
沈腾 宋芸桦
Hello
Mr.Billionaire
$首富¥
闫非 彭大魔


HWARZENEGGER
IT'S NOTHING PERSONAL
TERMINATOR 2
JUDGMENT DAY

一：男/女性最喜欢的电影/电影类型
男女分歧最大的几部电影

二：最热门电影
评分最高的十部电影

三：不同类型电影出现的频数

# movielens电影评分数据分析(上)

## ●数据从哪里来?

给定数据集-MovieLens 1M数据集

一组从20世纪90年末到21世纪初由MovieLens用户提供的电影评分数据。这些数据中包括电影评分、电影元数据（风格类型和年代）以及关于用户的人口统计学数据（年龄、邮编、性别和职业等）。

MovieLens 1M数据集含有来自6000名用户对4000部电影的100万条评分数据。分为三个表：评分、用户信息和电影信息。

ratings.dat　movie.dat　users.dat

user.dat 用户表

| | UserID | Gender | Age | Occupation | Zip-code |
|---|---|---|---|---|---|
| 0 | 1 | F | 1 | 10 | 48067 |
| 1 | 2 | M | 56 | 16 | 70072 |
| 2 | 3 | M | 25 | 15 | 55117 |
| 3 | 4 | M | 45 | 7 | 02460 |
| 4 | 5 | M | 25 | 20 | 55455 |

ratings.dat 评分表

| | UserID | MovieID | Rating | Timestamp |
|---|---|---|---|---|
| 0 | 1 | 1193 | 5 | 978300760 |
| 1 | 1 | 661 | 3 | 978302109 |
| 2 | 1 | 914 | 3 | 978301968 |
| 3 | 1 | 3408 | 4 | 978300275 |
| 4 | 1 | 2355 | 5 | 978824291 |

movie.dat 电影信息表

| | MovieID | Title | Genres |
|---|---|---|---|
| 0 | 1 | Toy Story (1995) | Animation\|Children's\|Comedy |
| 1 | 2 | Jumanji (1995) | Adventure\|Children's\|Fantasy |
| 2 | 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| 3 | 4 | Waiting to Exhale (1995) | Comedy\|Drama |
| 4 | 5 | Father of the Bride Part II (1995) | Comedy |

merge合并

# 1.数据分析：男女分歧最大的一部电影

| Gender<br>Title | F | M | diff |
|---|---|---|---|
| The Terminator(1984) | 1.000000 | 4.333333 | 3.333333 |
| Independence Day(1996) | 4.000000 | 1.000000 | 3.000000 |
| Country Life (1994) | 5.000000 | 2.000000 | 3.000000 |
| Neon Bible, The (1995) | 1.000000 | 4.000000 | 3.000000 |
| James Dean Story, The (1980) | 4.000000 | 1.000000 | 3.000000 |
| Enfer, L' (1994) | 1.000000 | 3.750000 | 2.750000 |
| Babyfever (1994) | 3.666667 | 1.000000 | 2.666667 |
| Stalingrad (1993) | 1.000000 | 3.593750 | 2.593750 |
| Woman of Paris, A (1923) | 5.000000 | 2.428571 | 2.571429 |
| Cobra (1925) | 4.000000 | 1.500000 | 2.500000 |

## 分析步骤

1.查看每一部电影不同性别的平均评分

2. 在数据中增加一列diff，求出平均评分差值

3. 在数据中按照差值进行从大到小的排序

# 步骤1 --查看每一部电影不同性别的平均评分

**方法1：使用分组聚合进行组内计算**

# 使用分组聚合求组内计算

groupby方法的参数及其说明

- 该方法提供的是分组聚合步骤中的拆分功能，能根据索引或字段对数据进行分组。其常用参数与使用格式如下。

*DataFrame.**groupby**(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)*

| 参数名称 | 说明 |
| --- | --- |
| by | 接收list，string，mapping或generator。用于确定进行分组的依据。无默认。 |
| axis | 接收int。表示操作的轴向，默认对列进行操作。默认为0。 |
| sort | 接收boolearn。表示是否对分组依据分组标签进行排序。默认为True。 |
| group keys | 接收boolearn。表示是否显示分组标签的名称。默认为True。 |
| squeeze | 接收boolearn。表示是否在允许的情况下对返回数据进行降维。默认为False。 |

# 使用分组聚合求组内计算

## agg函数参数及其说明

- agg支持对每个分组应用某函数，包括Python内置函数或自定义函数。
- 它们的参数说明如下表。

*DataFrame.**agg**(func, axis=0, *args, **kwargs)*

*DataFrame.**aggregate**(func, axis=0, *args, **kwargs)*

| 参数名称 | 说明 |
| --- | --- |
| func | 接收list、dict、function。表示应用于每行 / 每列的函数。无默认。 |
| axis | 接收0或1。代表操作的轴向。默认为0。 |

# 步骤1 --查看每一部电影不同性别的平均评分

**方法2：使用数据透视表**

# 1.3 数据分析

## ●1.3.1 查看每一部电影不同性别的平均评分

使用povit_table函数创建透视表-pivot_table函数常用参数及其说明

➢ 利用pivot_table函数可以实现透视表，pivot_table()函数的常用参数及其使用格式如下。

```
pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean',
fil
```

```
data_gender = data.pivot_table(values='Rating', index='Title', columns='Gender', aggfunc='mean')
```

| | |
|---|---|
| data | 接收DataFrame。表示创建表的数据。无默认。 |
| values | 接收字符串。用于指定想要聚合的数据字段名，默认使用全部数据。默认为None。 |
| index | 接收string或list。表示行分组键。默认为None。 |
| columns | 接收string或list。表示列分组键。默认为None。 |
| aggfunc | 接收functions。表示聚合函数。默认为mean。 |
| margins | 接收boolearn。表示汇总（Total）功能的开关，设为True后结果集中会出现名为“ALL”的行和列。默认为True。 |
| dropna | 接收boolearn。表示是否删掉全为NaN的列。默认为False。 |

# 1.3数据分析

## 1.3.2 查看每一部电影不同性别的平均评分

- pivot_table函数主要的参数调节

  - #index表示透视表的行
  - #columns表示透视表的列
  - #aggfunc表示对分析对象进行的分析，一般默认为求平均值，可以指定
  - #margins表示添加每行每列求和的值，默认不添加。

# 1.3数据分析

## ●1.3.2 计算评分分歧，找出男性和女性观众分歧最大的电影并进行排序

```
data_gender['diff'] = data_gender.F - data_gender.M
```

```
# 进行排序
data_gender_sorted = data_gender.sort_values(by='diff', ascending=False)
```

ascdending 参数用于控制升序或者降序

# 1.3数据分析

## ● 1.3.4 查看评分次数多的电影并进行排序

```
# 查看评分最多的电影
data_rating_num = data.groupby('Title').size()
```

先将 DataFrame 按电影标题分组，接下来利用size方法计算每组样本的个数，

```
# 评分最多的电影进行排序
data_rating_num_sorted = data_rating_num.sort_values(ascending=False)
```

1.想过滤掉评分条目数不足250条的电影

2.评分最高的十部电影

3.不同的电影风格出现的次数

# movielens电影评分数据分析(下)

# 1.3数据分析

## ●1.3.5 过滤掉评分条目数不足250条的电影

```
#过滤掉评分条目数不足250条的电影
 hot_movies=data_rating_num[data_rating_num>250]


#对评分数量进行排序，并取前20条数据
data_rating_num_sorted = hot_movies.sort_values(ascending=False)
```

# 1.3数据分析

## ●1.3.6 评分最高的十部电影

查看每一部电影被评论过的次数和被打的平均分。取出至少被评论过100次的电影按照平均评分从大到小排序，取最大的10部电影

按照电影名称分组，用agg函数通过一个字典{‘rating’ : [np.size, np.mean]}来按照key即rating这一列聚合

```
#查看评分最高的十部电影
movie_stats = data.groupby('Title').agg({'Rating':[np.size, np.mean]})
atleast_100 = movie_stats['Rating']['size'] >= 100
movie_stats[atleast_100].sort_values([('Rating', 'mean')], ascending=False)[:10]
```

# 1.1使用分组聚合进行组内计算

## 1.1.2使用agg方法聚合数据

- **agg和aggregate函数参数及其说明**

- agg，aggregate方法都支持对每个分组应用某函数，包括Python内置函数或自定义函数。同时这两个方法能够也能够直接对DataFrame进行函数应用操作。
- 在正常使用过程中，agg函数和aggregate函数对DataFrame对象操作时功能几乎完全相同，因此只需要掌握其中一个函数即可。它们的参数说明如下表。

*DataFrame.**agg**(func, axis=0, *args, **kwargs)*

*DataFrame.**aggregate**(func, axis=0, *args, **kwargs)*

| 参数名称 | 说明 |
| --- | --- |
| func | 接收list、dict、function。表示应用于每行 / 每列的函数。无默认。 |
| axis | 接收0或1。代表操作的轴向。默认为0。 |

# 1.1使用分组聚合进行组内计算

## 1.1.2使用agg方法聚合数据

- **agg方法求统计量**

- 可以使用agg方法一次求出当前数据中所有菜品销量和售价的总和与均值，如detail[['counts','amounts']].agg([np.sum,np.mean]))。
- 对于某个字段希望只做求均值操作，而对另一个字段则希望只做求和操作，可以使用字典的方式，将两个字段名分别作为key，然后将NumPy库的求和与求均值的函数分别作为value，如detail.agg({'counts':np.sum,'amounts':np.mean}))。
- 在某些时候还希望求出某个字段的多个统计量，某些字段则只需要求一个统计量，此时只需要将字典对应key的value变为列表，列表元素为多个目标的统计量即可，如detail.agg({'counts':np.sum,'amounts':[np.mean,np.sum]}))

# 1.1使用分组聚合进行组内计算

## 1.1.2使用agg方法聚合数据

- **agg方法与自定义的函数**

- 在agg方法可传入读者自定义的函数。
- 使用自定义函数需要注意的是NumPy库中的函数np.mean，np.median，np.prod，np.sum，np.std，np.var能够在agg中直接使用，但是在自定义函数中使用NumPy库中的这些函数，如果计算的时候是单个序列则会无法得出想要的结果，如果是多列数据同时计算则不会出现这种问题。
- 使用agg方法能够实现对每一个字段每一组使用相同的函数。
- 如果需要对不同的字段应用不同的函数，则可以和Dataframe中使用agg方法相同。

# 1.3数据分析

## ●1.3.7 查看不同年龄的分布情况并且采用直方图进行可视化

（1）查看用户的年龄分布：

```
users.Age.plot.hist(bins=30)
plt.title("Distribution of users' ages")
plt.ylabel('count of users')
plt.xlabel('age');
```

# 1.3数据分析

## ●1.3.7 在原数据中标记出用户位于的年龄分组

（1）#用pandas.cut函数将用户年龄分组：

```
labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
data['age_group'] = pd.cut(data.Age, range(0, 81, 10), labels=labels)
```

http://pandas.pydata.org/pandas-docs/stable/generated/pandas.cut.html

# 1.3数据分析

## ●1.3.8 电影评分表中计算不同类型电影的频数

1.对数据进行规整-movies

规整化处理的方法与步骤:

[1].以电影所属类型（genres）及符号’|’为依据进行分割;

[2].如果一个电影有多个类型（genres），将分割成多个列表（List）；

[3].将分割后得到的多个列表（Lists）转换为一个数据框（DataFrame）；

[4].将数据框的索引设置为movieId

```
movies_tidy1=pd.DataFrame(movies.Genres.str.split('|').tolist(),index=movies.MovieID)
```

# 1.3数据分析

## ●1.3.8 电影评分表中计算不同类型电影的频数

| MovieID | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1 | Animation | Children's | Comedy | None | None | None |
| 2 | Adventure | Children's | Fantasy | None | None | None |
| 3 | Comedy | Romance | None | None | None | None |

| | movieId | level_1 | 0 |
|---|---|---|---|
| 0 | 1 | 0 | Adventure |
| 1 | 1 | 1 | Animation |
| 2 | 1 | 2 | Children |
| 3 | 1 | 3 | Comedy |
| 4 | 1 | 4 | Fantasy |
| 5 | 2 | 0 | Adventure |
| 6 | 2 | 1 | Children |
| 7 | 2 | 2 | Fantasy |
| 8 | 3 | 0 | Comedy |
| 9 | 3 | 1 | Romance |

[5].按规整化数据的基本原则，采用stack（）函数进行重构，并重新设置行索引

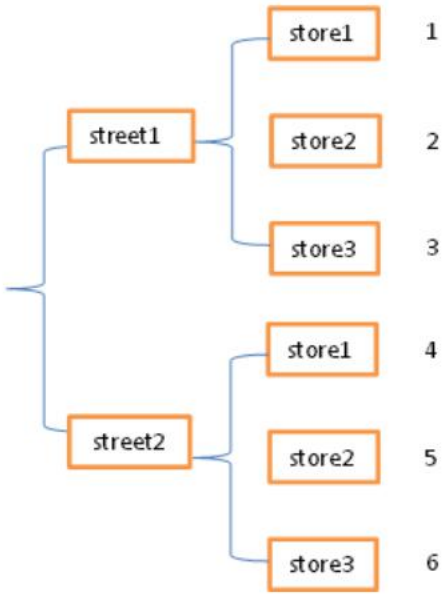
注：重新设置行索引后，原来的行索引保留为movieId列

```
movies_tidy2=movies_tidy1.stack().reset_index()
```

## ●1.3.8 电影评分表中计算不同类型电影的频数

在用pandas进行数据重排时，经常用到stack和unstack两个函数。stack的意思是堆叠，堆积，unstack即“不要堆叠”

常见的数据的层次化结构有两种，一种是表格，一种是“花括号”，即下面这样的l两种形式：




| | store1 | store2 | store3 |
|---|---|---|---|
| street1 | 1 | 2 | 3 |
| street2 | 4 | 5 | 6 |

# 1.3数据分析

## ●1.3.8 电影评分表中计算不同类型电影的频数

[6].删除level_1列，将columns为0的列重命名为genres,并重新定义数据框为movies_genres

```
movies_genres=movies_tidy2.drop('level_1',axis=1).rename(columns={0:'genres'})
movies_genres.head()
```

[7].将原movies数据中的Genres列替换成movies_genres，得到规整化处理后的movies数据

```
movies=pd.merge(movies.drop('Genres',axis=1),movies_genres)
movies.head()
```

# 1.3数据分析

## ●1.3.8 电影评分表中计算不同类型电影的频数

基于规整化处理后的movies和ratings，构建电影评分数据集movie_ratings

[1].去掉movies中的title列，得到新movies

[2].合并新movies与ratings，连接键为movieId

[3].得到新的数据框movie_ratings

[4].显示数据框movie_ratings的前5行

```
movie_ratings=ratings.merge(movies.drop('Title',axis=1),on='MovieID',how='inner')
movie_ratings.head()
```

```
merge: 合并数据集， 通过left， right确定连接字段，默认是两个数据集相同的字段
参数 说明
left 参与合并的左侧DataFrame
right 参与合并的右侧DataFrame
how 连接方式：‘inner’（默认）；还有，‘outer’、‘left’、‘right’
on 用于连接的列名，必须同时存在于左右两个DataFrame对象中，如果位指定，则以left和right列名的交集作为连接键
left_on 左侧DataFarme中用作连接键的列
right_on 右侧DataFarme中用作连接键的列
left_index 将左侧的行索引用作其连接键
right_index 将右侧的行索引用作其连接键
sort 根据连接键对合并后的数据进行排序，默认为True。有时在处理大数据集时，禁用该选项可获得更好的性能
suffixes 字符串值元组，用于追加到重叠列名的末尾，默认为（‘_x’,‘_y’）.例如，左右两个DataFrame对象都有‘data’，则
结果中就会出现‘data_x’，‘data_y’
copy 设置为False，可以在某些特殊情况下避免将数据复制到结果数据结构中。默认总是赋值
"""
```

# 1.3数据分析

## ●1.3.8 电影评分表中计算不同类型电影的频数

计算movies_ratings中不同类型电影的频数

[1].按genres列进行分组处理

[2].计算movieId的频数

[3].按movieId的频次进行降序

```
movie_ratings.groupby(['genres'])['MovieID'].size().sort_values(ascending=False)
```

# 谢 谢！ ！ ！