

**实验(实训)课程报告**

**课程 数据挖掘技术实践**

|  |  |  |  |
| --- | --- | --- | --- |
| 院　　别 | 信息技术与管理 | 专 业 | 计算机科学与技术 |
| 届 别 | 2022 | 班 级 | 计算机科学与技术（职高）一班 |
| 学生姓名 | 邱宇杰 | 学 号 | 202205650121 |
| 指导教师 | 钟敏娟 | 职 称 |  |

湖南财政经济学院

2024年 8 月

**实验(实训)报告4**

实验(实训)时间: 2024年 12月 1日 指导教师评分：

|  |  |
| --- | --- |
| 实验项目 | 实验四：电影智能推荐 |
| 组员名单 | 独立完成或姓名（学号）（操作性和验证性实验均独立完成，综合性实验3-5人一组） |
| 实验类型 | □基础性实验 ☑验证性实验 □综合性实验 |
| 实验(实训)时间 | 2024年 12月 1日 |

1. 实验目的和要求：

请基于671个用户对6993部电影的评分数据，利用关联规则挖掘算法构建推荐模型，实现影片智能推荐。

此次实验包含ratings和movie两个数据文件，ratings是不同用户对不同电影的评分，包括用户id、电影id、电影打分以及评分时间；movies是电影的基本信息，包括电影id、电影名字以及电影类型。

分析提示：

1. 读入数据文件；
2. 为了便于电影推荐，仅关注评分大于3的电影；
3. 整理数据，转换成用户所看的电影列表；
4. 利用mlxtend进行关联分析
5. 实验步骤（记录程序代码、分析实验结果）

import pandas as pd

import numpy as np

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import apriori, association\_rules

# 导入数据

ratings = pd.read\_csv(r'作业\电影智能推荐\ratings.csv')

movies = pd.read\_csv(r'作业\电影智能推荐\movies.csv')

# 筛选出评分大于3的记录

high\_ratings = ratings[ratings['rating'] > 3]

# 将电影ID和用户ID进行聚合，得到每个用户看过的电影列表

user\_movie\_lists = high\_ratings.groupby('userId')['movieId'].apply(list).reset\_index()

user\_movie\_lists.columns = ['userId', 'movies\_watched']

# 将用户看过的电影列表转换成布尔矩阵

te = TransactionEncoder()

te\_ary = te.fit\_transform(user\_movie\_lists['movies\_watched'])

df\_bool = pd.DataFrame(te\_ary, columns=te.columns\_)

# 使用 apriori 算法进行频繁项集挖掘

frequent\_itemsets = apriori(df\_bool, min\_support=0.05, use\_colnames=True)

# 生成关联规则

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1.0)

def get\_recommendations(watched\_movies, rules):

recommendations = set()

watched\_movies = watched\_movies[0] if isinstance(watched\_movies, np.ndarray) else watched\_movies

for movie in watched\_movies:

recs = rules[rules['antecedents'] == frozenset([movie])]['consequents']

for rec in recs:

recommendations.update(rec)

recommendations -= set(watched\_movies) # 去掉已经看过的电影

return list(recommendations)

# 示例用户的电影推荐

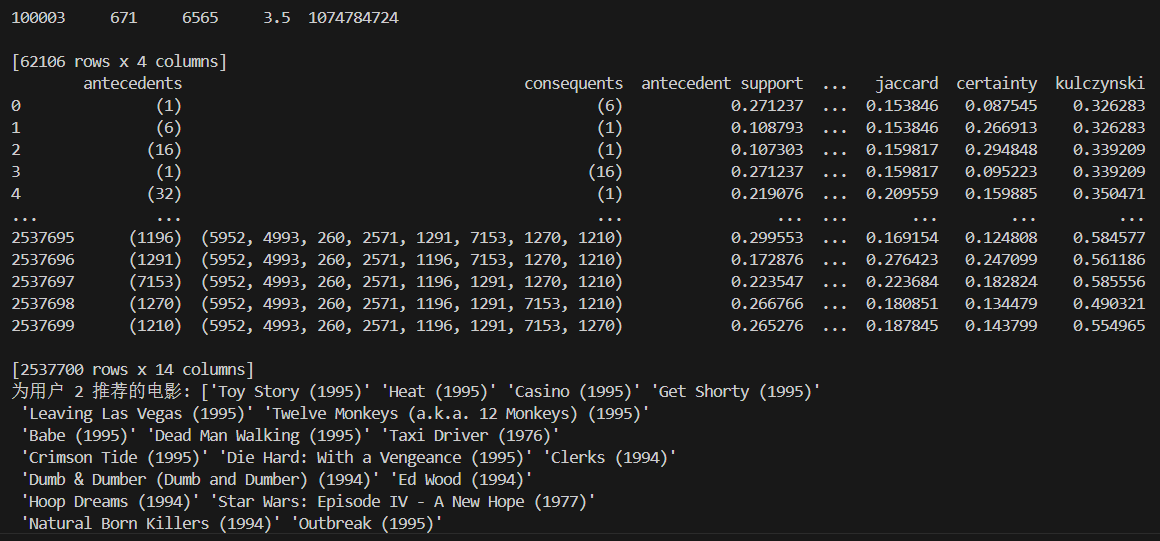
example\_user\_id = 2 # 假设我们要为用户ID为2的用户推荐电影

example\_user\_movies = user\_movie\_lists[user\_movie\_lists['userId'] == example\_user\_id]['movies\_watched'].values

recommended\_movies = get\_recommendations(example\_user\_movies, rules)

# 转换电影ID为电影名字

recommended\_movie\_names = movies[movies['movieId'].isin(recommended\_movies)]['title'].values

print(f"为用户 {example\_user\_id} 推荐的电影: {recommended\_movie\_names}")

1. 心得体会与建议

**通过这次实验我学到了很多**