中国地质大学

《数据采集与预处理》结课报告

—以最受欢迎电影评论分析为例

授课教师：樊俊青

学生姓名：钱晓楠

学生学号：20221003675

学生班号：195221

时　　间：2024 年 5 月 7 日

目录

一、项目背景

豆瓣电影是目前国内最受欢迎的电影评价网站之一，用户可以在该网站上进行电影评分、评论、收藏等操作。由于豆瓣电影上的数据非常丰富，因此对这些数据进行采集以及预处理、对收集到的数据进行分析已经可视化研究有助于我们深入了解电影市场的发展趋势和用户评价偏好，为电影制作和营销提供有益的参考。

随着社会的不断发展，电影产业也在不断壮大，越来越多的人开始重视电影的文化价值和商业价值。豆瓣电影上的评分和评论成为了衡量电影品质和受欢迎程度的重要标准之一。通过对豆瓣电影上的数据进行采集与预处理、分析和可视化研究，我们可以深入了解用户对电影的评价和偏好，挖掘电影市场的发展趋势和商业机会，为电影制作和营销提供更有针对性的建议和策略。

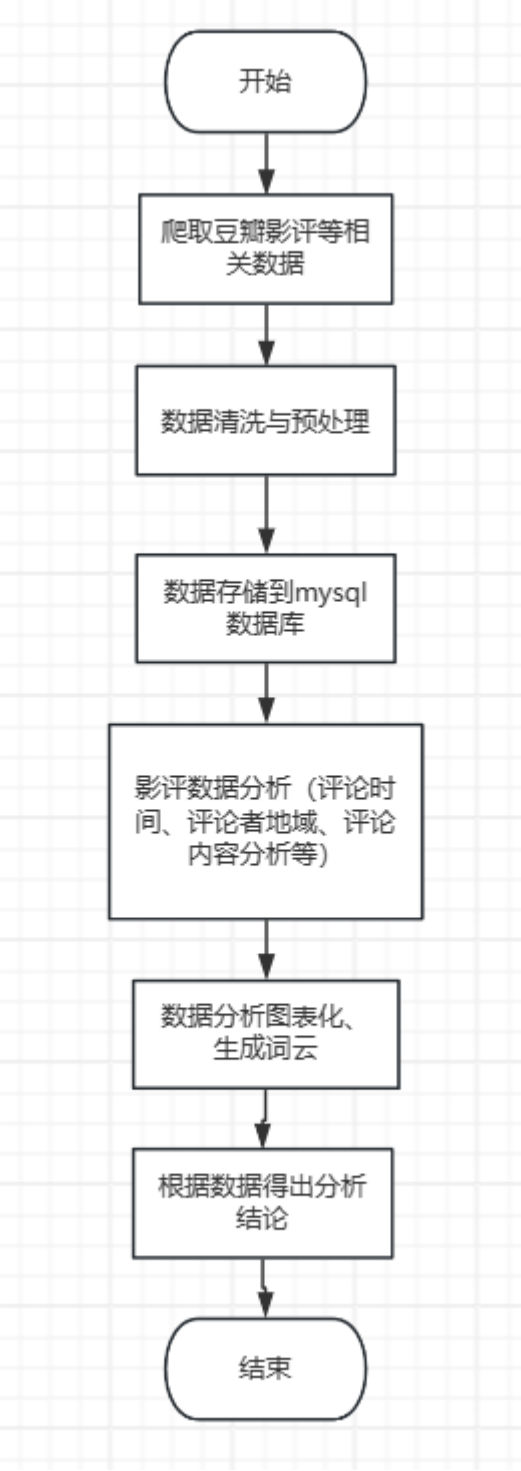
2019 年 2 月 5 日电影《流浪地球》正式在中国内地上映。该电影在举行首映的时候，口碑好得出奇，所有去看片的业界大咖都发出了同样赞叹。文化学者戴锦华说：“中国科电影元年开启了。”导演徐峥则说，“里程碑式的电影，绝对是世界级别的”。可是公映之后，《流浪地球》的豆瓣评分却 8.4 一路跌到了 7.9。影片页面排在第一位的，是一篇一星影评《流浪地球，不及格》。文末有 2.8 万人点了“有用”，3.6 万人点了“没用”。

关于《流浪地球》的观影评价，已经变成了一场逐渐失控的舆论混战，如“枪稿”作者灰狼所说，“关于它的舆论，已经演化成‘’政治正确、水军横行、自来水灭差评、道德绑架、战狼精神”。为了

对《流浪地球》的观影评价有个全面的了解，对《流浪地球》的豆瓣影评数据进行挖掘和分析。

本项目以国内提供最新的电影介绍及评论包括上映影片的影讯查询及购票服务的豆瓣电影为数据源，获取了 2019 年初争议比较大的《流浪地球》豆瓣短评，并对获取的短评数据进行了预处理和评分挖掘与分析。

项目实施步骤见下图：



1. 设计豆瓣电影爬虫程序，获取电影数据

获取数据的时候，需要模拟浏览器对网站进行请求，需要加入请求头，然后分析不同JSON数据包中的参数，发现具体的规律之后可以设置对应的程序进行获取数据集。如果IP频繁的访问网站不仅会给目标网站带来负载压力，还会被网站识别为恶意爬虫，所以设计爬虫程序的时候需要加入延时函数，增强爬虫的稳定性。

其次由于有时候获取数据的时候，会有有一些字段在某些电影中不存在，所以为了保障程序的稳定健壮的持续运行，需要设置智能化爬虫。初步的分析需要对字段的数据值进行一个判断，如果没有获取到数据，那么就自动赋值为空值，这样就可以避免程序中断。

2. 对爬取到的数据进行清洗和预处理

由于我们获取的大量的数据中，存在一些不规则的字段，我们需要对其进行结构化清洗，保证数据的有效性，便于后续的分析，其次数据中存在一些空值，需要进行处理，然后将其保存为一个新的数据。

其次，在处理时间字段的时候，需要将原本的数据类型转化为date类型，可以方便后续的数据分析，增加分析的维度，保障数据的有效进行。

3. 将清洗好的数据存储到MySQL数据库中

将预处理好的数据存入在MySQL中，便于后续的管理和调用数据，MySQL作为一个结构化的数据库，可以存储大量的数据，并且可以帮助我们采用SQL语句进行查询和数据分析，具有非常高效的特点。

4. 豆瓣电影影评数据字段多维度数据分析

采用数据分析思维，调用数据库中的结构化数据，从多个维度对豆瓣电影数据进行深入分析，比如评论者地域分析，评论时间维度分析，电影影评分析等。

5. 利用matplotlib可视化库，绘制各种多维度的图表

利用Python的matplotlib这个第三方可视化库，调用数据库中的数据，实现数据的可视化，便于我们进行发现规律，给用户或者其他人员提供决策性的支持依据。

6. 对分析结果进行论述，提供数据分析结论

对分析数据进行分析，提出相关性的结论，以及出现的规律性的结论。

二、数据采集

足够的数据量是保证企业进行数据分析和数据挖掘的基础，因此使用何种数据采集方法是获取大量数据的关键。当前的数据采集方法

主要有数据库采集、系统日志采集、网络爬虫采集、感知设备数据采集等，本项目拟通过Python网络爬虫作为主要的数据采集工具（或者其他工具），对豆瓣网站关于电影《流浪地球》的观影者的电影影评等关键信息进行爬取与存储。

具体采用的python爬虫技术为利用Xpath语言进行信息爬取，数据采集的网络爬虫总流程如下图 1 所示，具体影评数据提取部分流程如图 2 所示。

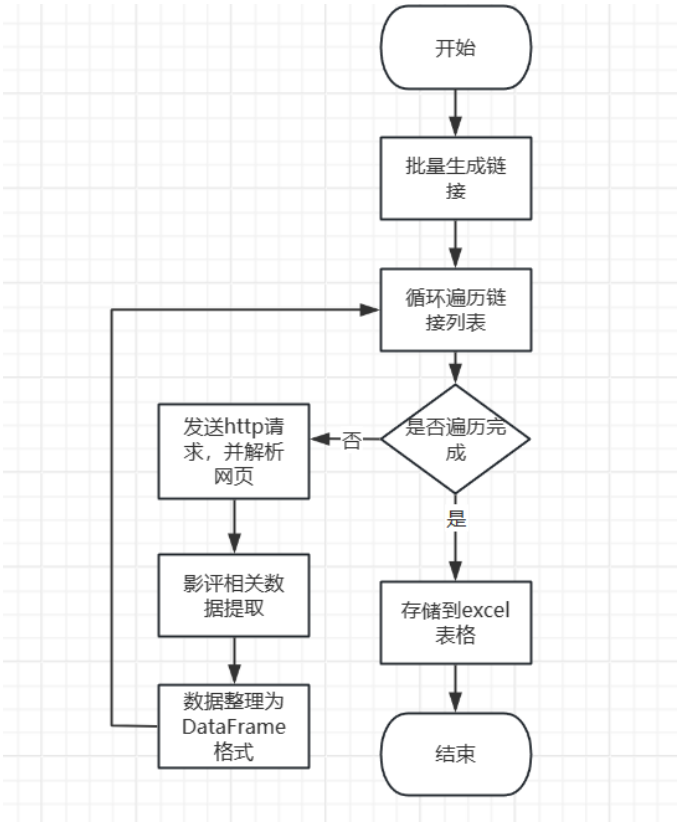

开始
批量生成链接
循环遍历链接列表
是否遍历完成
否
发送http请求，并解析网页
影评相关数据提取
数据整理为DataFrame格式
是
存储到excel表格
结束

图 1

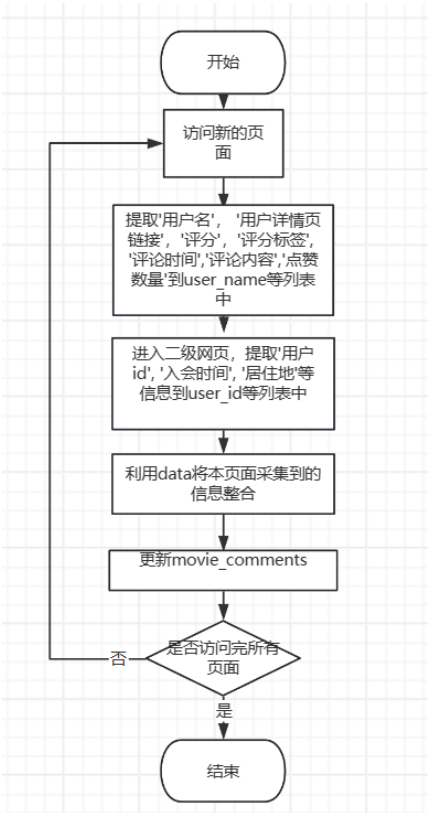

开始
访问新的页面
提取'用户名'，'用户详情页链接'，'评分'，'评分标签'，'评论时间','评论内容','点赞数量'到user_name等列表中
进入二级网页，提取'用户id'，'入会时间'，'居住地'等信息到user_id等列表中
利用data将本页面采集到的信息整合
更新movie_comments
是否访问完所有页面
否
是
结束

图 2

2.1 变量介绍

urls：包含了豆瓣电影页面的短评链接列表。

movie_comments：存储所有评论目录页的字段信息的DataFrame

'用户名': user_name,

'用户详情页链接': user_href,

'用户id': user_id,

'入会时间': user_time,

'居住地': address,

'评分': score,

'评分标签': score_label,

'评论时间': comment_time,

'评论内容': comment_content,

'点赞数量': number_like

Data:DataFrame类型，每次访问一个页面后将这一页提取到的影评信息整合起来。用于更新movie_comments

Data定义：

```
data = pd.DataFrame({
    '用户名': user_name,
    '用户详情页链接': user_href,
    '用户id': user_id,
    '入会时间': user_time,
```

```
        '居住地': address,
        '评分': score,
        '评分标签': score_label,
        '评论时间': comment_time,
        '评论内容': comment_content,
        '点赞数量': number_like
    })
```

2.2 详细设计

1、 对于每一页进行一次网页请求及解析，每每访问一次就可以得到一个长度为 20 的用户信息列表（'用户名'，'用户详情页链接'，'评分'，'评分标签'， '评论时间','评论内容','点赞数量'）

2、 由于用户id、入会时间以及用户居住地在用户详情页。所以我们再进入一个循环，遍历访问用户详情页链接列表，利用xpath语言提取'用户id', '入会时间', '居住地'，将其存储在另外三个列表中。

3、 访问完一个页面后，调用sleep函数设置时间间隔，防止被识别为爬虫程序。并将提取到的数据存储在data中，进而利用data更新movie_comments。当访问完所有页面后，信息提取完成。

4、 最后将网络爬取到的信息存到excel文件中。

2.3 技术细节

a) 批量生成链接： 使用循环结构批量生成豆瓣电影页面的短评链接，每次递增 20 条评论。

b) 发送请求： 使用 requests 库发送 HTTP 请求，并设置了请求头和 cookies，模拟浏览器行为。

c) 数据解析： 使用 XPath 语法解析 HTML 页面，定位所需数据的位置，并使用 etree 库进行解析。

d) 数据整理： 将提取的数据整理为 DataFrame 格式，方便后续的分析和处理。

e) 数据存储： 使用 pandas 库将 DataFrame 中的数据保存到 Excel 文件中，便于后续使用和分享。

三、数据清洗及预处理

3.1 数据类型转换

对通过爬虫所获取得到的评论数据集进行分析后，我们发现数据集中存在数据缺失，错误等问题，且需要进行部分数据类型的转换。

首先我们打印出movie_comments的基本信息：

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220 entries, 0 to 219
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   用户名       220 non-null    object
 1   用户详情页链接  220 non-null    object
 2   用户id      220 non-null    object
 3   入会时间      220 non-null    object
 4   居住地       220 non-null    object
 5   评分        220 non-null    object
 6   评分标签      220 non-null    object
 7   评论时间      220 non-null    object
 8   评论内容      220 non-null    object
 9   点赞数量      220 non-null    int64
dtypes: int64(1), object(9)
memory usage: 17.3+ KB
```

1、 由于用户信息（用户id、入会时间以及用户居住地）是通过二级链接查询，由列表追加得到的，在数据框展现为列表，故需转化，清洗前的效果参见图 1。

以处理'用户id'列为例，详细介绍清洗方法：

1) 定义一个 lambda 函数，对每个元素应用两次 .replace() 方法，去除字符串中的 "['\n " 和 " ']" 部分，将字符串中的特定部分替换为空字符串。这样可以清洗掉多余的空格和引号。

2) 使用.apply() 方法，该方法用于将一个函数应用到 movie_comments['用户id']中，将每一个用户id进行清洗。

3) 这样处理后，'用户id'列中的每个元素应该都变得更加干净。

由于居住地一栏有缺失值，故还要进行缺失值处理，如果居住地一栏为空，则用“缺失”填充。

清洗三列数据所用到的源代码：

```
#1.先处理列表对象的这三列：用户id、入会时间、居住地
movie_comments['用户id'] = movie_comments['用户id'].apply(lambda x:x.replace("['\\n","").replace(" ']",""))
movie_comments[' 入 会 时 间 '] = movie_comments[' 入 会 时 间 '].apply(lambda x:x.replace("['\\n ","").replace("加入 ']",'').replace("加入 ', '\\n ']","""))
# 同时处理了缺失值，填充为“缺失”
movie_comments[' 居 住 地 '] = movie_comments[' 居 住 地 '].apply(lambda x:x.replace("['","").replace("']",'').replace("[]","缺失"))
```

清洗后的数据如图 2 所示，可见数据成功被清洗为指定格式：

	C	D	E
	用户id	入会时间	居住地
	['\n xzfd ']	['\n 2008-03-04加入 ']	['北京']
	['\n 159673853 ']	['\n 2017-03-29加入 ']	['Antigua and Barbuc
	['\n tjz230 ']	['\n 2005-07-18加入 ']	['北京']
	['\n dreamfox ']	['\n 2008-03-12加入 ']	['北京']
	['\n lingrui1995 ']	['\n 2012-08-07加入 ']	['四川成都']
	['\n zhangzongqian '	['\n 2009-04-14加入 ']	['北京']
	['\n yimaxxduck ']	['\n 2009-12-01加入 ']	['上海']
	['\n nothingbut ']	['\n 2007-12-07加入 ']	[]
	['\n 114515320 ']	['\n 2014-12-06加入 ']	['北京']
	['\n chaneagle ']	['\n 2010-08-22加入 ']	['浙江杭州']
	['\n singlesinger '	['\n 2008-02-05加入 ']	['北京']
	['\n jishaoting ']	['\n 2007-09-08加入 ']	['北京']
	['\n yyy18n ']	['\n 2009-12-20加入 ']	['广东深圳']
	['\n alexcite ']	['\n 2009-10-03加入 ']	['北京']
	['\n 88439681 ']	['\n 2014-05-07加入 ']	['浙江杭州']
	['\n zhangxy1326 ']	['\n 2008-09-27加入 ']	['上海']
	['\n 1233038 ']	['\n 2006-08-12加入 ']	['北京']
	['\n MovieL ']	['\n 2006-03-29加入 ']	['广东佛山']
	['\n 3540441 ']	['\n 2009-01-25加入 ']	[]
	['\n problemchildrer	['\n 2009-06-19加入 ']	['广东广州']
	['\n darkwood ']	['\n 2006-07-22加入 ']	['上海']
	['\n diewithme ']	['\n 2009-07-18加入 ']	['北京']
	[]	[]	[]
	['\n graycat ']	['\n 2010-01-22加入 ']	['广东广州']
	['\n phoenix43 ']	['\n 2009-03-08加入 ']	['湖北武汉']
	['\n reave ']	['\n 2007-08-10加入 ']	['北京']
	[]	[]	[]
	['\n Uvo ']	['\n 2005-10-03加入 ']	[]
	['\n kidneytopo ']	['\n 2007-11-03加入 ']	['北京']

图 1

用户id	入会时间	居住地
1233038	2006-08-12	北京
xzfd	2008-03-04	北京
dreamfox	2008-03-12	北京
tjz230	2005-07-18	北京
yimaxxduck	2009-12-01	上海
nothingbut	2007-12-07	缺失
114515320	2014-12-06	北京
phoenix43	2009-03-08	湖北
chaneagle	2010-08-22	浙江
lingrui1995	2012-08-07	四川
zhangzongqian	2009-04-14	北京
yyy18n	2009-12-20	广东
singlesinger	2008-02-05	北京
jishaoting	2007-09-08	北京
Uvo	2005-10-03	缺失
alexcite	2009-10-03	北京
3540441	2009-01-25	缺失
88439681	2014-05-07	浙江
zhangxy1326	2008-09-27	上海
darkwood	2006-07-22	上海
159673853	2017-03-29	缺失
qijiuzhiyue	2005-10-09	缺失
diewithme	2009-07-18	北京
problemchildren	2009-06-19	广东
roayo	2007-08-10	北京

图 2

2、 接下来，为方便后续的数据分析工作，需要将带有时间意义的字符串转为时间类型。

#本身是一个对象，先转成字符串再转时间类型

movie_comments['评论时间'] = pd.to_datetime(movie_comments['评论时间'].apply(lambda x:str(x)))

3.2 将居住地全部改为省份

由于有的用户的居住地仅仅精确到省份，故为保持统一，我们这里将movie_comments[‘居住地’]中的元素全部改为省份。

具体操作很简单，保留前两位字符即可，如下所示：

```
#将居住地全部改为省份
movie_comments['居住地'] = movie_comments['居住地'].str[0:2]
```

3.3 缺失值处理

```
#3.缺失值处理
# 除了以上已填充“未知”的居住地外，无缺失值
print('缺失值个数：\n')
print(movie_comments.isnull().sum())
```

```
缺失值个数：

用户名          0
用户详情页链接      0
用户id         0
入会时间          0
居住地           0
评分            0
评分标签          0
评论时间          0
评论内容          0
点赞数量          0
dtype: int64
```

3.4 异常值处理

1、 评分异常值处理

观察发现评分数据包含异常值。首先，使用 .value_counts() 方法检查评分列中的值的分布情况。

利用print(movie_comments['评分'].value_counts())语句，以下是输出结果：

```
评分
3    63
4    52
5    46
2    27
1    21
-     5
Name: count, dtype: int64
```

对评分列中的异常值进行处理，决定删除评分为 '-' 的数据。

```
movie_comments = movie_comments[movie_comments['评分'] != '-']
```

再次检查评分列的值分布,确保异常值已被删除。

```
print(movie_comments[’评分’].value_counts())
```

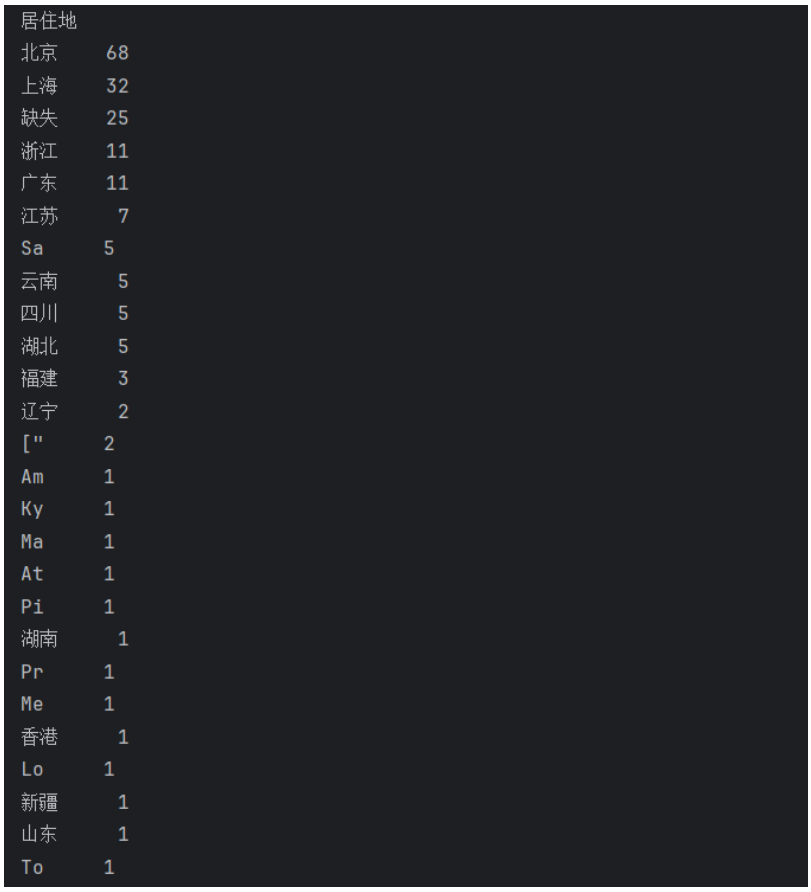
修改后结果：

```
评分
3    63
4    52
5    46
2    27
1    21
Name: count, dtype: int64
```

2、 居住地异常值处理

观察发现居住地数据也存在异常值，如包含字母的地名。首先，使用 .value_counts() 方法检查居住地列中的值的分布情况。

```
print(movie_comments['居住地'].value_counts())
```



接着，对含有字母的地名进行处理，将其替换为空字符串，然后将其替换为“缺失”，以将其归入缺失数据部分。

movie_comments['居住地'] = movie_comments['居住地'].str.replace(r'[^\u4e00-\u9fa5]','缺失')

这里使用 Pandas 中的 .str.replace() 方法来替换居住地列中的特定模式。r'[^\u4e00-\u9fa5]'是一个正则表达式模式,用于匹配居住地列中不属于汉字范围(\u4e00-\u9fa5)的字符。\u4e00 和 \u9fa5 是汉字的 Unicode 范围。

最后，再次检查居住地列的值分布，确保异常值已被处理。

print(movie_comments['居住地'].value_counts())

处理后数据：

```
居住地
北京      68
缺失      50
上海      32
浙江      11
广东      11
江苏       7
湖北       5
四川       5
云南       5
福建       3
辽宁       2
重庆       1
香港       1
新疆       1
山东       1
清迈       1
河南       1
曼谷       1
江西       1
```

3.5 重复值处理

首先检查movie_comments中完全重复的行的数量，通过duplicated()函数和numpy库中的sum()函数,其中duplicated() 函数是 pandas 库中用于检测和标记数据框中重复行的函数。它返回一个布尔类型的Series，表示每一行是否是重复的。Sum函数再统计重复行的行数。

结果表明原数据中有 6 个重复行。

```
dtype: int64
重复值的行数：6
评分
3     63
```

drop_duplicates() 是 pandas 中的一个函数，用于删除数据框中

的重复行。通过设置 inplace=True 参数，可以实现在原始数据框上直接进行修改，而不是返回一个新的数据框。利用drop_duplicates(inplace=True)函数删除所有字段都相同的行。

```
#4.重复值处理
# 输出所有值都重复的行数
print(f'重复值的行数：{np.sum(movie_comments.duplicated())}')
#去重
movie_comments.drop_duplicates(inplace=True) # 删除所有字段都相同的行
```

3.6 对影评进行文本数据预处理

由于本项目的重点是处理电影影评，属于文本类型数据，故我们这里采用了多种文本数据预处理方法。包括文本去重，机械压缩去词和短句删除，以及中文分词和停用词过滤等操作。

（1） 文本去重

1、文本去重的基本解释及原因

文本去重指的是从文本数据中删除重复的文本内容，保留唯一的文本实例。

其原因在于：

（1）提高数据处理效率：减少了重复数据的处理和分析时间，提高了处理效率。

（2）数据清洗：确保数据质量，避免重复的数据对分析结果产生误导。

（3）节省存储空间：去除重复文本可以节省存储空间，尤其对于大型数据集来说是十分重要的。

2、文本去重演算法的缺陷

效率问题：对于大规模数据集，一些传统的去重算法可能效率较低，

需要较长的处理时间。

精度问题：某些去重算法在处理短文本或语义相似但不完全相同的文本时，可能会产生误判，导致删除了重要信息。

3、文本去重选用的方法及原因

我们选用了基于简单的基于哈希值和文本内容比较的方法进行文本去重。这种方法简单高效，适用于我们的数据规模和要求。在代码中，我们使用了drop_duplicates方法来实现文本去重的操作。

```
movie_comments.drop_duplicates(subset=['评论内容'], inplace=True)
```

（2）机械压缩去词

1. 机械压缩去词思想

机械压缩去词是一种基于规则的文本处理方法，其思想是根据一定的规则对文本中的词语进行压缩和删除，以减少词汇的冗余和提取关键信息。

2. 机械压缩去词处理

在代码中，我们通过分词和替换重复出现的词语来实现机械压缩去词的操作。

```
import re
# 假设将重复出现两次以上的词进行压缩，可以利用正则表达式实现
movie_comments['评论内容'] = movie_comments['评论内容'].apply(lambda x: re.sub(r'\b(\w+)\b\s+\1\b', r'\1', x))
```

3. 压缩去词规则

我们采用了简单的规则，对重复出现的词语进行压缩。这样可以有效地减少冗余信息，提取关键信息。

（3）短句删除

1、短句删除的原因

短句删除指的是从文本数据中删除长度过短的句子或文本段落。它能够提高数据分析质量：短句往往不包含足够的信息，可能会引入噪音或干扰分析结果。它还能够简化模型建模：在一些文本分析任务中，过短的句子可能无法提供足够的语境，影响模型的建模效果。

2、文本评论字数确定

确定文本评论字数的阈值通常需要结合具体的应用场景和数据特点进行，一般而言，可以根据任务需求和数据分布情况来确定。例如，在一些情感分析任务中，较长的评论可能包含更多信息，而在某些分类任务中，短文本也可能具有较高的信息量。这里，我们视长度小于 5 的评论为短句，并进行删除。

（4）中文分词及停用词过滤

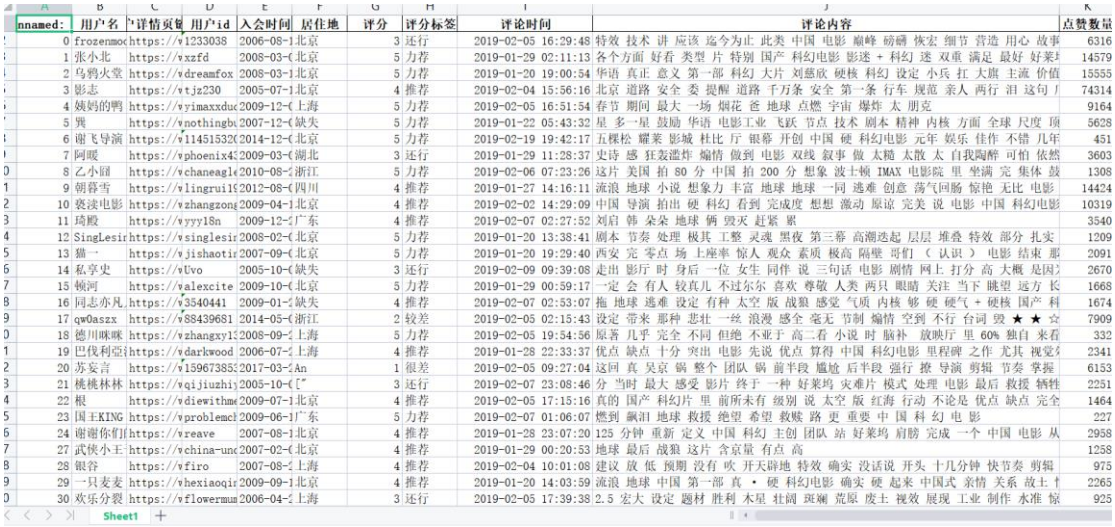
对预处理后的短评数据进行文本分词以及去除停用词操作，主要涉及涉及到中文分词库jieba以及停用词表。

在代码中，我们使用了jieba中文分词库和停用词表进行文本分词和停用词过滤操作。这样可以将评论内容分割成有意义的词语，并去除一些常见但无实际意义的词语，如“这部”、“一部”等。

```
import jieba

# 分词和去除停用词
stop_words_path="E:\edge_download\chinese-stop-words-list-master\chinese-stop-words-list-master\中文停用词库.txt"
with open(stop_words_path, 'r', encoding='utf-8') as f:
    stop_words = f.read().split()
stop_words = [' ', '\n', '这部', '.', '一部'] + stop_words
data_cut = movie_comments['评论内容'].apply(jieba.lcut)  # 分词
data_after = data_cut.apply(lambda x: [i for i in x if i not in stop_words])  # 去除停用词
movie_comments['评论内容'] = data_after.apply(lambda x: ' '.join(x))  # 将分词结果转换为字符串
```

进行简单分词的结果以及进行停用词过滤后的文本head如下图所示：

```
0    [1, ., 从, 特效, 和, 技术, 上, 讲, ，, 这, 应该, 是, 迄今为止, ...
1    [从, 各个方面, 来说, 都, 是, 一部, 好看, 的, 类型, 片, 。, 而, 最,...
2    [华语, 真正, 意义, 上, 的, 第一部, 科幻, 大片, ！, 刘慈欣, 的, 硬核,...
3    [“, 北京, 道路, 安全, 委, 提醒, 你, ：, 道路, 千万条, ，, 安全, 第...
4    [春节, 期间, 最大, 的, 一场, 烟花, ，, 是, 看, 你, 爸, 为, 你, 和...
Name: 评论内容, dtype: object
0    [特效, 技术, 讲, 应该, 迄今为止, 此类, 中国, 电影, 巅峰, 磅礴, 恢宏, ...
1    [各个方面, 好看, 类型, 片, 特别, 国产, 科幻电影, 影迷, +, 科幻, 迷, ...
2    [华语, 真正, 意义, 第一部, 科幻, 大片, 刘慈欣, 硬核, 科幻, 设定, 小兵,...
3    [北京, 道路, 安全, 委, 提醒, 道路, 千万条, 安全, 第一条, 行车, 规范, ...
4       [春节, 期间, 最大, 一场, 烟花, 爸, 地球, 点燃, 宇宙, 爆炸, 太, 朋克]
Name: 评论内容, dtype: object
```



nnamed:	用户名	户详情页	用户id	入会时间	居住地	评分	评分标签	评论时间	评论内容	点赞数量
0	frozenmoc	https://v	1233038	2006-08-1	北京	3	还行	2019-02-05 16:29:48	特效 技术 讲 应该 迄今为止 此类 中国 电影 巅峰 磅礴 恢宏 细节 营造 用心 故事	6316
1	张小北	https://v	xzfd	2008-03-(	北京	5	力荐	2019-01-29 02:11:13	各个方面 好看 类型 片 特别 国产 科幻电影 影迷 + 科幻 迷 双重 满足 最好 好莱	14579
2	乌鸦火堂	https://v	dreamfox	2008-03-1	北京	5	力荐	2019-01-20 19:00:54	华语 真正 意义 第一部 科幻 大片 刘慈欣 硬核 科幻 设定 小兵 扛 大旗 主流 价值	15555
3	影志	https://v	tjz230	2005-07-1	北京	4	推荐	2019-02-04 15:56:16	北京 道路 安全 委 提醒 道路 千万条 安全 第一条 行车 规范 亲人 两行 泪 这句	74314
4	姨妈的鸭	https://v	yimaxxduc	2009-12-(	上海	5	力荐	2019-02-05 16:51:54	春节 期间 最大 一场 烟花 爸 地球 点燃 宇宙 爆炸 太 朋克	9164
5	隽	https://v	nothingbu	2007-12-(	缺失	5	力荐	2019-01-22 05:43:32	星 多一星 鼓励 华语 电影工业 飞跃 节点 技术 剧本 精神 内核 方面 全球 尺度 顶	5628
6	谢飞导演	https://v	11451532(	2014-12-(	北京	5	力荐	2019-02-19 19:42:17	五棵松 耀莱 影城 杜比 厅 银幕 开创 中国 硬 科幻电影 元年 娱乐 佳作 不错 几年	451
7	阿暖	https://v	phoenix43	2009-03-(	湖北	3	还行	2019-01-29 11:28:37	史诗 感 狂轰滥炸 煽情 做到 电影 双线 叙事 做 太糙 太散 太 自我陶醉 可怕 依然	3603
8	乙小囧	https://v	chaneagle	2010-08-(	浙江	5	力荐	2019-02-06 07:23:26	这片 美国 拍 80 分 中国 拍 200 分 想象 波士顿 IMAX 电影院 里 坐满 完 集体 鼓	1308
9	朝暮雪	https://v	lingrui19	2012-08-(	四川	4	推荐	2019-01-27 14:16:11	流浪 地球 小说 想象力 丰富 地球 地球 一同 逃难 创意 荡气回肠 惊艳 无比 电影	14424
10	瓷泷电影	https://v	zhangzong	2009-04-1	北京	4	推荐	2019-02-02 14:29:09	中国 导演 拍出 硬 科幻 看到 完成度 想想 激动 原谅 完美 说 电影 中国 科幻电影	10319
11	琦殿	https://v	yyy18n	2009-12-(	广东	4	推荐	2019-02-07 02:27:52	刘启 韩 朵朵 地球 俩 毁灭 赶紧 累	3540
12	SingLesir	https://v	singlesir	2008-02-(	北京	5	力荐	2019-01-20 13:38:41	剧本 节奏 处理 极其 工整 灵魂 黑夜 第三幕 高潮迭起 层层 堆叠 特效 部分 扎实	1209
13	猫一	https://v	jishaotir	2007-09-(	北京	5	力荐	2019-01-20 19:29:40	西安 完 零点 场 上座率 惊人 观众 素质 极高 隔壁 哥们 《 认识 》 电影 结束 那	2091
14	私享史	https://v	Uvo	2005-10-(	缺失	3	还行	2019-02-09 09:39:08	走出 影厅 时 身后 一位 女生 同伴 说 三句话 电影 剧情 网上 打分 高 大概 是因	2670
15	顿河	https://v	alexcite	2009-10-(	北京	5	力荐	2019-01-29 00:59:17	一定 会 有人 较真儿 不过尔尔 喜欢 尊敬 人类 两只 眼睛 关注 当下 眺望 远方 长	1668
16	同志亦凡	https://v	3540441	2009-01-(	缺失	4	推荐	2019-02-07 02:53:07	拖 地球 逃难 设定 有种 太空 版 战狼 感觉 气质 内核 够 硬 硬气 + 硬核 国产 科	1674
17	qw0aszx	https://v	88439681	2014-05-(	浙江	2	较差	2019-02-05 02:15:43	设定 带来 那种 悲壮 一丝 浪漫 感全 毫无 节制 煽情 空到 不行 台词 毁 ★ ★ 空	7909
18	德川咪咪	https://v	zhangxy1	2008-09-(	上海	5	力荐	2019-02-05 19:54:56	原著 几乎 完全 不同 但绝 不亚于 高二看 小说 时 脑补 放映厅 里 60% 独自 来看	332
19	巴伐利亚	https://v	darkwood	2006-07-(	上海	4	推荐	2019-01-28 22:33:37	优点 缺点 十分 突出 电影 先说 优点 算得 中国 科幻电影 里程碑 之作 尤其 视觉	2341
20	苏妄言	https://v	15967385	2017-03-(	An	1	很差	2019-02-05 09:27:04	这回 真 吴京 锅 整个 团队 锅 前半段 尴尬 后半段 强行 撩 导演 剪辑 节奏 掌握	6153
21	桃桃林林	https://v	qijiuzhiy	2005-10-(	["	3	还行	2019-02-07 23:08:46	分 当时 最大 感受 影片 终于 一种 好莱坞 灾难片 模式 处理 电影 最后 救援 牺牲	2251
22	根	https://v	diewithme	2009-07-1	北京	4	推荐	2019-02-05 17:15:16	真的 国产 科幻片 里 前所未有 级别 说 太空 版 红海 行动 不论是 优点 缺点 完全	1464
23	国王KING	https://v	problemch	2009-06-1	广东	5	力荐	2019-02-07 01:06:07	燃到 飙泪 地球 救援 绝望 希望 救赎 路 更 重要 中 国 科 幻 电 影	227
24	谢谢你们	https://v	reave	2007-08-1	北京	4	推荐	2019-01-28 23:07:20	125 分钟 重新 定义 中国 科幻 主创 团队 站 好莱坞 肩膀 完成 一个 中国 电影 从	2958
27	武侠小王	https://v	china-unc	2007-02-(	北京	4	推荐	2019-01-29 00:20:53	地球 最后 战狼 这片 含京量 有点 高	1258
28	银谷	https://v	firo	2007-08-2	上海	4	推荐	2019-02-04 10:01:08	建议 放 低 预期 没有 吹 开天辟地 特效 确实 没话说 开头 十几分钟 快节奏 剪辑	975
29	一只麦麦	https://v	hexiaoqir	2009-09-1	北京	4	推荐	2019-01-20 14:03:59	流浪 地球 中国 第一部 真 · 硬 科幻电影 确实 硬 起来 中国式 亲情 关系 故土	2265
30	欢乐分裂	https://v	flowermun	2006-04-2	上海	3	还行	2019-02-05 17:39:38	2.5 宏大 设定 题材 胜利 木星 壮阔 斑斓 荒原 废土 视效 展现 工业 制作 水准 惊	925

Sheet1

四、数据分析

4.1 评分分布图

先进行评分统计再绘制饼图分析评分分布情况，根据各评论者的所评的星级给出打分情况，分值为 1-5。

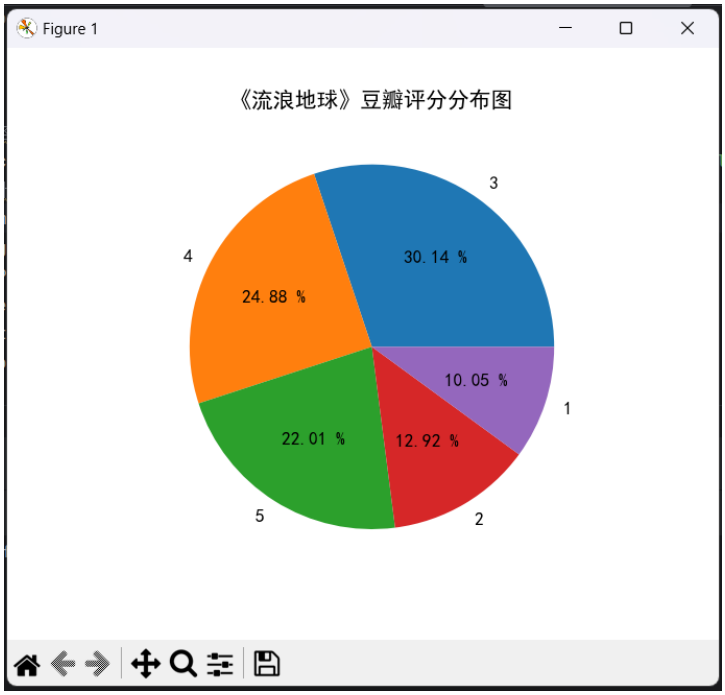
首先统计每一种分值的词频，再利用matplotlib库画出评分分部饼图。

分值词频如下：

```
评分
3    63
4    52
5    46
2    27
1    21
Name: count, dtype: int64
```

所用代码如下：

```
num = movie_comments['评分'].value_counts() #  统计词频
plt.figure(figsize=(4, 4))
plt.rcParams['font.sans-serif'] = 'Simhei'
plt.pie(num, autopct="%.2f %%", labels=num.index)
plt.title('《流浪地球》豆瓣评分分布图')
plt.show()
```



评分统计饼图

由数据以及评分统计饼图，可见大众对于《流浪地球》这部影片的总体评价是好的，评出 4-5 分的人数占比将近一半，大部分人给出 3 星的评价。

4.2 评论者的省份分布情况图

在已经进行完数据预处理的基础上，我们可以进一步分析评论的人数在省份分布上的情况。先进行省份人数统计再绘制饼图分析评分分布情况，省份人数统计可以通过输出movie_comment[‘居住地’]的词频实现。

整体思路为：首先统计每一种省份（这里包含了缺失值，所以实际

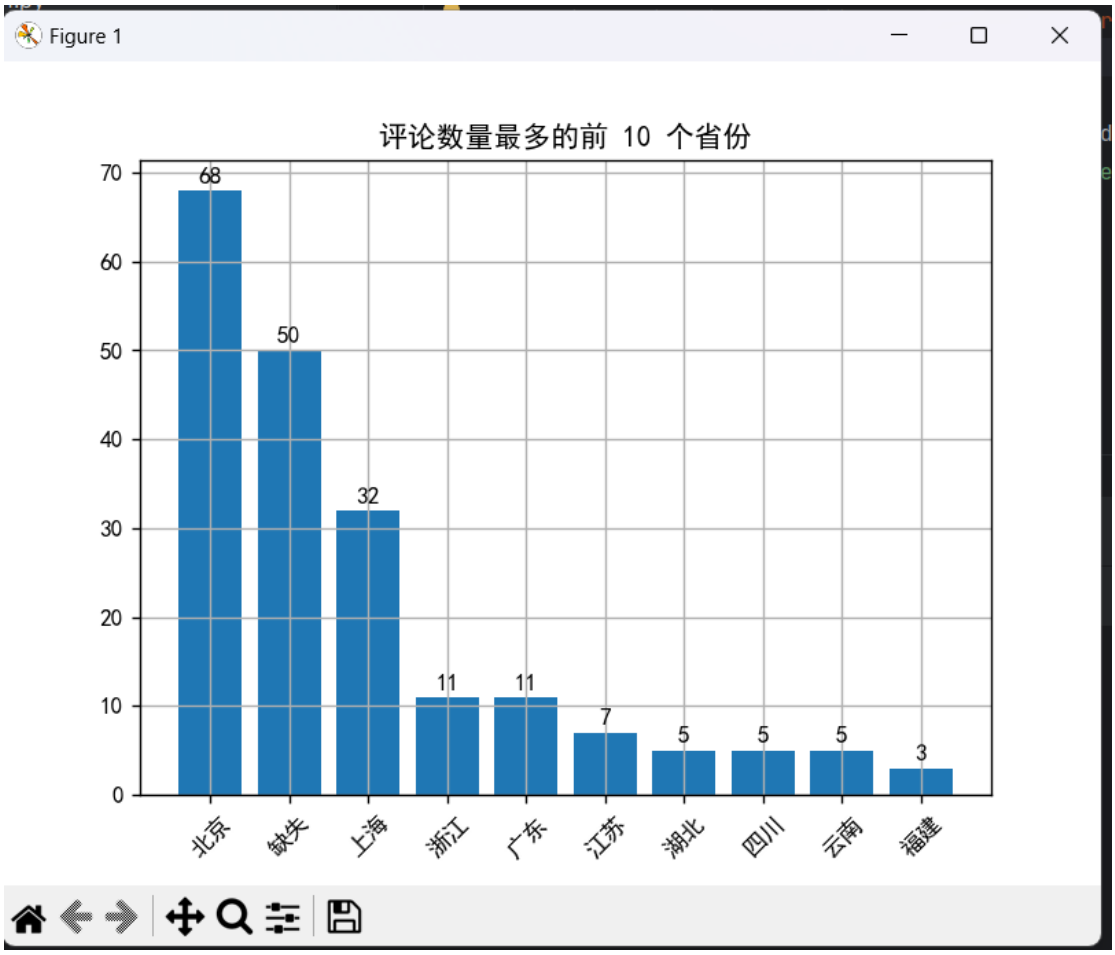
上只统计了九个省份）的词频，再利用matplotlib库画出评分分部饼图。

分值词频如下：

```
居住地
北京    68
缺失    50
上海    32
浙江    11
广东    11
江苏     7
湖北     5
四川     5
云南     5
福建     3
辽宁     2
重庆     1
香港     1
新疆     1
山东     1
清迈     1
河南     1
曼谷     1
江西     1
陕西     1
湖南     1
Name: count, dtype: int64
```

所用代码如下：

```
#评论者的城市分布情况
num = movie_comments['居住地'].value_counts()
print(num)
plt.rcParams['font.sans-serif'] = 'Simhei'
plt.bar(range(len(num[:10])), num[:10])
plt.xticks(range(len(num[:10])),                    num[:10].index,
```



评论最多的前十个省份图

可以看出北京、上海的用户是最多的，有将近一半的评论用户来自这两个城市。

4.3 统计评论中高频词汇生成词云图

对去除完停用词后的数据进行词频统计并绘制词云的操作，分析一下对于《流浪地球》这部电影，豆瓣的用户主要探讨的是什么主题内容。

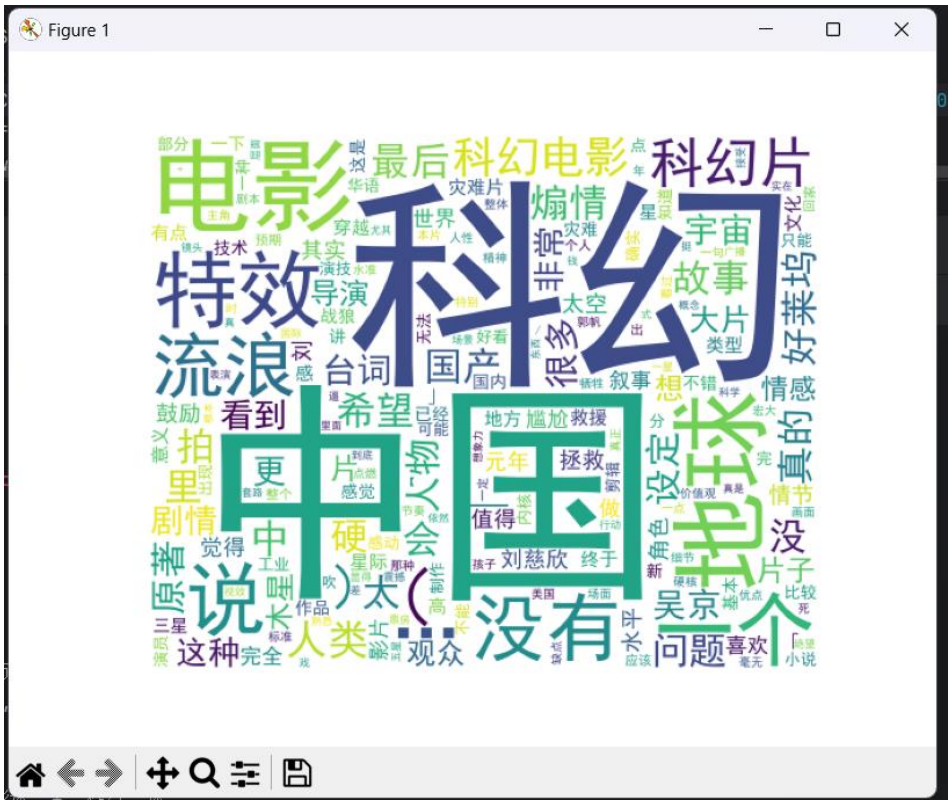
由于movie_comments[‘评论内容’]已经经过预处理，现在其中存放的是分词以后的结果。首先将电影评论内容合并为一个字符串，以便后续生成词云图时使用。通过.join()方法，将评论内容中的每条评论用空格连接成一个长字符串。

其次，创建一个词云对象wc，设置完词云的一些参数后，使用.generate()方法，传入之前合并的评论内容字符串text，生成词云图数据。

生成词云图的代码如下：

```
# 词云
text = ' '.join(movie_comments['评论内容'])  # 将评论内容合并为一个字符串
wc=WordCloud(font_path='./data/simhei.ttf',
background_color='White',width=500,height=400, mode="RGBA")
wc2 = wc.generate(text)
plt.imshow(wc2)
plt.axis('off')
```

生成词云图如下：



词云图

通过词云图，我们可以快速可知该部电影的关键词，此张词云图中最醒目的几个大词分别是：中国、科幻、特效、地球等等。

词云图直观展示了评论内容中词频最高词汇，字体大小代表了词频的高低。我们可以清楚看到，“中国""科幻""特效”等词汇字号最大，出现频次最高。词频的高低能够有效反映出观众评论的关注重点。本研究样本文本中这些高频词充分说明，观众在观看科幻题材电影《流浪地球》时，带着深深的爱国情怀以及对中国科幻影片的认可。词频分析直接展示了观众评论的主要话题，为我们进一步分析观众感知提供了重要线索

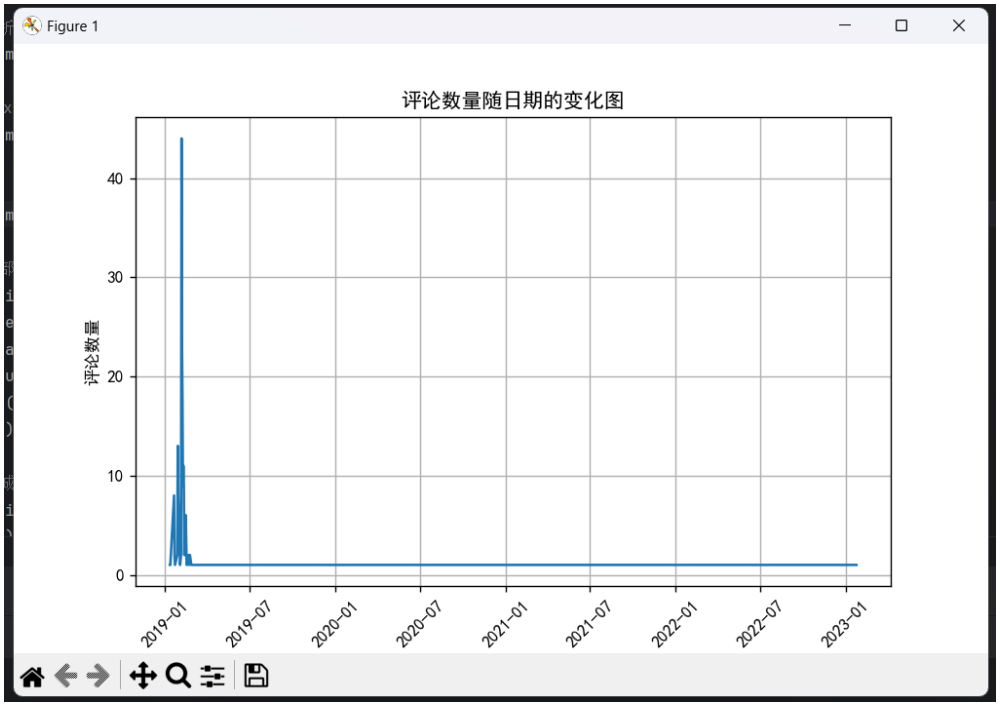
4.4 评论数量随日期的变化图

统计随着时间的推进，豆瓣用户对《流浪地球》发表评论的数量变

化情况。

首先按照‘评论时间’对movie_comments进行分组，这样每一组的大小就是该日期发表评论的总人数。

```
# 按日期分组统计评论数量
num=movie_comments.groupby(movie_comments['评论时间'].dt.date).size()
num2 = num.sort_index()
```



评论数量随日期变化图

可见，在电影初上映期发表评论的人数是最多的，随着日期的推移，发表评论的人数减少，并十分平稳。

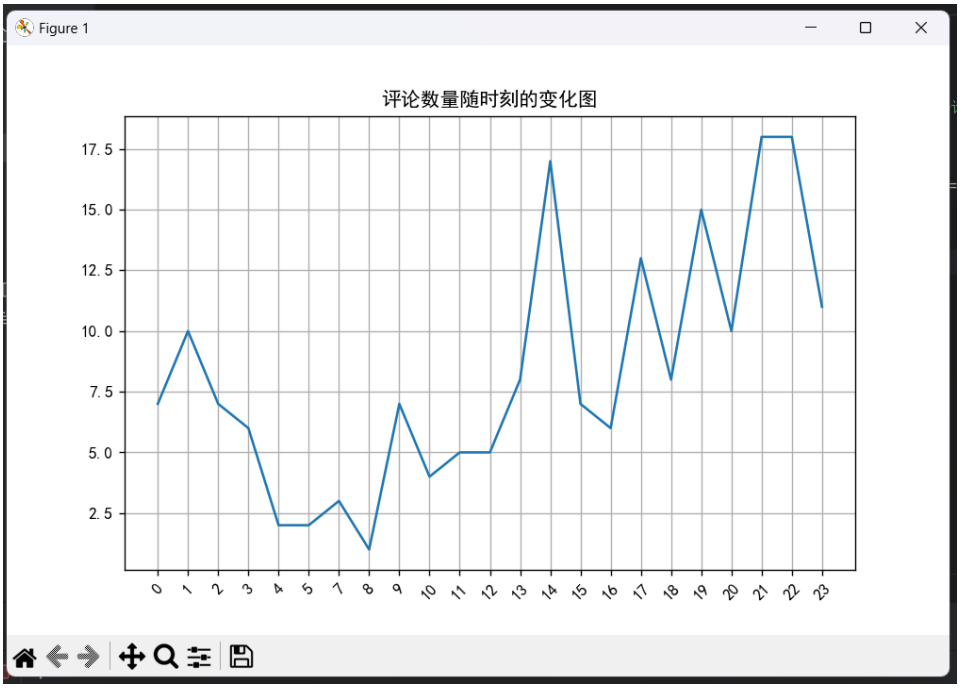
4.5 评论数量随时刻的变化图

由于movie_comments[‘评论时间’]已经经过预处理，均为datetime类

型，故通过.hour函数获取评论时间的时刻，进而画图进行统计。

并对存储时刻的列表num进行排序，最后画出评论数量随时刻的变化图。

```
#评论随时刻分布
num =pd.to_datetime(movie_comments['评论时间']).apply(lambda
x:x.hour).value_counts()
num2 = num.sort_index()
plt.figure(figsize=(8,5))
plt.plot(range(len(num2)), num2)
plt.xticks(range(len(num2)), num2.index, rotation=45)
plt.title('评论数量随时刻的变化图')
plt.grid()
plt.show()
```



评论数量随时刻变化图

豆瓣用户发布短评的时间主要集中在下午和晚上，下午的 13 点至 14 点和晚上的 21 点至 22 点比例尤为明显。随着时间向深夜推进，比例逐渐下降，凌晨 4 点达到最低值。这主要与用户的作息生活有关系。

4.6 对评论内容进行情感分析

1. 方法介绍

这里对评论内容的情感分析采用了基于朴素贝叶斯分类器的方法。朴素贝叶斯分类器是一种简单而有效的分类算法，在文本分类任务中被广泛应用。它基于贝叶斯定理和特征的独立假设，通过计算文本在给定类别下的概率来进行分类。

由于在预处理中已经实现了文本预处理以及分词和中文停用词过滤，所以为情感分析工作减轻了负担。

2. 代码设计思路

设计情感分析结果为通过评论内容得到的预测评分，后续还能通过真实用户评分实现对模型的评估。

读取电影评论数据集，并将其分为训练集和测试集，以便评估模型性能。

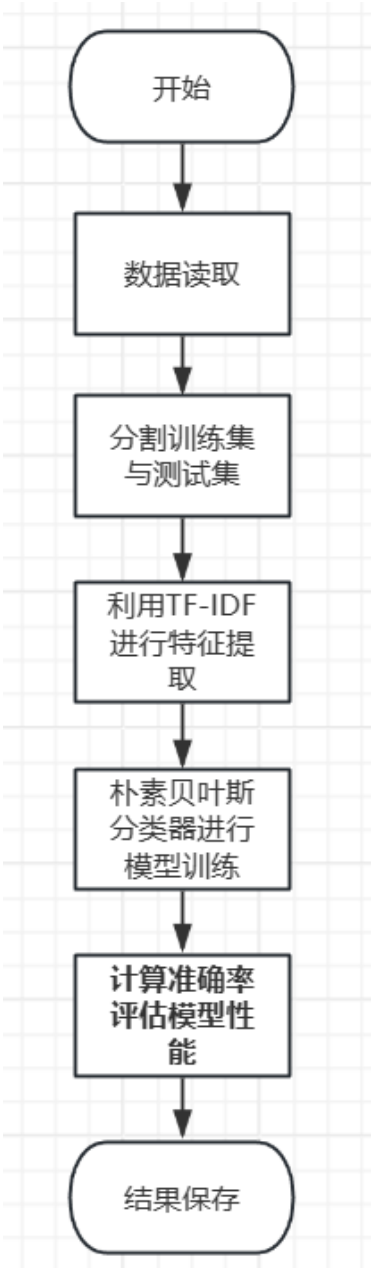
使用TF-IDF（Term Frequency-Inverse Document Frequency）进行特征提取，将文本数据转换为特征向量。TF-IDF是一种常用的文本特征提取方法，它考虑了词频和逆文档频率，能够更好地表示文本的重要程度。

使用朴素贝叶斯分类器对提取的特征向量进行训练，以构建情感分析模型。

对测试集进行预测，并计算模型的准确率。

将情感分析结果添加到原始数据集中，并保存到Excel表格中。

3. 代码流程



3.1 数据读取

使用pd.read_excel()函数读取名为"after.xlsx"的Excel表格，该表格包含了电影评论数据。

3.2 分割训练集与测试集

使用train_test_split()函数将数据集分割为训练集和测试集，其中80%用于训练，20%用于测试。

将评论内容（X_train和X_test）与评分（y_train和y_test）分开。

3.3 特征提取

使用TF-IDF（Term Frequency-Inverse Document Frequency）进行特征提取，通过TfidfVectorizer类将文本数据转换为特征向量。设定max_features=5000，限制提取的特征数量为 5000，以减少计算复杂度。

3.4 模型训练与评估

使用朴素贝叶斯分类器（MultinomialNB）进行情感分析模型的训练。通过fit()方法将训练集的特征向量和对应的评分进行模型训练。

使用训练好的模型对测试集进行预测，并通过准确率（accuracy_score）评估模型的性能。

3.5 结果保存

将情感分析结果添加到原始数据集中，使用predict()方法将评论内容转换为情感分析结果。

将更新后的数据集保存为名为"after1.xlsx"的Excel表格，使用to_excel()函数，并设置index=False以不保存行索引。

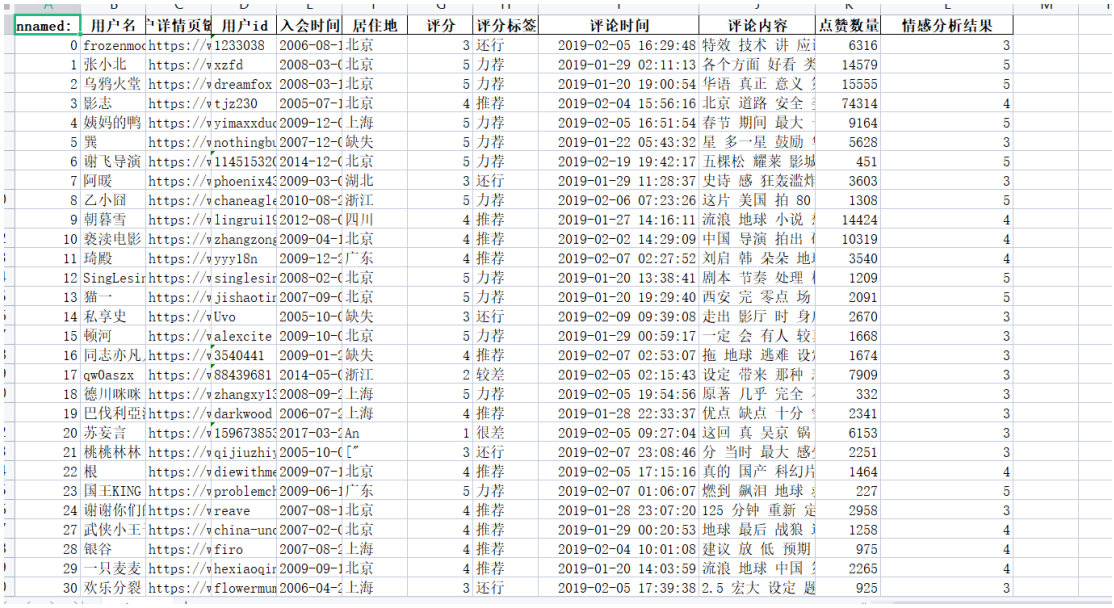
```
# 读取表格
movie_comments = pd.read_excel("C:\\Users\\1234\\Desktop\\after.xlsx")
# 分割训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(movie_comments['评论内容'], movie_comments['评分'], test_size=0.2, random_state=42)
# 特征提取
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
# 训练朴素贝叶斯分类器
clf = MultinomialNB()
clf.fit(X_train_vec, y_train)
# 预测并计算准确率
y_pred = clf.predict(X_test_vec)
accuracy = accuracy_score(y_test, y_pred)
print(f'准确率：{accuracy}')
# 将情感分析结果加入到数据集中
movie_comments['情感分析结果'] = clf.predict(vectorizer.transform(movie_comments['评论内容']))
# 保存到 Excel 表格
movie_comments.to_excel("C:\\Users\\1234\\Desktop\\after1.xlsx", index=False)
```

结果展示

准确率：

```
准确率：0.2894736842105263
```

将情感分析后的数据保存到excel表格中：



nnamed:	用户名	详情页链	用户id	入会时间	居住地	评分	评分标签	评论时间	评论内容	点赞数量	情感分析结果
0	frozenmoc	https://v	1233038	2006-08-1	北京	3	还行	2019-02-05 16:29:48	特效 技术 讲 应	6316	3
1	张小北	https://v	xzfd	2008-03-(	北京	5	力荐	2019-01-29 02:11:13	各个方面 好看 类	14579	5
2	乌鸦火堂	https://v	dreamfox	2008-03-1	北京	5	力荐	2019-01-20 19:00:54	华语 真正 意义	15555	5
3	影志	https://v	tjz230	2005-07-1	北京	4	推荐	2019-02-04 15:56:16	北京 道路 安全	74314	4
4	姨妈的鸭	https://v	yimaxxduc	2009-12-(	上海	5	力荐	2019-02-05 16:51:54	春节 期间 最大	9164	5
5	巽	https://v	nothingbu	2007-12-(	缺失	5	力荐	2019-01-22 05:43:32	星 多一星 鼓励	5628	3
6	谢飞导演	https://v	114515320	2014-12-(	北京	5	力荐	2019-02-19 19:42:17	五棵松 耀莱 影城	451	5
7	阿暖	https://v	phoenix43	2009-03-(	湖北	3	还行	2019-01-29 11:28:37	史诗 感 狂轰滥炸	3603	3
8	乙小囧	https://v	chaneagle	2010-08-2	浙江	5	力荐	2019-02-06 07:23:26	这片 美国 拍 80	1308	5
9	朝暮雪	https://v	lingruil9	2012-08-(	四川	4	推荐	2019-01-27 14:16:11	流浪 地球 小说	14424	4
10	褒淡电影	https://v	zhangzong	2009-04-1	北京	4	推荐	2019-02-02 14:29:09	中国 导演 拍出	10319	4
11	琦殿	https://v	yyy18n	2009-12-2	广东	4	推荐	2019-02-07 02:27:52	刘启 韩 朵朵 地	3540	4
12	SingLesir	https://v	singlesir	2008-02-(	北京	5	力荐	2019-01-20 13:38:41	剧本 节奏 处理	1209	5
13	猫一	https://v	jishaotir	2007-09-(	北京	5	力荐	2019-01-20 19:29:40	西安 完 零点 场	2091	5
14	私享史	https://v	Uvo	2005-10-(	缺失	3	还行	2019-02-09 09:39:08	走出 影厅 时 身	2670	3
15	顿河	https://v	alexcite	2009-10-(	北京	5	力荐	2019-01-29 00:59:17	一定 会 有人 较	1668	3
16	同志亦凡	https://v	3540441	2009-01-2	缺失	4	推荐	2019-02-07 02:53:07	拖 地球 逃难 设	1674	3
17	qw0aszx	https://v	88439681	2014-05-(	浙江	2	较差	2019-02-05 02:15:43	设定 带来 那种	7909	3
18	德川咪咪	https://v	zhangxy13	2008-09-2	上海	5	力荐	2019-02-05 19:54:56	原著 几乎 完全	332	3
19	巴伐利亞	https://v	darkwood	2006-07-2	上海	4	推荐	2019-01-28 22:33:37	优点 缺点 十分	2341	3
20	苏妄言	https://v	1596738533	2017-03-2	An	1	很差	2019-02-05 09:27:04	这回 真 吴京 锅	6153	3
21	桃桃林林	https://v	qijiuzhiy	2005-10-(	"	3	还行	2019-02-07 23:08:46	分 当时 最大 感	2251	3
22	根	https://v	diewithme	2009-07-1	北京	4	推荐	2019-02-05 17:15:16	真的 国产 科幻片	1464	4
23	国王KING	https://v	problemcl	2009-06-1	广东	5	力荐	2019-02-07 01:06:07	燃到 飙泪 地球	227	5
24	谢谢你们	https://v	reave	2007-08-1	北京	4	推荐	2019-01-28 23:07:20	125 分钟 重新 定	2958	3
27	武侠小王	https://v	china-und	2007-02-(	北京	4	推荐	2019-01-29 00:20:53	地球 最后 战狼	1258	4
28	银谷	https://v	firo	2007-08-2	上海	4	推荐	2019-02-04 10:01:08	建议 放 低 预期	975	4
29	一只麦麦	https://v	hexiaoqir	2009-09-1	北京	4	推荐	2019-01-20 14:03:59	流浪 地球 中国	2265	4
30	欢乐分裂	https://v	flowermun	2006-04-2	上海	3	还行	2019-02-05 17:39:38	2.5 宏大 设定 题	925	3

情感分析结论以及改进方向

由正确率可见，本次情感分析的结果并不良好，还有很大的改进空间。

通过查阅资料得知，情感分析可以有以下改进方向：

（1）进一步优化模型参数，如调整TF-IDF特征提取器的参数（如max_df、min_df、ngram_range等），尝试不同的分类器或集成学习方法，以提高模型的准确率和泛化能力。

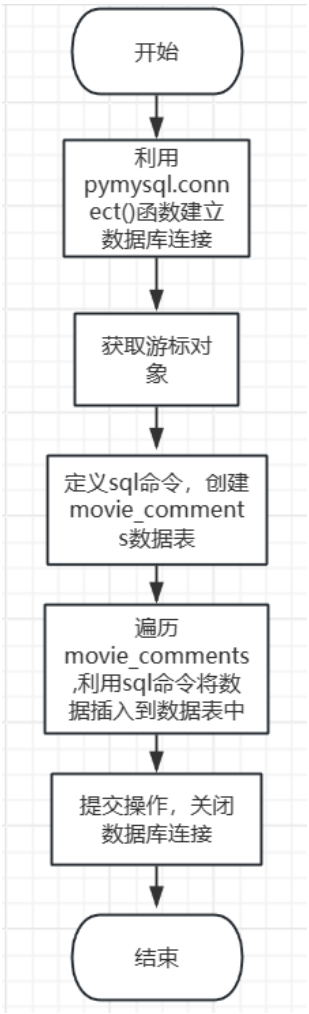
（2）考虑使用更大规模的数据集进行训练，以改善模型的性能和鲁棒性。

（3）对模型预测错误的样本进行分析，进一步优化数据预处理方法或模型设计，以提升模型在特定情境下的性能。

五、将数据存入数据库

这里采用pymysql库进行python与数据库的连接，进而将数据存入到mysql数据库中。

流程步骤：



其中，创建数据表的命令如下：

定义 SQL 命令，创建名为 movie_comments 的数据表。

```
create_table_sql = '''
CREATE TABLE IF NOT EXISTS movie_comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    user_name VARCHAR(255),
    user_id VARCHAR(255),
    join_time VARCHAR(255),
    address VARCHAR(255),
    score VARCHAR(10),
    score_label VARCHAR(255),
    comment_time VARCHAR(255),
    comment_content TEXT,
    like_count INT
)
'''
```

AUTO_INCREMENT: 自增主键，每插入一条记录，id 字段会自动递增。

各个字段的数据类型和长度：

user_name, user_id, join_time, address, score, score_label, comment_time: VARCHAR 类型，最大长度为 255。

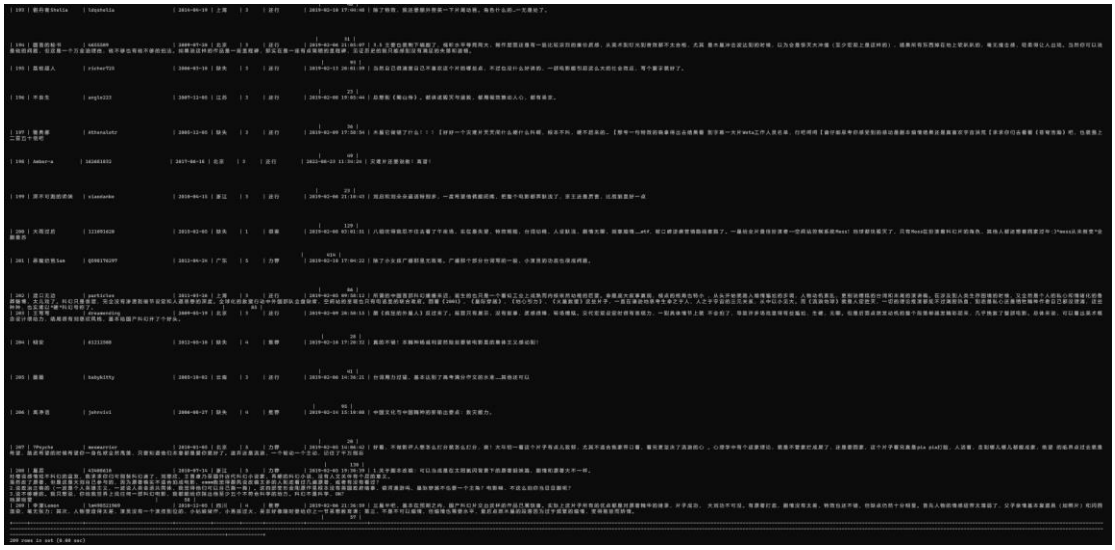
comment_content: TEXT 类型，用于存储较长的评论内容。

like_count: INT 类型，存储点赞数量。

```
# 建立数据库连接
db = pymysql.connect(host='localhost',
                     port=3306,
                     user='root',
                     password='123456',
                     database='豆瓣',
                     charset='utf8mb4',
                     )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 创建数据表
create_table_sql = '''
CREATE TABLE IF NOT EXISTS movie_comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_name VARCHAR(255),
    user_id VARCHAR(255),
    join_time VARCHAR(255),
    address VARCHAR(255),
    score VARCHAR(10),
    score_label VARCHAR(255),
    comment_time VARCHAR(255),
    comment_content TEXT,
    like_count INT
)
'''
cursor.execute(create_table_sql)
# 插入数据
for index, row in movie_comments.iterrows():
    insert_sql = '''
    INSERT INTO movie_comments (user_name, user_id, join_time, address, score,
score_label, comment_time, comment_content, like_count)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
    '''
    cursor.execute(insert_sql, (
        row['用户名'], row['用户id'], row['入会时间'], row['居住地'], row['评分'],
row['评分标签'], row['评论时间'], row['评论内容'],
        row['点赞数量']))
# 提交操作
db.commit()
# 关闭数据库连接
cursor.close()
db.close()

print('数据已保存到MySQL数据库中')
```

成功将数据全部存入数据库中：

由于评论文本过长，故仅缩小查看最后的数据栏：



六、难点及解决方法

1、最开始决定使用的网络爬虫技术为BeautifulSoup，但是由于使用依旧不是特别熟练，在提取用户居住地时遇到了困难，写的BeautifulSoup语句提取不出信息，故最后决定改用xpath语言进行提取。

2、一开始进行网络爬取信息时，采集不到用户id、入会时间以及用户居住地的信息（采集到这里时代码总是报错），后面意识到一个网页包含 20 个用户的详情页链接，应该采用遍历的方式，所以创建三个列表保存用户id、入会时间以及用户居住地的信息。

3、利用kettle将数据存入MySQL数据库时，excel数据的输入多次出现了问题，例如数据类型不匹配等等，经过多次尝试后决定直接在

Pycharm中建立数据库连接。

4、在进行数据分析时，程序常常由于各种各样的原因报错，由于技术不是很熟练，很多用法都是在网上学习到的，通过不断的调试与查阅资料debug，最后成功绘制出图表。

七、结论与展望

7.1 结论

随着社会的不断发展，电影产业也在不断壮大，越来越多的人开始重视电影的文化价值和商业价值。豆瓣电影上的评分和评论成为了衡量电影品质和受欢迎程度的重要标准之一。

本文通过网络爬虫对豆瓣网上的《流浪地球》影片进行数据采集并进行去重、去空以及文本分词和中文停用词过滤的数据预处理，将非结构化的数据清洗为结构化的数据，并存储到MySQL数据库中。对清洗后的数据进行多维度的分析和可视化展示，我们得到了对用户发布评论时间、用户居住地和评论内容等方面的深入理解和洞见。

详细来说，通过多维度分析以及情感处理，我们得到了以下结论：

（1）大众对于《流浪地球》这部影片的总体评价是好的，评出 4-5 分的人数占比将近一半，大部分人给出 3 星的评价。

（2）通过对评论者的居住地进行统计分析，可以看出北京、上海的用户是最多的，有将近一半的评论用户来自这两个城市。

（3）词云图中的高频词充分说明，观众在观看科幻题材电影《流浪

地球》时，带着深深的爱国情怀以及对中国科幻影片的认可。

（4）通过统计分析用户的评论时间，我们可以知道：在电影初上映期发表评论的人数是最多的，随着日期的推移，发表评论的人数减少，并十分平稳；豆瓣用户发布短评的时间主要集中在下午和晚上，下午的 13 点至 14 点和晚上的 21 点至 22 点比例尤为明显。随着时间向深夜推进，比例逐渐下降，凌晨 4 点达到最低值。这主要与用户的作息生活有关系。

（5）利用现有评论内容进行情感分析，预测用户给出的评分值，可以用来预估观众对影片的认可度。

7.1 展望

本文使用传统的机器学习方法对豆瓣影评进行特征提取和情感分析，但许多地方存在不足。例如：由于豆瓣是最大的国内用户评论的平台，但是受豆瓣平台的限制，对于每部影片只能获取一定量数据，相对比于正常的大模型来说，本文所使用的数据量相对较少，导致无法全部涵盖用户的评论数据。

大数据技术在海量的影评数据面前具有广大的应用前景。同时，本项目有很多潜在的展望，特别是结合机器学习和深度学习技术，可以进一步提升数据分析的深度和准确性，以及为用户提供更好的电影推荐和分析服务。可能的展望方向：

- 情感分析模型优化：可以开发更复杂的情感分析模型，例如使用深度学习的自然语言处理技术，如循环神经网络（RNN）或者Transformer模型，以更准确地捕捉用户评论中的情感和态度。

这可以提高对用户评分的预测准确性，更好地了解用户对电影的态度和情感倾向

- 个性化推荐系统：结合用户历史评分数据和评论内容，可以构建个性化的电影推荐系统。通过机器学习算法，如协同过滤、基于内容的过滤、或深度学习中的推荐模型，可以为用户推荐更符合其口味和兴趣的电影，提高用户体验。

- 时间序列分析：对评论时间的分析可以进一步扩展，例如使用时间序列分析方法来预测未来电影评论的趋势和高峰期，有助于影院制定更好的上映和营销策略。

- 主题建模：利用主题建模技术，如Latent Dirichlet Allocation (LDA)，可以从评论内容中提取出隐藏的主题和话题，进一步了解用户对电影的关注点和讨论焦点。

- 用户行为分析：结合用户的地理位置信息和其他行为数据，可以进行更深入的用户行为分析，了解不同地区和人群对电影的喜好和观影习惯，为电影制作方和发行方提供更精准的市场分析和定位。

- 多模态数据分析：结合评论文本以外的数据，如用户的观影历

史、社交媒体活动等，可以进行多模态数据分析，进一步深化对用户喜好和态度的理解，为电影产业的决策提供更全面的参考。

综合来看，结合机器学习和深度学习技术，可以使得该项目更加智能化、个性化，并且能够提供更精准的数据分析和服务，为电影产业的发展和用户的需求提供更好的支持和帮助。

八、个人心得

通过完成本次项目实践，我对数据采集以及预处理技术有了更加深刻的了解与掌握。本项目综合到了课程所学的大部分技术，我在实践过程中同时也是在复习温顾课程中学到的种种知识，所以这个综合项目对我的技术学习起到了很好的巩固作用，同时还锻炼到了我的整体性思维已经统筹能力。在数据采集的过程中，爬虫技术是有一套固定的规则的，按照这个框架一步步来，细心完成；其次数据预处理操作则是对数据进行观察分析，找到需要预处理的点，再利用已有技术进行逐个攻克，这对预处理技术的掌握也有一定要求；数据分析过程不是课堂上的重点，所以我在网上查阅学习了许多相关资料，对基本的数据分析类型以及方法有了初步的了解及掌握，这里我仅仅采用了图表分析的方法，希望以后能够学习到能多分析的方法和技术。

同时，由于本次项目是关于电影影评，我还体会到了理解数据采集与预处理广泛的应用性和必要性，影评的采集有许多应用场景，比如可以分析大众对电影的喜好程度、情感偏好，其得出的结论可以为

人们的生活带来许多便利。通过对影评进行评论还能结识更多志同道合的影迷，丰富自己兴趣爱好的同时也提升了艺术鉴赏能力。

总而言之，我很感谢这次项目实践的机会，虽然完成过程中错误频出，我也困惑苦恼、焦虑烦躁过，但是努力带来的回报是肯定的。

九、参考文献

[1]赵秀华,曾一果.基于文本挖掘的历史题材动画电影评论情感分析——以《长安三万里》为例[J].东南传播,2024(01):35-39.DOI:10.13556/j.cnki.dncb.cn35-1274/j.2024.01.024.

[2] 葛霓琳, 凡甲甲. 基于朴素贝叶斯和支持向量机的评论情感分析[J]. 计算机与数 字工程, 2020. 48(07): 1700 ~ 1704

[3]张月梅,刘媛华.基于K近邻和随机森林的情感分类研究[J].计算机与数字工程,2020,48(02):367-371.

[4]徐军,丁宇新,王晓龙.使用机器学习方法进行新闻情感自动分类[J].中文信息学报,2007,21(6):95-100.XU Jun,DING Yunxin,WANG Xiaolong. Using machine learning method to classify news emotion automati cally[J.Chinese journal of information fechnology,2007,21(6):95-100

[5]王雪.基于爬虫技术的电影评论信息获取及可视化设计[J].河南科技,2021,40(18):14-16.

[6]郭二强,李博.大数据环境下基于Python的网络爬虫技术[J.计算机产品与流通,2017(12):82.

[7]米洪.大数据采集与预处理.人民邮电出版社，2019

[8]刘国利. 基于评论文本和神经网络的电影票房预测方法研究[D].北京交通大学,2023.DOI:10.26944/d.cnki.gbfju.2022.000832.

[9]停用词集合(哈工大停用词表、四川大学机器智能实验室停用词库、百度停用词表)[EB/OL].http://www.datatang.com/data/19300， 2016

[10]包淑华,石盈鑫.大数据条件下国产电影影评的情感分析[J].呼伦贝尔学院学报,2022,30(02):126-131.

[11] 邓慈云,余国清.基于朴素贝叶斯的影评情感分析研究[J].智能计算机与应用,2023,13(02):210-212+217.

[12] Blei D M. Probabilistic topic models[J]. Communications of the ACM，2012, 55 (04): 77-84.

[13] 陈建平,陈志德,席进爱.大数据技术和应用.清华大学出版社,2020

附录：源代码

基于xpath.py:

```
import pandas as pd
import requests
from lxml import etree
import time
import openpyxl
import jieba
import itertools
```

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import pymysql
def trans_star(score):
    """转换评论星级"""
    if score == 'allstar10 rating':
        return '1 星'
    elif score == 'allstar20 rating':
        return '2 星'
    elif score == 'allstar30 rating':
        return '3 星'
    elif score == 'allstar40 rating':
        return '4 星'
    elif score == 'allstar50 rating':
        return '5 星'
    else:
        return '未知'

# 批量生成网页链接
urls = []
for i in range(29):
    url =
```

```
'https://movie.douban.com/subject/26266893/comments?start='+str(i*
20)+'&limit=20&status=P&sort=new_score'
        urls.append(url)
# 批量采集数据
movie_comments = pd.DataFrame()  # 用于存储所有评论目录页的字
段信息

for i in urls:  # 循环上面批量生成的网页
        #print(i)

        # 发送请求
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36'
        }
        cookies = {
            'Cookie': 'bid=2Xn2TzWr_Zw;
__utmz=30149280.1663724334.1.1.utmcsr=(direct)|utmccn=(direct)|ut
mcmd=(none); __gads=ID=b7f53a5b83e5e704-
22da5e63aad6007a:T=1663894139:RT=1663894139:S=ALNI_MbbFS2p9
6tG4R74PQrad89K2ai9VA; ll="118281";
```

```
__gpi=UID=000009d8ef520d13:T=1663894139:RT=1666575669:S=ALNI_
MY4J1DYTuvimFIyxk5IinlfidSqnA; push_noty_num=0;
push_doumail_num=0; dbcl2="216837743:t92rRERRJB4"; ck=zlas;
__utmc=30149280; __utmv=30149280.21683; ap_v=0,6.0;
_pk_ref.100001.8cb4=%5B%22%22%2C%22%22%2C1666588216%2C%2
2https%3A%2F%2Fmovie.douban.com%2Fsubject%2F25845392%2Fcom
ments%3Fstatus%3DP%22%5D;
_pk_id.100001.8cb4=b67eccd466165bb3.1666582708.2.1666588216.16
66583309.; _pk_ses.100001.8cb4=*;
__utma=30149280.1729025570.1663724334.1666574970.1666588217.
6; __utmt=1; __utmb=30149280.2.10.1666588217'}
    rq = requests.get(i, headers=headers, cookies=cookies)

    # 网页解析
    dom = etree.HTML(rq.text)

    # 提取数据
    user_name =
dom.xpath('//*[@id="comments"]/div/div[2]/h3/span[2]/a/text()')  #
用户名
    score =
dom.xpath('//*[@id="comments"]/div/div[2]/h3/span[2]/span[2]/@clas
```

```
s')  # 评分
        score_label =
dom.xpath('//*[@id="comments"]/div/div[2]/h3/span[2]/span[2]/@title
')  # 评分标签
        comment_time = dom.xpath(
            '//*[@id="comments"]/div/div[2]/h3/span[@class="comment-
info"]/span[@class="comment-time "]/@title')  # 评论时间
        comment_content =
dom.xpath('//*[@id="comments"]/div/div[2]/p/span/text()')  # 评论内
容
        number_like =
dom.xpath('//*[@id="comments"]/div/div[2]/h3/span[1]/span/text()')
# 点赞数量
        user_href =
dom.xpath('//*[@id="comments"]/div/div[2]/h3/span[2]/a/@href')  #
用户详情页链接

        # 创建空列表，存储循环读取的数据
        user_id = []
        user_time = []
        address = []
```

```
    # 因为用户id，入会时间，用户地址在用户详情页中
    # 对二级链接（用户详情页）进行循环，读取数据
    for i in user_href:
        # 发送请求
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36'
        }
        cookies = {
            'Cookie': 'bid=2Xn2TzWr_Zw;
__utmz=30149280.1663724334.1.1.utmcsr=(direct)|utmccn=(direct)|ut
mcmd=(none); __gads=ID=b7f53a5b83e5e704-
22da5e63aad6007a:T=1663894139:RT=1663894139:S=ALNI_MbbFS2p9
6tG4R74PQrad89K2ai9VA; ll="118281";
__gpi=UID=000009d8ef520d13:T=1663894139:RT=1666575669:S=ALNI_
MY4J1DYTuvimFIyxk5IinlfidSqnA; push_noty_num=0;
push_doumail_num=0; dbcl2="216837743:t92rRERRJB4"; ck=zlas;
__utmc=30149280; __utmv=30149280.21683; ap_v=0,6.0;
_pk_ref.100001.8cb4=%5B%22%22%2C%22%22%2C1666588216%2C%2
2https%3A%2F%2Fmovie.douban.com%2Fsubject%2F25845392%2Fcom
ments%3Fstatus%3DP%22%5D;
```

```
_pk_id.100001.8cb4=b67eccd466165bb3.1666582708.2.1666588216.16
66583309.;                                    _pk_ses.100001.8cb4=*;
__utma=30149280.1729025570.1663724334.1666574970.1666588217.
6; __utmt=1; __utmb=30149280.2.10.1666588217'}
            user_rq = requests.get(i, headers=headers, cookies=cookies)

            # 网页解析
            dom_1 = etree.HTML(user_rq.text)

            # 读取数据

user_id.append(dom_1.xpath('//*[@id="profile"]/div/div[2]/div[1]/div/d
iv/text()[1]'))   # 用户id

user_time.append(dom_1.xpath('//*[@id="profile"]/div/div[2]/div[1]/di
v/div/text()[2]'))   # 入会时间

address.append(dom_1.xpath('//*[@id="profile"]/div/div[2]/div[1]/div/a
/text()'))   # 用户地址

            # 设置强制等待时间
            time.sleep(0.01)
```

```
        # 数据整理
        data = pd.DataFrame({
            '用户名': user_name,
            '用户详情页链接': user_href,
            '用户id': user_id,
            '入会时间': user_time,
            '居住地': address,
            '评分': score,
            '评分标签': score_label,
            '评论时间': comment_time,
            '评论内容': comment_content,
            '点赞数量': number_like
        })
        # 数据合并
        movie_comments = pd.concat([movie_comments, data])
movie_comments.reset_index(drop=True, inplace=True) # 重设索引
movie_comments.to_excel("E:\\Python_file\\中促杯案例_流浪地球\\pre.xlsx",index = False)
print('数据已保存到xls表格中')
```

```
# 建立数据库连接
db = pymysql.connect(host='localhost',
                                port=3306,
                                user='root',
                                password='123456',
                                database='豆瓣',
                                charset='utf8mb4',
                                )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 创建数据表
create_table_sql = '''
CREATE TABLE IF NOT EXISTS movie_comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_name VARCHAR(255),
    user_id VARCHAR(255),
    join_time VARCHAR(255),
    address VARCHAR(255),
    score VARCHAR(10),
    score_label VARCHAR(255),
```

```
    comment_time VARCHAR(255),
    comment_content TEXT,
    like_count INT
)
'''

cursor.execute(create_table_sql)

# 插入数据
for index, row in movie_comments.iterrows():
    insert_sql = '''
    INSERT INTO movie_comments (user_name, user_id, join_time, address, score, score_label, comment_time, comment_content, like_count)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
    '''
    cursor.execute(insert_sql, (
        row['用户名'], row['用户id'], row['入会时间'], row['居住地'], row['评分'], row['评分标签'], row['评论时间'], row['评论内容'],
        row['点赞数量']))

# 提交操作
```

```
db.commit()

# 关闭数据库连接
cursor.close()
db.close()

print('数据已保存到MySQL数据库中')
```

预处理.py:

```
import pandas as pd
import numpy as np
import re
import jieba
import pymysql
#读取表格
movie_comments = pd.read_excel("E:\\Python_file\\中促杯案例_流浪地球\\movie_comments.xlsx")
movie_comments.info()  # 输出数据各列信息

## 数据预处理部分 ##

#处理用户id、入会时间和居住地
```

```
movie_comments['用户id'] = movie_comments['用户id'].apply(lambda x: x.replace("['\\n            ", "").replace(" ']", ""))
movie_comments['入会时间'] = movie_comments['入会时间'].apply(lambda x: x.replace("['\\n                ", "").replace("加入  ']", '').replace("加入  ', '\\n            ']","").replace("[]", "缺失"))
movie_comments['居住地'] = movie_comments['居住地'].apply(lambda x: x.replace("['", "").replace("']", '').replace("[]", "缺失"))
movie_comments['居住地'] = movie_comments['居住地'].str[0:2]

#将时间改为日期类型
movie_comments['评论时间'] = pd.to_datetime(movie_comments['评论时间'].apply(lambda x: str(x)))

#处理评分
movie_comments['评分'] = movie_comments['评分'].str[7]

#处理缺失值和重复值
print(f'重复值的行数：{np.sum(movie_comments.duplicated())}')
movie_comments.drop_duplicates(inplace=True)

#异常值处理
#  处理评分异常
```

```
print(movie_comments['评分'].value_counts())
movie_comments = movie_comments[movie_comments['评分'] != '-']
print(movie_comments['评分'].value_counts())

# 处理用户居住地异常
print(movie_comments['居住地'].value_counts())
movie_comments['居住地'] = movie_comments['居住地'].str.replace(r'[^\u4e00-\u9fa5]','缺失')
print(movie_comments['居住地'].value_counts())

#在这里加入数据库连接，并将movie_comments存入数据库‘豆瓣’中的movie_data表中
# 数据库连接信息
host = 'localhost'
user = 'root'
password = '123456'
database = '豆瓣'

# # 创建连接
# connection = pymysql.connect(host=host, port=3306,user=user, password=password, database=database, charset='utf8mb4', cursorclass=pymysql.cursors.DictCursor)
```

```
#
# try:
#         # 将 DataFrame 转换为字典列表
#         dict_list = movie_comments.to_dict(orient='records')
#         # 创建 SQL 插入语句
#                         sql_insert = "INSERT INTO movie_info
(user_name,user_href,user_id,user_time,address,score,score_label,com
ment_time,comment_content,number_like) VALUES ( %(用户名)s, %(用
户 详 情 页 链
接)s, %(user_href)s,%(user_href)s,%(user_id)s,%(user_time)s,%(adress)s,
%(score)s,%(score_label)s,%(comment_time)s,%(comment_content)s,%(
number_like)s);"
#
#         # 逐行执行 SQL 插入语句
#         with connection.cursor() as cursor:
#                 for record in dict_list:
#                         cursor.execute(sql_insert, record)
#                 connection.commit()
#
# except pymysql.MySQLError as e:
#         print(f"Error: {e}")
#
```

```
# finally:
#         # 关闭数据库连接
#         connection.close()



#文本预处理
#删除重复评论
movie_comments.drop_duplicates(subset=['评论内容'], inplace=True)
#删除机械压缩的短句
# 假设将重复出现两次以上的词进行压缩，可以利用正则表达式实现
movie_comments['评论内容'] = movie_comments['评论内容'].apply(lambda x: re.sub(r'\b(\w+)\b\s+\1\b', r'\1', x))
#删除过短的评论
movie_comments = movie_comments[movie_comments['评论内容'].apply(lambda x: len(x) >= 5)]

#分词和去除停用词
stop_words_path = "E:\edge_download\chinese-stop-words-list-master\chinese-stop-words-list-master\中文停用词库.txt"
with open(stop_words_path, 'r', encoding='utf-8') as f:
    stop_words = f.read().split()
```

```
stop_words = [' ', '\n', '这部', '.', '一部'] + stop_words
data_cut = movie_comments['评论内容'].apply(jieba.lcut)  # 分词
data_after = data_cut.apply(lambda x: [i for i in x if i not in stop_words])
# 去除停用词
movie_comments['评论内容'] = data_after.apply(lambda x: ' '.join(x))  # 将分词结果转换为字符串

#存储预处理完之后的数据
movie_comments.to_excel("E:\\Python_file\\中促杯案例_流浪地球\\after.xlsx")
```

分析.py:

```
import pandas as pd
import numpy as np
import re
import jieba
import pymysql
#读取表格
movie_comments = pd.read_excel("E:\\Python_file\\中促杯案例_流浪地球\\movie_comments.xlsx")
movie_comments.info()  # 输出数据各列信息

## 数据预处理部分 ##
```

```
#处理用户id、入会时间和居住地
movie_comments['用户id'] = movie_comments['用户id'].apply(lambda x: x.replace("['\\n            ", "").replace(" ']", ""))
movie_comments['入会时间'] = movie_comments['入会时间'].apply(lambda x: x.replace("['\\n                ", "").replace("加入  ']", '').replace("加入  ', '\\n            ']","""))
movie_comments['居住地'] = movie_comments['居住地'].apply(lambda x: x.replace("['", "").replace("']", '').replace("[]", "缺失"))
movie_comments['居住地'] = movie_comments['居住地'].str[0:2]

#将时间改为日期类型
movie_comments['评论时间'] = pd.to_datetime(movie_comments['评论时间'].apply(lambda x: str(x)))

#处理评分
movie_comments['评分'] = movie_comments['评分'].str[7]

#处理缺失值和重复值
print(f'重复值的行数：{np.sum(movie_comments.duplicated())}')
movie_comments.drop_duplicates(inplace=True)
```

```
#异常值处理
# 处理评分异常
print(movie_comments['评分'].value_counts())
movie_comments = movie_comments[movie_comments['评分'] != '-']
print(movie_comments['评分'].value_counts())

# 处理用户居住地异常
print(movie_comments['居住地'].value_counts())
movie_comments['居住地'] = movie_comments['居住地'].str.replace(r'[^\u4e00-\u9fa5]','缺失')
print(movie_comments['居住地'].value_counts())

#在这里加入数据库连接，并将movie_comments存入数据库‘豆瓣’中的movie_data表中
# 数据库连接信息
host = 'localhost'
user = 'root'
password = '123456'
database = '豆瓣'

# # 创建连接
# connection = pymysql.connect(host=host, port=3306,user=user,
```

```
password=password, database=database, charset='utf8mb4',
cursorclass=pymysql.cursors.DictCursor)
#
# try:
#     # 将 DataFrame 转换为字典列表
#     dict_list = movie_comments.to_dict(orient='records')
#     # 创建 SQL 插入语句
#                 sql_insert = "INSERT INTO movie_info
(user_name,user_href,user_id,user_time,address,score,score_label,com
ment_time,comment_content,number_like) VALUES ( %(用户名)s, %(用
户 详 情 页 链
接)s, %(user_href)s,%(user_href)s,%(user_id)s,%(user_time)s,%(adress)s,
%(score)s,%(score_label)s,%(comment_time)s,%(comment_content)s,%(
number_like)s);"
#
#     # 逐行执行 SQL 插入语句
#     with connection.cursor() as cursor:
#         for record in dict_list:
#             cursor.execute(sql_insert, record)
#         connection.commit()
#
# except pymysql.MySQLError as e:
```

```
#         print(f"Error: {e}")
#
# finally:
#         # 关闭数据库连接
#         connection.close()



#文本预处理
#删除重复评论
movie_comments.drop_duplicates(subset=['评论内容'], inplace=True)
#删除机械压缩的短句
# 假设将重复出现两次以上的词进行压缩，可以利用正则表达式实现
movie_comments['评论内容'] = movie_comments['评论内容'].apply(lambda x: re.sub(r'\b(\w+)\b\s+\1\b', r'\1', x))
#删除过短的评论
movie_comments = movie_comments[movie_comments['评论内容'].apply(lambda x: len(x) >= 5)]


#分词和去除停用词
stop_words_path = "E:\edge_download\chinese-stop-words-list-master\chinese-stop-words-list-master\中文停用词库.txt"
```

```
with open(stop_words_path, 'r', encoding='utf-8') as f:
    stop_words = f.read().split()
stop_words = [' ', '\n', '这部', '.', '一部'] + stop_words
data_cut = movie_comments['评论内容'].apply(jieba.lcut)  # 分词
data_after = data_cut.apply(lambda x: [i for i in x if i not in stop_words])
# 去除停用词
movie_comments['评论内容'] = data_after.apply(lambda x: ' '.join(x))  # 将分词结果转换为字符串

#存储预处理完之后的数据
movie_comments.to_excel("E:\\Python_file\\中促杯案例_流浪地球\\after.xlsx")


import pandas as pd
import numpy as np
import re
import jieba
import itertools
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# 读取表格
movie_comments =
pd.read_excel("C:\\Users\\1234\\Desktop\\after.xlsx")
# 情感分析部分
# 分割训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(movie_comments['评论
内容'], movie_comments['评分'], test_size=0.2, random_state=42)

# 特征提取
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# 训练朴素贝叶斯分类器
clf = MultinomialNB()
clf.fit(X_train_vec, y_train)

# 预测并计算准确率
```

```
y_pred = clf.predict(X_test_vec)
accuracy = accuracy_score(y_test, y_pred)
print(f'准确率：{accuracy}')

# 将情感分析结果加入到数据集中
movie_comments['情感分析结果'] = clf.predict(vectorizer.transform(movie_comments['评论内容']))

# 保存到 Excel 表格
movie_comments.to_excel("E:\\Python_file\\中促杯案例_流浪地球\\after1.xlsx", index=False)

# 画图统计部分
num = movie_comments['评分'].value_counts()  # 统计词频
plt.figure(figsize=(4, 4))
plt.rcParams['font.sans-serif'] = 'Simhei'
plt.pie(num, autopct="%.2f %%", labels=num.index)
plt.title('《流浪地球》豆瓣评分分布图')
plt.show()

# 评论者的城市分布情况
num = movie_comments['居住地'].value_counts()
```

```
print(num)
plt.rcParams['font.sans-serif'] = 'Simhei'
plt.bar(range(len(num[:10])), num[:10])
plt.xticks(range(len(num[:10])), num[:10].index, rotation=45)
plt.title('评论数量最多的前 10 个省份')
# 在每个柱子上显示城市的评论数量
for i, (city, count) in enumerate(zip(num[:10].index, num[:10])):
    plt.text(i, count + 0.5, str(count), ha='center', va='bottom')
plt.grid()
plt.show()

#评论随日期分布
# 按日期分组统计评论数量
num = movie_comments.groupby(movie_comments['评论时间'].dt.date).size()
num2 = num.sort_index()

# 绘制评论数量随日期的变化图
plt.figure(figsize=(8,5))
plt.plot(num2.index, num2.values)
plt.xticks(rotation=45)
plt.title('评论数量随日期的变化图')
```

```
plt.xlabel('日期')
plt.ylabel('评论数量')
plt.grid()
plt.show()

#评论随时刻分布
num  =pd.to_datetime(movie_comments[' 评 论 时 间 ']).apply(lambda
x:x.hour).value_counts()
num2 = num.sort_index()
plt.figure(figsize=(8,5))
plt.plot(range(len(num2)), num2)
plt.xticks(range(len(num2)), num2.index, rotation=45)
plt.title('评论数量随时刻的变化图')
plt.grid()
plt.show()

#不同省份对电影的评分情况分析
tmp = pd.DataFrame(0, index=movie_comments[' 评 分
'].astype(str).drop_duplicates().sort_values(),
                   columns=num[:5].index)
tmp.sort_index(inplace=True)
for i, j, k in zip(movie_comments['评分'], movie_comments['居住地'],
```

```
movie_comments['点赞数量']):
        # print(i, j, k)
        if j in num[:5].index:
            tmp.loc[str(i), j] += 1


plt.figure(figsize=(8, 4))
(n, m) = tmp.shape
plt.rcParams['axes.unicode_minus'] = False
for i in range(0, m):
        plt.plot(range(n), tmp.iloc[:, i])
        plt.fill_between(range(n), tmp.iloc[:, i], alpha=0.5)
plt.legend(tmp.columns)
plt.xticks(range(n), tmp.index, rotation=45)
plt.title('省份与评分关系图')
plt.grid()
plt.show()


# 词云
text = ' '.join(movie_comments['评论内容'])  # 将评论内容合并为一个字符串
wc = WordCloud(font_path='./data/simhei.ttf', background_color='White', width=500, height=400, mode="RGBA")
```

```
wc2 = wc.generate(text)
plt.imshow(wc2)
plt.axis('off')
plt.show()
```

存入数据库.py:

```
import pandas as pd
import pymysql
#读取表格
movie_comments = pd.read_excel("E:\\Python_file\\中促杯案例_流浪地球\\movie_comments1.xlsx")
#  建立数据库连接
db = pymysql.connect(host='localhost',
                     port=3306,
                     user='root',
                     password='123456',
                     database='豆瓣',
                     charset='utf8mb4',
                     )

#  使用cursor()方法获取操作游标
cursor = db.cursor()
```

```
# 创建数据表
create_table_sql = '''
CREATE TABLE IF NOT EXISTS movie_comments (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_name VARCHAR(255),
    user_id VARCHAR(255),
    join_time VARCHAR(255),
    address VARCHAR(255),
    score VARCHAR(10),
    score_label VARCHAR(255),
    comment_time VARCHAR(255),
    comment_content TEXT,
    like_count INT
)
'''

cursor.execute(create_table_sql)

# 插入数据
for index, row in movie_comments.iterrows():
    insert_sql = '''
    INSERT INTO movie_comments (user_name, user_id, join_time,
```

```
address, score, score_label, comment_time, comment_content,
like_count)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
        '''
        cursor.execute(insert_sql, (
            row['用户名'], row['用户id'], row['入会时间'], row['居住地'],
row['评分'], row['评分标签'], row['评论时间'], row['评论内容'],
            row['点赞数量']))

# 提交操作
db.commit()

# 关闭数据库连接
cursor.close()
db.close()

print('数据已保存到MySQL数据库中')
```