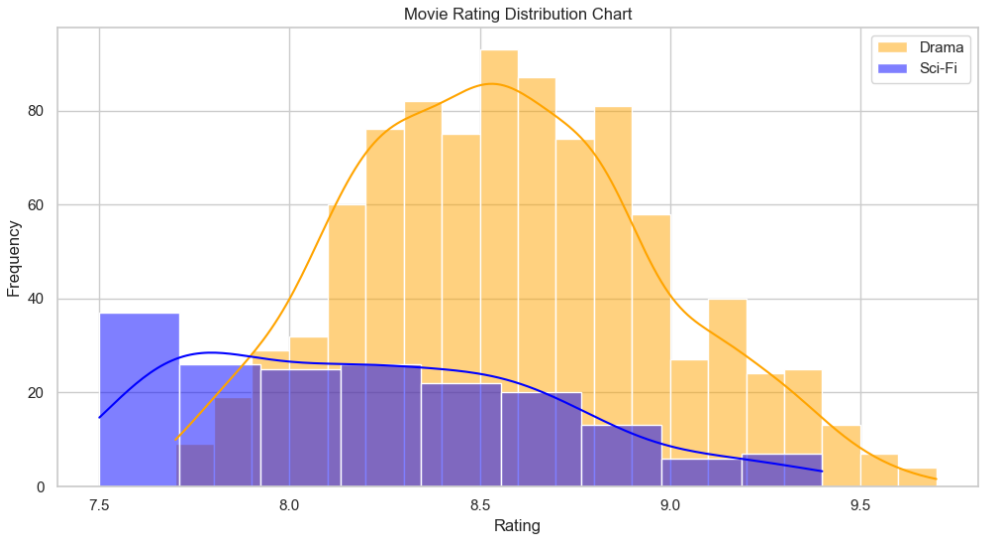
豆瓣科幻电影评分相关数据可视化和分析

华子涵 10225501409

选题

科幻电影作为一种引人入胜的电影类型，深受观众喜爱。然而，作为一位热衷于科幻的影迷，我不禁注意到，不仅一部我喜爱的科幻电影在豆瓣上收到了我认为与其实际质量不符的评分。这引发我提出两个问题：低分高质量的现象是否普遍存在于科幻电影中？如果是的话，什么造成了观众对科幻电影评分存在一定的偏颇？在这个背景下，我迫切想了解问题的答案，以探讨观众对科幻电影评价的普遍趋势及其可能的解释。

初步探索



我爬取了豆瓣top10%的科幻电影和相对更受欢迎的剧情电影的评分并进行比较。初步数据分析表明，与剧情电影相比，科幻电影的评分整体偏低0.3分左右。由此可见，低分高质量的现象确实存在于科幻电影中。我猜测，科幻作为小众电影题材，观影门槛的提高可能是一个关键因素。

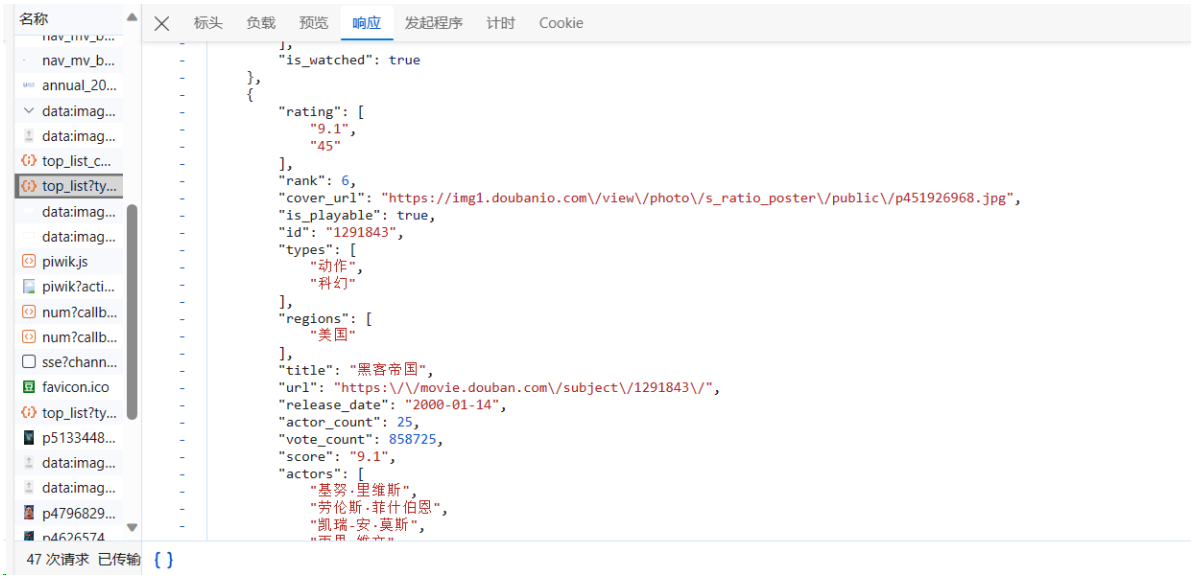
数据获取

我想到了几个因素可能与观众对科幻电影的评分有关：观影（评论）时间、用户所在地区、用户观影历史、用户对科幻电影总体的评价。因此我尝试对上述数据进行爬取。

豆瓣爬取数据是一项复杂而艰难的任务，主要原因有以下几点：

1. **反爬机制：** 豆瓣在这几年不断增进了强大的反爬机制，采取了一系列手段来防止数据抓取，包括限制请求频率、使用验证码、检测爬虫行为等。许多网上过往成功的爬取案例我发现都不再有效。
2. **动态加载内容：** 豆瓣采用了动态加载技术，通过JavaScript在页面加载完成后再加载部分内容，而不是简单地从页面的HTML代码中提取信息。
3. **用户登录限制：** 豆瓣限制了未登录用户的访问权限。要获取更全面的数据，我通过设置一系列的user_agent和cookie，来实现登录模块、处理用户身份验证和Cookie管理，增加了获取数据的复杂性。

1.获取豆瓣top10%科幻电影编号



在对网页进行分析时我发现电影排行榜的信息存储在json格式的页面中。豆瓣在这里采用了前后端分离的架构，前端通过异步请求获取 JSON 数据，而不是直接在 HTML 中嵌入数据。

```
def __init__(self):

    self.headers = {
        'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36',
        'Referer':'https://movie.douban.com/typerank?type_name=%E7%A7%91%E5%B9%BB&type=17&interval_id=100:90&action=',
        'Cookie':'bid=fXh3tiHb2zo; _ga_RXNMP372GL=GS1.1.1687533043.1.1.1687533052.51.0.0; _pk_id.100001.4cf6=e3a470da9c31bf12.1687533068.; __utm
    }
    self.requestt()
```

1.设置伪装。目的是通过在HTTP请求头中提供一个虚假的用户代理信息，使服务器认为请求是从特定浏览器或设备发出的。这样做的主要目的有以下几点：通过设置伪装，可以让程序的请求头看起来像是来自真实浏览器的请求。这有助于模拟人类用户的访问行为，减小被豆瓣识别为爬虫的风险。

```
def request(self):

    num = 0
    # 设置json页码规律
    for page in range(10):
        # 目标网页
        url = 'https://movie.douban.com/j/chart/top_list?type=17&interval_id=100%3A90&action=&start={}&limit=20'.format(num)
        num += 20

        self.responsee = requests.get(url, headers=self.headers)

        self.parse()
```

2.找到排行榜的json页码规律，每页存储20个电影信息。循环遍历电影排行榜以获得电影编号。

```
def parse(self):

    print(self.responsee.json())

    for i in self.responsee.json():
        # 获取连接
        keynum = i['url']
        # 提取编码
        keynum = re.findall('https://movie.douban.com/subject/(\d+)/', keynum)[0]

        self.write.writerow([keynum])
```

3.通过分析刚才的json信息，用xpath提取出想要的编码，写入编码.csv。

2.获取短评

```
self.write.writerow(['用户链接', '用户名', '星级', '评论内容', '时间', '有用数', '地区'])
self.headers = {
    # 设置伪装， 注意cookie
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36',
    'Cookie': 'bid=oromBooD0_Q; __gads=ID=1e85c8ee63650842-228f5a192cd8004f:T=1668070627:RT=1668070627:S=ALNI_Ma'
```

1.设置伪装。

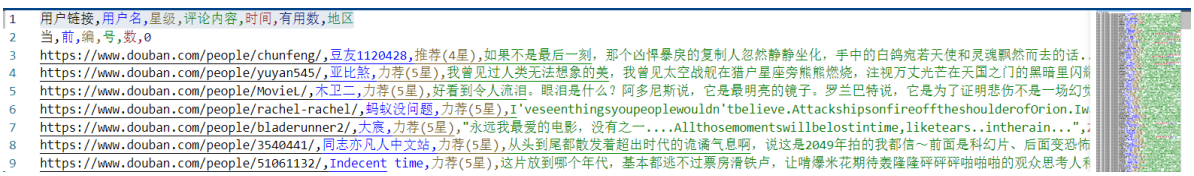
```
a = 0
# 遍历编号文件中的数据
for l in reader:
    print('当前编号数{}'.format(a))
    self.write.writerow(['当前编号数{}'.format(a)])
    a += 1
    print(l[0])

    # 从好评、差评等不同页面获取评价，保证数据的多样性
    for page in range(3):
        if page == 0:
            num = 0
            for i in range(3):
                # 构建URL，获取好评页面评论
                url = 'https://movie.douban.com/subject/{}/comments?percent_type=h&start={}&limit=200&status=P&sort=new_score'.format(l[0], num)
                num += 200
                self.parse(url)
        elif page == 1:
            num = 0
            for i in range(3):
```

2.找到短评的页码规律。根据爬取到的科幻电影的编号，读取每部电影的编号数并将其填入url进行按页爬取。

```
def parse(self, url):
    # 发送HTTP请求获取页面内容
    response = requests.get(url, headers=self.headers)
    aimtext = response.text
    text = etree.HTML(aimtext)
    # 获取所有用户数据
```

3.获取页面内容。


```
# 遍历评论区域
text = text.xpath('//div[@id="comments"]/div')[:-1]
for item in text:
    try:
        # 提取用户链接、用户名、有用数、评论内容等信息
        link = item.xpath('.//h3/span[2]/a/@href')[0]
        name = item.xpath('.//h3/span[2]/a/text()')[0]
        youyongshu = item.xpath('.//h3/span[1]/span/text()')[0] + '有用'
        content = item.xpath('.//span[@class="short"]/text()')[0]
        content = re.sub('[\n\t\r\s]', '', content)
        try:
            xing = item.xpath('.//h3/span[2]/span[contains(@class, "rating")]/@class')[0].split(' ')[0][-2]
            xingji = item.xpath('.//h3/span[2]/span[contains(@class, "rating")]/@title')[0] + '(' + xing + '星'+ ")"
        except:
            xingji = '无'
        time = item.xpath('.//h3/span[2]/span[@class="comment-time "]/text()')[0]
        time = re.sub('[\s\t\n\r]', '', time)
        try:
            region = item.xpath('.//h3/span[2]/span[@class="comment-location"]/text()')[0]
            region = re.sub('[\s\r\t\n]', '', region)
        except:
            region = '无'
        region = re.sub('[\s\r\t\n]', '', region)
        # 输出信息并写入文件
        print(link, name, xingji, content, time, youyongshu, region)
        self.write.writerow([link, name, xingji, content, time, youyongshu, region])
    except:
```



4.用xpath提取出想要的信息，存储到短评.csv中。

3.获取观影历史：科幻观影数量、科幻电影平均评分、总观影数量

1.删除短评.csv中重复的用户主页链接，提取用户编号。

```
用户链接
https://movie.douban.com/people/chunfeng/collect?start={}&sort=time&rating=all&filter=all&mode=grid
https://movie.douban.com/people/yuyan545/collect?start={}&sort=time&rating=all&filter=all&mode=grid
https://movie.douban.com/people/MovieL/collect?start={}&sort=time&rating=all&filter=all&mode=grid
https://movie.douban.com/people/rachel-rachel/collect?start={}&sort=time&rating=all&filter=all&mode=grid
https://movie.douban.com/people/bladerunner2/collect?start={}&sort=time&rating=all&filter=all&mode=grid
https://movie.douban.com/people/3540441/collect?start={}&sort=time&rating=all&filter=all&mode=grid
https://movie.douban.com/people/51061132/collect?start={}&sort=time&rating=all&filter=all&mode=grid
```

2.找出用户看过的电影页面的页码规律，将用户编号填入完成url，存储到用户链接_新形式.csv中。

3.循环遍历用户链接_新形式.csv，以获取每个观众的观影历史。

```
try:
    proxy = random.choice(self.proxies)
    # 发送HTTP请求以获取评论页面的内容
    response = requests.get(url, headers=self.headers, proxies={'http': proxy, 'https': proxy})
    print(f"Response Status Code: {response.status_code}")  # 添加这一行
    aimtext = response.text
    text = etree.HTML(aimtext)

    if text is None:
        print("未成功获取到 text 对象")
        return None

    # 提取包含已观看电影数量的字段
    movie_count_element = text.xpath('//title/text()')
    if movie_count_element:
        movie_count_text = movie_count_element[0]
        # 用正则式提取数字
        match = re.search(r'\((\d+)\)', movie_count_text)
```

4.提取每个观众的总观影数量，并计算已观看电影页面的循环次数，以进一步获取每个观众自己的科幻观影数量和科幻电影平均评分。

```
for movie_item in movie_items:
    try:
        movie_count += 1

        # 找到评分标签
        rating_tags = movie_item.xpath('.//span[contains(@class, "rating") and contains(@class, "t")]/@class')

    # 检查是否找到评分标签
        if rating_tags:
            rating_value = int(re.search(r'\d+', ' '.join(rating_tags)).group())
            # 将评分添加到总数
            movie_rating += rating_value

            # 使用 XPath 表达式获取电影名字和判断是否为科幻电影
            title_element = movie_item.xpath('.//li[@class="title"]/a/em/text()')[0]
            genre = movie_item.xpath('.//li[@class="intro"]/text()')[0]

            # 判断是否包含 "科幻"
            is_sci_fi = "科幻" in genre
            # 打印结果
            if is_sci_fi:
                sci_fi_movie_count += 1
                sci_fi_movie_rating += rating_value
```

5.用xpath提取已观看的电影类别是否为科幻，进行统计。

很遗憾地是由于豆瓣网页信息的限制，我最终只拿到了部分评论和评论者的信息。

数据预处理

1. **数据清洗：**

   填补地址空缺值：部分用户的所在地区未能获取。由于数据集过于庞大，所以我选择直接忽略存在地区属性缺失的元组。

2. **数据集成：**

   冗余数据处理：同一观众很可能在不同电影下重复评论。为了方便后续的统计分析，只保留一个即可。

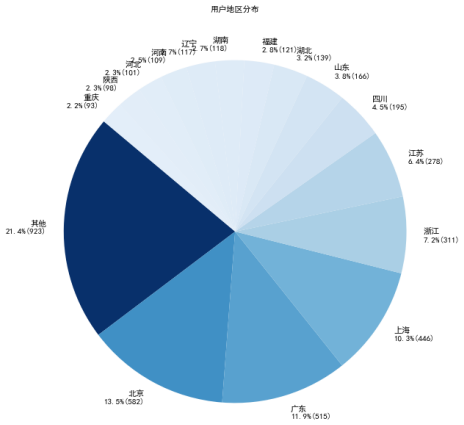
3. **数据变换：**

   数据类型变换：提取到的时间格式形如2009-10-0801:14:48，无法直接进行数据分析，需要进行数据类型变换以便分析。

探索性分析

结构化数据

1. **地区：**



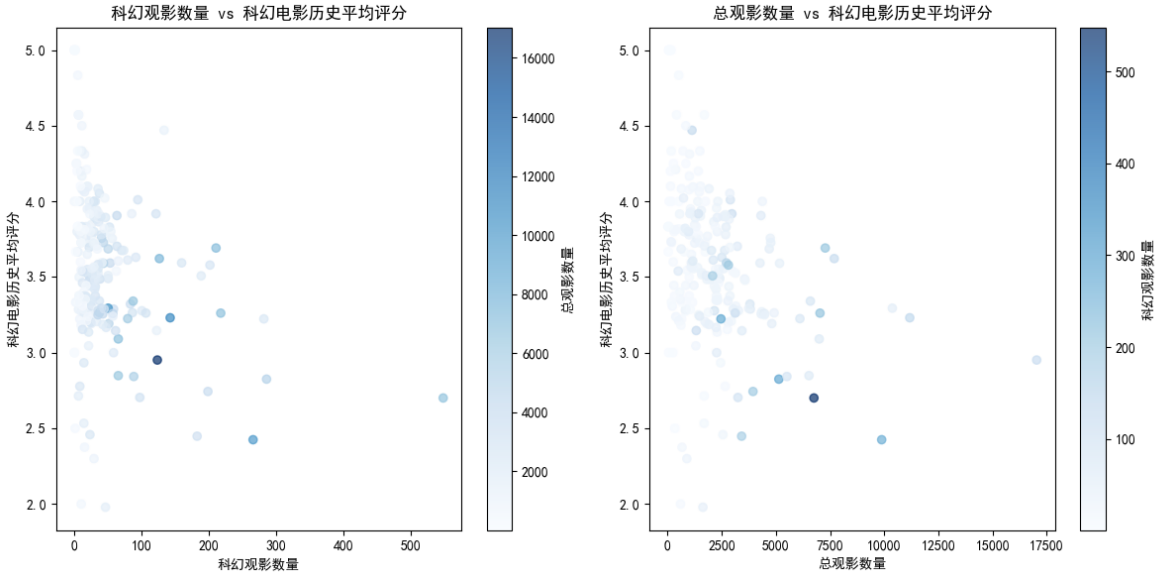
1. **主要地区分布：** 北京、广东、上海等地区的评分者数量相对较多，占比较高。
2. **区域多样性：** 评分者分布涵盖了全国各地，显示了科幻电影在不同地区的受欢迎程度。
3. **地区差异：** 不同地区的评分者数量差异较大，可能反映了科幻电影在不同地区的受欢迎程度或电影市场的发展水平。

2. **观影时间**

1. **高峰年份：** 评论数量的高峰主要出现在2012年左右，这一时期可能与多部热门科幻电影上映、豆瓣平台用户增长等因素有关。
2. **稳定阶段：** 自2017年以后，科幻电影评论数量呈现出相对稳定的趋势，这可能反映了科幻电影在这段时间内的持续受欢迎，并且豆瓣平台用户维持在一个较高水平。

3.**用户观影历史&科幻类别观影历史：**

	平均值	中位数	标准差
科幻观影数量	42.778281	26	59.395980
总观影数量	2136.140271	1684	2114.130039
科幻电影历史平均评分	3.571766	3.555556	0.503250



1. **观众科幻电影数量分布：**平均每位观众观看了约42部科幻电影；一半的观众观看的科幻电影数量少于26部；观众观看科幻电影数量的离散程度相对较大。
2. **观众观看过的所有电影数量分布：**观众总观影数量的离散程度相对较大。
3. **观众对科幻电影的平均评分分布：**观众对科幻电影的平均评分相对较稳定。
4. **综合分析：**

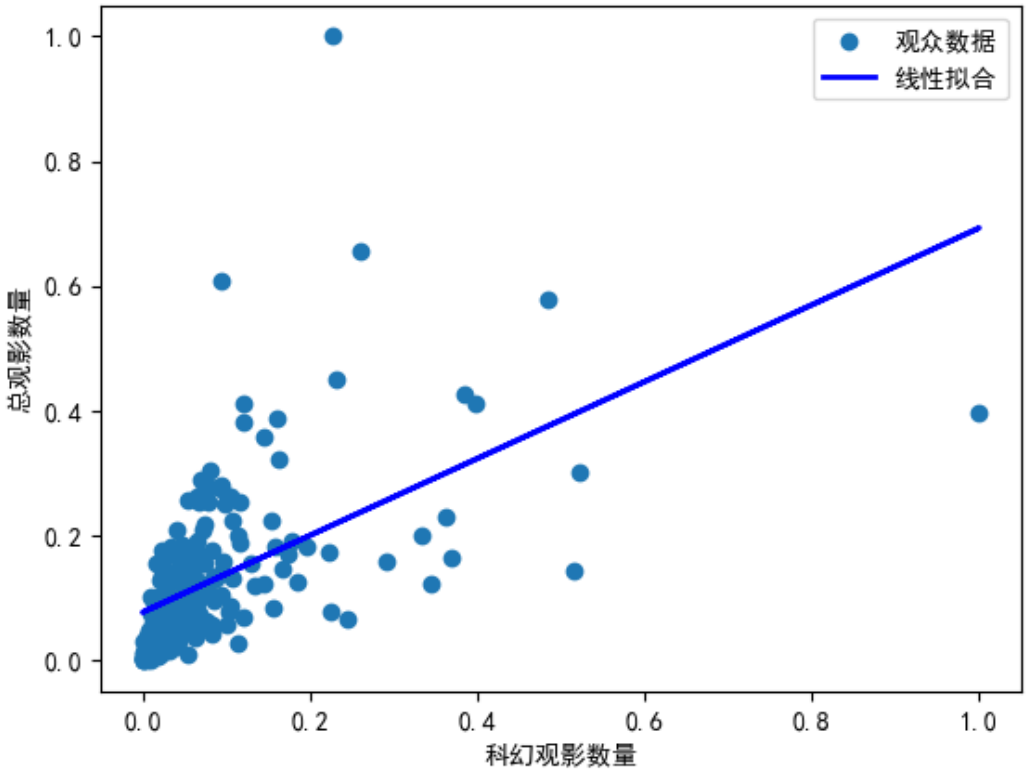
   观影习惯差异： 观众在观看科幻电影的数量上存在较大的个体差异。这可能反映了观众在对电影类型的偏好上存在差异。但是我认为无法确定这样的差异是否是总观影数量的差异造成的，需要进一步的数据分析。

   平均评分相对稳定： 科幻电影历史平均评分的标准差相对较小，这可能表示观众在平均水平上对科幻电影有较为一致的评价。

由于上述数据不能够直接判断用户观影历史对评分的影响，需要对数据进一步探索。

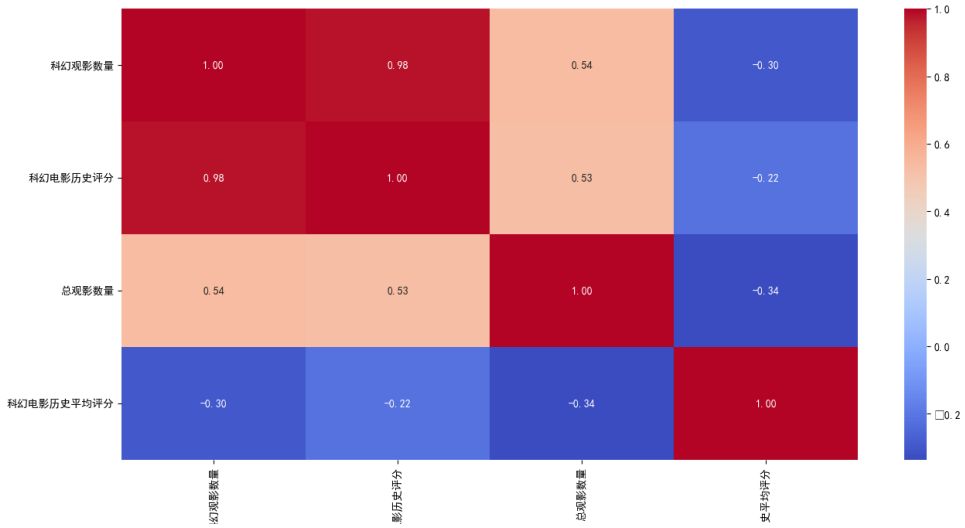
特征工程

数据归一化：采用Min-Max 归一化方法将科幻类别观影数量和总观影数量归一化到 [0, 1] 区间，分析两个数量之间的相对大小关系。



线性拟合模型：y = 0.6158 * x + 0.0773

不难看出拟合直线的斜率小于1，表示着观众总体观影数量相对较大，但其中包含的科幻电影观影数量相对较少。这可能暗示着科幻电影是一个相对小众的题材，只有部分观众对其有较高的兴趣。这与我之前的猜测相符合。



科幻电影历史平均评分与科幻电影历史评分、科幻观影数量的相关性系数分别为-0.22、-0.3，表明存在一定程度的负相关性。通常情况下，如果观众倾向于给予科幻电影较高的评分，那么科幻观影数量可能会略微减少，反之亦然。

这一现象可能有多种解释，其中一个较为合理的解释是：科幻观影数量较少的观众往往更专注于观看评分较高、质量较高的科幻电影，因此更愿意为这些影片给予较高的评分；而科幻观影数量较多的观众由于涉及的电影更为广泛，因此其平均评分可能受到质量参差不齐的影响，因而相对较低。这与我们的直觉相符合。

数据建模

为了更准确地反映观众的整体倾向和品味，我引入了观影历史加权评分的概念。该评分考虑了观众对科幻电影的历史观影数量和评分，通过权重调整体现了观众的整体科幻电影品味。这种权重调整机制使最终评分不仅取决于当前电影的个体评分，还考虑了观众对该类型电影的整体喜好和经验。

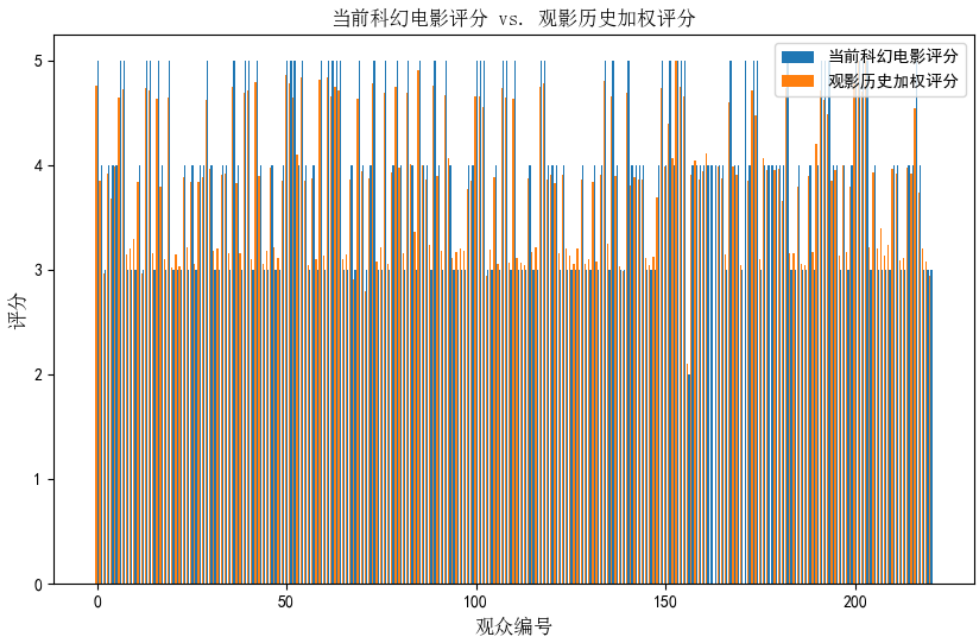
最终评分=(1−α)×个体评分+α×观影历史评分

其中：

- 最终评分：考虑观影历史权重后的最终评分。
- 个体评分：观众对当前电影的个体评分。
- 观影历史评分 = ((∑科幻电影历史评分) / 科幻电影观影次数)，也就是科幻电影历史平均评分
- 观影历史权重系数 α，其取值范围为 0 到 1，表示观影历史在最终评分中的权重比例。

考虑到科幻电影的普遍评分较低，这一权重调整机制有助于平衡观众的个体倾向和对科幻电影整体质量的考量。通过这样的调整，我们可以更全面地理解观众对科幻电影的评价，从而提高评分的准确性和代表性。

我先将α的值设为0.2，对观众的评分重新进行运算。（可惜的是之前爬取数据的时候没有考虑到这一步，因此这里的个体评分随机生成，主要由3-5分构成）



可以看出虽然4-5分段的评分大部分都下降了0.1-0.2分左右，但是3分段基本都上升0.1分左右。在科幻电影评分为3分段为主的背景下，这个公式很可能使得一部科幻电影的评分小幅度上升，结果比较理想。

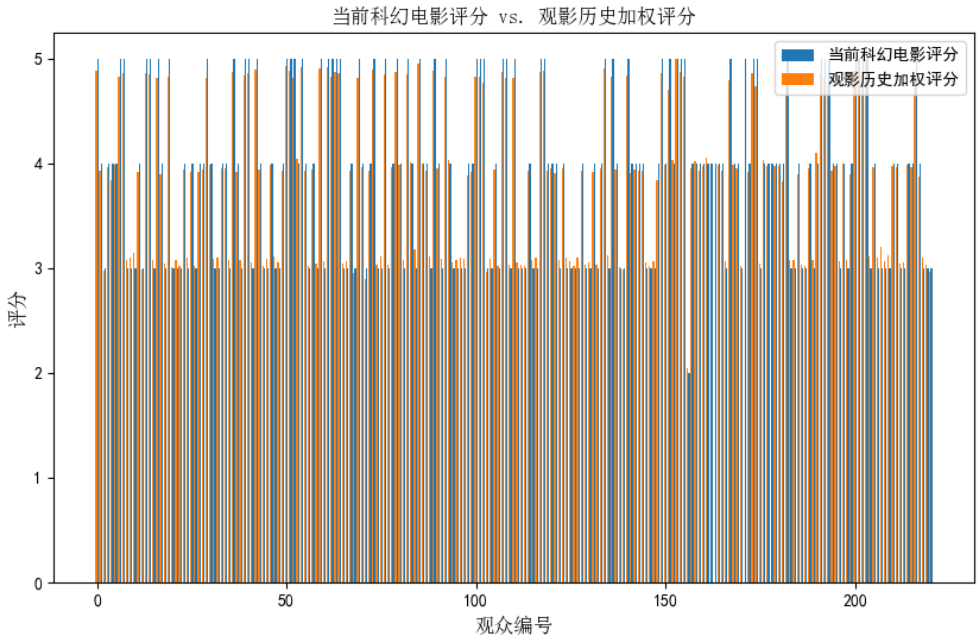
为了寻找更合适的α，使得最终评分能够反映观众的整体倾向的同时更准确地拟合实际数据，我选择使用线性回归模型，并通过网格搜索进行超参数调优。通过网格搜索，可以尝试不同的alpha值，从而找到在验证集上表现最好的超参数。通过上述操作选择出α，最终评分的计算得以在考虑观众整体喜好和经验的同时，更好地拟合实际数据，达到更合适的权衡。

```python
# 创建Lasso回归模型
lasso_model = Lasso()

# 定义观影历史权重系数 alpha 的取值范围
alphas = [0.1, 0.2, 0.3, 0.4, 0.5]

# 创建参数网格
param_grid = {'alpha': alphas}

# 创建均方误差评分对象
mse_scorer = make_scorer(mean_squared_error)

# 创建 GridSearchCV 对象
grid_search = GridSearchCV(lasso_model, param_grid, scoring=mse_scorer, cv=5)

# 准备特征和目标变量
X = train_df[["科幻电影历史平均评分"]].fillna(train_df["科幻电影历史平均评分"].mean())
y = train_df["当前科幻电影评分"]

# 执行网格搜索
grid_search.fit(X, y)
```

```
ore/daseINTRO/final/final-6-数据建模/调整参数.py
最佳参数: {'alpha': 0.1}
最佳均方误差: 0.6440786155409044
PS C:\Users\15910\Desktop\FCMU\sophomore\daseINTRO\final>
```

在我的问题中，我发现最优参数α总是对应于取值范围中的最小值 。可能原因是特征值只选取了一个，模型过于简单导致。这并不意味着较小的α始终是最佳选择，因为具体选择需要根据实际需求和模型性能进行综合考虑。因此在这个情况下，我选择了[0.1，0.2，0.3，0.4，0.5]作为取值范围。

调整后的α=0.1对观众的评分重新运算后的结果如下图所示。



最终评分的计算中，与初始的个体评分相比，不难观察到变化的缩小。相较于α=0.2 时，现在最终评分的计算更加充分地考虑了用户的实际评分。事实上，用户的实际评分是电影评分中最为重要且应当最为依赖的数据。

结论

通过这次实验，我发现了观众对科幻电影评分存在的问题，对豆瓣科幻电影评分相关数据有了更多的了解，并且提出了一种基于观影历史的调整机制，使评分在结合实际数据的同时反映观众的整体倾向和品味。这种方法不仅能够帮助我们更好地理解观众对科幻电影的评价，也为科幻电影乃至小众电影的评分系统提供了一种改进的思路。

过程中遇到的问题

问题：在数据获取阶段，爬虫频繁遭遇豆瓣的403错误。

原因：该问题的发生主要源于爬虫对豆瓣服务器的请求频率超出了豆瓣的访问频率限制和数量限制，导致服务器拒绝进一步的访问。

解决方法：尽管我尝试了多种方法，包括使用time.sleep降低请求频率等等，但由于豆瓣逐步更新其反爬虫机制，最终未能完全解决这个问题。虽然我找到了一些高质量的代理IP，但由于数据量庞大且获取耗时较长，这些IP也很快被豆瓣封禁。

问题：在特征工程阶段，数据处理的结果不够理想。

原因：获取的数据数量不够；爬取的数据不够全面：首先我获取的短评是从热门界面进行爬取的，所以评分者倾向于阅片量充足的豆瓣活跃用户，阅片量偏低的普通用户样本量不够；其次，我应该再针对每位观众对top10%科幻电影的平均评分进行爬取和计算，而非简单地根据科幻电影平均评分进行计算，从而忽略了最重要的元素——科幻电影的质量的影响。

吸取的经验：在之后的数据科学实验中，我需要在开始更完整地思考实验过程需要的数据。

问题：在数据建模阶段，计算最终评分时个体评分缺失。

原因：之前没有爬取需要的数据，且ip被封无法继续爬取。

解决方法：根据观众对科幻电影的评分规律随机生成了个体评分。