《机器学习》

-推荐系统实验手册



华为技术有限公司

目录

[1 实验介绍 2](#_Toc55223708)

[1.1 实验目的 2](#_Toc55223709)

[1.2 实验清单 2](#_Toc55223710)

[1.3 开发平台介绍 3](#_Toc55223711)

[1.4 背景知识 3](#_Toc55223712)

[1.4.1 协同过滤算法介绍 3](#_Toc55223713)

[1.4.2 基于物品的协同过滤算法 4](#_Toc55223714)

[2 电影推荐实验 6](#_Toc55223715)

[2.1 实验介绍 6](#_Toc55223716)

[2.1.1 简介 6](#_Toc55223717)

[2.1.2 实验目的 6](#_Toc55223718)

[2.2 实验环境要求 6](#_Toc55223719)

[2.3 实验总体设计 6](#_Toc55223720)

[2.4 实验过程 6](#_Toc55223721)

[2.4.1 代码和数据准备 6](#_Toc55223722)

[2.4.2 查看数据 7](#_Toc55223723)

[2.4.3 数据预处理 9](#_Toc55223724)

[2.4.4 创建相关矩阵进行电影推荐 10](#_Toc55223725)

[2.4.5 保存结果至OBS 12](#_Toc55223726)

[2.5 实验小结 13](#_Toc55223727)

[2.6 思考题参考答案 13](#_Toc55223728)

[2.6.1 查看用户的评分信息 13](#_Toc55223729)

[2.6.2 查看电影信息 14](#_Toc55223730)

# 实验介绍

推荐系统（RS）主要是指应用协同智能（collaborative intelligence）做推荐的技术。推荐系统的两大主流类型是基于内容的推荐系统和协同过滤（Collaborative Filtering）。另外还有基于知识的推荐系统（包括基于本体和基于案例的推荐系统）是一类特殊的推荐系统，这类系统更加注重知识表征和推理。

本章实验将会学习使用协同过滤技术分析用户对电影的评分数据，并基于这个数据建立一个推荐系统，根据用户输入的一部感兴趣的电影，为其推荐其他可能感兴趣的电影。此案例中，我们使用的数据集是用户对电影的评分数据，包含用户数据、评分数据、电影数据。

本章实验难度：中级。

中级实验：①电影推荐案例实验

## 实验目的

本案例将掌握如何使用机器学习算法全流程构建一个电影推荐系统的方案。通过本实验的学习，您将能够：

掌握如何使用华为云ModelArts Notebook上传数据、执行Python代码；

掌握如何载入、查阅、清洗、合并用户的数据，并计算物品相似度矩阵；

掌握如何使用物品的相似度矩阵，为客户进行推荐其他物品。

## 实验清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| 电影推荐案例 | 基于电影评分数据，使用推荐算法进行电影推荐 | 中级 | Python3 | ModelArts |

## 开发平台介绍

ModelArts是面向开发者的一站式AI开发平台，为机器学习与深度学习提供海量数据预处理及半自动化标注、大规模分布式Training、自动化模型生成，及端-边-云模型按需部署能力，帮助用户快速创建和部署模型，管理全周期AI工作流。

具体内容请参考平台介绍PPT。

## 背景知识

### 协同过滤算法介绍

协同过滤(Collaborative Filtering)作为推荐算法中最经典的类型，包括在线的协同和离线的过滤两部分。所谓在线协同，就是通过在线数据找到用户可能喜欢的物品，而离线过滤，则是过滤掉一些不值得推荐的数据，比如推荐值评分低的数据，或者虽然推荐值高但是用户已经购买的数据。

协同过滤的模型一般为m个物品，n个用户的数据，只有部分用户和部分数据之间是有评分数据的，其它部分评分是空白，此时我们要用已有的部分稀疏数据来预测那些空白的物品和数据之间的评分关系，找到最高评分的物品推荐给用户。

一般来说，协同过滤推荐分为三种类型。

第一种是基于用户(user-based)的协同过滤

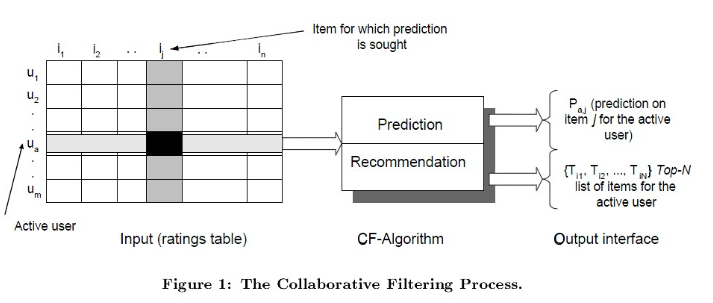
第二种是基于物品(item-based)的协同过滤

第三种是基于模型(model based)的协同过滤

基于用户(user-based)的协同过滤主要考虑的是用户和用户之间的相似度，只要找出相似用户喜欢的物品，并预测目标用户对对应物品的评分，就可以找到评分最高的若干个物品推荐给用户。

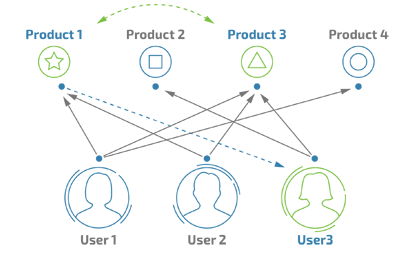
而基于物品(item-based)的协同过滤和基于用户的协同过滤类似，只不过这时我们转向找到物品和物品之间的相似度，只有找到了目标用户对某些物品的评分，那么我们就可以对相似度高的类似物品进行预测，将评分最高的若干个相似物品推荐给用户。比如你在网上买了一本机器学习相关的书，网站马上会推荐一堆机器学习，大数据相关的书给你，这里就明显用到了基于物品的协同过滤思想。

基于用户的协同过滤和基于物品的协同过滤对比：基于用户的协同过滤需要在线找用户和用户之间的相似度关系，计算复杂度肯定会比基于基于物品的协同过滤高。但是可以帮助用户找到新类别的有惊喜的物品。而基于物品的协同过滤，由于考虑的物品的相似性一段时间不会改变，因此可以很容易的离线计算，准确度一般也可以接受，但是推荐的多样性来说，就很难带给用户惊喜了。



### 基于物品的协同过滤算法

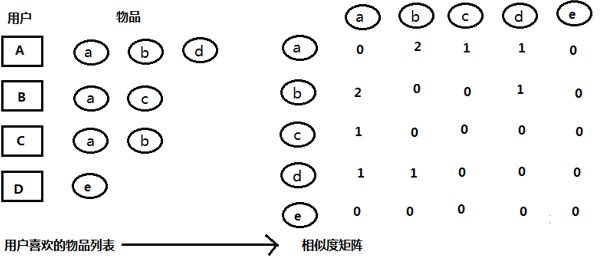
假设某天你看了电影《蚁人》，那么影院的APP可能会给你推荐电影《复仇者联盟》。因为机器经过判断得出这两者相似度很高，你既然会喜欢《蚁人》那么喜欢《复仇者联盟》的概率会很大。因此，**基于物品的协同过滤算法就是给用户推荐那些和他们之前喜欢的物品相似的物品。由于本实验采用的是基于物品的协同过滤算法，所以将着重介绍此算法。而基于**用户的协同过滤与基于模型的协同过滤可自行查阅资料。



基于物品的协同过滤

**计算物品间的相似度：这里我们使用余弦相似度算法（Ochiai coefficient落和系数）。**

**倒排：过滤出完全没关系的物品和。**



倒排

通过计算得到 =0.82，=0.58，=0.58

计算用户对物品的兴趣度

用户对物品的感兴趣程度计算公式：

此公式与上面基于用户的公式非常相像。表示与j物品最相似的个物品的集合。表示用户喜欢的物品集合。表示物品之间的相似度。是用户对物品的兴趣。（对于隐反馈数据集，如果用户对物品有过行为，即可令=1。）

**4. Top-N分析**  
同上，计算出之后，对其排序，取前几名作为推荐物品推荐给用户。

# 电影推荐实验

## 实验介绍

### 简介

本实验用协同过滤技术分析用户对电影的评分数据，并基于这个数据建立一个推荐系统，根据用户输入的一部感兴趣的电影，为其推荐其他可能感兴趣的电影。此案例中，我们使用的数据集是用户对电影的评分数据，包含用户数据、评分数据、电影数据。

### 实验目的

掌握协同过滤算法的基本原理。

掌握推荐系统的整体流程。

## 实验环境要求

华为云ModelArts

## 实验总体设计

## 实验过程

### 代码和数据准备

案例配置信息填写

案例中需要将运行结果上传至OBS中，因此，在实验之前需创建自己的obs桶，并且桶所在区域与所创建的notebook属于一个区域。我们需要设置以下相关参数（使用自己真实的桶名和唯一ID替换掉\*号）：

BUCKET\_NAME ： 自己的OBS桶名

UNIQUE\_ID : 唯一ID，填写自己的学号或者IAM子账号名称相关代码、数据和模型都已准备好存放在OBS中，执行下面一段代码即可将其拷贝到Notebook中。

BUCKET\_NAME = '\*\*\*\*\*\*\*\*\*\*\*'

#UNIQUE\_ID = '\*\*\*\*\*\*\*\*\*\*\*'

#OBS\_BASE\_PATH = BUCKET\_NAME + '/' + UNIQUE\_ID

OBS\_BASE\_PATH = BUCKET\_NAME

初始化ModelArts SDK

from modelarts.session import Session

session = Session()

下载源代码和数据

这一步准备案例所需的源代码和数据，相关资源已经保存在OBS中，我们通过ModelArts SDK将资源下载到本地，并解压到当前目录下。解压后，当前目录包含ml-100k目录，存有数据集。

session.download\_data(bucket\_path="ai-course-common-20-bj4/movie\_recommendation/movie\_recommendation.tar.gz", path="./movie\_recommendation.tar.gz")

# 使用tar命令解压资源包

!tar xf movie\_recommendation.tar.gz

输出结果：



导入基本工具库

执行下面方框中的这段代码，可以导入本次实验中使用的Python开发基本工具库。numpy是数据分处理工具,pandas是文件读取和数据处理工具，scipy是一个科学计算库，这里导入了cosine, correlation两种距离计算方法。此段代码只是引入Python包，无回显（代码执行输出）。

# import same usefull libraries

import numpy as np

import pandas as pd

from sklearn.metrics import pairwise\_distances

from scipy.spatial.distance import cosine, correlation

### 查看数据

使用pandas库导入用户的个人信息

用户数据的字段描述如下：

user\_id：用户ID

age：用户年龄

sex：性别

occupation：职业

zip\_code：邮编

评分数据的字段描述如下：

user\_id：用户ID

movide\_id：电影ID

rating：评分

unix\_tiemstamp：评分时间

电影数据的字段描述如下：

user\_id：用户ID

movide\_id：电影ID

rating：评分

unix\_tiemstamp：评分时间

查看用户信息

（1）定义用户信息

# 用户信息

users\_cols = ['user\_id', 'age', 'sex', 'occupation', 'zip\_code']

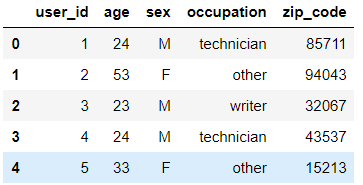
users = pd.read\_csv('./ml-100k/u.user', sep='|', names=users\_cols, parse\_dates=True)

（2）打印用户信息

打印前5个用户的个人信息，可以看到用户个人信息包含用户ID（user\_id）、年龄(age)、性别(sex)、职业(occupation)、邮编(zip\_code)。

users.head()

输出结果：



前5个用户的个人信息

（3）打印数据表格的大小，可以看到这是一个 943x5的矩阵， 其中943代表有943个用户，5代表每个用户有5项信息。

users.shape

输出结果：



查看用户的评分信息

参照查看用户信息的实现，请自行编码实现以下步骤：

（1）使用Pandas库导入用户的评分信息。

（2）打印前5个评分信息，可以看到评分信息包含用户ID（user\_id）、电影ID(movide\_id)、评分(rating)、评分时间(unix\_tiemstamp)。

（3）打印数据表格的大小，可以看到这是一个 10000x4的矩阵， 其中10000代表有10000条评论，4代表每个评论有5项信息。

查看电影信息

参照查看用户信息的实现，请自行编码实现以下步骤：

（1）使用pandas库导入电影的信息。

（2）打印前5个电影信息，可以看到电影信息包含电影ID(movide\_id)、电影名称(title)、发布时间(release\_date)、视频发布时间(video\_release\_date)、评论网站URL链接(imdb\_url)。

（3）打印数据表格的大小，可以看到这是一个 1682x5的矩阵， 其中1682代表有1682部电影，5代表每部电影有5项信息。

### 数据预处理

数据合并

（1）把电影数据表、评论数据表、用户信息数据表进行合并，最后得到一张数据信息总表。

# Merging movie data with their ratings

movie\_ratings = pd.merge(movies, ratings)

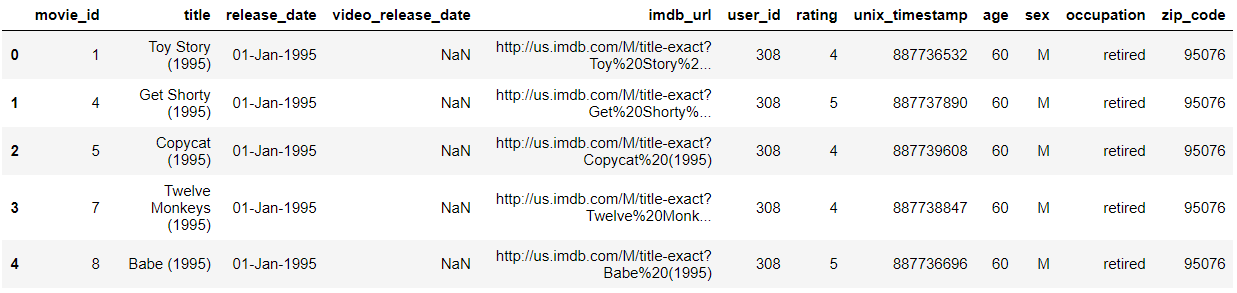
# merging movie\_ratings data with the User's dataframe

df = pd.merge(movie\_ratings, users)

（2）查看数据信息总表

df.head()

输出结果：



前5个用户的数据信息总表

（3）打印数据总表的大小，可以看到这是一个 10000x12的矩阵， 其中10000代表有10000条评论，12代表每条评论有12项属性，包括电影ID，电影信息，用户ID，评分，用户信息等。

df.shape

输出结果：



数据清洗

（1）去除一些无效或不需要的信息，比如video\_release\_date、imdb\_url、unix\_timestamp。

# pre-processing

# dropping colums that aren't needed

df.drop(df.columns[[3, 4, 7]], axis=1, inplace=True)

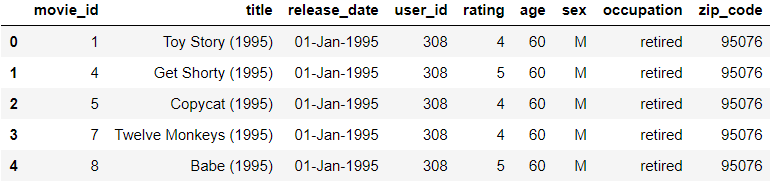
ratings.drop("unix\_timestamp", inplace=True, axis=1)

movies.drop(movies.columns[[3, 4]], inplace=True, axis=1)

（2）查看新的数据信息总表。

df.head()

输出结果：



前5个用户的新数据信息总表

### 创建相关矩阵进行电影推荐

创建用户-电影评分矩阵

（1）根据评分数据表(ratings),创建用户-电影评分矩阵。

# Pivot Table(This creates a matrix of users and movie\_ratings)

ratings\_matrix = ratings.pivot\_table(index=['movie\_id'], columns=['user\_id'], values='rating').reset\_index(drop=True)

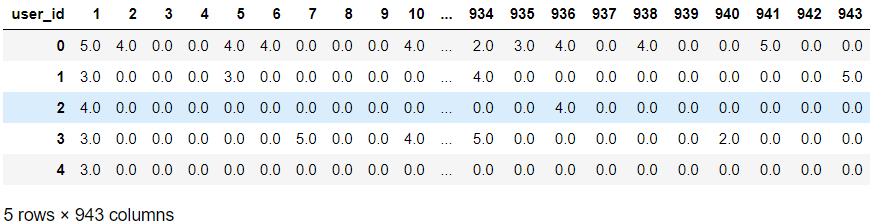
ratings\_matrix.fillna(0, inplace=True)

cmu = ratings\_matrix

（2）查看用户-电影评分矩阵，其中每一行代表每部电影来自所有用户的评分；每一列代表每个用户对所有电影的评分。

cmu.head()

输出结果：



用户-电影评分矩阵

打印数据总表的大小，可以看到这是一个 1682x943的矩阵， 其中1682代表有1682部电影，943代表有943个用户。

创建电影的相似矩阵

（1）根据943位用户对每部电影的评分，创建1682部电影的相似矩阵，矩阵大小为1682x1682的形状。

# Cosine Similarity(Creates a cosine matrix of similaraties ..... which is the pairwise distances

# between two items )

​

movie\_similarity = 1 - pairwise\_distances(ratings\_matrix.values, metric="cosine")

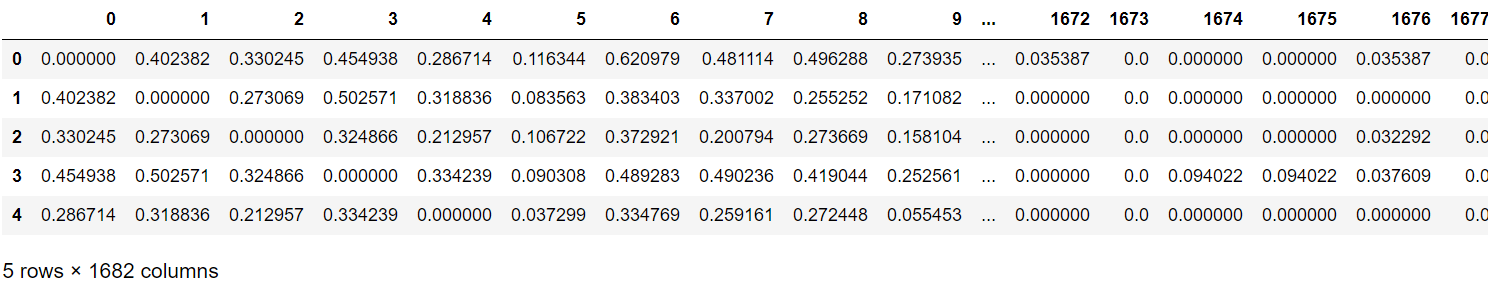
np.fill\_diagonal(movie\_similarity, 0)

ratings\_matrix = pd.DataFrame(movie\_similarity)

（2）查看电影相似矩阵，以第三行，第二列为例，数值为0.273069，这个值代表第二部电影与第三部电影的相似度。

ratings\_matrix.head()

输出结果：



电影相似矩阵

（3）查看形状

ratings\_matrix.shape

输出结果：



根据电影的相似矩阵，推荐电影

（1）当用户查看了 Copycat (1995)，那么根据电影的相似矩阵，推荐与 Copycat (1995) 近似分数比较高的电影。具体如下：根据电影名 Copycat (1995)， 查询电影信息表 (movies)中的index序号。

# user\_inp=input('Enter the reference movie title based on which recommendations are to be made: ')

user\_inp = "Copycat (1995)"

inp = movies[movies['title'] == user\_inp].index.tolist()

inp = inp[0]

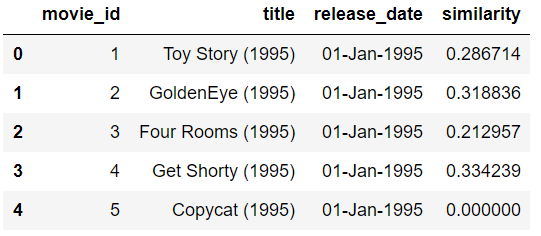
（2）根据电影index序号，去查电影的相似矩阵，得到1682部电影的相似值，并打印表格中的5部电影的相似值。

movies['similarity'] = ratings\_matrix.iloc[inp]

movies.columns = ['movie\_id', 'title', 'release\_date', 'similarity']

movies.head(5)

输出结果：



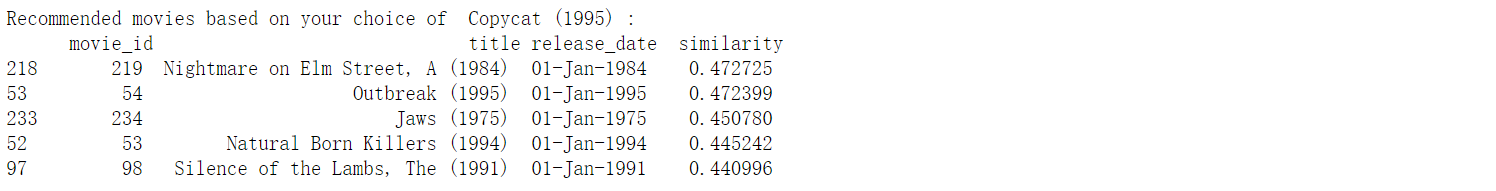
5部电影的相似值

（3）把相似值进行排序，并打印最相似的5部电影

recommended\_movies = movies.sort\_values(["similarity"], ascending=False)[1:6]

​print("Recommended movies based on your choice of ", user\_inp, ": \n", recommended\_movies)

输出结果：



最相似的5部电影

### 保存结果至OBS

我们将推荐的前5部电影的信息保存到文本文件中，并上传到OBS，以便以后查看。

写入本地文件

将电影的信息写入到文本文件中。会打印成功保存的信息。

import os

​if not os.path.exists('results'):

os.mkdir('results') # 创建本地保存路径

​with open('./results/recommended\_movies.txt', 'w') as f:

f.write(str(recommended\_movies)) # 写入本地文本文件

print('Successfully saved!')

输出结果：

Successfully saved!

上传文件至OBS

使用ModelArts SDK上传本地文件至OBS。可以看到上传成功的日志。

session.upload\_data(bucket\_path=OBS\_BASE\_PATH + '/movie\_recommendation/results/', path='./results/recommended\_movies.txt')

输出结果：

Successfully upload file ./results/recommend\_movies.txt to OBS professional-construction/movie\_recommendtion/results

注意：该案例所在的OBS存储路径下，results目录下，有模型文件recommended\_movies.txt。

## 实验小结

本实验基于ModelArts平台使用协同过滤算法中基于物品的协同过滤实现了电影推荐案例。

## 思考题参考答案

### 查看用户的评分信息

（1）使用Pandas库导入用户的评分信息

# Ratings

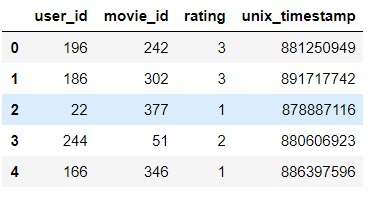
rating\_cols = ['user\_id', 'movie\_id', 'rating', 'unix\_timestamp']

ratings = pd.read\_csv('./ml-100k/u.data', sep='\t', names=rating\_cols)

（2）打印前5个评分信息，可以看到评分信息包含用户ID（user\_id）、电影ID(movide\_id)、评分(rating)、评分时间(unix\_tiemstamp)

ratings.head()

输出结果：



前5个用户的评分信息

（3）打印数据表格的大小，可以看到这是一个 10000x4的矩阵， 其中10000代表有10000条评论，4代表每个评论有5项信息

ratings.shape

输出结果：



### 查看电影信息

（1）使用pandas库导入电影的信息

# Movies

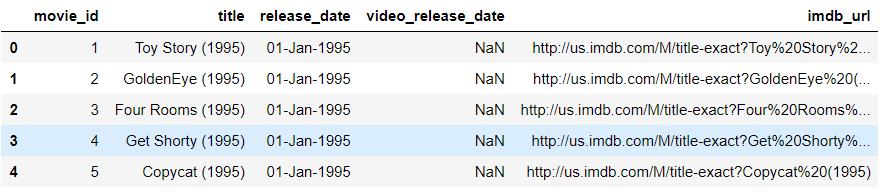
movie\_cols = ['movie\_id', 'title', 'release\_date', 'video\_release\_date', 'imdb\_url']

movies = pd.read\_csv('./ml-100k/u.item', sep='|', names=movie\_cols, usecols=range(5), encoding='latin-1')

（2）打印前5个电影信息，可以看到电影信息包含电影ID(movide\_id)、电影名称(title)、发布时间(release\_date)、视频发布时间(video\_release\_date)、评论网站URL链接(imdb\_url)

movies.head()

输出结果：



前5部电影信息

（3）打印数据表格的大小，可以看到这是一个 1682x5的矩阵， 其中1682代表有1682部电影，5代表每部电影有5项信息

movies.shape

输出结果：

