

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： 本硕博2301班**

**学 号： U202315763**

**姓 名： 王家乐**

**指导教师： 崔金华**

**报告日期： 2025.5.27**

**计算机科学与技术学院**

**目录**

[实验五 推荐系统算法及其实现 1](#_Toc27024)

[1.1 实验目的 1](#_Toc28937)

[1.2 实验内容 1](#_Toc18191)

[1.3 实验过程 2](#_Toc6296)

[1.3.1 编程思路 2](#_Toc7771)

[1.3.2 遇到的问题及解决方式 2](#_Toc32363)

[1.3.3 实验测试与结果分析 2](#_Toc10468)

[1.4 实验总结 2](#_Toc11632)

# 实验五 推荐系统算法及其实现

## **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. 推荐算法中，利用minhash算法对效用矩阵进行降维处理

## **1.2 实验内容**

给定Recommend-data.zip电影数据集，包含电影评分文件（ratings.csv，原始电影评分数据，仅供参考），训练集train\_set.csv和测试集test\_set.csv（原始电影评分拆分出来的两个数据集），movies.csv电影信息数据。

根据给定的数据集，完成以下推荐系统任务：

1）**基于用户-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。实现基于协同过滤的推荐系统算法，完成对单个指定的userID用户进行推荐，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。此外，利用测试集数据，计算推荐算法的SSE（误差平方和），在测试集中包含100条用户-电影评分记录，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，获得SSE。

1. **基于用户-用户的协同过滤推荐优化方法。**此方法采用minhash算法对效用矩阵进行降维处理，从而得到相似度矩阵。注意minhash采用jaccard方法计算相似度，需要对效用矩阵进行01处理，这儿将0.5-2.5的评分置为0，3.0-5.0的评分置为1。
2. **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。实现基于内容的推荐系统算法，完成对某个用户-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：

其中，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。此外，对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，计算获得SSE值。

**4）基于内容的推荐优化方法。**该方法使用minhash算法对基于内容推荐算法的相似度计算进行降维，把最小哈希的模块作为一种近似度的计算方式，从而得到相似度矩阵。注意minhash采用jaccard方法计算相似度，特征矩阵应为01矩阵，因此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。

备注：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，最小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。

## **1.3 实验过程**

### 1.3.1 编程思路

**1. 基于用户-用户的协同过滤推荐算法**

（1）数据加载：从 data/ 目录中加载 movies.csv, train\_set.csv, test\_set.csv 三个数据文件：movies.csv：电影信息；train\_set.csv：训练集用户评分；test\_set.csv：测试集用户评分。

（2）构建用户评分数据结构：user\_ratings：{user\_id: {movie\_id: rating}}；user\_avg：{user\_id: 平均评分}。计算用户的平均评分：对于用户 u，其评分集合为 ，则平均评分为： 。

（3）用户评分归一化：对每个用户的评分进行中心化归一化处理（即每个评分减去用户的平均评分，）。

（4）计算用户之间的相似度：对每对用户u、v，计算他们对共同电影的评分余弦相似度。设共同评分电影集合为，则 。

（5）构建用户相似度矩阵：计算所有用户两两之间的相似度，保存在 sim\_matrix中；每个用户按相似度排序其邻居。

（6）评分预测：对用户u和电影i，使用其最相似的k个用户的评分预测该电影的评分。设邻居集合为，预测评分为： 。

（7）评估模型性能：对测试集中每个用户-电影对，计算平方误差；累加求出SSE（）。

**2. 基于用户-用户的协同过滤推荐优化方法**

（1）数据加载：从 data/ 目录中加载 movies.csv, train\_set.csv, test\_set.csv 三个数据文件：movies.csv：电影信息；train\_set.csv：训练集用户评分；test\_set.csv：测试集用户评分。

（2）构建用户评分数据结构：user\_ratings：{user\_id: {movie\_id: rating}}；user\_avg：{user\_id: 平均评分}；binary\_user\_ratings：{user\_id: 0/1}。评分二值化（用于后续 MinHash，规则：评分>=3.0记作1，否则记作0）。

（3）生成MinHash签名：对每个用户的喜欢的电影集合（即 binary\_rating == 1）生成MinHash签名，使用多个哈希函数，每个用户生成一个长度为num\_hashes的签名向量，记录每个哈希函数作用下的最小哈希值，对每个用户u，其签名为： 。

（4）计算用户相似度矩阵：使用Jaccard相似度即两个签名向量中相同位置相等的数量比上签名长度， 。

（5）对于用户u未评分的电影i，设邻居集合为，则预测评分为： 。

（6）评估模型性能：对测试集中每个用户-电影对，计算平方误差；累加求出SSE（）。

**3. 基于内容的推荐算法**

（1）数据加载：从 data/ 目录中加载 movies.csv, train\_set.csv, test\_set.csv 三个数据文件：movies.csv：电影信息；train\_set.csv：训练集用户评分；test\_set.csv：测试集用户评分。

（2）构建用户评分数据结构：user\_ratings：{user\_id: {movie\_id: rating}}；user\_avg：{user\_id: 平均评分}。计算用户的平均评分：对于用户 u，其评分集合为 ，则平均评分为： 。

（3）构建TF-IDF向量：把genres按 | 分词（比如 "Action|Thriller" -> ["Action", "Thriller"]）；TF： ， 是电影d中类型t出现的次数；IDF：，N是总电影数，df(t)是类型t出现的电影数；TF-IDF向量：。

（4）计算余弦相似度： , 是电影i、j的TF-IDF向量，使用嵌套的循环的暴力计算（可替换为 sklearn 的 cosine\_similarity）。

（5）评分预测：根据用户之前看过的电影及其评分，通过与目标电影的相似度加权平均，预测用户对未看电影的评分。 ，为用户u看过的电影集合，sim(i,m)为电影i与目标电影m的相似度。

（6）模型评估：对测试集中每个用户-电影对，计算平方误差；累加求出SSE（）。

**4. 基于内容的推荐优化方法**

（1）数据加载：从 data/ 目录中加载 movies.csv, train\_set.csv, test\_set.csv 三个数据文件：movies.csv：电影信息；train\_set.csv：训练集用户评分；test\_set.csv：测试集用户评分。

（2）构建用户评分数据结构：user\_ratings：{user\_id: {movie\_id: rating}}；user\_avg：{user\_id: 平均评分}。计算用户的平均评分：对于用户 u，其评分集合为 ，则平均评分为： 。

（3）生成MinHash签名：对每个电影的特征集合生成MinHash签名，使用多个哈希函数，每个用户生成一个长度为num\_hashes的签名向量，记录每个哈希函数作用下的最小哈希值，对每个电影m，其签名为： 。

（4）计算电影相似度矩阵：使用Jaccard相似度即两个签名向量中相同位置相等的数量比上签名长度， 。

5）评分预测：根据用户之前看过的电影及其评分，通过与目标电影的相似度加权平均，预测用户对未看电影的评分。 ，为用户u看过的电影集合，sim(i,m)为电影i与目标电影m的相似度。

（6）模型评估：对测试集中每个用户-电影对，计算平方误差；累加求出SSE（）。

### 1.3.2 遇到的问题及解决方式

**1. 数据稀疏性问题：**  
 在构建用户-电影评分矩阵时发现，大部分用户只对少数电影进行了评分，导致评分矩阵非常稀疏，影响了相似度计算的准确性。  
 解决方式：  
 尝试对评分矩阵进行填补处理，但最终决定采用只计算有共同评分项的用户对或电影对，从而在一定程度上缓解了稀疏性影响。此外，使用最小哈希（MinHash）方法也有效减少了稀疏数据带来的计算开销。

**2.** MinHash**算法实现复杂度高**：  
 初次实现MinHash算法时，对哈希函数的构造方式不明确，导致签名结果不准确，进而影响了Jaccard相似度的计算。  
 解决方式：  
 通过查阅相关资料，明确了哈希函数的形式 h(x) = (a\*x + b) % c，并设置多组互异的a、b参数，保证哈希函数的独立性。同时验证每组MinHash结果的合理性。

**3.** TF-IDF**向量构造及归一化：**  
 在基于内容的推荐算法中，对genres字段进行TF-IDF编码时发现，不同电影的类别数量不一，导致向量维度不一致。  
 解决方式：  
 统一所有电影的类别集合作为特征空间，并对每部电影建立与该特征空间对应的一致长度的TF-IDF向量，填充为0的项表示该电影不具备该特征。

**4. 相似度矩阵计算耗时严重**：  
 在计算所有用户之间、或所有电影之间的相似度矩阵时，算法时间复杂度高，尤其是使用余弦相似度计算时耗时更明显。  
 解决方式：  
 尝试利用向量化工具（如NumPy）进行矩阵级计算，同时限制相似度矩阵中只保留Top-K相似项，从而加快整体处理速度。

### 1.3.3 实验测试与结果分析

**1. 基于用户-用户的协同过滤推荐算法**

代码运行结果如下：

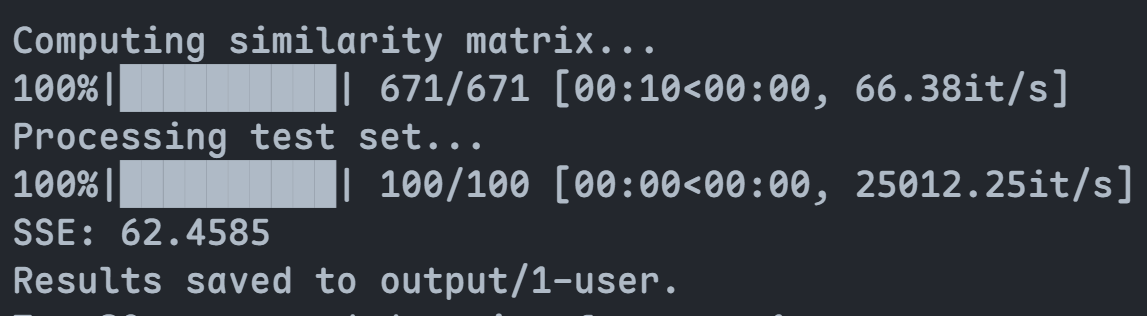


图1-1 代码运行结果

为用户1选取最相近的30个用户推荐10部电影：

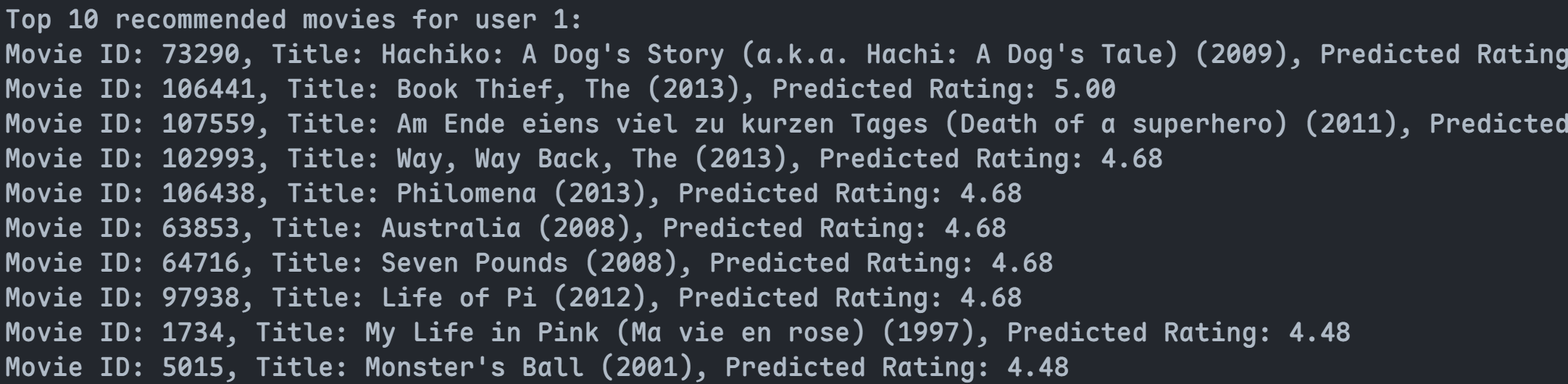


图1-2 用户1推荐结果

对于测试集，取相似用户数量为30，SSE为62.4585：



图1-3 测试集结果

**2. 基于用户-用户的协同过滤推荐优化方法**

代码运行结果如下：

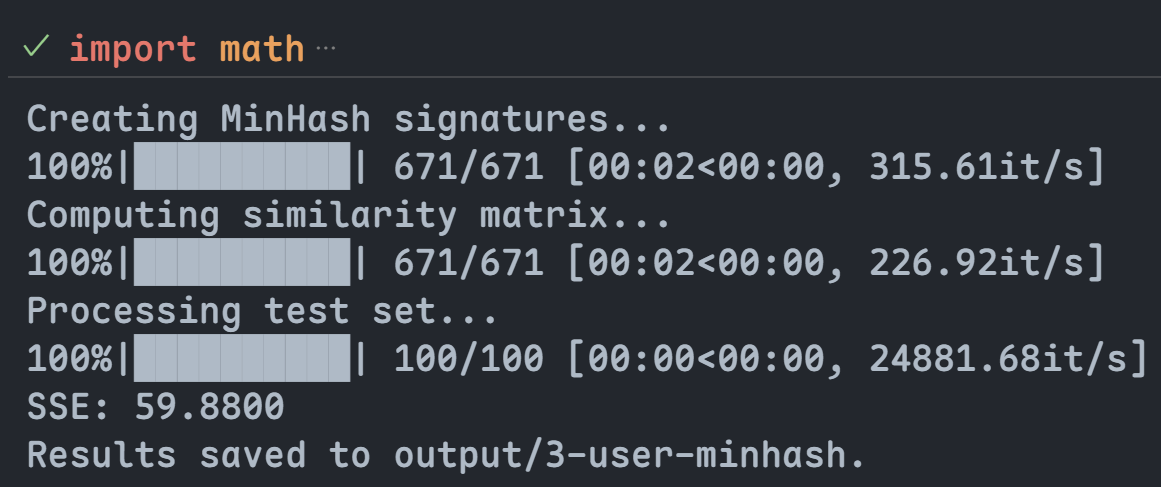


图2-1 代码运行结果

对于测试集，取相似用户数量为30，SSE为59.8800，预期评分与实际评分较为接近：

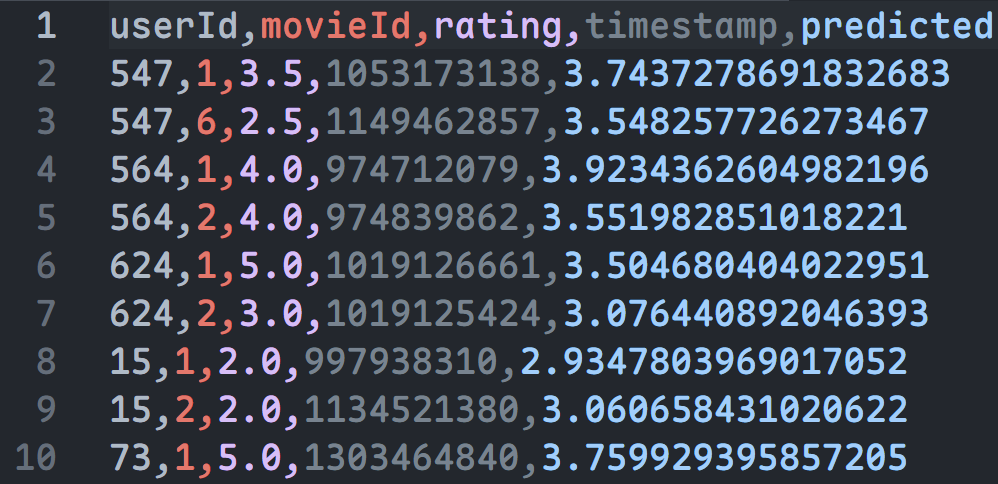


图2-2 测试集结果

**3. 基于内容的推荐算法**

代码运行结果如下：

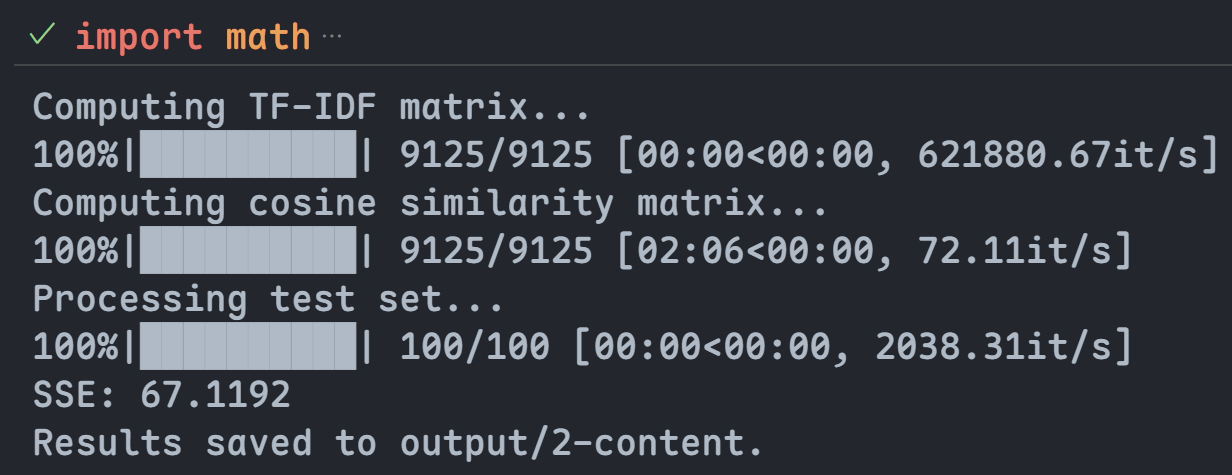


图3-1 代码运行结果

对于测试集，SSE为67.1192，预期评分与实际评分较为接近：

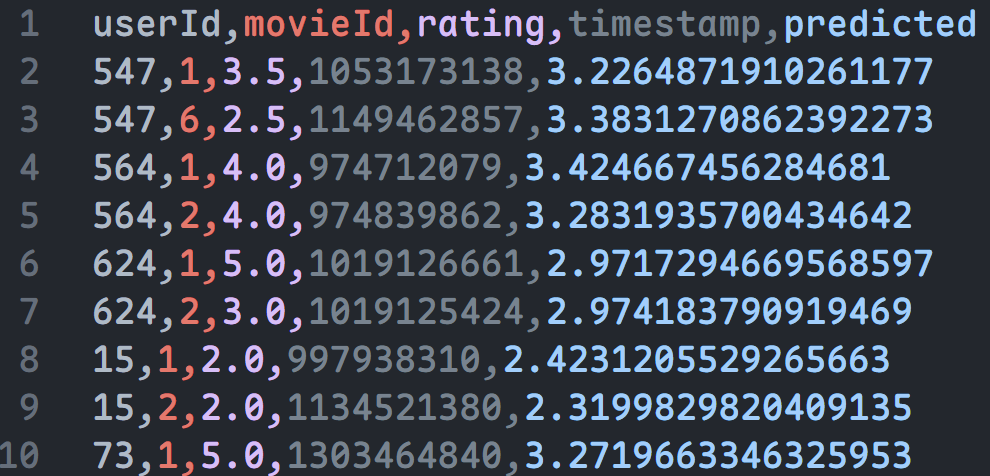


图3-2 测试集结果

**4. 基于内容的推荐优化算法**

代码运行结果如下：

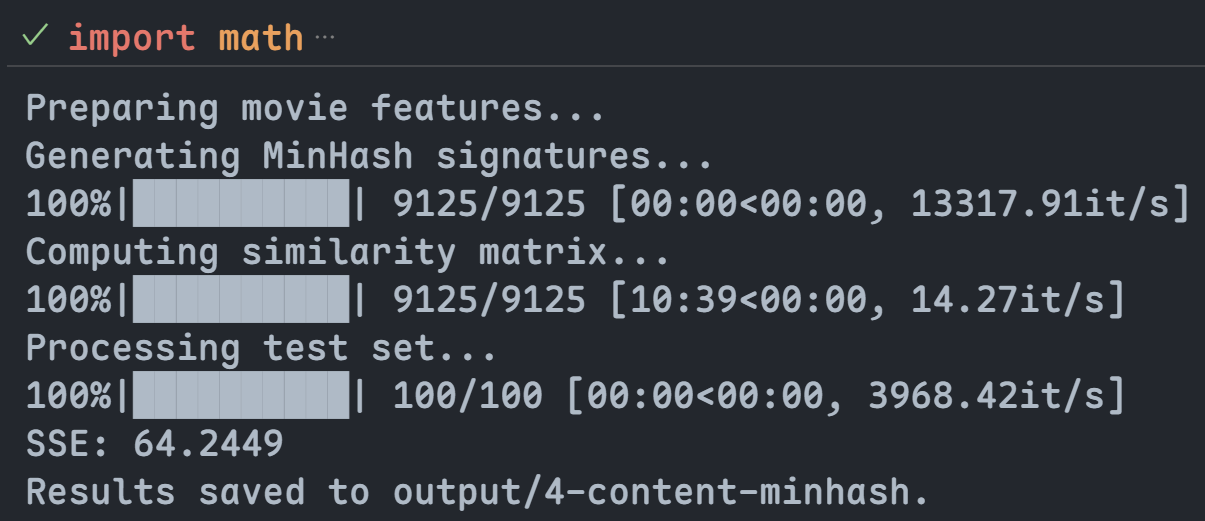


图4-1 代码运行结果

对于测试集，SSE为64.2449，预期评分与实际评分较为接近：

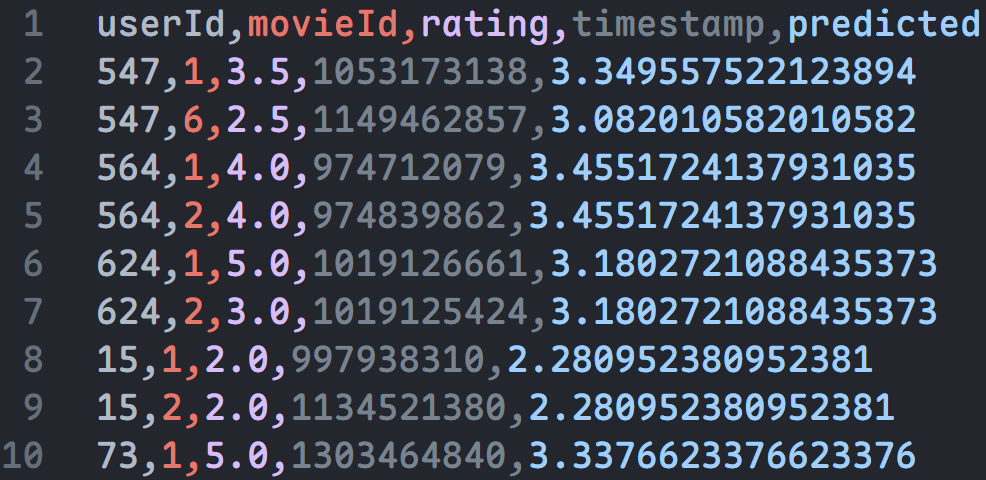


图4-2 测试集结果

1. **总结**

对四种算法进行实现和测试，结合100条用户-电影评分的测试数据，得到了以下实验结果：

| 算法 | SSE | 分析 |
| --- | --- | --- |
| 基于用户协同过滤 | 62.4585 | 挖掘用户间兴趣相似性 |
| 基于用户协同过滤（MinHash优化） | 59.8800 | 降低了计算开销 |
| 基于内容推荐 | 67.1192 | 推荐结果更具个性化 |
| 基于内容推荐（MinHash优化） | 62.2499 | 对新电影推荐更快 |

表5-1 总结

## **1.4 实验总结**

本次实验内容是推荐系统的实现，分为基于用户的协同过滤算法和基于内容的推荐算法，分别以用户间的相似性和电影间的相似性为出发点，来对目标用户的喜好进行预测，这也是实际中推荐系统的重要方法。

但是在实现过程中，我也发现了这两种算法存在的问题。对于基于用户的协同过滤算法，如果用户的相似用户中看过某部电影的人较少，那么在不对公式进行修正的情况下，对于这部电影的评分就会被这少数人极大影响，从而造成结果的失真。而基于内容的推荐算法存在的问题是，电影的分类并不能很好地作为用户评分的参考标准，即用户评分时不会以分类作为单一标准，还可能有电影情节、特效制作等方面的标准，仅以电影分类为预测标准是有失偏颇的。因此，在真实的推荐系统中，我们不仅需要增加推荐的相关指标，也需要将基于用户、基于内容两种方式进行结合，以期找到更精准的推荐结果。

本次实验的主要收获为：掌握了协同过滤与基于内容推荐两类主流推荐算法的实现细节与差异；了解了MinHash算法对高维稀疏矩阵降维处理的实际效果；

学会了使用多种相似度度量方法（余弦相似度、Jaccard相似度、Pearson相关系数）；提升了数据结构组织、向量化编程、哈希函数构造等实际编程能力。