Pandas 实战 Kaggle 百万级影评数据集之 10 大问题探索...

今天继续探索 twitter 电影数据集，昨天我们已经对这个数据集完成特征工程处理，三张表分别关于电影、用户和电影评分。在探索前，我们设想几个有趣的问题，循着好奇心，更容易坚持下去，看完今天这篇 EDA 实战。

问题包括：

1. 29 类电影中，猜测下哪几类影片数是最多的？
2. 从上世纪初到现在，电影的产出数是平稳的还是线型增长，或者指数增长？
3. 喜剧片、动作片、爱情片、惊悚片你心目中的 TOP10 榜单是怎样的？ 根据 twitter 80 多万影评，挖出的TOP榜单又是怎样的？
4. 近 100 年，所有电影的 TOP10 榜单里有我们熟知的肖申克救赎，阿甘正传吗？
5. 近 100 年，最烂的垃圾篇 BAD10 榜单里都有哪些部电影被不幸入选？
6. 哪些电影是最有槽点的，被人们茶余饭后津津乐道呢？
7. 有哪些时期人们的吐槽兴致大增？
8. 哪些影迷最能吐槽吗？他们的 twitter ID 也被挖出来了！
9. 他们的吐槽数能有几千条吗？
10. 他们的评论严厉吗？平均评论得分是多少？

今天，通过 twitter 电影数据集进行有趣的数据分析，——回答以上 8 个问题。

今天课程的完整 notebook 代码，下载地址链接: 链接: https://pan.baidu.com/s/1oR1Ok41gfY9M4QRzRu4w 提取码: ryp6

今天课程使用的包如下：

复制
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pyecharts.charts import Bar,Grid,Line,Pie
import pyecharts.options as opts
from pyecharts.globals import ThemeType

from snapshot_phantomjs import snapshot # pyecharts导出图片使用到的库
from pyecharts.render import make_snapshot
```

1 哪几类影片数最多？

重新熟悉下三个 DataFrame 的前五行：

复制
```
movies2.head()
```

复制
```
users.head()
```

复制
```
ratings.head()
```

昨天已经得出最百花齐放的前 10 类电影：

复制
```
top10
```

复制
```
[('Mystery', 2649),
 ('Adventure', 3116),
 ('Documentary', 3224),
 ('Horror', 4288),
 ('Crime', 4723),
 ('Action', 5175),
 ('Romance', 5987),
 ('Thriller', 7307),
 ('Comedy', 10741),
 ('Drama', 17589)]
```

排名第一的是 Drama （戏剧）类电影，一共有 17589 部，那么它的出厂时间跨度呢？

复制
```
movies2.drop('index',axis=1,inplace=True) # 删除 index 列
movies2
```

复制
```
mdrama = movies2[movies2['Genre'].str.contains('Drama')]
mdrama

mdrama.sort_values(by='year')
```

看到 Drama 电影最早出厂追溯到上世纪 1909 年， 最近出厂年份 2019 年，整整110年。

2 近百年电影的产出数如何变化？

接下来，我们每十年，对数据完成下采样。

复制
```
tmp = mdrama.copy()
tmp.loc[:,'yeardt'] = pd.to_datetime(mdrama['year'],format='%Y')
tmp
tmpdt_index = pd.DatetimeIndex(tmp['yeardt'].dt.date)
tmpdt_index # 先创建 DatetimeIndex 索引对象

tmp.loc[:,'Movie Count'] = 1
tmp10 =  tmp.set_index(tmpdt_index).resample('10Y')['Movie Count'].sum().
to_frame()
```

```
tmp10
```

对 `resample` 我们在专栏 day36 ： Pandas 12 个实用小技巧里已经讲到，不会的读者可以返回去看看。

最终得到的结果如下所示：

直观展示以上数据：

复制

```
c = (
    Bar()
    .add_xaxis(tmp10.index.year.to_list())
    .add_yaxis("电影数", tmp10['Movie Count'].to_list(), category_gap=0,co
lor='blue')
   .set_global_opts(title_opts=opts.TitleOpts(title="每10年Drama类电影生产数"
))

)

c.render_notebook()
```

折线图：

复制

```
x_data = list(map(str,tmp10.index.year.to_list()))
y_data = tmp10['Movie Count'].to_list()
l0 = (
    Line()
    .add_xaxis(xaxis_data=x_data)
    .add_yaxis(
        series_name="",
        y_axis=y_data,
        symbol="emptyCircle",
        areastyle_opts=opts.AreaStyleOpts(opacity=1, color="#C67570")
    )

)

l0.render_notebook()
```

将上面对 Drama 类电影的分析过程，整理为一个函数，分别分析其他9类电影的产量变化情况。

复制

```
### 整理以上脚本，分析其他前10种类的电影

def mgenre_ana(mg):
    print(mg)
    mdrama = movies2[movies2['Genre'].str.contains(mg)]
    #mdrama
    mdrama.sort_values(by='year')
    tmp = mdrama.copy()
    tmp.loc[:,'yeardt'] = pd.to_datetime(mdrama['year'],format='%Y')
    #tmp
    tmpdt_index = pd.DatetimeIndex(tmp['yeardt'].dt.date)
    tmpdt_index # 先创建 DatetimeIndex 索引对象
    tmp.loc[:,'Movie Count'] = 1
    tmp10 =  tmp.set_index(tmpdt_index).resample('10Y')['Movie Count'].su
m().to_frame()
    #tmp10
    c = (
    Bar()
    .add_xaxis(tmp10.index.year.to_list())
    .add_yaxis("电影数", tmp10['Movie Count'].to_list(), category_gap=0,co
lor='blue')
   .set_global_opts(title_opts=opts.TitleOpts(title="每10年%s电影生产数"%(mg
,)))
    )

    x_data = list(map(str,tmp10.index.year.to_list()))
    y_data = tmp10['Movie Count'].to_list()
    l = (
    Line()
    .add_xaxis(xaxis_data=x_data)
    .add_yaxis(
        series_name="",
        y_axis=y_data,
        symbol="emptyCircle",
        areastyle_opts=opts.AreaStyleOpts(opacity=1, color="#C67570")
      )
    )
    return [c,l]
```

运行这个 notebook 前，请自行下载安装文件 phantomjs，下载地址：https://phantomjs.org/download.html

下载后一定要放置在于此 jupyter notebook 同一个安装路径下。

复制

```
top10_genre_names = [x for x,y in top10]
for g in top10_genre_names:
    c,l = mgenre_ana(g)
    make_snapshot(snapshot, c.render(), "%s.png"%(g,))
    make_snapshot(snapshot, l.render(), "%sline.png"%(g,))
```

其他 9 类电影每10年的电影产量。

Action 类：

Adventure 类：

Comedy 类：

Crime 类：

Documentary 类：

Horror 类：

Mystery 类：

Romance 类：

Thriller 类：

3 每类电影的TOP榜单

代码如下，看到 Pandas 的可读性很强，没有一个 for 循环。

```
# 先连接movies2 和 ratings 两个DataFrame
def rating_top10(mg,topn=10):
    print(mg)
    mdrama = movies2[movies2['Genre'].str.contains(mg)]
    mmerge = mdrama.merge(ratings,on='Movie ID')
    mcount = mmerge['Movie ID'].value_counts()
    mcount2 = mcount.sort_values()
    #mcount2
    mask = mcount2 > 100
    # 看到有的影片只有1次被评论，显然这种平均分求前topn的可信度不高，我们设置一个电影被评论次数超过100次的topn电影
    tmp = mmerge.set_index('Movie ID')# index 对齐
    mmerge2 = tmp[mask]
    mpivot = mmerge2.pivot_table(columns='Movie ID',values=['Rating'],aggfunc=[np.mean])
    #mpivot
    mmelt = mpivot['mean'].melt(value_name ='Rating') # 宽表变为长表，符合我们的习惯
    #mmelt
    mtop10 = mmelt.sort_values('Rating')[-topn:] # 返回得分最高的前topn部电影
    return mtop10.merge(movies,on='Movie ID', how='left') # 连接表得到完整的电影得分前topn电影信息
```

调用函数：

```
top10_genre_names = [x for x,y in top10]
for g in top10_genre_names:
    topn = 50
    result = rating_top10(g,topn)
    result
    result.to_csv('./score_topn/%s-top%d.csv'%(g,topn),encoding='utf-8')
```

得到 10 个 csv 文件，保存着TOP50 的榜单。

Action 类电影的TOP10榜单：

Movie ID	Rating	Movie Title	Genre
468569	9.280967	The Dark Knight (2008)	Action|Crime|Drama|Thriller
103064	9.103004	Terminator 2: Judgment Day (1991)	Action|Sci-Fi
4154796	9.038913	Avengers: Endgame (2019)	Action|Adventure|Fantasy|Sci-Fi
1375666	9.012629	Inception (2010)	Action|Adventure|Sci-Fi|Thriller
3863552	9.01005	Bajrangi Bhaijaan (2015)	Action|Comedy|Drama
5074352	8.982196	Dangal (2016)	Action|Biography|Drama|Sport
172495	8.855072	Gladiator (2000)	Action|Adventure|Drama
110413	8.790083	L茅on (1994)	Action|Crime|Drama|Thriller
4154756	8.764341	Avengers: Infinity War (2018)	Action|Adventure|Sci-Fi

Comedy 类：

Movie ID	Rating	Movie Title	Genre
5512872	9.985836	Be Somebody (2016)	Comedy|Drama|Romance
3863552	9.01005	Bajrangi Bhaijaan (2015)	Action|Comedy|Drama
88763	8.945946	Back to the Future (1985)	Adventure|Comedy|Sci-Fi
118799	8.862573	La vita bella (1997)	Comedy|Drama|Romance|War

Movie ID	Rating	Movie Title	Genre
1675434	8.851211	The Intouchables (2011)	Biography|Comedy|Drama
70735	8.839286	The Sting (1973)	Comedy|Crime|Drama
45152	8.824427	Singin' in the Rain (1952)	Comedy|Musical|Romance
2380307	8.740077	Coco (2017)	Animation|Adventure|Comedy|Family|Fantasy|Music|Mystery
435761	8.732632	Toy Story 3 (2010)	Animation|Adventure|Comedy|Family|Fantasy

更多榜单就不再放到文章里了，欢迎手动执行 notebook 脚本，获得 10 个榜单 csv 文件。

4 10部最佳影片

整个 3 万多部影片中，分析出评分最佳的 10 部影片。

复制

```
# 先连接movies2 和 ratings 两个DataFrame
def all_rating_top10(topn=10,reversed=True):
    mdrama = movies2
    mmerge = mdrama.merge(ratings,on='Movie ID')
    mcount = mmerge['Movie ID'].value_counts()
    mcount2 = mcount.sort_values()
    #mcount2
    mask = mcount2 > 100
    # 看到有的影片只有1次被评论，显然这种平均分求前topn的可信度不高，我们设置一个电影被评论次数超过100次的topn电影
    tmp = mmerge.set_index('Movie ID')# index 对齐
    mmerge2 = tmp[mask]
    mpivot = mmerge2.pivot_table(columns='Movie ID',values=['Rating'],aggfunc=[np.mean])
    #mpivot
    mmelt = mpivot['mean'].melt(value_name ='Rating') # 宽表变为长表，符合我们的习惯
    #mmelt
    if reversed is True:
        mtop10 = mmelt.sort_values('Rating')[-topn:] # 返回得分最高的前topn部电影
        return mtop10.merge(movies,on='Movie ID', how='left') # 连接表得到完整的电影得分前topn电影信息

    mbad10 = mmelt.sort_values('Rating')[:topn]
    return mbad10.merge(movies,on='Movie ID', how='left') # 连接表得到完整的电影得分前topn电影信息
```

调用：

复制

```
TOP10 = all_rating_top10()
TOP10
```

复制

```
TOP10c = TOP10.sort_values(by='Rating',ascending=False) #这样看着更习惯
TOP10c
```

绘制柱状图：

复制

```
x = TOP10c['Movie Title'].to_list()
y = TOP10c['Rating'].round(2).to_list()
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('得分',y,category_gap='50%')
    .reversal_axis()
    .set_global_opts(title_opts=opts.TitleOpts(title="电影得分TOP榜单"),
                     xaxis_opts=opts.AxisOpts(min_=8.0,name='得分'),
                     toolbox_opts=opts.ToolboxOpts(),)
)
grid = (
    Grid(init_opts=opts.InitOpts(theme=ThemeType.DARK))#MACARONS
    .add(bar, grid_opts=opts.GridOpts(pos_left="30%"))
)
grid.render_notebook()
```

5 10部最垃圾的影片

复制

```
BAD10 = all_rating_top10(reversed=False) # 人类历史最垃圾的10部电影
BAD10
```

绘制柱状图:

复制

```
x = BAD10['Movie Title'].to_list()
y = BAD10['Rating'].round(2).to_list()
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('得分',y,category_gap='50%')
    .reversal_axis()
    .set_global_opts(title_opts=opts.TitleOpts(title="电影得分BAD榜单"),
                     xaxis_opts=opts.AxisOpts(min_=3.0,name='得分'),
                     toolbox_opts=opts.ToolboxOpts(),)
)
grid = (
    Grid(init_opts=opts.InitOpts(theme=ThemeType.DARK))#MACARONS
    .add(bar, grid_opts=opts.GridOpts(pos_left="30%"))
)
grid.render_notebook()
```

6 电影被吐槽数TOP榜单

```
mmerge = movies2.merge(ratings,on='Movie ID')
mcount = mmerge['Movie ID'].value_counts()
mcount3 = mcount.sort_values()
mcount4 = mcount3[-10:].reset_index()
mcount4.columns=['Movie ID','Rating Count']
mcount5 = mcount4.merge(movies,on='Movie ID', how='left')
mcount5
```

绘制柱状图：

```
x = mcount5['Movie Title'].to_list()
y = mcount5['Rating Count'].round(2).to_list()
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('吐槽数',y,category_gap='50%')
    .reversal_axis()
    .set_global_opts(title_opts=opts.TitleOpts(title="电影被吐槽TOP榜单"),
                     xaxis_opts=opts.AxisOpts(min_=8.0,name='吐槽数'),
                     toolbox_opts=opts.ToolboxOpts(),)
)
grid = (
    Grid(init_opts=opts.InitOpts(theme=ThemeType.DARK))
    .add(bar, grid_opts=opts.GridOpts(pos_left="30%"))
)
grid.render_notebook()
```

7 电影被吐槽最多TOP月份

```
ratings.loc[:,'ym'] = ratings['rating_dt'].map(lambda x: 100*x.year + x.m
onth)
ratings
ratingmonth = ratings['ym'].value_counts()
rating_as_month = ratingmonth.sort_values()[-10:]
rating_as_month
```

结果：

```
201311    15872
201306    15989
201307    16341
201501    16501
201310    16656
201312    16897
201303    17089
201308    18652
201401    21209
201309    22635
Name: ym, dtype: int64
```

绘制柱状图：

```
x = rating_as_month.index.to_list()
y = rating_as_month.to_list()
bar = (
    Bar()
    .add_xaxis(x)
    .add_yaxis('年月',y,category_gap='50%')
    .reversal_axis()
    .set_global_opts(title_opts=opts.TitleOpts(title="电影被吐槽数"),
                     xaxis_opts=opts.AxisOpts(min_=8.0,name='吐槽数'),
                     toolbox_opts=opts.ToolboxOpts(),)
)
grid = (
    Grid(init_opts=opts.InitOpts(theme=ThemeType.DARK))
    .add(bar, grid_opts=opts.GridOpts(pos_left="30%"))
)
grid.render_notebook()
```

8 最能吐槽的影迷TOP榜单

```
user_ratings = ratings['User ID'].value_counts()
user_ratings_sort = user_ratings.sort_values()
user_ratings2 = user_ratings_sort.reset_index()
user_ratings2.columns = ['User ID', 'Rating Count']
user_ratings2[-10:].merge(users,on='User ID',how='left')
```

9 评论平均得分

```
user_pivot = ratings.pivot_table(columns='User ID',values=['Rating'],aggf
unc=[np.mean])
user_pivot2 = user_pivot['mean']
user_pivot2
user_melt = user_pivot2.melt(value_name='Rating Score')
mask5 = user_melt['User ID'].isin(user_ratings2[-10:]['User ID'])
user_melt2 = user_melt[mask5].sort_values(by='Rating Score')
user_melt2.merge(user_ratings2[-10:],on='User ID',how='left')
```

小结

今天的知识点有很多个，涉及很多数据分析常用的经典的方法，强烈建议大家动手执行下今天的notebook，体会里面常用的方法：

- merge
- pivot_table
- melt
- sort_values
- value_counts
- Series.map

- isin
- drop_duplicates
- set_index
- reset_index
- pyecharts 绘图常用方法

下一章

互动评论

说点什么

评论

The Scrapper 1 个月前

完成

鼓掌

存