

**课 程 实 验 报 告**

**课程名称： 大数据分析**

**专业班级： CS1806**

**学 号： U201814670**

**U201814672**

**姓 名： 李田田**

**何灵姗**

**指导教师： 崔金华**

**报告日期： 2020.12.30**

**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](#_Toc59543291)

**[1.1实验目的](#_Toc59543292)** [1](#_Toc59543292)

**[1.2 实验内容](#_Toc59543293)** [1](#_Toc59543293)

**[1.3 实验过程](#_Toc59543294)** [3](#_Toc59543294)

[1.3.1 编程思路 3](#_Toc59543295)

[1.3.2 遇到的问题及解决方式 5](#_Toc59543296)

[1.3.3 实验测试与结果分析 6](#_Toc59543297)

**[1.4 实验总结](#_Toc59543298)** [8](#_Toc59543298)

# 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. 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。**

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**。

**选做（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 编程思路

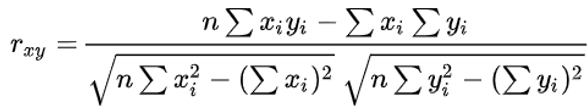
1. 基于用户的协同过滤推荐算法

建立用户模型：用训练集中的评分数据构造用户-电影效用矩阵。

获取相似用户集合：对每个用户进行推荐时，分别调用pearson函数计算出该用户与其他所有用户的相似度，取相似度最高的前K个用户。其中：

Pearson相似度计算：

用户x与用户y的相似度计算公式：



其中xi表示用户x对电影i的评分，yi表示用户y对电影i的评分，n为电影总数。

预测该用户所有未评分电影的评分：对于每一部未评分电影，预测评分由两部分相加组成。一部分是该用户的所有已评分电影的平均打分，一部分是补充打分。补充打分来自那K个相似用户，是该相似用户集中每个用户对于当前电影的补充打分的一个加权平均，权重依赖于用户的相似度。每个用户对当前电影的补充打分是他的直接打分减去他自己的平均打分。具体计算公式如下：

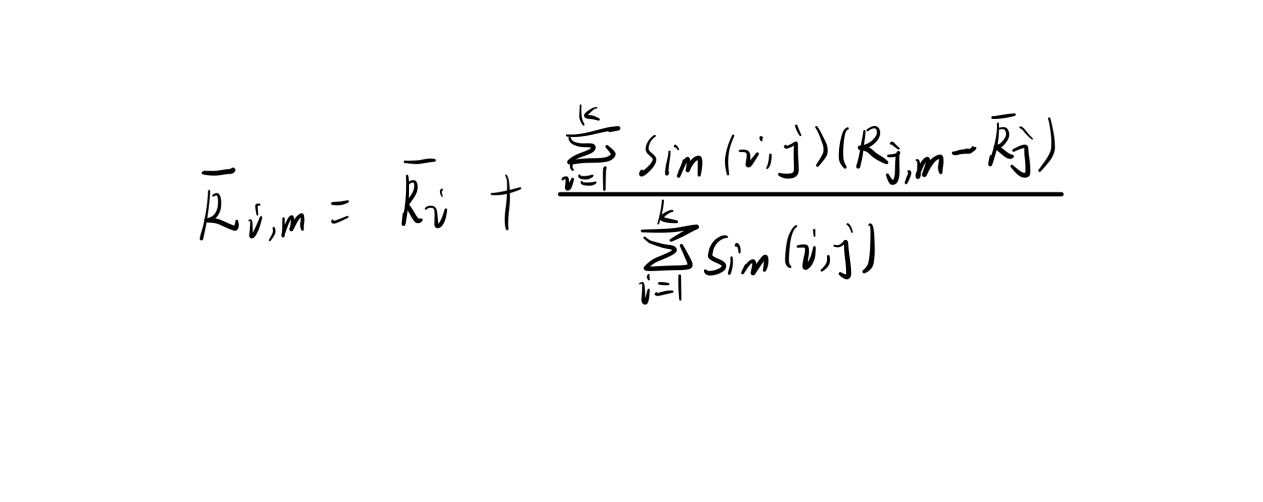


图1.1 预测评分公式

其中，Ri,m表示用户i对电影m的预测评分，Ri表示用户i的所有已评分电影的平均打分，sim(i, j)表示用户i和用户j的相似度，Rj,m表示用户j对电影m的预测评分，Ri表示用户i的所有已评分电影的平均打分。

取前k个与该用户相似度最高的用户，对于这k个用户中与该用户相似度大于0的用户套用该公式，小于0的舍弃。

选取前n个电影进行推荐时，这些电影都是与其相似的k个用户看过的，但是该用户还未看过。

误差计算：对测试集中的100条用户-电影记录预测评分，与真实评分一起带入下列公式计算。



2. 基于内容的推荐算法

对数据集movies.csv中的电影类别进行分词，获取所有m个特征值。先生成电影与类别的矩阵，如果该电影对应于这个类别，则矩阵[电影，类别]=1，利用该矩阵来计算生成tf-idf矩阵。此处tf取特征值在该电影类别中的出现频率，也就是该电影对应一行中相加的和的倒数，idf为总电影数n与包含某特征值的电影数的商，再取以10为底的对数，也就是对应矩阵[,类别]这一列的和除以总行数，再取以10为底的对数就是idf值。而tf-idf矩阵生成就是矩阵中对应位置矩阵[电影，类别]处tf值乘以idf值。

对每个电影，调用余弦相似度函数根据tf-idf矩阵计算出该电影于其他电影的相似度，生成电影相似度矩阵。

对测试集中的每条用户-电影记录，根据下列公式预测评分：

其中，sim(i)为第i个电影与当前用户-电影的相似度，score’(i)为计算集合中第i个电影的分数。计算集合不一定为K，因为对于相似度为负数的电影不纳入计算。当n为0时，score为该用户所有已打分电影的平均值。

为用户推荐电影时，利用该公式通过用户已经打分的电影来依次计算电影相似度矩阵中每一个电影的预测打分(除去用户已看过的电影)，对其进行排序，选出其中前k个电影进行推荐。

误差计算：同上。

### 1.3.2 遇到的问题及解决方式

1. 基于用户的协同过滤算法

（1） 在写基于用户的协同过滤算法时，一开始利用流程图上给出的公式计算出来的sse结果为73左右，误差有些大。再查找资料后，对该公式进行了修正，得到1.3.1中所用的公式。

（2）计算用户之间的相似度时，因为矩阵计算量比较大，在用自己编写的pearson相似度函数进行计算时，程序的开销很大，需要很长时间才能运行出来。调用库里的函数之后，进行了重新的编写，得到的新程序运行比较快。

（3）对于用户i，假如与其相近的k个用户都未看过电影m，但是测试集里要求用户i对电影m进行评分，这就会造成无法评分，而只能将K值变大来求得该评分。

2. 基于内容的推荐算法

（1） 遇到的问题：余弦相似度矩阵计算时间太长

解决方式：直接调用库函数cosine\_similarity()计算余弦相似度矩阵

（2）遇到的问题：给用户推荐电影时不知道应该怎么计算排名

解决方式：利用用户已经观看的电影所打出的分数以及给出的预测分数公式，对所有电影进行预测打分，选出排名前k个预测打分高的电影，输出电影名字进行推荐。

（3）遇到的问题：计算所得的SSE误差平方和太少，只有10的-27次方

解决方式：发现是分值代入出错，代入了应该预测电影的分数，修改之后，代入已观看的电影打出的分数，SSE误差平方和为67。

### 1.3.3 实验测试与结果分析

1. 基于用户的协同过滤算法

改变k值，得到不同的SSE的值，作图如下所示，从图中可以看出当k值达到95时，得到的SSE的值最小，为50.87。K最小取39时才能完成测试集的计算。

图1.2 不同的k值对应的SSE折线图

对于用户1，根据最相似前80个用户进行推荐，为该用户推荐出的10个电影如下图所示，包括movieid、rating、moviename、genres。



图1.3对用户1推荐结果

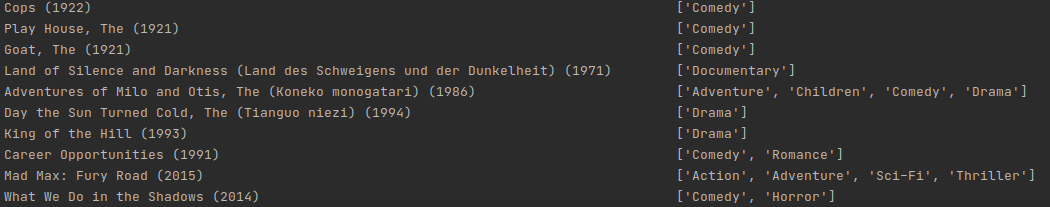


图1.4对用户1推荐结果

2. 基于内容的推荐算法

测试集预测打分结果：

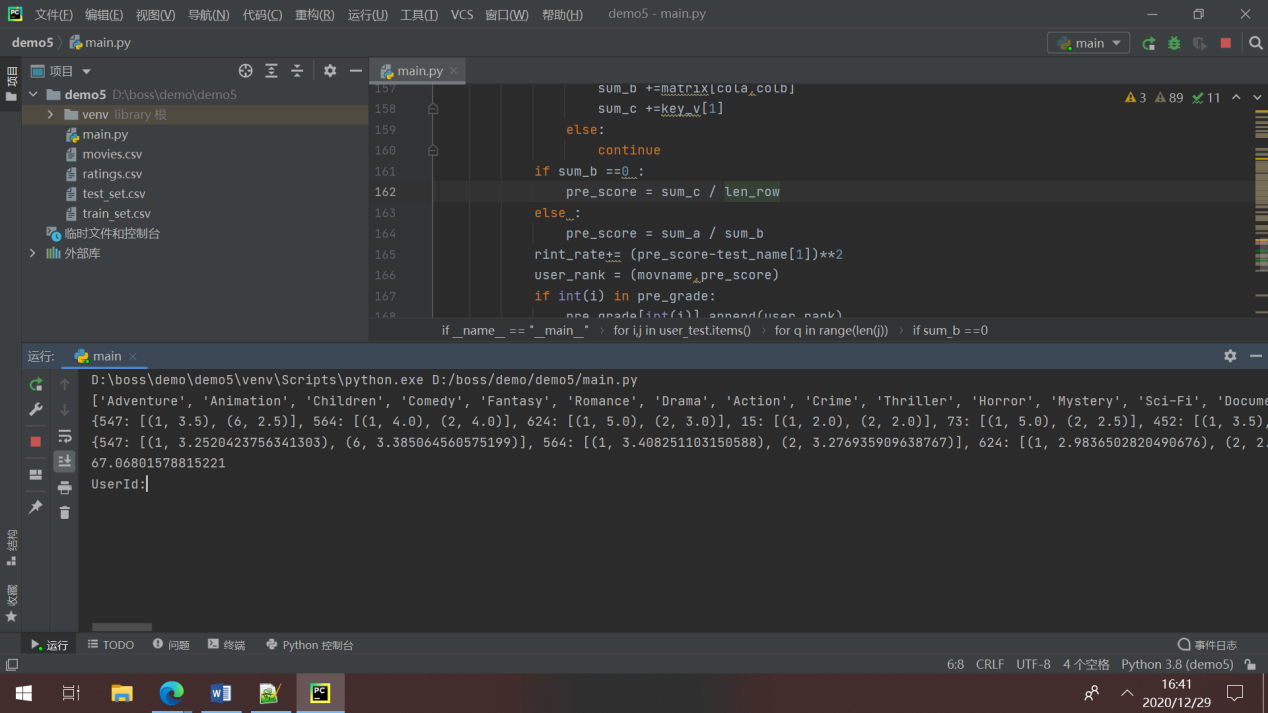


图1.5 预测打分结果

结果分析：由图可知预测打分结果中有的分数预测和实际分数接近，有的预测分数与实际分数相差一定值，最终SSE误差平方和计算为67.068

测试为用户ID为2的推荐前10个电影：

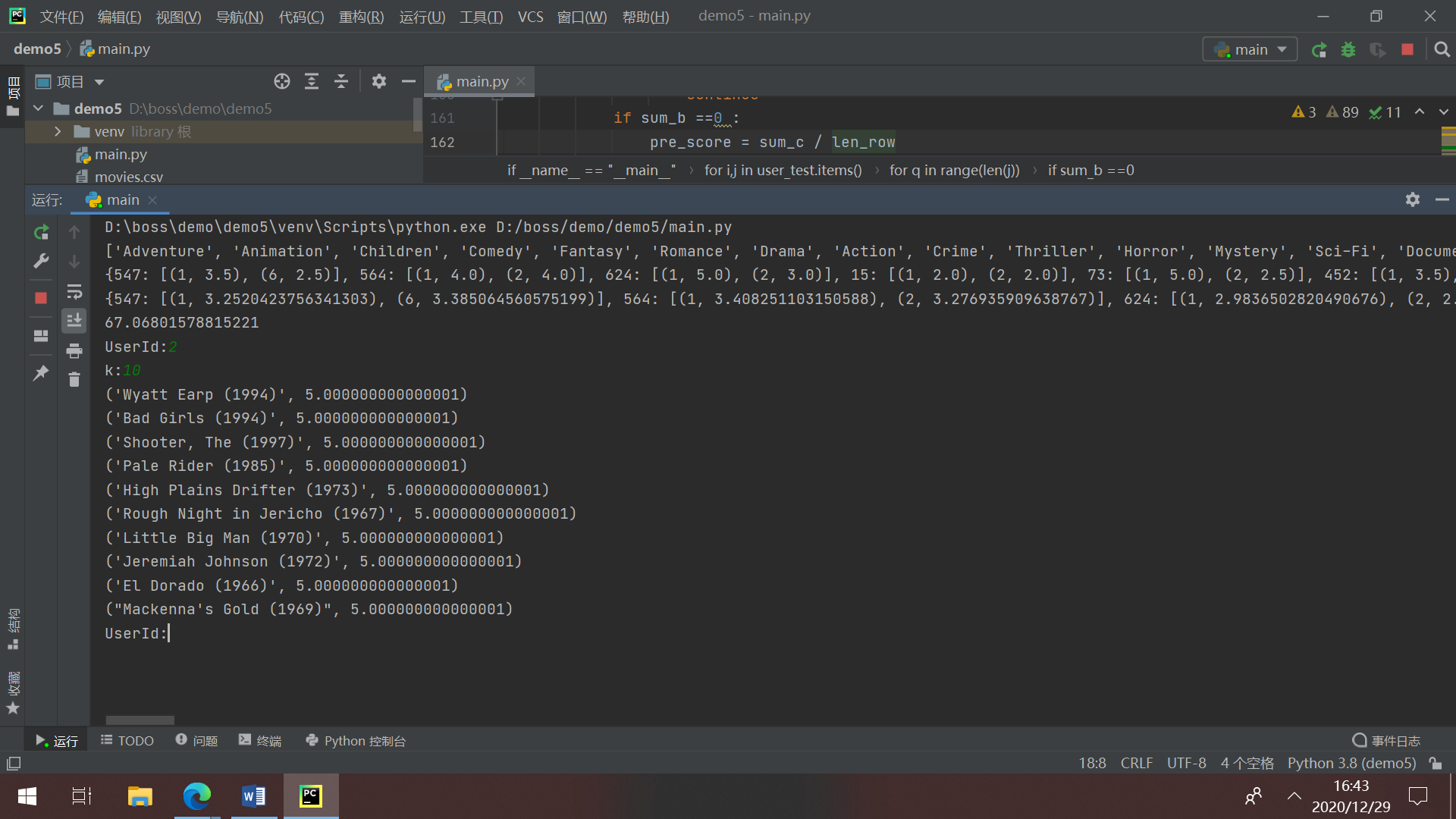


图1.6对用户2推荐结果

## **1.4 实验总结**

**李田田：**

刚开始写实验一时，对于这些数据用列表还是字典、对于其中每一个元素是一个还是几个特征的集合进行了考虑。最后选取了用户字典，其格式为user\_rate[用户id]=[(电影id,电影评分)...]，电影字典其格式为movie\_user[电影id]=[用户id1,用户id2...]，以便于完成后续的查找工作。在写代码的时候也遇到了很多问题，一开始对程序计算出的结果七十几表示很怀疑，感觉总有一些大，经过和队友讨论，通过对公式进行优化，尽量减少单个用户对自己看过的电影的评分很高的影响，最后得到的SSE比之前小很多。同时，自己也进一步实验比对k值的选取对SSE值的影响。在整个实验的过程中，一点点理解算法，理解基于用户的协同过滤和基于物品的协同过滤和基于内容的推荐算法的区别与共同之处，觉得推荐系统与生活的联系十分紧密，当然实际中大多数的应用的推荐算法都采用混合推荐，以提高推荐的准确性。

除此之外，对于测试集进行计算SSE时发现了该算法的一些问题所在让我对该算法有了更深入的理解。总之，在本次实验中收获良多。

**何灵姗：**

一开始看了实验二的要求后，整个人都懵，有点没看懂要干什么，后来和队友交流了后，明白了要做什么，但是感觉很难，自己会做不出来。不过后来明白要怎么做了，就开始一步一步写代码。整个实验下来感觉最麻烦的莫过于文件读取和生成矩阵了，光是读取movies.csv的代码就修改了一定的时间，生成了电影类别矩阵后，计算tf-idf矩阵就容易很多，一开始余弦相似度矩阵生成是自己写的一段代码，写完了跑了一下，跑了几十分钟，最终选择还是调用库函数比较香。生成了电影相似度矩阵后再预测分数还有推荐电影就容易了很多，中间因为写的自己都有点乱了，把预测分数的公式中代入的分数值搞错了，计算出来的SSE和队友的SSE相差很大，后来检查才发现是值带错了。最终写完代码后，和队友的电影推荐不一样，不是很明白原因。总的来说，和队友边交流边写代码，有很多不懂的地方都通过交流解决了。