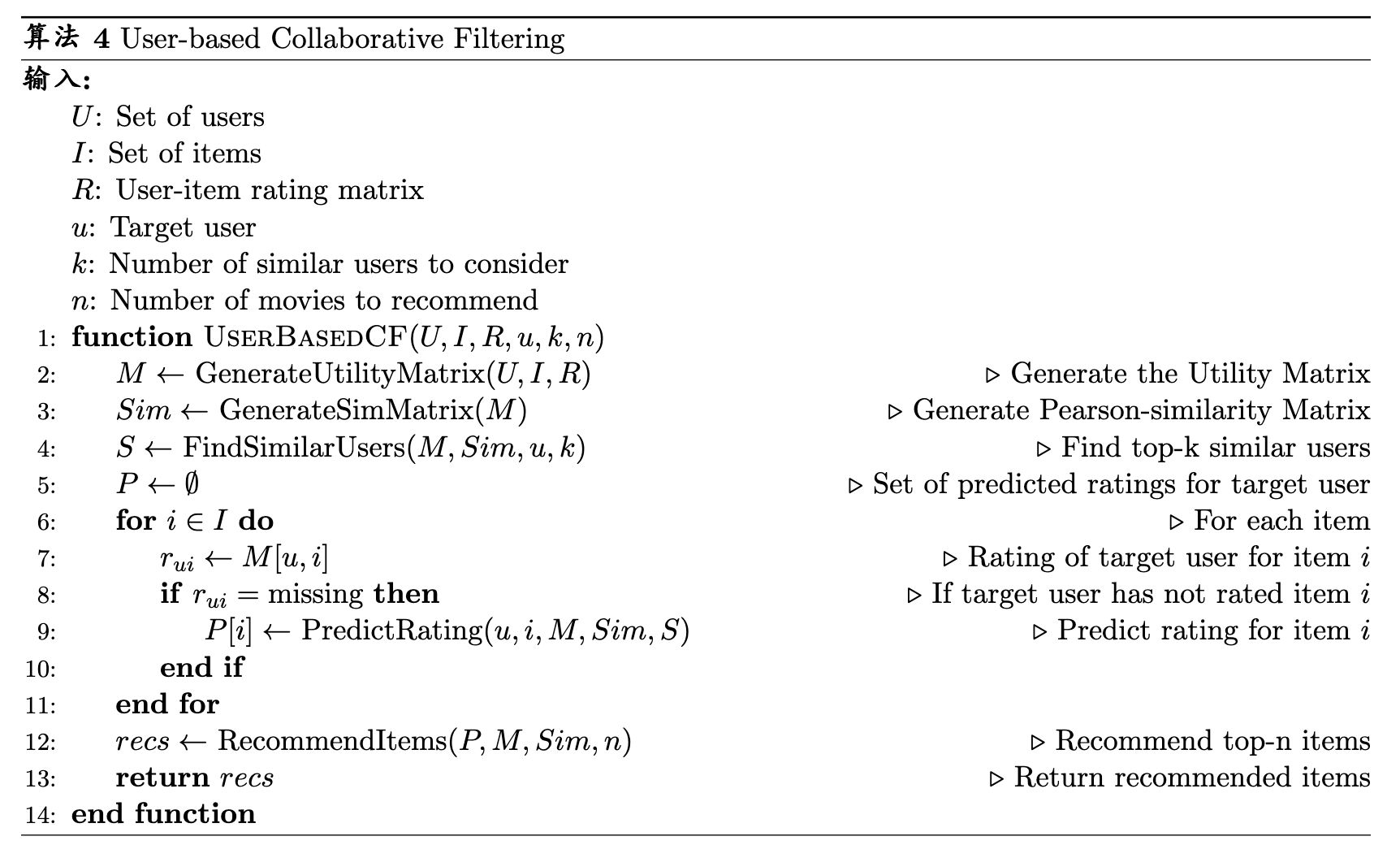


图1-2 基于用户的协同过滤推荐算法伪代码

1. **用户-电影效用矩阵构建**

矩阵的构建包含对数据集的处理，train\_set.csv数据集的格式如图1-3所示，其中第一列代表用户id，第二列代表电影id，第三行是用户对电影的评分。构建矩阵时按行遍历整个文件，构建train\_data[user\_id][movie\_id] = rating的矩阵，最后将空值赋值为0即可。

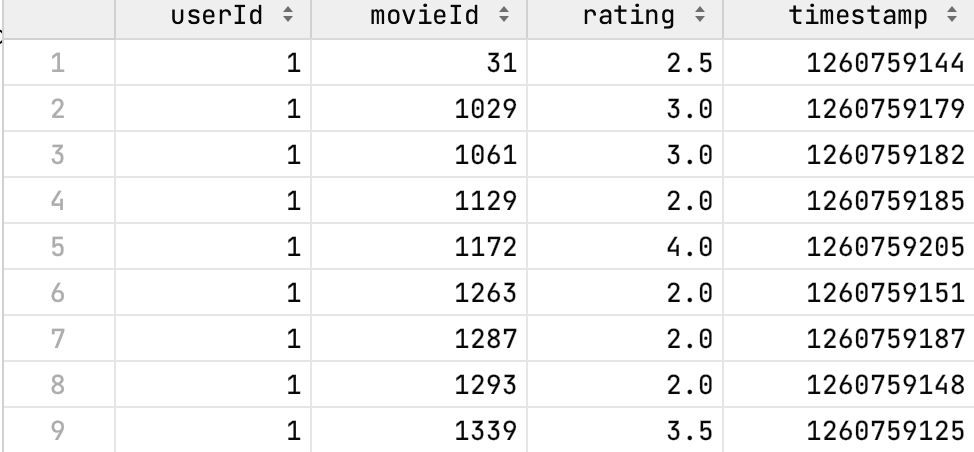


图1-3 数据集截图



图1-4 构建效用矩阵代码

1. **用户相似度矩阵构建，寻找k个相似用户**

用户相似度的计算采用皮尔森相似度（Pearson Similarity），在实现上只需调用封装好的corr函数即可，此处不多赘述。

相似度计算完成后，将目标用户与其他用户对相似度进行排序，排序前k个用户设为相似用户，返回相似用户对电影对所有评分。代码如1-5所示：



图1-5 寻找k个相似用户代码

1. **预测目标用户对某部电影的评分**

预测评分的计算方式采用评分与相似度加权平均，这样做是基于相似度越高的用户对某部电影的喜好越相似的假设，该用户评分权重也越大。计算公式如下所示：

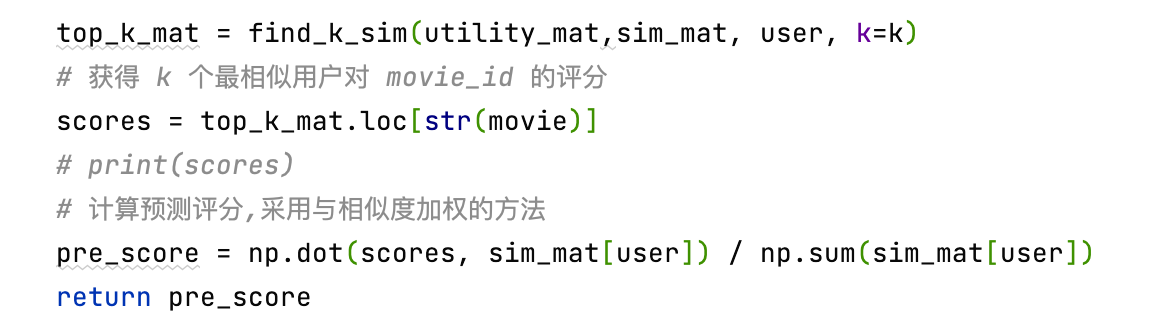


图1-6 预测评分代码

**5）向目标用户推荐电影**

推荐电影的过程中，首先遍历目标用户所有未评分的电影，通过加权平均的方式预测其评分。然后将预测评分进行排序，排序前n的电影即为推荐的电影，返回这些电影id即可。特别地，我规定至少超过30个人对此电影打过分才认为该电影有预测对价值。



图1-6 向目标用户推荐电影代码

**6）引入Minihash进行降维**

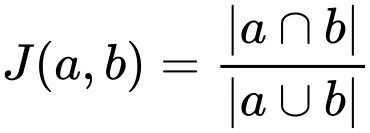
由于用户-电影效用矩阵维度较大（9066\*671），可以使用minihash对其进行降维。这样的操作一方面可以减少生成相似度矩阵的时间，同时可以处理大规模的稀疏数据，符合本实验的数据集特征。[3]

1. 生成签名矩阵（Signature Matrix）

首先对效用矩阵进行处理，如果评分为0 - 2.5分之间，则将其置为0，否则将其置为1。在实现上我调用了datasketch库中的MiniHash函数，在01处理完成后重新遍历行索引，选择值为1的更新minihash，将其作为特征值。由此生成了签名矩阵，其维度由hash函数的数量决定。

1. 生成相似度矩阵

对签名矩阵的相似度计算采用Jaccard相似度，其计算公式如下展示，在实现上调用datasketch库中的Jaccard函数，能够大幅提升运算效率。



生成的相似度矩阵维度为671\*671，同不使用minihash的推荐算法相同，故预测评分（predict），推荐电影（recommend\_movie）等函数可以直接复用。



图1-7 minihash代码展示

**1.3.2.2基于内容的推荐算法（Content-Based Filtering）**

1. **算法描述**

首先下面我简单描述算法的运行流程，算法伪代码如图1-8所示：[4]

A）构建电影-类别Tf-idf特征矩阵

B）计算不同电影之间的相似度，构建相似度矩阵。

C）根据该用户对其他电影的评分预测对目标电影的评分。

D）根据预测评分向某用户推荐电影。

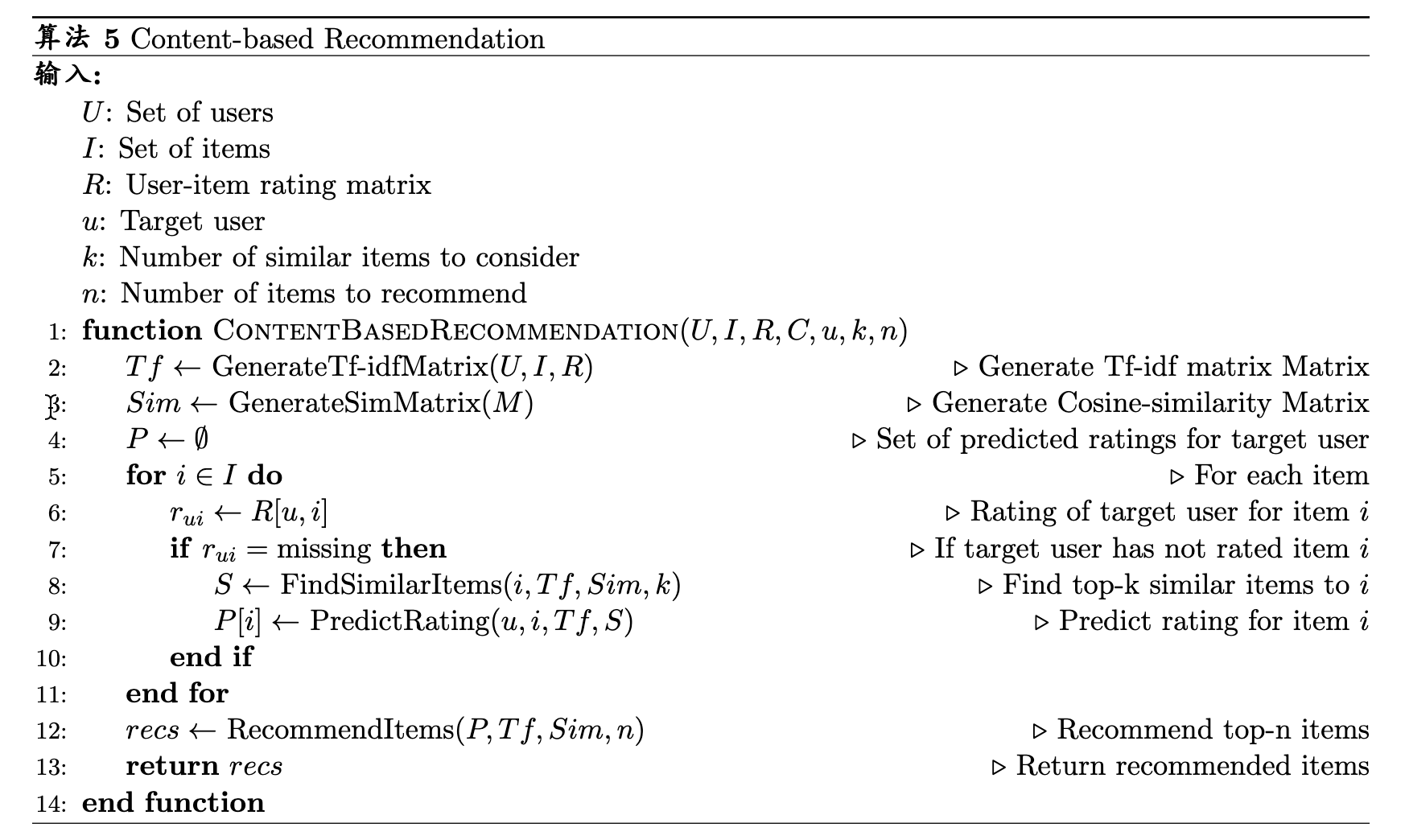
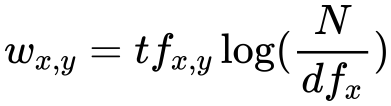


图1-8 基于内容的推荐算法伪代码

1. **电影-类别特征矩阵与相似度矩阵构建**

TF-IDF（Term Frequency-Inverse Document Frequency）是常用的特征表示方法，其计算公式如下展示。本实验中构建tf-idf特征矩阵，通过电影的类别特征来判断其相似度。相似度的计算采用余弦相似度（Cosine Similarity）。



在实现中，数据集的维度为（9125\*9125），如果使用公式进行计算，耗时将远超预期，所以在搜索资料之后我使用了sk-learn库中的函数，大大提升了运算效率，代码如图1-9所示。

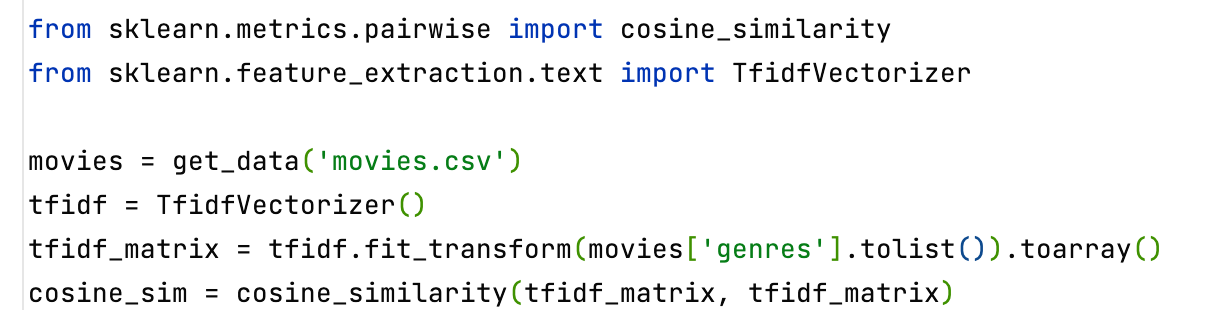


图1-9 特征矩阵构建代码

1. **预测某位用户目标电影的评分**

在预测评分时，首先寻找该用户已完成评分的电影，获得目标电影同已评分电影的相似度，如果是相似电影（相似度 > 0）就记录下来，最后通过相似度与评分的加权平均计算得到预测的分数。预测分数计算公式如下：

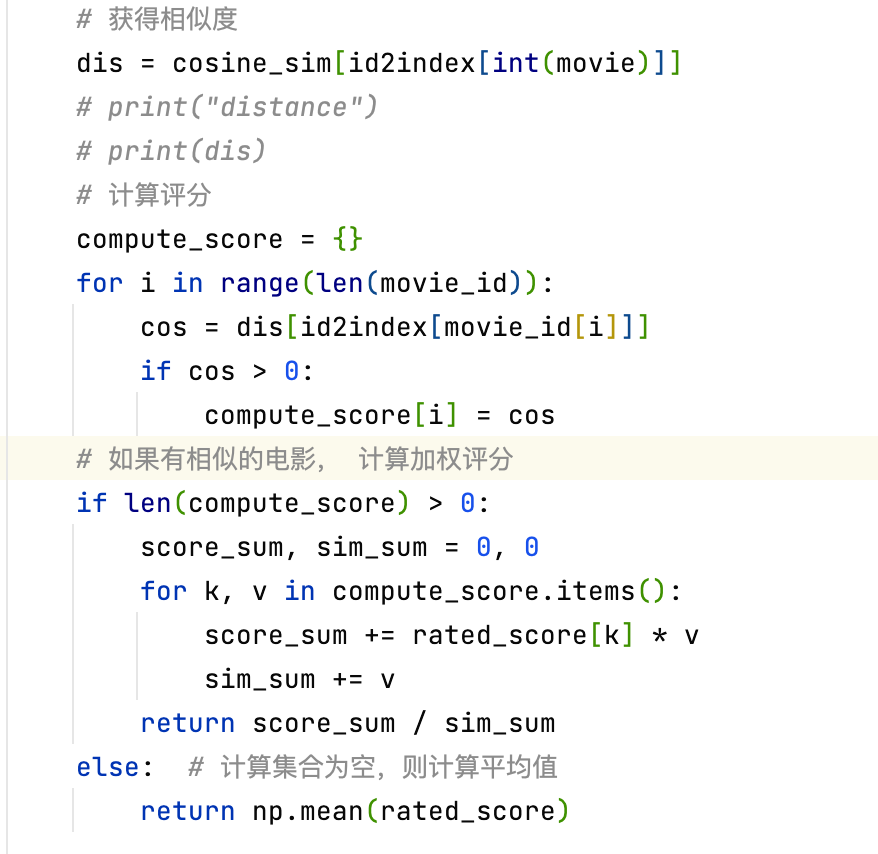


图1-10 预测目标电影评分代码

1. **向目标用户推荐电影**

该部分同基于用户的推荐算法类似，即对目标用户所有未评分的电影进行预测，推荐预测评分前n的电影，此处不再赘述。

1. **引入Minihash进行降维**

基于内容的推荐算法同样可以应用minihash进行降维。和基于用户的推荐算法相比，minihash的对象变成了电影-类别的特征矩阵，降维的方式和实现与基于用户的推荐算法类似，相似度同样采用Jaccard相似度进行计算。

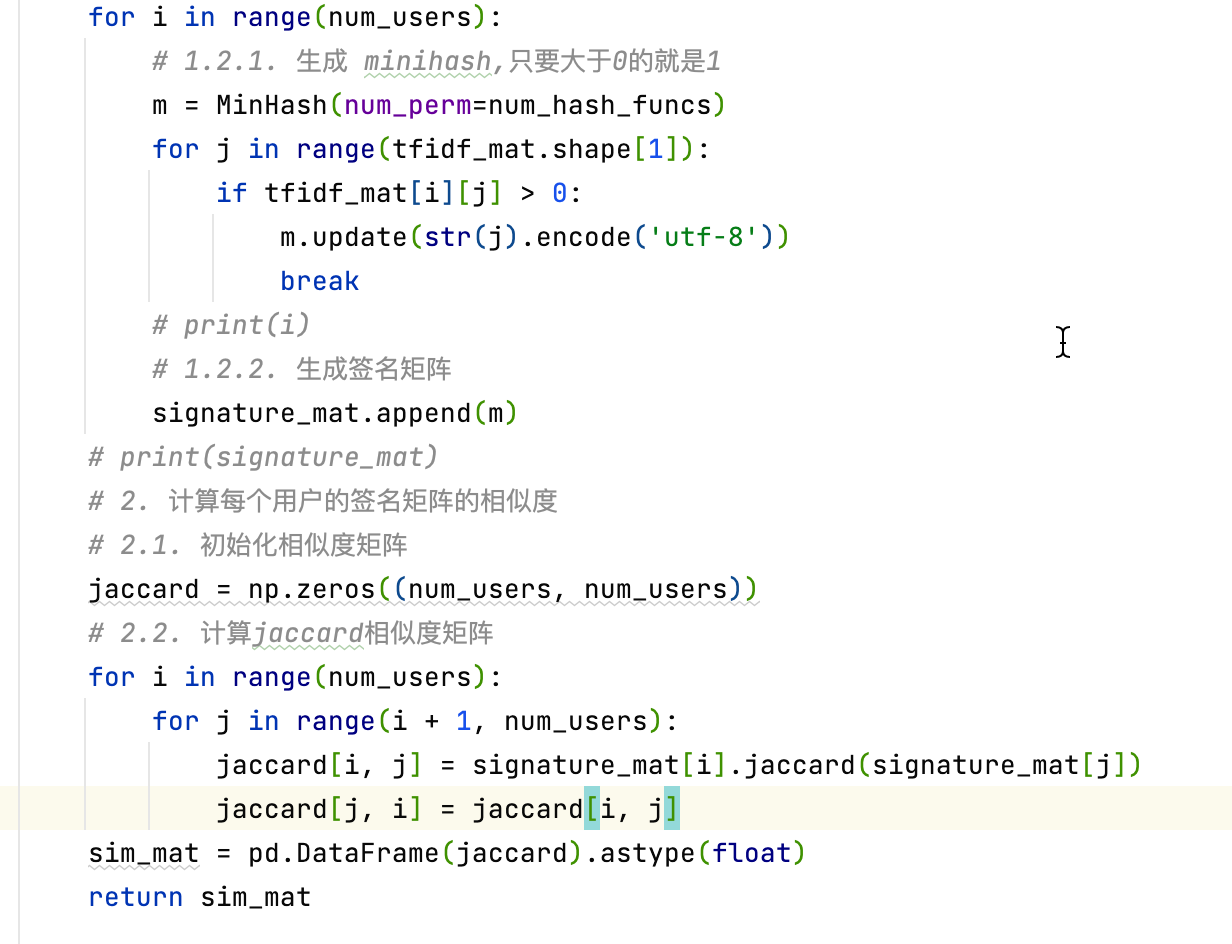
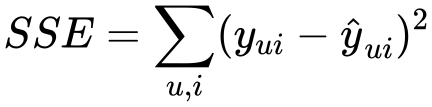


图1- 11 Minihash代码

1. **性能评估**

为了测试算法的精确度，引入评估指标sse（误差平方和），其越小代表预测评分和实际评分差异越小，推荐算法的预测效果越好。sse的计算公式如下，代码如图1-12 所示。



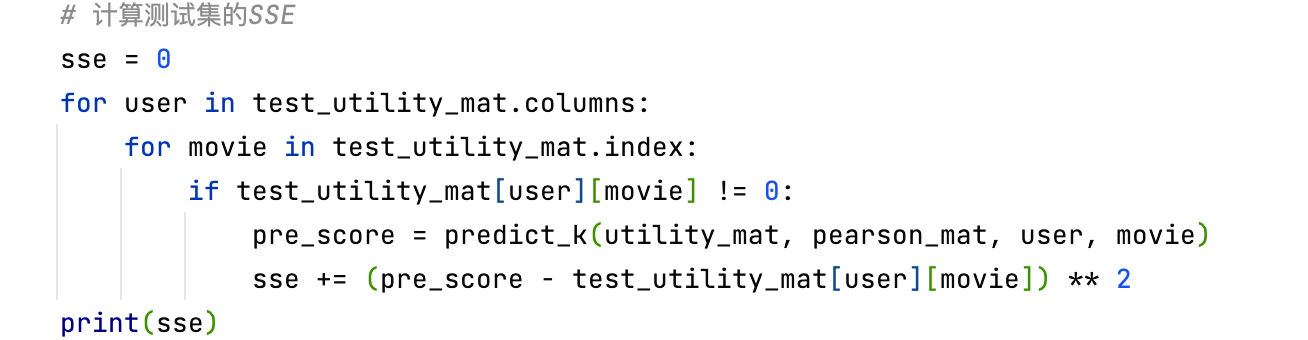


图1-12 计算SSE代码

## **1.3.3 实验测试**

**1）实验环境**

本实验使用python进行开发，版本为3.11.3。使用的库为numpy，os，time，pandas，datasketch，sk-learn，matplotlib等。

实验的目录设置和基本参数如图1-13所示：相似用户设置为100个，推荐电影数设置为10，hash函数设置为100个。

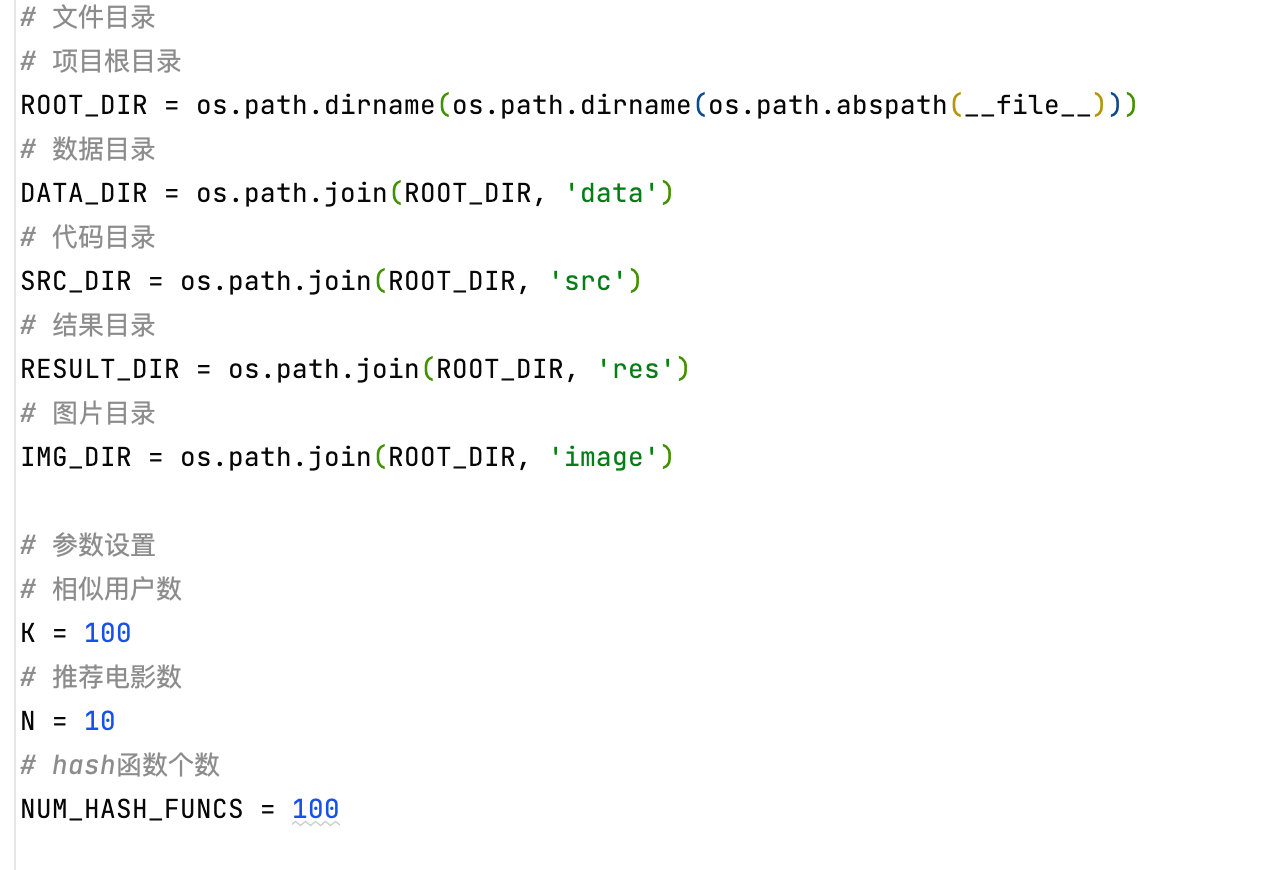


图1-13 基本参数图

1. **基于用户的协同过滤推荐算法（User-based Collaborative Filtering）测试**
2. 基本算法测试

首先确定该算法实现上的正确性，下面是基于用户的推荐算法对用户id为12的用户推荐的10部电影的id，如图1-14所示，其中第一列为用户id，第二列为电影id，第三列为评分

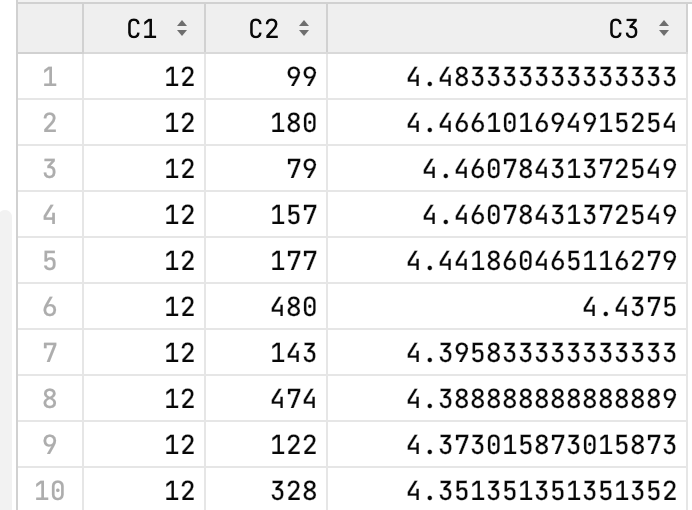


图1-14 基于用户的推荐功能测试，用户id为12

1. 引入minhash测试

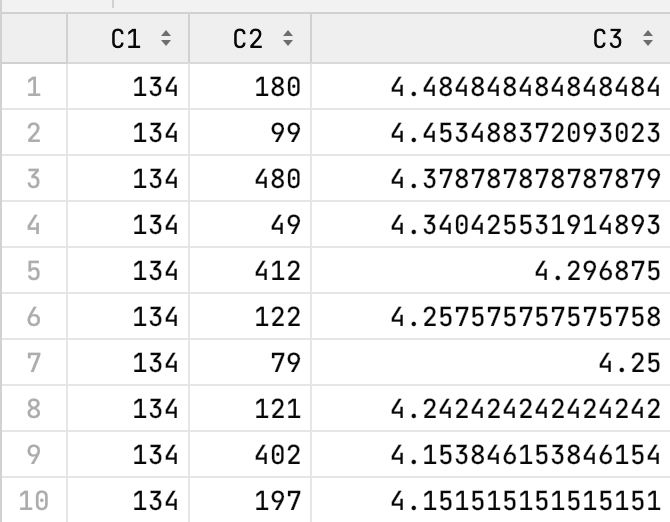


图1-15 minihash推荐功能测试，用户id为134

1. **基于内容的推荐算法（Content-Based Filtering）测试**
2. 基本算法测试

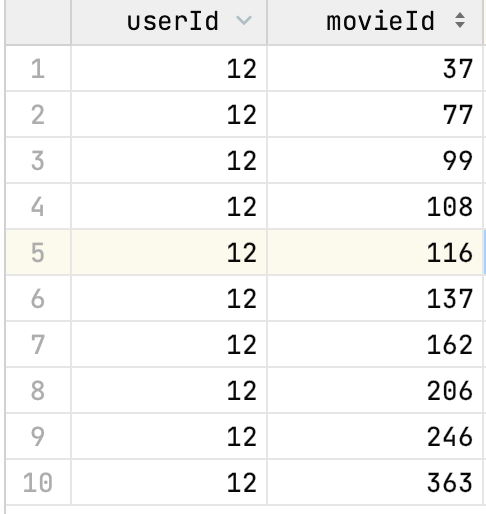


图1-16 基于内容的推荐功能测试，用户id为12

B）引入minhash测试

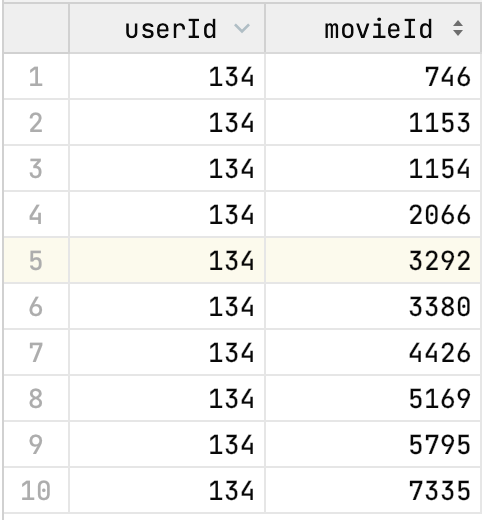


图1-17 minihash推荐功能测试，用户id为134

无论是基于用户还是基于内容的，推荐算法都能够完成推荐，从结果来看，由于计算评分的方式不同，最后的推荐结果也不相同。

1. 算法性能评估

对于上述四种情况分别计算sse，结果如图1-18所示。

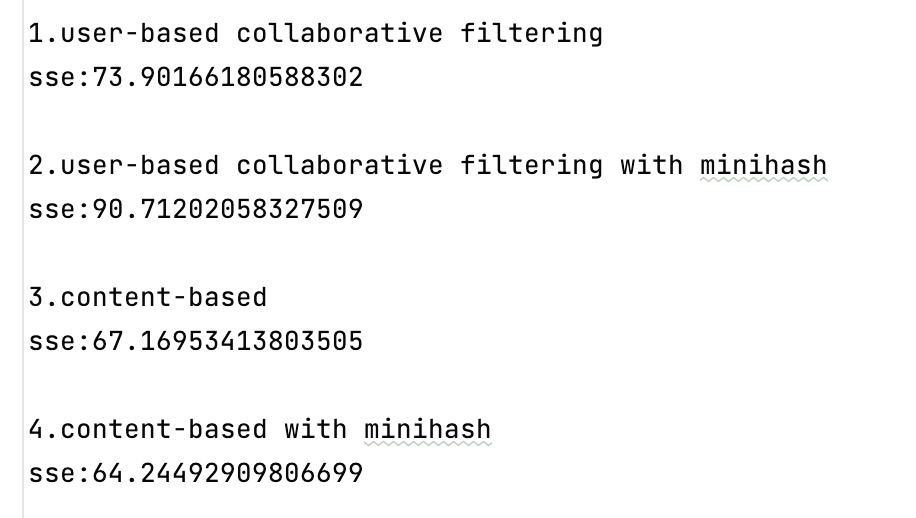


图1-18 不同算法sse值

在实验过程中，加入minihash算法的耗时相对更长，这可能是由于数据集本身并不大，降维虽然可以减少在生成相似度矩阵时的时间，但是多耗费了生成签名矩阵的时间，总体来说耗时增加。

在基于用户的推荐算法中，我发现加入minihash之后算法的sse值增加，这意味着预测效果并不好，在查阅资料后我认为这可能跟hash函数的数量有关（此处的hash函数数量为10），适当提升hash函数的数量可以提升预测结果，后续的对比实验也印证了我的观点，最小的sse达到了70左右。

在基于内容的推荐算法中，sse的值降低了，我认为这是minihash对于大规模稀疏数据的处理能力更好，通过minihash计算出来的相似度矩阵更能衡量电影之间的相似度，从而达到了更好的预测效果。

## **1.3.4 对比实验及分析**

为了更好的探究模型的细节，我做了两组对比实验，第一组实验是在基于用户的推荐算法实验中，对相近用户数k进行更替[5]；第二组实验是在基于内容的推荐算法中，选用不同的hash函数个数。

1. **选用不同的k（k代表相似用户的数量）**

为了测试不同k值下算法预测性能的好坏，我在[50,300]区间内以10为步长选择k值，计算算法的sse值。实验结果如图1-19所示。

从实验结果来看，k值的选取在[100,150]之间比较合适。如果k值太小，会导致算法仅考虑了少数相似用户，并不能完全代表用户喜好，所以预测结果较差。如果k值太大，不仅会带来更大的计算复杂度，而且会导致算法局限于用户的特定偏好而忽略一些潜在的偏好。事实上，k值的选取是一个相对的概念，应该根据数据集的规模和特征灵活确定。

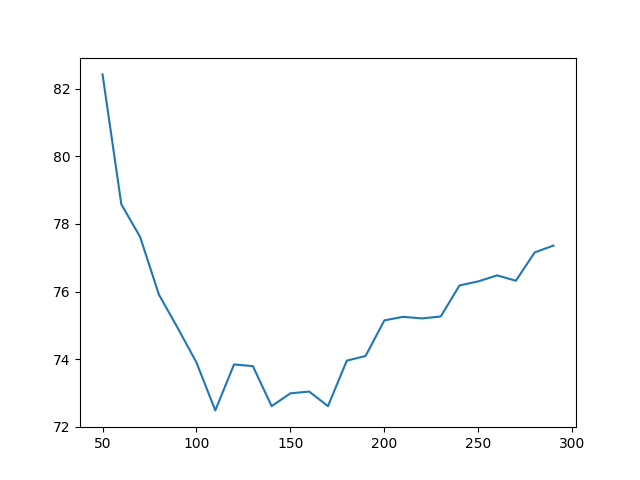
****

图1-19 不同k值下的推荐系统性能评估（sse）

1. **选用不同的hash函数数量**

为了测试minihash过程中，不同hash函数数量对于系统预测性能的影响，我在[10,210]之间以步长50取不同的hash函数数量，计算算法的sse值，实验结果如图1-20所示。

从实验结果来看，随着hash函数数量的增加，sse值逐渐降低，但是下降的趋势越来越缓慢。这可能是因为随着hash函数数量的增多，其发生哈希冲突的可能性就越小，在相似度计算时产生的误差降低，预测效果也越好。但是数量太多，在生成签名矩阵时计算复杂度非常高，影响算法的性能。

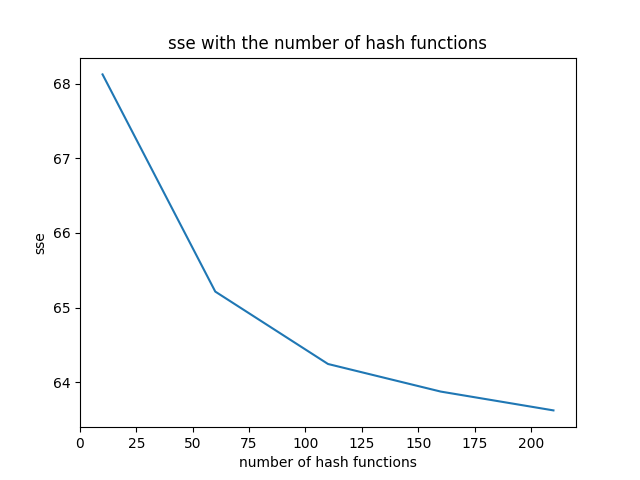


图1-19 不同hash函数数量的推荐系统性能评估（sse）

## **1.4 实验总结**

## **1.4.1 实验中遇到的问题及解决**

**1）Minihash运行时间过长**

在加入minihash时，我最使用自带的hash算法和jaccard相似度计算公式进行矩阵构建，在运行时发现生成签名矩阵和相似度矩阵的过程耗时非常长，达到了minute级别，在请教助教和同学之后我调用了datasktech库中的相关函数，运算效率极大的提升，这可能是因为其运算的底层逻辑为CPython，无需经过Python解释器，性能提升很大。

**2）Minihash构建相似度矩阵**

在使用Jaccard构建相似度矩阵时，我的维度并未与非minihash算法进行统一，比如在基于用户的推荐算法中，原本的相似度矩阵维度为671\*671（与用户数相同），而我构建的维度为9025\*9025（与电影数相同），这也导致了我后续运行时的错误。经过排查我很顺利地解决了错误，这属于我对算法的理解不够透彻，维度的确定的确是非常重要的。

**3）索引报错**

在寻找k个相似用户时代码出现了报错图1-20所示的报错，这代表着索引未找到。在仔细审视代码后，我发现程序在最初构建效用矩阵时，使用的索引为user\_id，其格式为str字符串，而我在预测函数中提供的输入为int格式，这样小的疏忽困扰了我很久，虽然解决方法非常简单，但是这也提醒我一定要注意代码前后的连贯性，不能顾此失彼。

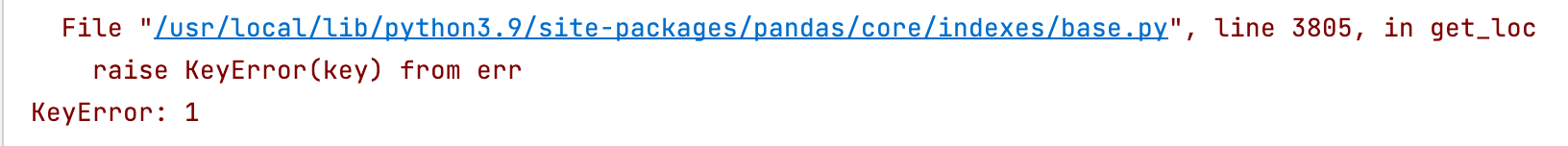


图1-20 报错展示

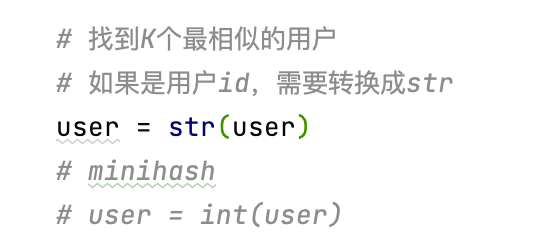


图1-21 解决方案

## **1.4.2 实验总结**

本次大作业我完成了两种不同的推荐算法，还熟悉了Minihash进行降维的处理操作。不同的推荐算法侧重点不同，基于用户的推荐算法简单直观，可解释性强，但是受用户差异影响较大，且容易忽视用户自身的个性化喜好；基于内容的推荐算法不依赖用户喜好，独立性较强，但是太过依赖物品特征的提取，如果面对难以提取特征的复杂场景，其效果可能受到影响。这些算法基于不同的假设，采用不同的计算方式，让我感受到了对同一问题的发散性理解可以带来更多创新的火花。

算法的优化也同样可以从数据集特征入手，比如在本实验中无论是用户-电影矩阵，还是电影-类别矩阵，都属于稀疏矩阵，而minihash对于稀疏矩阵的处理能力更强，所以在原有推荐算法的基础上加入minihash可以带来不错的优化。

借此机会我还想谈谈对于课程的理解。大数据分析这门课程涵盖了多个大数据应用领域的常见算法，不仅有PageRank，Map-Reduce，Apriori这些实验涉及的算法，还有Stream Data，LSH，SVD等更加复杂，更加奇妙的大数据分析算法。对于每一个算法，老师都会从原理讲起，从数学推导讲起，结合实际应用帮助我们快速理解算法的本质。我认为这是一门开拓眼界的课程。

老师的一句话至今让我印象深刻，“你对大数据的理解，应当区别于你在生活中遇到的普通数据”，当数据规模变得不可想象时，概率采样，哈希等近似计算的策略能够减少计算复杂度，同时保持不错的性能，这些策略对于普通规模的数据来说无关紧要，但对于大数据来说至关重要，这也在不断鞭策我，尝试以更多元的视角去对待大数据，处理与分析大数据。

# 大作业推荐系统实验 参考文献

[1] "Content-based filtering vs Collaborative filtering." ResearchGate, Accessed [19 May2023],https://www.researchgate.net/figure/Content-based-filtering-vs-Collaborative-filtering-Source\_fig5\_323726564.

[2] Recommendation System: User-Based Collaborative Filtering." GrabNGo Info, Medium,11July2019,https://medium.com/grabngoinfo/recommendation-system-user-based-collaborative-filtering-a2e76e3e15c4.

[3] "MinHash." Wikipedia, The Free Encyclopedia, 19 May 2023, <https://en.wikipedia.org/wiki/MinHash.>

[4] 尤雷·莱斯科夫（Jure Leskovec） / 阿南德·拉贾拉曼（Anand Rajaraman） .斯坦福数据挖掘教程（第3版）.人民邮电出版社.2021

[5] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." Recommender systems handbook . Boston, MA: springer US, 2010. 1-35.