```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan  2 20:36:48 2018

@author: lanlandetian
"""

import UserCF
import UserCF_IIF
import ItemCF
import ItemCF_IUF
import random
import Evaluation
import LFM

import imp
imp.reload(UserCF)
imp.reload(ItemCF)
imp.reload(ItemCF_IUF)
imp.reload(Evaluation)
imp.reload(LFM)


def readData():
    data = []
    fileName = './u.data'
    fr = open(fileName,'r')
    for line in fr.readlines():
        lineArr = line.strip().split()
        data.append([lineArr[0], lineArr[1], 1.0])
    return data


def SplitData(data,M,k,seed):
    test = []
    train = []
    random.seed(seed)
    for user, item,rating in data:
        if random.randint(0,M-1) == k:
            test.append([user,item,rating])
        else:
            train.append([user, item,rating])
```

```
    return train, test


# 将列表形式数据转换为dict形式
def transform(oriData):
    ret = dict()
    for user,item,rating in oriData:
        if user not in ret:
            ret[user] = dict()
        ret[user][item] = rating
    return ret

if __name__ == '__main__':
    data = readData()
    numFlod = 5
    precision =0
    recall = 0
    coverage = 0
    popularity =0
    for i in range(0,numFlod):
        [oriTrain,oriTest] = SplitData(data,numFlod,i,0)
        train = transform(oriTrain)
        test = transform(oriTest)

        W = UserCF.UserSimilarity(train)
        rank = UserCF.Recommend('1',train,W)
        result = UserCF.Recommendation(test.keys(), train, W)

        # W = UserCF_IIF.UserSimilarity(train)
    #    rank = UserCF_IIF.Recommend('1',train,W)
        # result = UserCF_IIF.Recommendation(test.keys(), train, W)

    #    W = ItemCF.ItemSimilarity(train)
    #    rank = ItemCF.Recommend('1',train,W)
#        result =  ItemCF_IUF.Recommendation(test.keys(),train, W)

#        W = ItemCF_IUF.ItemSimilarity(train)
    #    rank = ItemCF_IUF.Recommend('1',train,W)
#        result =  ItemCF_IUF.Recommendation(test.keys(),train, W)

#        [P,Q] = LFM.LatentFactorModel(train, 10,30, 0.02, 0.01)
#        rank = LFM.Recommend('2',train,P,Q)
```

```
                                                mainCFuser
#       result = LFM.Recommendation(test.keys(), train,P,Q)


        N = 10
        precision += Evaluation.Precision(train,test, result,N)
        recall += Evaluation.Recall(train,test,result,N)
        coverage += Evaluation.Coverage(train, test, result,N)
        popularity += Evaluation.Popularity(train, test, result,N)

    precision /= numFlod
    recall /= numFlod
    coverage /= numFlod
    popularity /= numFlod


     #输出结果
    print('user based method results are below:')
    print('precision = %f' %precision)
    print('recall = %f' %recall)
    print('coverage = %f' %coverage)
    print('popularity = %f' %popularity)
```