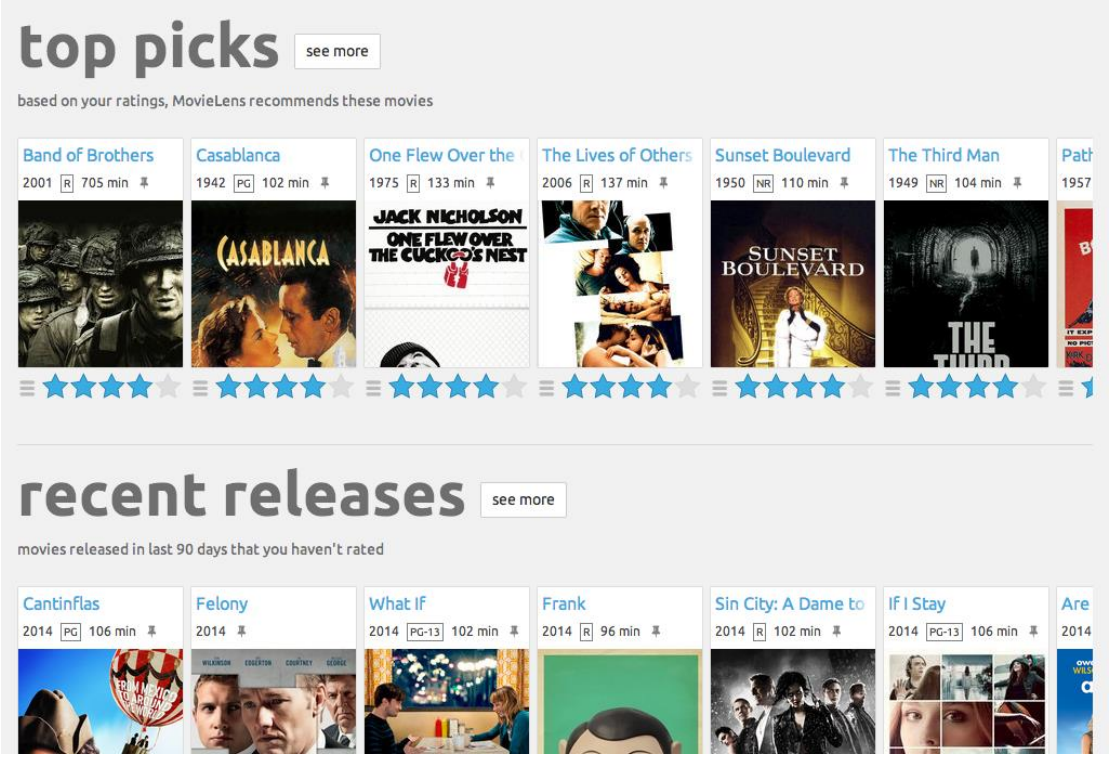


构建基于协同过滤的电影推荐系统

——基于 MovieLens 20m 电影数据集

报告人：吴凡璐

学校：中央财经大学

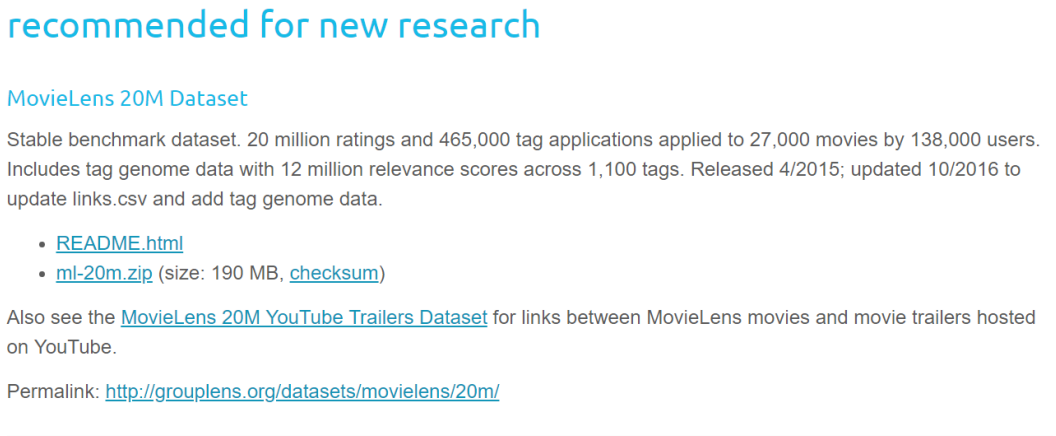
一、 项目背景

基于机器学习的推荐系统近年来在音乐、电影、书籍、电商等各个领域都十分流行，它为人们寻找符合自己需要的产品提供了便利，也帮助商家有针对性地推销产品，提高产品销量。推荐系统能够根据用户的喜好来推荐符合他们口味的产品，帮助他们找到不知道是否存在或不知道如何寻找到的产品。本项目将建立一个针对电影的协同过滤推荐系统，利用网络上公开的 MovieLens 20m 电影推荐数据集，进行相关的数据挖掘。

二、 数据描述

1、数据来源

本项目数据来源于 grouplens（https://grouplens.org/datasets/movielens/）网站上发布的电影评论网站 MovieLens（http://movielens.org）上的电影评论数据，对应不同的数据量有不同的数据集可供选择，本项目选择是 MovieLens 20M，包含 138493 位用户对 27278 部电影的 20000263 项电影的评分(1-5 分)，电影标签数为 465564 个，数据采集自网站 movielens.umn.edu，时间段为 1995.01-2015.03。


recommended for new research

MovieLens 20M Dataset

Stable benchmark dataset. 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags. Released 4/2015; updated 10/2016 to update links.csv and add tag genome data.

- README.html
- ml-20m.zip (size: 190 MB, checksum)

Also see the MovieLens 20M YouTube Trailers Dataset for links between MovieLens movies and movie trailers hosted on YouTube.

Permalink: http://grouplens.org/datasets/movielens/20m/

该数据集中包含以下子数据集：

- **ratings.csv** (userId, movieId, rating,timestamp)
- **movies.csv** (movie, title, genres)
- **tags.csv** (userId, movieId, tag, timestamp)
- **links.csv** (movieId, imdbId, tmdbId)
- **genome_score.csv** (movieId, tagId, relevance)
- **genome_tag.csv** (tag, tagId)

本项目采用 **ratings** 和 **movies** 表进行分析和建模。

2、变量描述

（1）Movies 表

movies 表记录了网站上所有有评分或标签的 27278 部电影的基本信息，包括 3 个字段：

表 1：movies 表基本信息

字段名	含义	备注
movieId	电影编号	与其他表的 movieId 一致
title	电影名称	名称+上映年份
genres	电影种类	可属于多个类别

在 hive 中展示前 4 条记录如下：

```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
```

（2）ratings 表

ratings 表记录了网站上给定时间段内用户做出的 20000263 条电影评级信息，有 4 个字段：

表 2：ratings 表基本信息

字段名	含义	备注
movieId	电影编号	与其他表的 movieId 一致
userId	用户编号	该用户的用户名编号
rating	电影评分	该用户对电影的评分（0.5~5 星）
timestamp	时间戳	评论发布的时间

在 hive 中展示前 4 条记录如下：

```
userId,movieId,rating,timestamp
1,2,3.5,1112486027
1,29,3.5,1112484676
1,32,3.5,1112484819
1,47,3.5,1112484727
```

三、 利用 hive 进行探索性分析

在进行建模之前，我们首先对数据进行探索性分析。在 hive 中建立 wmv 数据库后，导入外部表 movie 和 rating，利用数据库技术对数据进行探索。

首先获取按照用户评分数量排序的电影的最高和最低评分，以及计算得出的

平均得分。可以看到 1995 年到 2015 年间，获得评论数最多的电影是 *Pulp Fiction*（中文名：《低俗小说》）,是一部摄于 1994 年的犯罪片，电影获得的评级跨度较大，最高分为 5 分，最低分为 0.5 分，平均得分为 4.17 分。紧随其后的是《阿甘正传》、《沉默的羔羊》等经典电影。

```
name    maxs    mins    avgs    usercount
Pulp Fiction (1994)     5.0     0.5     4.17    67310
Forrest Gump (1994)     5.0     0.5     4.03    66172
"Shawshank Redemption   5.0     0.5     4.45    63366
"Silence of the Lambs   5.0     0.5     4.18    63299
Jurassic Park (1993)    5.0     0.5     3.66    59715
Time taken: 113.788 seconds, Fetched: 5 row(s)
```

在网站上，不同的用户参与电影评级的积极性也不同，我们调查了评分数量靠前的用户，将他们定义为活跃用户，以下 hive 查询结果显示了最活跃（评分数量多）的前 10 名用户以及他们评分的次数。由图可见，最活跃的用户是 ID 为 118205 的用户，评论次数高达 9254 次，是个“电影发烧友”。

```
rating.userid   cnt
118205  9254
8405    7515
82418   5646
121535  5520
125794  5491
74142   5447
34576   5356
131904  5330
83090   5169
59477   4988
```

我们对最活跃用户 118205 进行了进一步挖掘，查询了他评分高于 4 分的电影名称及评分。在他评分较高的电影里，可以看到《勇敢的心》、《教父》、《杀死一只知更鸟》等知名电影。

```
userid  movieid rating  name
118205  50      4.5     "Usual Suspects
118205  110     5.0     Braveheart (1995)
118205  111     4.5     Taxi Driver (1976)
118205  246     4.5     Hoop Dreams (1994)
118205  296     4.5     Pulp Fiction (1994)
118205  594     5.0     Snow White and the Seven Dwarfs (1937)
118205  596     4.5     Pinocchio (1940)
118205  599     5.0     "Wild Bunch
118205  750     4.5     Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
118205  858     5.0     "Godfather
118205  892     4.5     Twelfth Night (1996)
118205  898     4.5     "Philadelphia Story
118205  899     5.0     Singin' in the Rain (1952)
118205  900     5.0     "American in Paris
118205  912     5.0     Casablanca (1942)
118205  919     5.0     "Wizard of Oz
118205  923     5.0     Citizen Kane (1941)
118205  947     4.5     My Man Godfrey (1936)
118205  953     5.0     It's a Wonderful Life (1946)
118205  954     4.5     Mr. Smith Goes to Washington (1939)
118205  955     4.5     Bringing Up Baby (1938)
118205  965     4.5     "39 Steps
118205  994     5.0     Big Night (1996)
118205  1172    5.0     Cinema Paradiso (Nuovo cinema Paradiso) (1989)
118205  1207    5.0     To Kill a Mockingbird (1962)
118205  1213    4.5     Goodfellas (1990)
118205  1221    4.5     "Godfather: Part II
118205  1222    4.5     Full Metal Jacket (1987)
118205  1226    5.0     "Quiet Man
118205  1254    5.0     "Treasure of the Sierra Madre
118205  1272    4.5     Patton (1970)
```

四、 利用 python 进行可视化

除了用户层面的微观探索，我们在宏观时间维度上也进行了数据分析。主要使用 python 进行可视化。

下图是对用户评论过的电影数量和电影平均得分随时间变化趋势的折线图。在数据处理方面，将 ratings 表中的时间戳 timestamp 格式化为年份后结合 movieId 和 rating 字段进行统计。第一张图是 1995-2015 这 10 年间该网站上所有用户评论过的电影数量变化图，随着年份增长，电影评论数也在增长，2009 年增长速率最大。第二张图是所有种类的电影 10 年间的平均得分，可见平均得分几乎没有太大波动，也没有明显的变化趋势，平均打分在 3.40-3.75 之间变化。

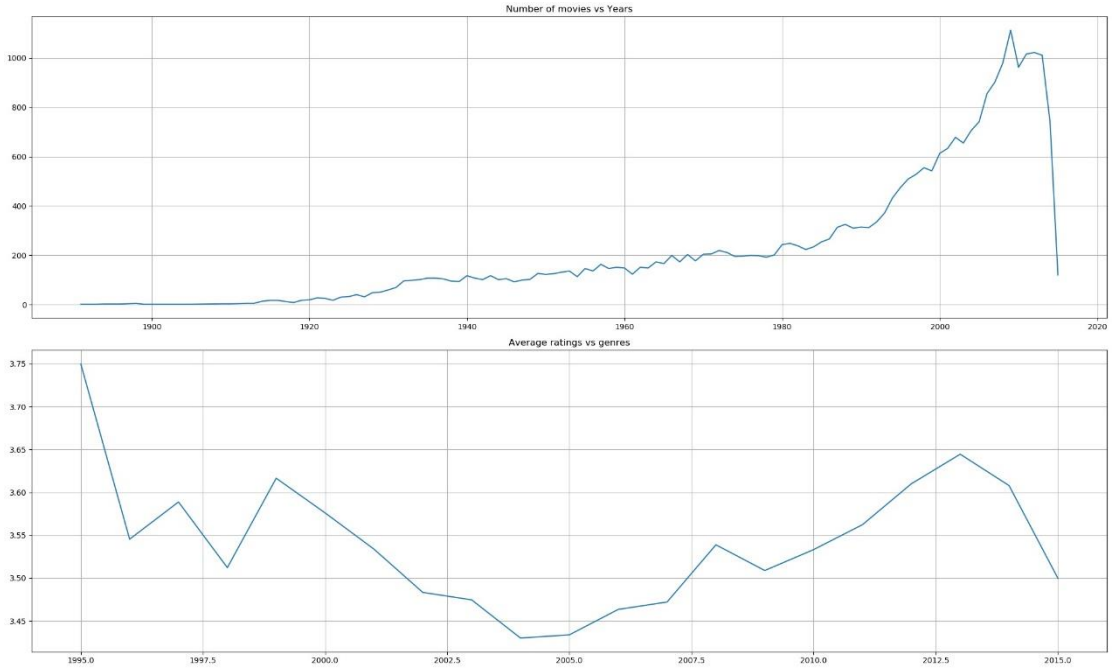


图 1：电影数量和电影平均评分随时间变化图

本次数据集中的电影分为多个种类，有动作、冒险、儿童、喜剧等多种风格，还有的电影没有给出类型信息，具体如下所示：

- Action
- Adventure
- Animation
- Children's
- Comedy
- Crime
- Documentary
- Drama
- Fantasy
- Film-Noir
- Horror
- Musical
- Mystery
- Romance
- Sci-Fi
- Thriller
- War
- Western
- (no genres listed)

下图是统计 10 年间放映的不同种类电影数量的柱形图，由图可见，最大众

化，数量最多的电影类型是戏剧（Drama）,数量第二的电影类型是喜剧（Comedy），接下来是动作片（Action），惊悚片（Thriller）等，数量最少的是黑色电影（Film-Noir），是一种主题为犯罪、刑侦，基调黑暗阴沉的相对小众的电影类型。

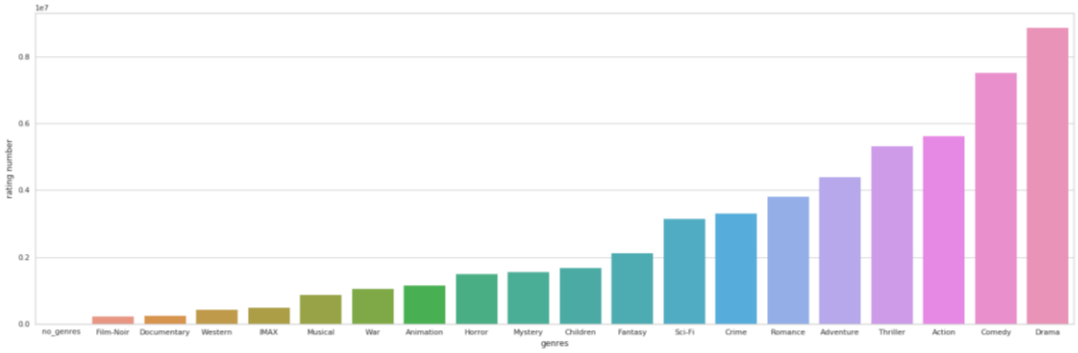


图 2：不同类型电影数量柱形图

接下来我们统计了不同类型电影的热度排名，根据评论数量将电影从高到低排序得到如下柱形图。前 2 名排名和电影数量的排名一致，分别是戏剧（Drama）和喜剧（Comedy），但是热度第三高的电影类型是浪漫爱情片。数量较多冒险类型（Adventure）电影收获的评论数并没有很高，但是数量很少的纪实片（Documentary）获得评论数却排名靠前，可能是纪实片反映了当时的社会现实，激发了人们的广泛讨论。

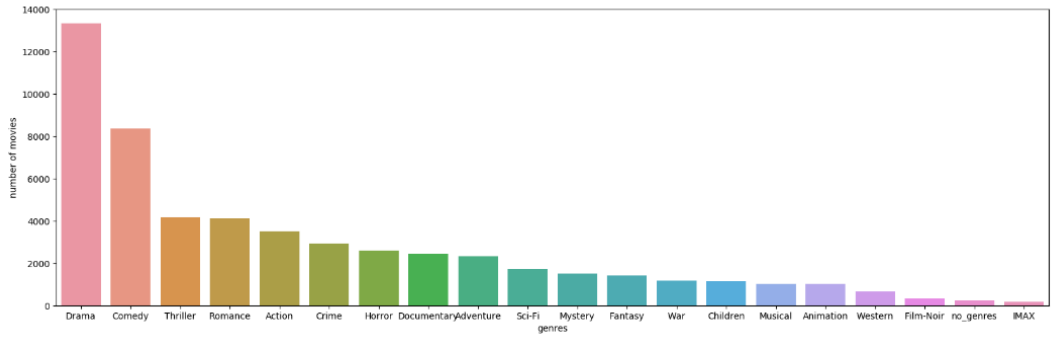


图 3：不同类型电影获得评论数量柱形图

不同种类的电影数量，收获的评论数都相差如此之大，那么电影的评分分布具体是怎样的呢？什么样的电影容易获得更高的评分，什么类型的电影最不容易获得人们的肯定？我们对不同种类电影的评分绘制了箱线图。如图所示，平均来看，戏剧（Drama），犯罪片（Crime）、神秘片（Mystery）、战争片（War）等更受人们青睐，而恐怖片（Horror）评分不仅相对较低，评分波动最大，得到的最低分也是所有电影中最低的，观众们对恐怖片更加“挑剔”。

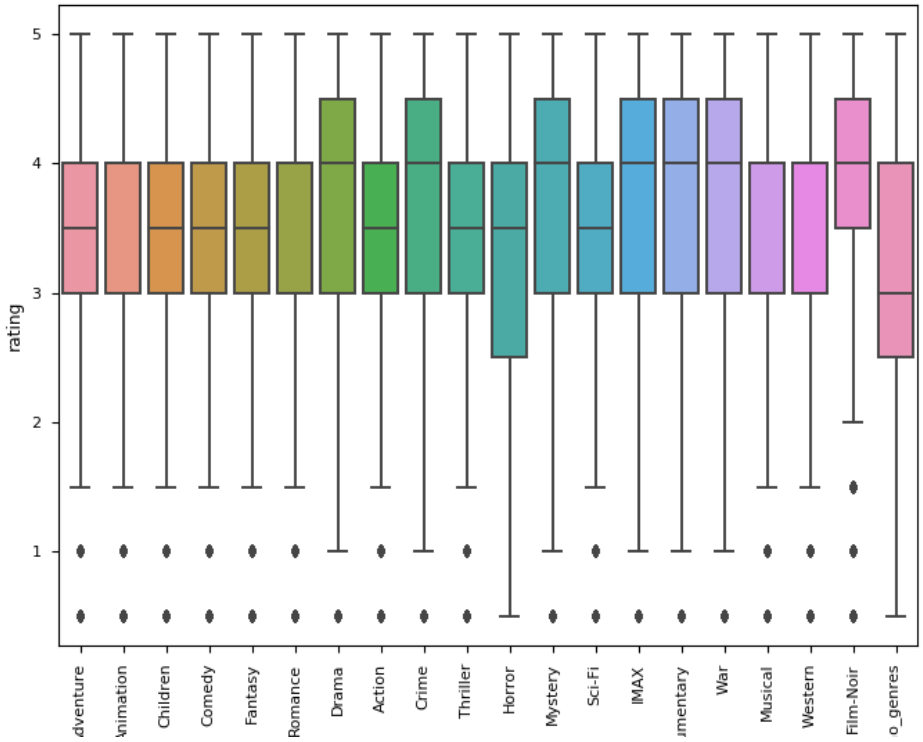


图 4：不同类型电影评分箱线图

五、 利用 pyspark 构建基于协同过滤的推荐系统

我们采用评分数据 ratings 表构建基于协同过滤的推荐系统，将符合用户需求的电影推荐给相应用户。在进行建模之前，用 pyspark 统计了 ratings 表的基本信息，如下图所示，ratings 表包括 138493 个用户对 26744 部电影做出的 2000263 条评论，和网站给出的数据集介绍一致。

```
>>> numRatings = ratingRDD.count()
>>> numUsers = ratingRDD.map(lambda r:r[0]).distinct().count()
>>> numMovies = ratingRDD.map(lambda r:r[1]).distinct().count()
>>> print ('Ratings dataset has %d ratings from %d users on %d movies.' %(numRatings,numUsers,numMovies))
Ratings dataset has 20000263 ratings from 138493 users on 26744 movies.
```

接下来我们采用 MLlib 库中的 ALS 算法来创建模型。将数据导入为 RDD 后，随机地将 80%的数据分割为训练集，将 20%的数据分割为测试集并进行缓存。统计得到，有 16002392 条记录被分到训练集，3997871 条记录被分到测试集。

```
>>> RDD1,RDD2=ratingRDD.randomSplit([0.8,0.2])
>>> trainingRDD=RDD1.cache()
>>> testRDD =RDD2.cache()
>>> trainingRDD.count()
16002392
>>> testRDD.count()
3997871
```

通过 ALS 算法在训练集上训练模型，将训练集中的用户 ID，电影 ID 和评分以及 rank（模型中潜在因素的数量）以及要运行的迭代次数输入训练集。本次训练设置 rank=10,迭代次数为 10，进行训练后，我们获得最活跃用户 118205 的 top5 电影预测，这 5 部电影的预测评分为：

表 3：118205 号用户预测得分 top5 的电影信息

电影 ID	电影名称	预测评分
60880	Family Game（1983）	6.17
56779	I Don't Want to Sleep Alone (Hei yan quan) (2006)	5.88
92161	Viva (2007)	5.85
128812	Drew: The Man Behind the Poster (2013)	5.80
74159	Ethan Mao (2004)	5.77

为了对模型进行评估，我们对 testRDD 进行预测，并与测试集中真实的评分进行对比。首先删除测试集中已有的评分，以备输入模型，处理后的测试集前两条记录如下，每条记录只包括电影 ID 和用户 ID。

```
>>> testUserMovieRDD=testRDD.map(lambda x: (x[0],x[1]))
>>> testUserMovieRDD.take(2)
[(1, 47), (1, 151)]
```

进行预测后得到测试集的预测得分，取前两条可见，ID 为 74884 的用户对 ID 为 1608 和 1288 的两部电影的预测评分分别是 3.19 和 4.16。

```
>>> predictionsTestRDD=model.predictAll(testUserMovieRDD).map(lambda r:((r[0],r[1]),r[2]))
>>> predictionsTestRDD.take(2)
[((74884, 1608), 3.1909520459207723), ((74884, 1288), 4.1685355704720894)]
```

将 testRDD 转换成键值对的形式，和预测的评分组合到一起得到新的 RDD，抽取 5 条记录查看。如下图所示，评分为 4 的电影得到的预测得分为 3.78，评分为 3.5 的电影得到的预测得分为 3.99。预测评分和真实评分较为接近。

```
>>> ratingsPredictions.take(5)
[((1, 253), (4.0, 3.7847680851088956)), ((1, 589), (3.5, 3.991870241942097)), ((1, 2173), (4.0, 3.520218710989021)), ((1, 2253), (3.5, 3.369308651622368)), ((1, 3037), (3.
5, 3.62149732122953))]
```

在所有预测结果中，也会产生一些不好的预测，我们将评分低于 1 的电影预测为评分高于 4 认定为一个坏预测。在坏预测中抽取两条记录可见，模型将两个原评分为 0.5 的电影分别预测成了 4.75 和 4.22，预测与现实相差甚远。

```
>>> badPredictions.take(2)
[((106348, 2706), (0.5, 4.755916537803875)), ((108608, 8366), (0.5, 4.222108254549917))]
```

不过，在总体为 2000263 的预测中，坏预测的个数只有 1763 个。

```
>>> badPredictions.count()
1763
```

从总体上评估模型，使用均方误差来衡量预测与真实之间的差距，均方误差（MSE）越低，模型效果越好。计算得到的均方误差为 0.54，模型有待改进，可通过调节 rank 和 numIteration 的数值进行优化，由于平台过于拥挤未能进行多次尝试。

```
>>> MSE=ratingsPredictions.map(lambda r: (r[1][0]-r[1][1])**2).mean()
>>> print ('Mean Squared Error ='+str(MSE))
Mean Squared Error =0.5407632503945364
```

六、 总结与反思

本次项目初步尝试在 pyspark 上构建推荐系统，使用 RDD 作为 API 用 MLlib 库建立模型。此外，利用 python 单机进行了基本的可视化分析，主要对电影类型的相关统计信息进行了挖掘，由于时间所限没有进行进一步的用户画像，没有使用 tags 表绘制不同类型电影的标签词云图。更丰富和更高级的可视化和模型选择，参数调优是本次项目有待改进的地方，也是以后会跟进的方向。