豆瓣电影数据分析报告

19313039 管理学院 甘寓宁

一、 分析数据

2018 豆瓣电影评论.csv

二、 调用的库

Pandas:处理整体数据

Matplotlib:数据可视化

Jieba:分词

Wordcloud:生成词云

三、 数据导入及初步处理

```
1. CSV_FILE_PATH =r'C:\Users\gynda\Desktop\2018 豆瓣电影评
   论.csv'            #获取 csv 路径
2. df = pd.read_csv(CSV_FILE_PATH)
          #读取 csv 文件
3. df['year'] =None
4. all_type = df['time'].str.split('/').apply(pd.Series) #获取评价年
   份
5. df['year']=all_type[0]
```

读取 csv 到 dataframe 类型的 df 中，并通过文本分割从评论时间中取出年份，写入 dataframe 中

四、 功能展示

1 评级统计

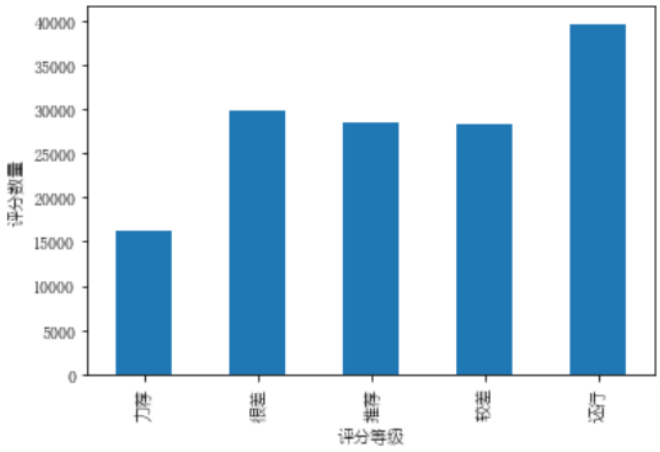
函数名：`rating_stat(df)`

功能描述：统计评论总数、有评级的评论总数、缺失评级的评论数、不同评级（力荐、很差、推荐、较差、还行）的评论数

输入：df

输出：评论总数、有评级的评论总数、缺失评级的评论数、不同评级的评论数及其柱状图

```
评论总数：  146942
有评级的评论总数：  142485
缺失评级的评论数：  4457
rating
力荐     16248
很差     29906
推荐     28465
较差     28220
还行     39646
dtype: int64
```



分析：数据中共有 146942 条评论，其中有 4457 条评论没有评级；“还行”最多，“力荐”最少，“很差”、“推荐”、“较差”数量较为相近

代码实现：

```
def rating_stat(df):
    '''
    功能：输入电影数据 dataframe,输出评论总数、带有评级的评论总数、前两者之差、不同等级评价数量统计
    '''
    num_rating = df.groupby(['rating']).size()
    print('评论总数：',df.shape[0])
    print('有评级的评论总数：',sum(num_rating))
    print('缺失评级的评论数：',df.shape[0]-sum(num_rating))
    print(num_rating)
    num_rating.plot.bar()
    plt.xlabel("评分等级")
    plt.ylabel("评分数量")
    plt.show()
```

2 评分统计

函数名：`rating_label(df)`

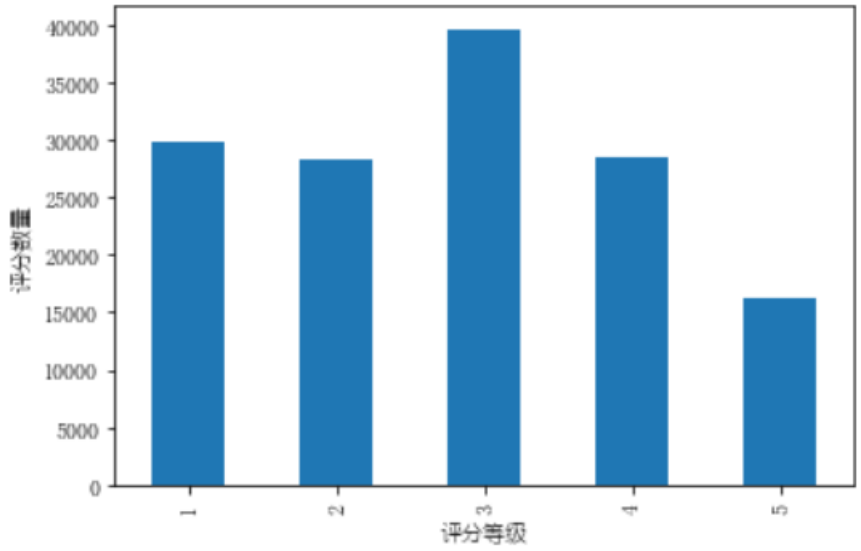
功能描述：将评级从“很差”到“力荐”分别设置为 1-5 分，设置为 df 的属

性，统计电影均分

输入：df

输出：不同评分的评论数及柱状图，平均评分

```
rating_label
1    29906
2    28220
3    39646
4    28465
5    16248
dtype: int64
电影评分均值：2.81
```



代码实现：

```
1. def rating_label(df):
2.     '''
3.     功能：输入电影数据 dataframe,为不同评价等级赋分，很差->力荐 对应 1->5，之后输出平均分数
4.     '''
5.     df['rating_label'] =None
6.     df.loc[(df['rating']=='很差'),'rating_label']=1
7.     df.loc[(df['rating']=='较差'),'rating_label']=2
8.     df.loc[(df['rating']=='还行'),'rating_label']=3
9.     df.loc[(df['rating']=='推荐'),'rating_label']=4
10.    df.loc[(df['rating']=='力荐'),'rating_label']=5
11.    num_label = df.groupby(['rating_label']).size()
12.    print(num_label)
13.    print('电影评分均值：%0.2f'%df['rating_label'].mean())
14.    num_label.plot.bar()
```

```
    plt.xlabel("评分等级")
    plt.ylabel("评分数量")
    plt.show()
```

3 评论统计

函数名：`comment_time(df)`

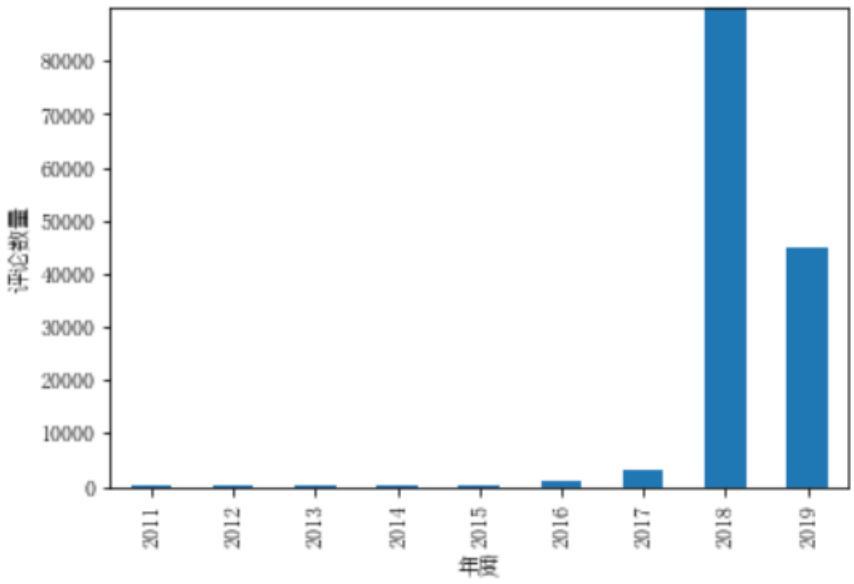
功能描述：统计电影数量，每部电影的评论数，不同时间的评论数

输入：df

输出：电影总数，评论最少的电影的评论数，评论最多的电影的评论数，分时间评论数及柱状图

```
电影总数：  427
评论最少的电影的评论数：  29
评论最多的电影的评论数：  500

year
2011      104
2012      204
2013      199
2014       83
2015      344
2016     1029
2017     3180
2018    96853
2019    44946
dtype: int64
```



分析：数据集中电影总数为 42 部，爬取的评论绝大多数来自于 2018 年、2019 年，其中 2018 年占了全体近 2/3。评论最多的电影的评论数为 500，且有多

部电影都为 500，猜测为爬虫设定抓取上限为 500。

代码实现：

```
def comment_time(df):
    '''
    功能：输入电影数据 dataframe,输出电影数量，评论最少的电影的评论数，评论最多电影的评论数，分年度的评论数统计
    '''
    num_movie = df.groupby(['movie_url']).size()    #根据 url 判断电影
    print('电影总数：',len(num_movie))
    print('评论最少的电影的评论数：',num_movie.values.min())
    print('评论最多的电影的评论数：',num_movie.values.max())
    num_movie_year=df.groupby(['year']).size()
    display(num_movie_year)
    num_movie_year.plot.bar()
    plt.ylim(0,90000)
    plt.yticks(np.arange(0,90000,10000))
    plt.ylabel("评论数量")
    plt.xlabel("年度")
    plt.show()
```

4 每部电影评分统计

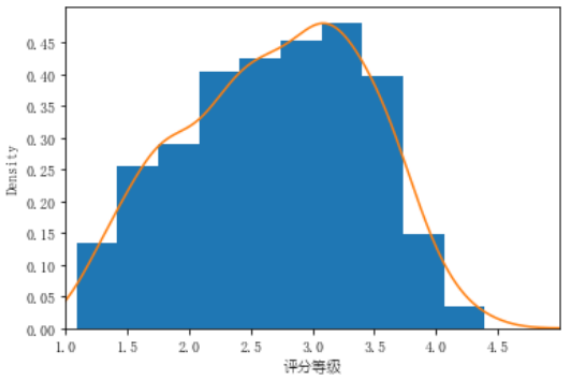
函数名：`rating_individual(df)`

功能描述：统计每部电影的平均评分

输入：df

输出：电影评分中位数、评分最高电影 url、最高电影评分、评分最低电影 url、最低电影评分、评分分布图

```
电影评分中位数：2.75
评分最高电影url：https://movie.douban.com/subject/26933248/
最高电影评分：4.39
评分最低电影url：https://movie.douban.com/subject/30352998/
最低电影评分：1.09
```



分析：评分中位数与所有电影评分的均值相近；落在 3-3.5 分的电影数量最多，4-4.5 分的最少；评分分布图左侧比重显然大于右侧，从 3.5 分往上电影数量骤降，而从 3 分到 1 分电影数量的下降较为平稳。

代码实现：

```
def rating_individual(df):
    '''
    功能：输入电影数据 dataframe,统计每部电影评分，输出评分中位数，评分最高电影的评分及 url，评分最低电影的评分及 url，评分分布图
    '''
    num_movie = df.groupby(['movie_url']).size()
    title=[] #电影 url 列表
    for i in num_movie.index:
        title.append(i)
    score=[] #电影评分
    for i in title:
    #遍历 dataframe,获取和 title 中名称匹配的评分
        sum=0
        for tup in zip(df['movie_url'], df['rating_label']):

            if i==tup[0]:
                if tup[1]!=None:
                    sum=sum+tup[1]
        score.append(sum/num_movie[i])
    score_movie=pd.Series(data=score,index=title)
    print('电影评分中位数：%0.2f'%score_movie.median())
    print('评分最高电影 url：%s'%score_movie.idxmax())
    print('最高电影评分：%0.2f'%score_movie.max())
    print('评分最低电影 url：%s'%score_movie.idxmin())
    print('最低电影评分：%0.2f'%score_movie.min())
    score_movie.plot.hist(density=1)
    plt.xlim(1,5)
    plt.xticks(np.arange(1,5,0.5))
    plt.yticks(np.arange(0,0.5,0.05))
    plt.ylabel("电影数量")
    plt.xlabel("评分等级")
    score_movie.plot.kde()
    plt.show()
```

5 每个用户的评论统计

函数名：`comment_individual(df)`

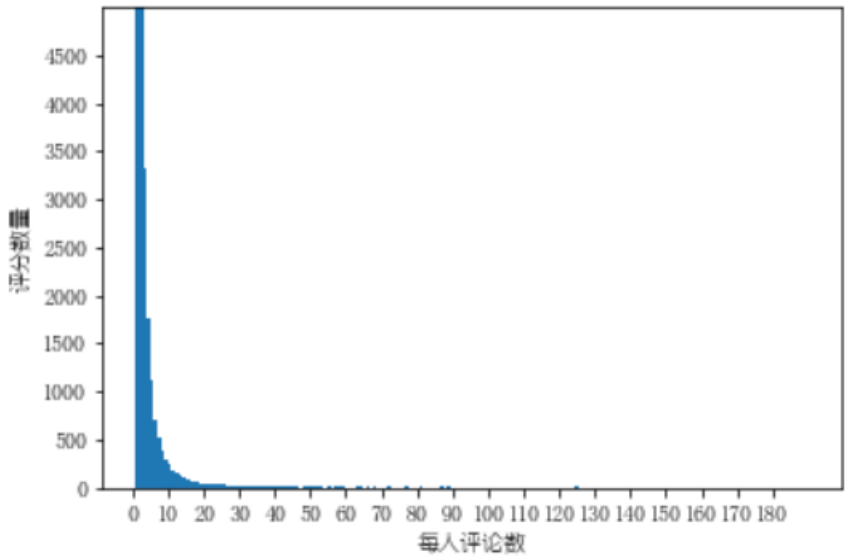
功能描述：统计每个用户的评论数量

输入：df

输出：评论用户总数、平均每人评论数、单人最多评论数

```
user_url
https://www.douban.com/people/100000149/          1
https://www.douban.com/people/100003157/          1
https://www.douban.com/people/100009133/          1
https://www.douban.com/people/100009418/          1
https://www.douban.com/people/100014788/          1
                                                  ..
https://www.douban.com/people/zzyy5221313/        1
https://www.douban.com/people/zzz.aoe/            1
https://www.douban.com/people/zzzhaox/            1
https://www.douban.com/people/zzzyao19960706/     5
https://www.douban.com/people/zzzzzzzz2/          3
Length: 73897, dtype: int64

评论用户总数： 73897
每人评论数均值：1.99
单人最多评论数：190
```



分析：评论的用户总数为 73897，平均每人评论 1.99 次，最多的人评论了 190 次，绝大多数落在 0-5 次内。

代码实现：

```
1. def comment_individual(df):
2.     '''
3.     功能：输入电影数据 dataframe,统计每个用户的评论数,输出评论的用户总数，每人评论数均值，单人最多评论数，评论数分布图
4.     '''
```

```
    num_user = df.groupby(['user_url']).size()
    display(num_user)
    print('评论用户总数：',len(num_user))
    print('每人评论数均值：%0.2f'%num_user.values.mean())
    print('单人最多评论数：%d'%num_user.values.max())
    num_user.plot.hist(bins=190)
    plt.ylim(0,5000)
    plt.xticks(np.arange(0,190,10))
    plt.yticks(np.arange(0,5000,500))
    plt.ylabel("评分数量")
    plt.xlabel("每人评论数")
    plt.show()
```

6 导出评论到 txt

函数名 ： get_comment(df),get_commment_low(df), get_commment_high(df)

功能描述：将所有评论/1-2 分评论/4-5 分评论分别导入到 all.txt/low.txt/high.txt 中。

输入：df（函数内输入评论 txt 路径）

输出：all.txt/low.txt/high.txt

代码实现：

```
def get_comment(df):
    '''
    功能：输入电影数据 dataframe，将所有评论写入 all.txt
    '''
    file=open(r"C:\Users\gynda\Desktop\all.txt",'w',encoding='utf-8')   #打开 all.txt(可自定义路径)
    for tup in zip(df['rating_label'], df['comment']):
        file.write((str(tup[1])))
        file.write('\n')
    file.close()

def get_commment_low(df):
    '''
    功能：输入电影数据 dataframe，将所有 1 分评论与 2 分评论写入 low.txt
    '''
    file=open(r"C:\Users\gynda\Desktop\low.txt",'w',encoding='utf-8')   #打开 low.txt(可自定义路径)
    for tup in zip(df['rating_label'], df['comment']):
        if tup[0]==1 or tup[0]==2:
            if tup[1]!=None:
```

```
                file.write((str(tup[1])))
                file.write('\n')
    file.close()

def get_commment_high(df):
    '''
    功能：输入电影数据 dataframe，将所有 4 分评论与 5 分评论写入 low.txt
    '''
    file=open(r"C:\Users\gynda\Desktop\high.txt",'w',encoding='utf-8')   #打开 high.txt(可自定义路径)
    for tup in zip(df['rating_label'], df['comment']):
        if tup[0]==4 or tup[0]==5:
            if tup[1]!=None:
                file.write((str(tup[1])))
                file.write('\n')
    file.close()
```

7 评论词频分析

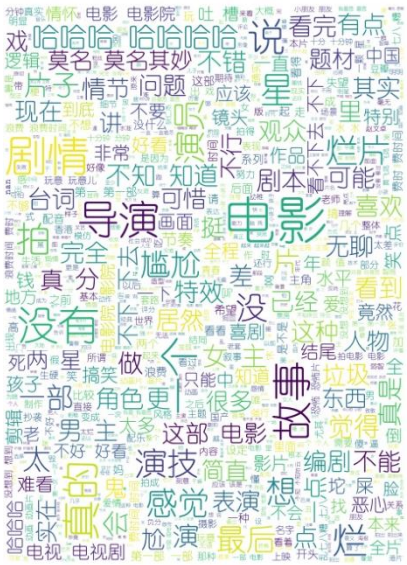
函数名：ciyun_comment_all(df), ciyun_comment_low(df), ciyun_comment_high(df)

功能描述：将对应的评论 txt（由 get_comment 等一系列函数获得）通过停用词进行分割，以背景图片为基础得到评论的词云图。

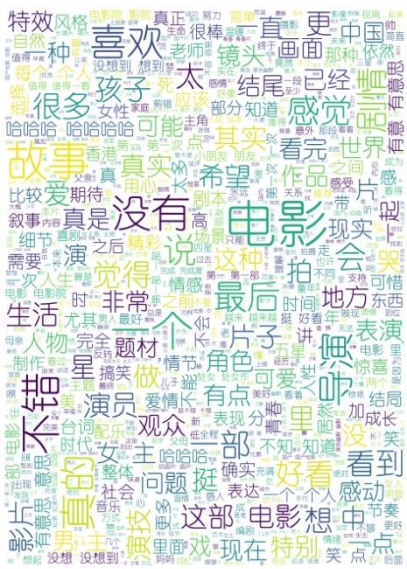
输入：df（函数内输入背景图片路径、评论 txt 路径、停用词路径）

输出：

全部评论



低分评论



高分评论

分析：低分评论明显含有更多负面情感的词；全部评论中负面情感的词数量也不少，侧面印证了电影评分均值较低的事实。