# A 程序代码

from collections import defaultdict

from surprise import Dataset, Reader

from surprise import Dataset

from surprise import SVD

from surprise.model\_selection import KFold

import os

import matplotlib.pyplot as plt

import numpy as np

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn import preprocessing

import pandas as pd

#设置中文编码

plt.rcParams['font.sans-serif']=['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

'''

构造用户评分矩阵

'''

#统计总用户数和总电影数,构造用户评分矩阵

def URmat\_fuc(rmu\_data ):

    sum\_users =rmu\_data.userID.unique().shape[0]

    sum\_movies =int( rmu\_data["movieID"].max())

    # print(sum\_users , sum\_movies)

    #创建用户评分矩阵

    UR\_mat = np.zeros((sum\_users,sum\_movies))

    for row in RMU\_data.itertuples() :

        UR\_mat[int(row[1])-1, int(row[2])-1] = int(row[3])

    return UR\_mat

def search\_name(movies\_data  , name , ru\_mat):

    title\_flag = input("search by title:")

    # #矩阵中用户总数  矩阵列 、行

    sum\_users = np.size(ru\_mat,1)

    sum\_movies = np.size(ru\_mat ,0)

    #保存平均评分

    aver\_lis = []

    for i in range(sum\_movies):

        movie\_averpoint = sum(ru\_mat[i])/sum\_users

        aver\_lis.append(movie\_averpoint)

    #movies\_data增加一列，最终输出 电影ID 、 电影标题 、 电影类别 、 电影平均评分

    movies\_data['averpoint'] = aver\_lis[:3883]

    #根据movies\_data查找具有该类别的电影，再根据平均评分排序截取前十数据返回

    #先查找电影名称、再查找电影类型，均无则返回错误告知

    if title\_flag == 'y'  or title\_flag=='Y':

        reco\_movies = movies\_data[movies\_data['title'].str.contains(pat=name)]

    else :

        reco\_movies = movies\_data[movies\_data['genres'].str.contains(pat=name)]

    reco\_movies=reco\_movies.sort\_values(by = 'averpoint' , ascending =False)[:10]

    return reco\_movies

def precision\_recall\_at\_k(predictions, k=10, threshold=3.5):

    '''返回为每个用户推荐k个物品时的准确率和召回率。'''

    # 首先将预测值映射至每个用户

    user\_est\_true = defaultdict(list)

    for uid, \_, true\_r, est, \_ in predictions:

        user\_est\_true[uid].append((est, true\_r))

    precisions = dict()

    recalls = dict()

    for uid, user\_ratings in user\_est\_true.items():

        # Sort user ratings by estimated value

        user\_ratings.sort(key=lambda x: x[0], reverse=True)

        # Number of relevant items

        n\_rel = sum((true\_r >= threshold) for (\_, true\_r) in user\_ratings)

        # Number of recommended items in top k

        n\_rec\_k = sum((est >= threshold) for (est, \_) in user\_ratings[:k])

        # Number of relevant and recommended items in top k

        n\_rel\_and\_rec\_k = sum(((true\_r >= threshold) and (est >= threshold))

                              for (est, true\_r) in user\_ratings[:k])

        # Precision@K: Proportion of recommended items that are relevant

        precisions[uid] = n\_rel\_and\_rec\_k / n\_rec\_k if n\_rec\_k != 0 else 1

        # Recall@K: Proportion of relevant items that are recommended

        recalls[uid] = n\_rel\_and\_rec\_k / n\_rel if n\_rel != 0 else 1

    return precisions, recalls

def SVD\_rec( ):

    # 指定文件所在路径

    file\_path = os.path.expanduser('ratings.csv')

    # 告诉文本阅读器，文本的格式是怎么样的

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

    # 加载数据

    data = Dataset.load\_from\_file(file\_path, reader=reader)

    #十折交叉验证，n\_splits=n 就是n折

    kf = KFold(n\_splits=10)

    algo = SVD()

    #十组测试集中每组的平均准确率

    accuracy = []

    recall = []

    #将原始数据分成10等分，进行十折交叉验证

    for trainset, testset in kf.split(data):

        #对训练集使用SVD算法进行训练

        algo.fit(trainset)

        #使用训练好的SVD算法获得测试集的预测值

        predictions = algo.test(testset)

        #计算该组的所有测试集用户的准确率和召回率

        precisions, recalls = precision\_recall\_at\_k(predictions, k=5, threshold=4)

        # 计算测试集的平均准确率和召回率

        accuracy.append( sum(prec for prec in precisions.values()) / len(precisions) )

        recall.append( sum(rec for rec in recalls.values()) / len(recalls) )

    return accuracy ,recall

def new\_userreco(users\_data , loginID):

    gender\_map = {'F':0 , 'M':1}

    users\_data['gender'] = users\_data['gender'].map(gender\_map)

    users\_data = users\_data.drop(['zip-code'],axis=1)

    users\_mat = users\_data.iloc[:,:].values

    users\_mat=preprocessing.scale(users\_mat)

    #构造相似度矩阵

    newuser\_similar = cosine\_similarity(users\_mat , dense\_output=True)

    NU\_silimar\_df = pd.DataFrame(newuser\_similar)

    #截取该用户与其他用户的相似度

    user\_similar=NU\_silimar\_df.iloc[loginID-1]

    #user\_similar中的列名称即为用户ID-1 ， 先获取用户ID-1

    user\_similar\_dict = user\_similar.to\_dict()

    #与logID用户相似的用户存放字典中，再根据字典中的value值排序取其前三

    user\_similar\_dict = sorted(user\_similar\_dict.items() , key=lambda x: x[1] , reverse=True )[1:4]

    #记录当下推荐的电影数

    sum\_recmovies = 0

    #新建推荐电影ID空字典

    rec\_moviesID  = []

    for x in range(3):

        #获取的是相似用户ID-1

        similar\_userID = user\_similar\_dict[x][0]

        #当下第i个相似用户的相似度

        simila = user\_similar\_dict[x][1]

        #在用户评分矩阵中，该相似用户所在的行就是相似用户ID-1=similar\_userID

        similuser\_rating = UR\_mat[similar\_userID]

        #推荐10部电影，从当前第i个相似用户的用户评分表中找到看过的电影，并计算电影推荐得分，公式：用户相似度\*0.7+电影评分\*0.3

        for i in range(3882):

            if sum\_recmovies < 10:

                #矩阵是从0开始，真实movieID要加1

                if similuser\_rating[i] != 0:

                    score = simila\*0.7+0.3\*similuser\_rating[i]

                    rec\_moviesID.append(([i+1] , score))

                    sum\_recmovies +=1

                    #每次加入新推荐电影，就对字典排序，为推荐电影满10个时插入更高评分的电影准备

                    rec\_moviesID =  sorted(rec\_moviesID, key=lambda x: x[1] , reverse=True )

            else:

                if similuser\_rating[i] != 0:

                    score = simila\*0.7+0.3\*similuser\_rating[i]

                    if rec\_moviesID[9][1] < score:

                        #删除字典中电影得分最少的元组，加入评分更高的元组，使推荐电影的评分在所有的电影中都是最高的

                        rec\_moviesID[9]=([i+1] , score)

                        rec\_moviesID = sorted(rec\_moviesID , key=lambda x: x[1] , reverse=True )

    #以上推荐的10部电影ID获取完成

    #根据推荐电影ID获取电影详细信息存储，并输出

    rec\_movies = []

    for i in range(len(rec\_moviesID)):

        moviesID = rec\_moviesID[i][0]

        moviesID =moviesID[0]-1

        rec\_movie = list(movies\_dat.iloc[moviesID])

        rec\_movies.append(rec\_movie)     # 电影信息

        #添加电影评分

        col = UR\_mat[[moviesID]]

        rec\_movie.append( col[(0<col)].mean() )

    rec\_movies=rec\_movies[:10]

    rec\_movies = sorted(rec\_movies , key=lambda x: x[3] , reverse=True )

    return   rec\_movies

#画图

def show\_acc\_recall(accuracy ,  recall):

    #准确率和召回率折线图

    plt.plot(accuracy, marker = 'o', label = '准确率')

    plt.plot(recall, marker = 'x', label = '召回率')

    plt.xlabel('测试集')

    plt.ylabel('百分比')

    plt.title('准确率和召回率折线图')

    plt.legend()

    plt.show()

    #十折交叉验证的平均准确率

    name\_list = ['平均准确率' , '平均召回率']

    num\_list = [sum(accuracy)/len(accuracy) , sum(recall)/len(recall)]

    plt.bar(range(len(num\_list)), num\_list,color='rgb',tick\_label=name\_list)

    plt.show()

#显示电影信息

def show\_detail(arr\_movie):

    for movie in arr\_movie:

        print(movie,'\n')

if \_\_name\_\_ == "\_\_main\_\_":

    #读取文件数据，三表合一，将性别转换为数字，去除 时间戳 、电影名称、邮编属性

    movies\_dat = pd.read\_csv('movies.CSV' , sep=","  , names=["movieID" , "title" , "genres"])

    ratings\_dat = pd.read\_csv('ratings.CSV' , sep="," , names=["userID" , "movieID" , "rating" , "timestamp"])

    users\_dat = pd.read\_csv('users.CSV' , sep="," , names=["userID" , "gender" , "age" , "occupation" , "zip-code"])

    #读取处理好的文件

    RMU\_data = pd.read\_csv('new\_RMU.CSV' , sep=',',names =['userID' , 'movieID' , 'rating ' , 'genres'  , 'gender' , 'age' , 'occupation'] )

    #获取前2000条用户数据

    UR\_mat=URmat\_fuc(RMU\_data)

    if input("搜索电影名称或电影类型（y）?：") == 'y':

        name = input("请输入要搜索的内容：")

        rec\_movies=search\_name(movies\_dat  , name ,UR\_mat)

        print(rec\_movies)

    elif input("新用户（y）？：") == 'y':

        loginID = input("请输入你的登入ID：")

        reco\_movie=new\_userreco(users\_dat , int(loginID))

        show\_detail(reco\_movie)

    #使用SVD算法，计算准确度和召回率并画图

    else:

        fig , ax = plt.subplots(1,1)

        ax.set\_title('相似度矩阵可视化' , fontsize = 12)

        # ax.plot(x,y,label='userID')

        # ax.ylabel('userID')

        ax.matshow(UR\_mat)

        plt.show()

        accuracy , recall =SVD\_rec()

        show\_acc\_recall(accuracy , recall)