**图片包含 游戏机, 房间, 食物

描述已自动生成 南昌大学实验报告**

学生姓名： 周华俊 学 号： 8008122013 专业班级： 软件工程2201班

实验类型：□ 验证■ 综合 ☑ 设计 □ 创新 实验日期： 2024.12.13 实验成绩：

**一、实验项目名称**

关联规则、文本推荐

**二、实验目的**

1. 熟悉pandas、sklearn库
2. 掌握频繁集生成
3. 掌握关联规则
4. 掌握TF-IDF概念
5. 掌握协同滤波推荐

**三、实验任务**

参照课堂讲解案例，完成movies.xlsx数据的频繁集和关联规则推理。

**四、主要仪器设备及耗材**

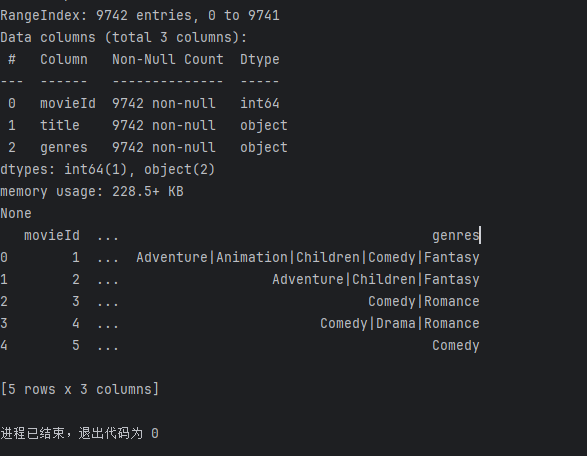
软件：pycharm。

**五、实验步骤**

1 .老样子，第一步导入库和加载数据以及观察数据基本结构

import pandas as pd  
# 加载数据集  
file\_path = r'C:\Program Files\JetBrains\fordaterevise\movies.xlsx' # 确保路径正确  
movies\_df = pd.read\_excel(file\_path)  
  
# 查看数据结构  
print(movies\_df.info()) # 查看每列的类型和是否有缺失值  
print(movies\_df.head()) # 查看每列的类型和是否有缺失值

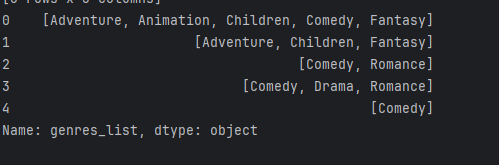
输出结果如下：



1. 确认数据没有缺失值或重复值后，进行预处理：将 genres 列的多类别拆分为单独类别（哑变量矩阵）并检查并处理可能存在的异常值或格式问题。

# 分解 genres 列  
movies\_df['genres\_list'] = movies\_df['genres'].str.split('|')  
# 检查分解后的结果  
print(movies\_df['genres\_list'].head())

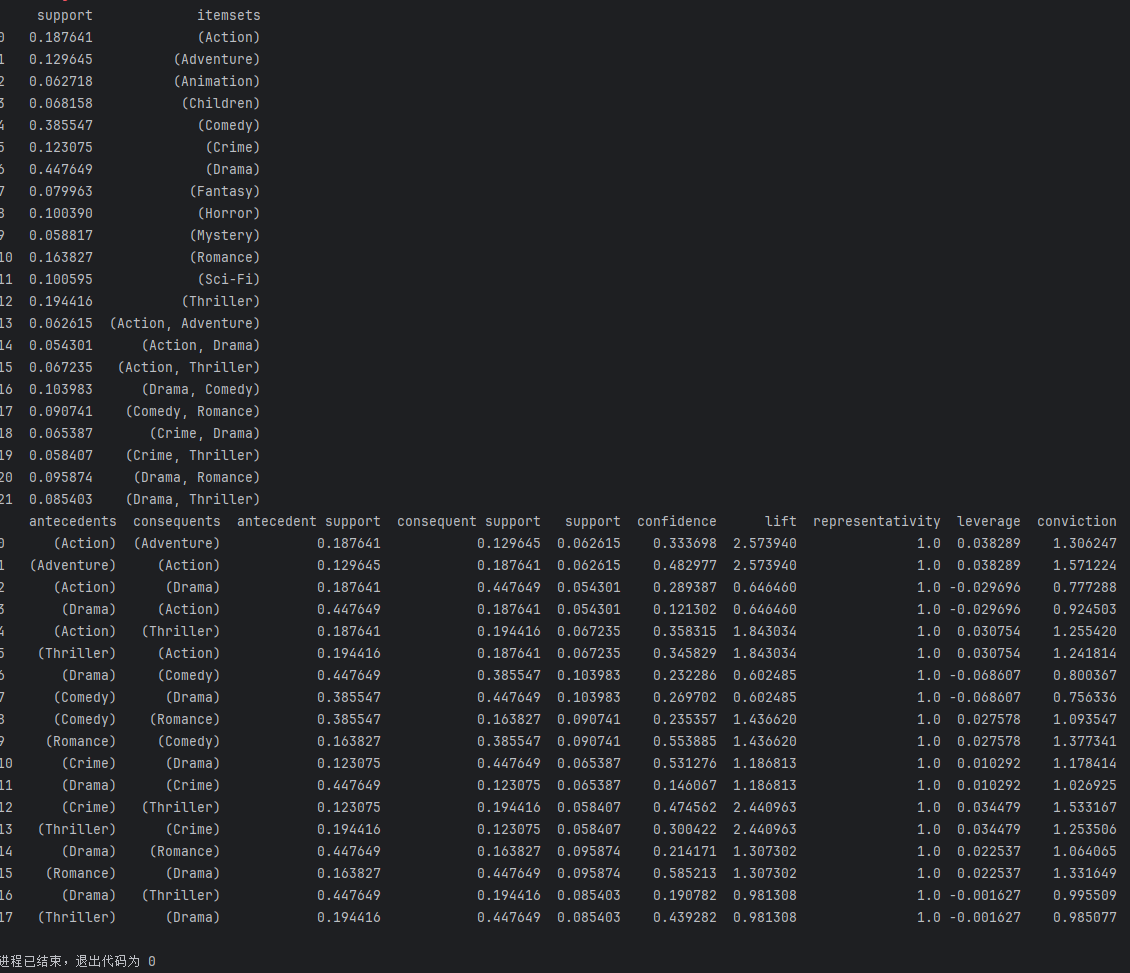
输出结果如下：



1. 使用分解后的 genres\_list 来生成频繁项集和关联规则

#生成频繁项集和关联规则  
from mlxtend.frequent\_patterns import apriori, association\_rules  
# 将 genres 转化为独热编码  
genres\_df = movies\_df['genres'].str.get\_dummies('|')  
# 使用 Apriori 算法生成频繁项集  
frequent\_itemsets = apriori(genres\_df, min\_support=0.05, use\_colnames=True)  
# 获取事务总数  
num\_itemsets = len(movies\_df)  
# 生成关联规则  
rules = association\_rules(frequent\_itemsets, num\_itemsets=num\_itemsets, metric="lift", min\_threshold=0.5)  
# 打印结果  
print(frequent\_itemsets)  
print(rules.to\_string())

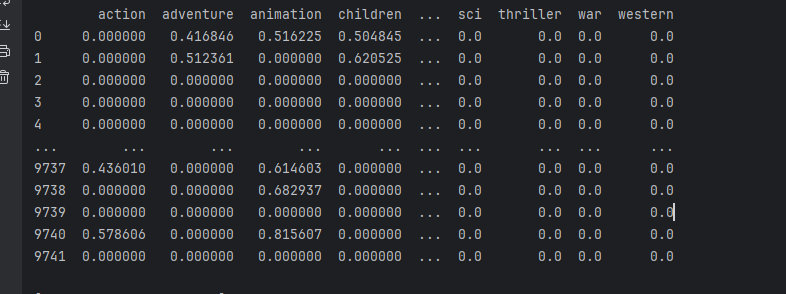
输出结果如下：



1. TD-IDF实现

#TF-IDF实现  
from sklearn.feature\_extraction.text import TfidfVectorizer  
  
# 之前进行了分解生成哑变量矩阵，通过以下步骤恢复 `genres` 列为原始格式  
movies\_df['genres'] = movies\_df['genres'].apply(lambda x: ' '.join(x) if isinstance(x, list) else str(x))  
genres\_str = movies\_df['genres']  
vectorizer = TfidfVectorizer(stop\_words=None)  
tfidf\_matrix = vectorizer.fit\_transform(genres\_str)  
  
# 查看 TF-IDF 结果  
tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), columns=vectorizer.get\_feature\_names\_out())  
print(tfidf\_df)

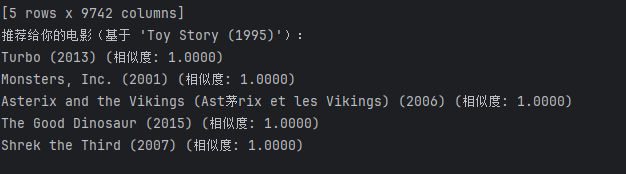
输出结果如下：



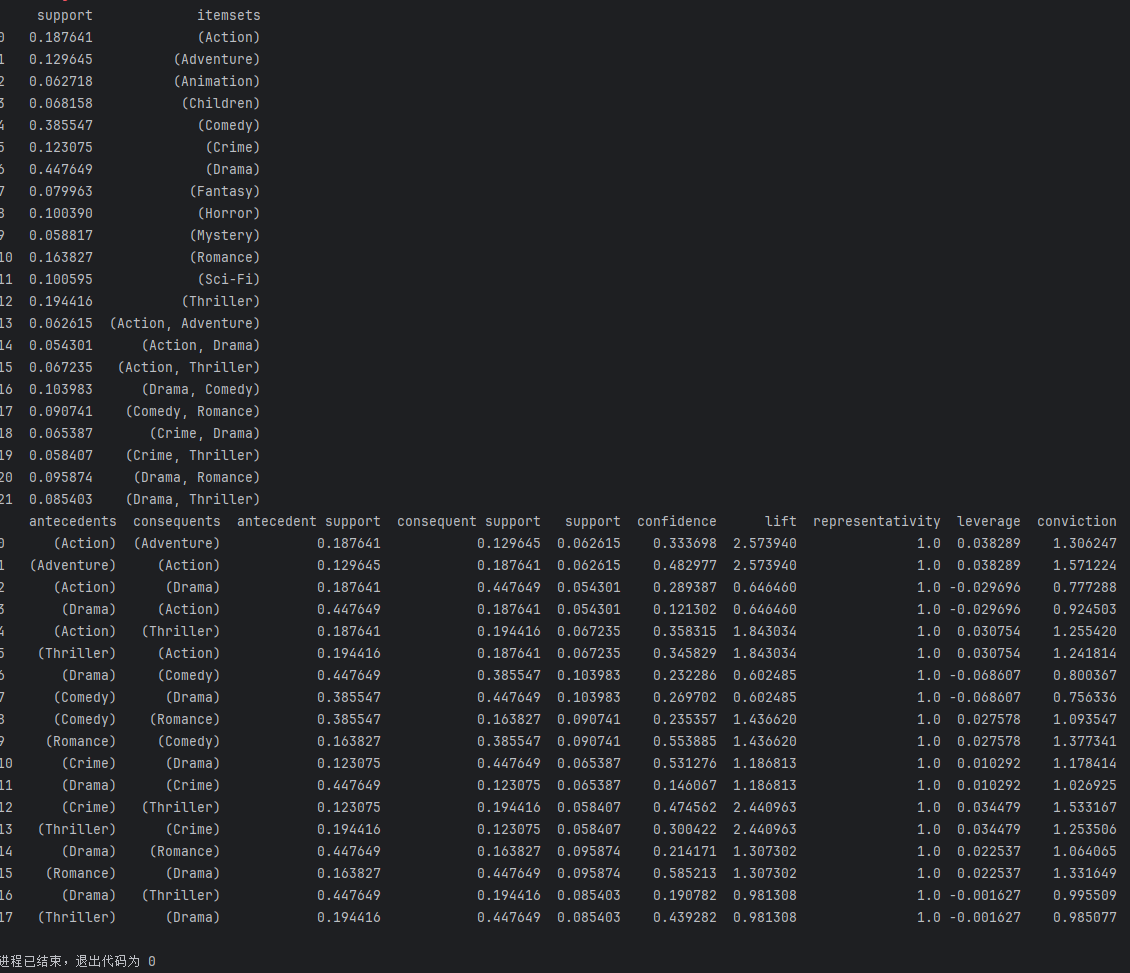
1. 完成基于电影内容的协同过滤推荐

from sklearn.metrics.pairwise import cosine\_similarity  
# 计算电影之间的余弦相似度  
cosine\_sim = cosine\_similarity(genres\_encoded)  
# 将相似度矩阵转换为DataFrame，方便查看  
cosine\_sim\_df = pd.DataFrame(cosine\_sim, index=movies\_df['title'], columns=movies\_df['title'])  
  
# 打印电影间的相似度矩阵（可以打印前几行查看）  
print(cosine\_sim\_df.head())  
# 建一个推荐函数，基于电影的名称，推荐最相似的电影  
def recommend\_movies(movie\_title, cosine\_sim\_df, top\_n=5):  
 # 获取指定电影的相似度评分  
 sim\_scores = cosine\_sim\_df.loc[movie\_title] # 使用 .loc 通过行索引访问电影的相似度评分  
 # 排序，得到与电影相似的其他电影的索引  
 sim\_scores = sim\_scores.sort\_values(ascending=False)  
 # 获取前top\_n个电影（除去自己）  
 similar\_movies = sim\_scores.iloc[1:top\_n + 1]  
 # 打印推荐的电影  
 print(f"推荐给你的电影（基于 '{movie\_title}'）：")  
 for idx, score in similar\_movies.items():  
 print(f"{idx} (相似度: {score:.4f})")  
# 举例：推荐与"Toy Story"相似的电影  
recommend\_movies('Toy Story (1995)', cosine\_sim\_df, top\_n=5)

输出结果如下：



**六、实验数据及处理结果**



**七、思考讨论题或体会或对改进实验的建议**

思考讨论题和建议：

频繁集生成和关联规则对于电影数据的分析有何重要性？可以讨论不同频繁项集之间的关联规则，以及这些规则如何帮助理解电影之间的关联性和用户喜好。

TF-IDF在电影推荐中的作用是什么？可以探讨TF-IDF如何帮助提取关键信息并衡量文本中的重要性，以及如何将其应用于电影类型数据的分析。

协同滤波推荐是一种常见的推荐算法，通过计算物品之间的相似度来进行推荐。可以讨论余弦相似度在推荐系统中的应用，以及如何根据相似度矩阵实现基于内容的推荐。

对改进实验的建议：

在频繁项集生成和关联规则推理中，可以尝试调整支持度和置信度的阈值，以获取不同级别的频繁项集和关联规则，从而更好地理解电影数据之间的关联关系。

在TF-IDF实现中，可以进一步探讨不同的文本处理技术，如词干提取、停用词过滤等，以提高TF-IDF的效果和准确性。

在协同滤波推荐部分，可以考虑引入其他相似度度量方法，如皮尔逊相关系数或欧氏距离，来比较不同推荐结果的准确性和多样性。

**八、参考资料**

1. Pang-Ning Tan, Michael Steinbach etc. Introduction to Data Mining, Second Edition, Pearson, 2019.
2. 李锐，李鹏，曲亚东等译. 机器学习实战. 高等教育出版社.2013.
3. <http://www.cse.msu.edu/~ptan/dmbook/software/>.（课本实验代码）