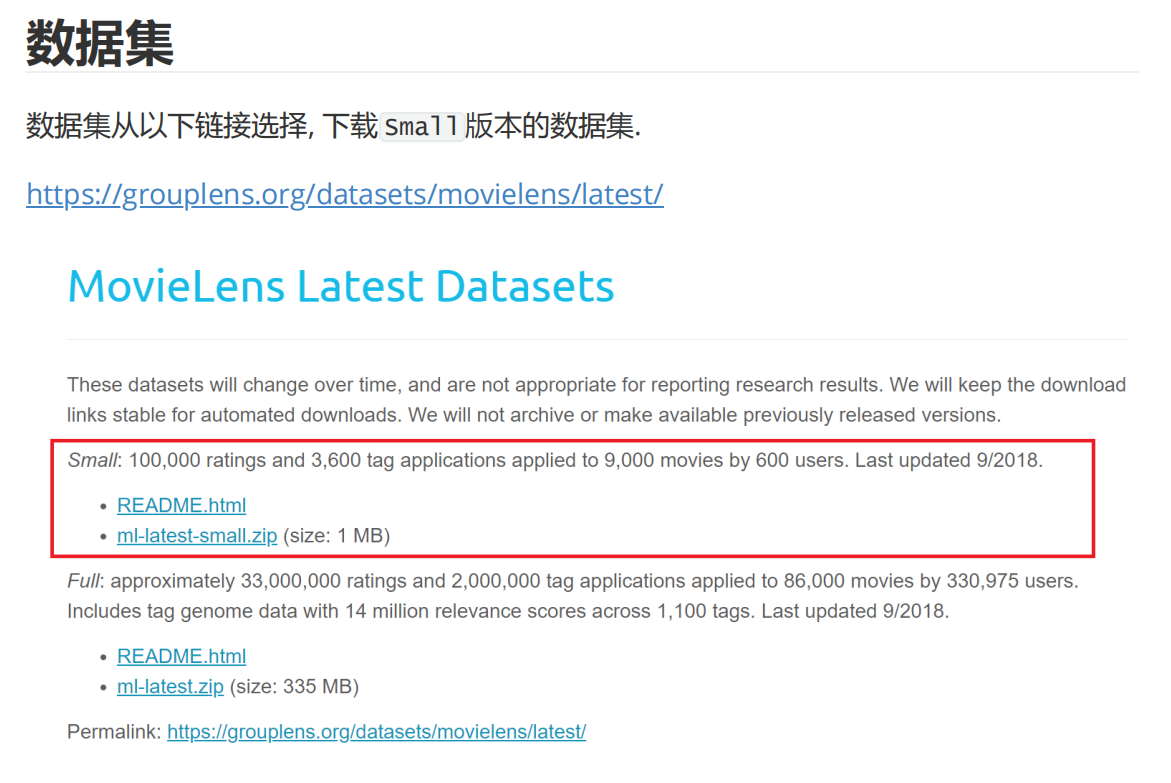
**推荐算法报告**

**学号: 3065645646 姓名: 张大炮 班级: 3班**

1. **数据**



1. 算法

基于KNN和余弦相似度的用户相似度计算，取把数据集分为0.7训练集，0.3测试集，推荐前50个估计分数最高的电影，最后计算召回率和精确率。

使用python surprise库进行计算

from surprise import Dataset

from surprise import Reader

from surprise import KNNBasic, KNNWithMeans

from surprise.model\_selection import train\_test\_split

from surprise import accuracy, Prediction

# 加载数据集

# data = Dataset.load\_builtin('ml-100k')

reader = Reader(line\_format='user item rating', sep=',', rating\_scale=(1.0, 5.0))

data = Dataset.load\_from\_file('./ml-latest-small/ratings.csv', reader)

# 创建协同过滤算法模型

sim\_options = {'name': 'cosine', 'user\_based': True}  # 使用余弦相似度进行用户间的相似度计算

model = KNNBasic(sim\_options=sim\_options)

# 划分数据集为训练集和测试集

trainset, testset = train\_test\_split(data, train\_size=0.7, test\_size=0.3, shuffle=True)

import numpy as np

# 训练模型

model.fit(trainset)

# 预测

predictions = model.test(testset)

计算总的召回率和精确度，MAE和RMSE

all\_precision = 0

all\_recall = 0

all\_rmse = 0

all\_mae = 0

cnt = 0

from tqdm import tqdm

for user\_id in tqdm(range(1, 611)):  # 目标用户ID

    user\_id = str(user\_id)

    testset\_user\_id\_movies = []

    for item in predictions:

        if item.uid == user\_id:

            testset\_user\_id\_movies.append(item)

    # print(testset\_user\_id\_movies)

    items\_to\_recommend = 50  # 推荐物品数量

    # 按实际值排序

    movies\_real = testset\_user\_id\_movies.copy()

    movies\_real = sorted(movies\_real, key=lambda x: x.r\_ui, reverse=False)

    movies\_real\_id = [int(item.iid) for item in movies\_real]

    movies\_real\_id = sorted(movies\_real\_id)

    # print(movies\_real\_id)

    # 按估计值排序

    movies\_est = testset\_user\_id\_movies.copy()

    movies\_est = sorted(movies\_est, key=lambda x: x.est, reverse=False)

    movies\_est\_id = [int(item.iid) for item in movies\_est[:items\_to\_recommend]]

    movies\_est\_id = sorted(movies\_est\_id)

    # print(movies\_est\_id)

    all\_mse += sum([(item.r\_ui - item.est)\*\*2 for item in movies\_est[:items\_to\_recommend]])

    all\_mae += sum([abs(item.r\_ui - item.est) for item in movies\_est[:items\_to\_recommend]])

    # 计算相同数量

    hit = 0

    index\_movies\_est\_id = 0

    index\_movies\_real\_id = 0

    length = min(len(movies\_est\_id), len(movies\_real\_id))

    cnt += length

    while index\_movies\_est\_id != length and index\_movies\_real\_id != length:

        if movies\_est\_id[index\_movies\_est\_id] == movies\_real\_id[index\_movies\_real\_id]:

            hit += 1

            index\_movies\_est\_id += 1

            index\_movies\_real\_id += 1

        elif movies\_est\_id[index\_movies\_est\_id] < movies\_real\_id[index\_movies\_real\_id]:

            index\_movies\_est\_id += 1

        else:

            index\_movies\_real\_id += 1

    # print(hit)

    precision = hit / len(movies\_est\_id)

    recall = hit / len(movies\_real\_id)

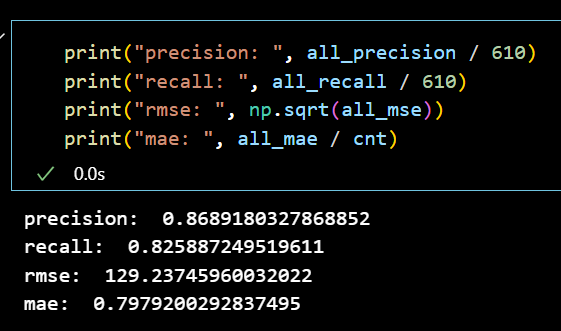
    all\_precision += precision

    all\_recall += recall

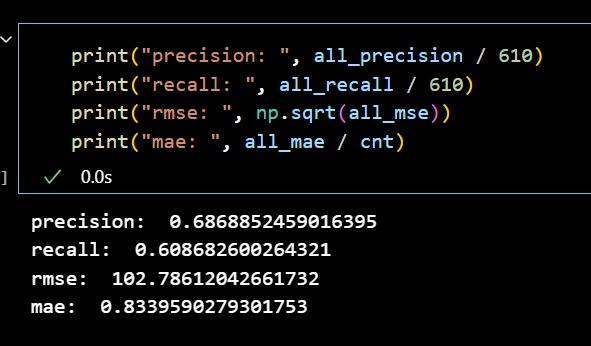
    # print(f"{user\_id} precision = {precision:3f}, recall = {recall:3f}")

1. 结果

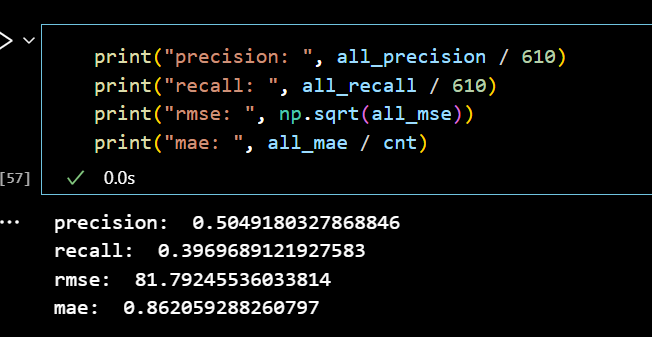
当推荐前50个（总量约为610\*50=30500）最佳评分电影时，精确度，召回率，RMSE和MAE为



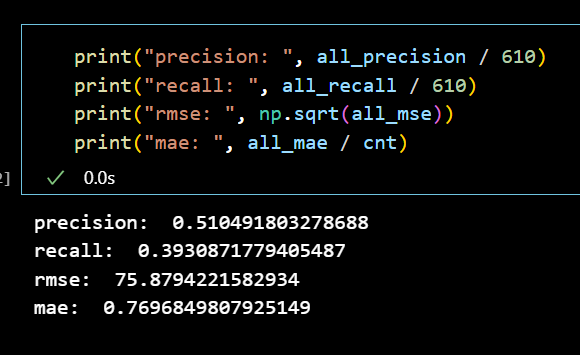
20（总量约为610\*20=12200）：



10（总量约为610\*10=6100）：



切换模型为KNNWithZScore即正态化的KNN模型，使用余弦相似度计算前10推荐电影：



MAE和RMSE都相对下降了，这是好的表现，但精确度和召回率没有太大变化，后续实验中使用pearson相关系数和msd相关，也没有得到更好结果。

1. 结论

可见模型的稳定性较好，但精确度堪忧，在无法在少量推荐的结果中得到较好的结果，但在大量推荐的情况下仍可以做到较高的精确度。