课程设计 2：豆瓣电影 Top250 爬取与可视化（BeautifulSoup）

一、准备过程

豆瓣是一个涵盖电影、图书、音乐等领域的综合性社交平台，特别是其“Top 250”电影排行榜，成为了广大电影爱好者了解热门影片的重要途径之一。豆瓣 Top 250 涵盖了全球范围内的高评分电影，数据内容包含电影的名称、导演、演员、评分、上映年份等。本实验旨在通过编写 Python 爬虫程序，自动化地抓取豆瓣电影 Top 250 中的电影信息，并将其保存为 CSV 文件，便于进一步的数据分析和可视化处理。

电影信息被封装在多个 标签中，具有清晰的 HTML 结构。此外，页面采用了分页显示，每页显示 25 部电影，因此需要通过循环来爬取所有 250 部电影的数据。

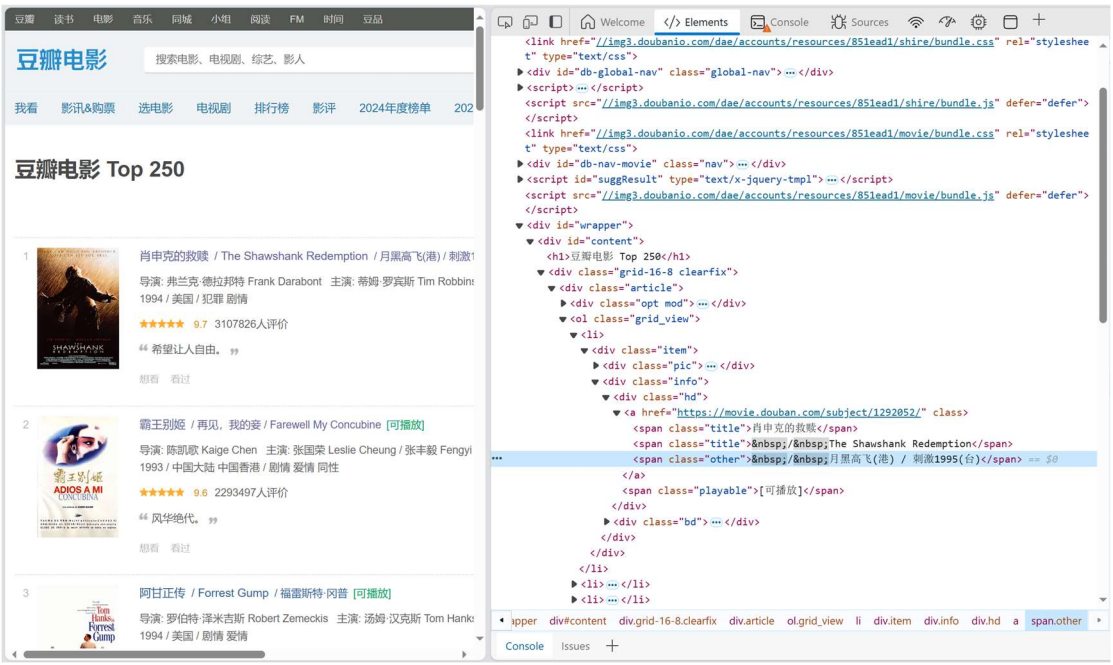

图 1：网站详情

为了实现网页数据的爬取，我们需要选择适当的网络请求库和 HTML 解析库。requests 库用于向 Douban 网站发送 HTTP 请求，获取网页内容；BeautifulSoup 库：用于解析网页内容。BeautifulSoup 能够将网页内容转换为一个树形结构，便于提取需要的数据。pandas 库：用于将抓取的数据存储为 DataFrame 格式，便于后续操作，并最终将数据导出为 CSV 文件。

二、实现过程

首先导入必要的库，pandas 用来数据处理，matplotlib，seaborn 用来数据可视化，warnings 用来屏蔽警告信息，最后一行用来设置绘图的字体。

```
import requests
```

```
from bs4 import BeautifulSoup
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import matplotlib
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```

下面是请求头信息，由于网站有一定的反爬机制，需要获取本机上的cookie字段。

```
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 \
            (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
Edg/131.0.0.0',
            'cookie':'ll="118237"; bid=ndsCNgM2Va0;
__yadk_uid=MjIrZflXmtsuUK6zCMw60EdmxKOOVUVI; \
            _vwo_uuid_v2=DF0D8D6009ECE426A43DEBF72F575CBF1|95687af1ed48
07eed9035df4f8cdccd1; \
            _pk_id.100001.4cf6=4ee74b6aa27cfeae.1735995274.;
__utmc=30149280; __utmc=223695111; \
            dbcl2="285789918:SvwbzhdqflE"; ck=_uvz;
_pk_ref.100001.4cf6=%5B%22%22%2C%22%22%2C1736071155%2C%22https%3A%2F%2F
accounts.douban.com%2F%22%5D; \
            _pk_ses.100001.4cf6=1;
__utma=30149280.1610677333.1649208538.1735996614.1736071156.13; \
            __utmb=30149280.0.10.1736071156;
__utmz=30149280.1736071156.13.3.utmcsr=accounts.douban.com|utmccn=(refe
rral)|utmcmd=referral|utmcct=/; \
            __utma=223695111.1610677333.1649208538.1735996614.173607115
6.11; __utmb=223695111.0.10.1736071156; \
            __utmz=223695111.1736071156.11.3.utmcsr=accounts.douban.com
|utmccn=(referral)|utmcmd=referral|utmcct=/; \
            push_noty_num=0; push_doumail_num=0'}
```

首先定义了一个解析函数，该函数接收一个参数 soup，即通过BeautifulSoup解析过的网页内容（HTML）。在函数中，我们初始化了8个空列表，分别用于存储电影的各类信息：

list_title：存储电影名称，list_director：存储导演信息，list_year：存储上映年份，list_country：存储国家/地区，list_kind：存储电影类型，list_star：存储电影评分，list_comment：存储评论数量，list_sum：存储电影简介。

```
# 解析爬取的内容
douban = pd.DataFrame()
def analyse(soup):
```

```
    list_title, list_director, list_year, list_country = [],[],[],[]
    list_kind, list_star, list_comment, list_sum = [],[],[],[]

    # 分析电影名
    all_title = soup.find_all('span', attrs={'class': 'title'})
    for title2 in all_title:
        title = title2.string
        if '/' not in title:
            list_title.append(title)

    # 基本信息
    all_info = soup.find_all('p',attrs={'class':''})
    for info in all_info:
        if '导演' in info.text:
            director_info = info.text.split('导演:')[1].split('主')[0]
            list_director.append(director_info)
            year_info = info.text.split('\n')[2].split('\xa0')[0][-4:]
            list_year.append(year_info)
            country_info = info.text.split('\xa0')[-3]
            list_country.append(country_info)
            kind_info = info.text.split('\xa0')[-1].split('\n')[0]
            list_kind.append(kind_info)

    # 评分
    all_star = soup.find_all('span',attrs={'class':'rating_num'})
    for star in all_star:
        star = star.text
        list_star.append(star)

    # 评价
    all_comment = soup.find_all('span')
    for i in all_comment:
        if '评价' in i.text:
            comment = i.text[:-3]
            list_comment.append(comment)

    # 简介
    all_sum = soup.find_all('span', attrs={'class': 'inq'})
    for i in all_sum:
        summery = i.text
        list_sum.append(summery)
    df = pd.DataFrame([list_title, list_director, list_year,
list_country, list_kind, list_star, list_comment, list_sum]).T
        return df
```

使用 for start in range(0, 251, 25)进行分页循环。电影页面每页展示 25 部电影，因此 start 值的范围是 0 到 251，步长为 25，每次循环处理 25 部电影。response.text 包含了从 Douban 返回的 HTML 页面内容。BeautifulSoup(response.text, 'html.parser')将 HTML 内容解析为 BeautifulSoup 对象，可以方便地提取其中的结构化数据。

调用 analyse(soup)函数解析该页面的数据，analyse 函数返回一个包含电影信息的 DataFrame。pd.concat([douban, page_])将当前页面抓取的数据（page_）与之前已抓取的数据（douban）进行合并。这样，每次抓取一页数据后，所有数据都会被累加到 douban 中。当所有页面的数据抓取完成后，调用 douban.to_csv('douban.csv', index=False)将最终合并的 DataFrame 保存为 CSV 文件。

```
for start in range(0,251,25):
    # 获取网页内容
    response =
requests.get(f'https://movie.douban.com/top250?start={start}',
headers=headers)
    # 解析网页内容
    soup = BeautifulSoup(response.text, 'html.parser')
    page_ = analyse(soup)
    douban = pd.concat([douban,page_])
    print(f'page{start} finished!')
# 保存为 csv 文件
douban.to_csv('douban.csv')
```

三、实验结果

pd.read_csv('douban.csv')：读取之前保存的 douban.csv 文件，将其加载为 pandas DataFrame 对象。drop(columns='Unnamed: 0')：删除名为 Unnamed: 0 的列。重命名列名。数据中包含错误的年份数据，统一将其改为为 2026。

```
# 数据清洗
douban = pd.read_csv('douban.csv').drop(columns='Unnamed: 0')
columns = ['名称','导演','年份','国家','类型','评分','评论数量','简介']
douban.columns = columns

douban['年份'][douban['年份'] == '国大陆)'] = 2026
douban['年份'] = douban['年份'].astype(int)

def country_clean(country):
    return country.split(' ')
douban['国家'] = douban['国家'].apply(country_clean)
douban['类型'] = douban['类型'].apply(country_clean)
```

简单的数据清洗之后的数据如下表所示：

	名称	导演	年份	国家	类型	评分	评论数量	简介
0	肖申克的救赎	弗兰克·德拉邦特 Frank Darabont	1994	[美国]	[犯罪, 剧情]	9.7	3106453	希望让人自由。
1	霸王别姬	陈凯歌 Kaige Chen	1993	[中国大陆, 中国香港]	[剧情, 爱情, 同性]	9.6	2292392	风华绝代。
2	阿甘正传	罗伯特·泽米吉斯 Robert Zemeckis	1994	[美国]	[剧情, 爱情]	9.5	2311764	一部美国近现代史。
3	泰坦尼克号	詹姆斯·卡梅隆 James Cameron	1997	[美国, 墨西哥]	[剧情, 爱情, 灾难]	9.5	2353234	失去的才是永恒的。
4	千与千寻	宫崎骏 Hayao Miyazaki	2001	[日本]	[剧情, 动画, 奇幻]	9.4	2400773	最好的宫崎骏，最好的久石让。

图 2：爬取信息详情

按照国家和电影类型重新整理，这一步的目的是便于下面的数据可视化。

```
# 按照国家和电影类型重新整理
year_country = pd.DataFrame(columns=['年份','国家'])
for i in range(250):
    year = douban.loc[i]['年份']
    country = douban.loc[i]['国家']
    for j in country:
        year_country.loc[len(year_country)] = {'年份':year, '国家': j}

year_kind = pd.DataFrame(columns=['年份','类型'])
for i in range(250):
    year = douban.loc[i]['年份']
    kind = douban.loc[i]['类型']
    for j in kind:
            year_kind.loc[len(year_kind)] = {'年份':year, '类型': j}
```

豆瓣电影前 250 年份分布可视化如下：

```
# 数据可视化
colors = sns.color_palette('rocket',n_colors=7)
dis_year = sns.displot(douban, x='年份', binwidth=3,
                       height=5, aspect=1.5, color=colors[1])
plt.grid(True, alpha=0.3)
dis_year.set_xlabels(size =12)
plt.title('豆瓣电影前 250 年份分布')
```

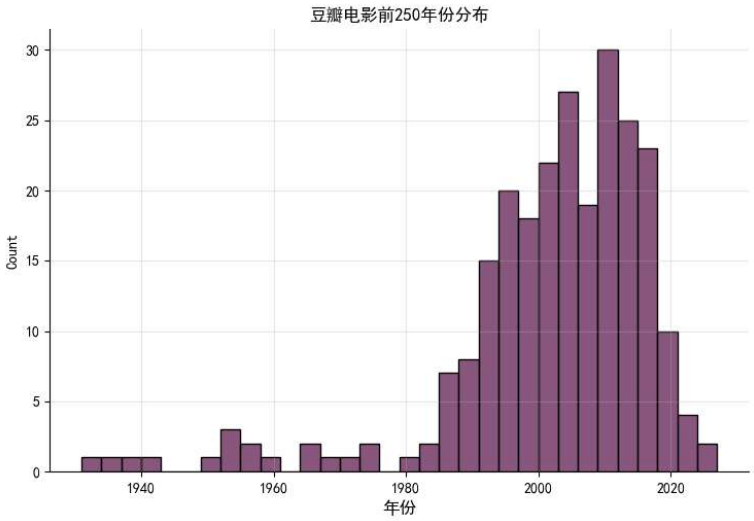

图 3：电影前 250 分布

豆瓣电影前 250 类型分布可视化如下：

```
sns.displot(x='年份', data=year_kind, hue='类型',
            binwidth=3, height=5, aspect=1.5)
plt.grid(True, alpha=0.3)
plt.gca().yaxis.set_major_locator(matplotlib.ticker.MaxNLocator(integer
=True))
plt.title('豆瓣电影前 250 类型分布')
```

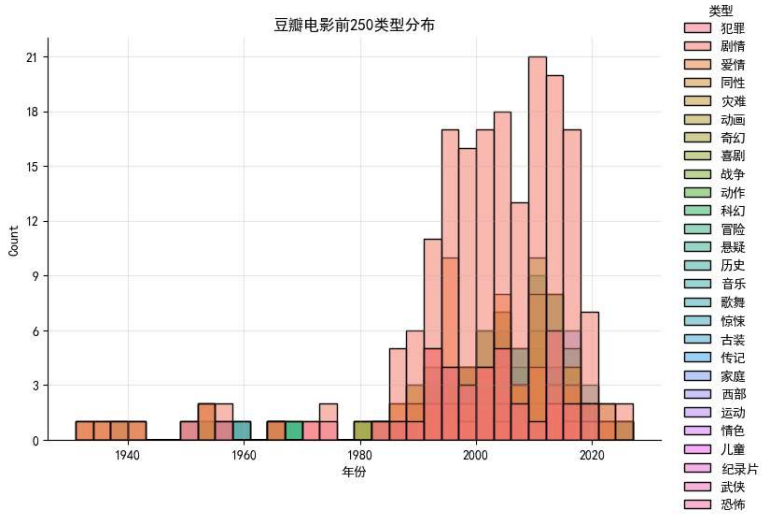

图 4：电影前 250 类型分布

豆瓣电影前 250 国家分布可视化如下：

```
sns.displot(x='年份', data=year_country, hue='国家',
            binwidth=3, height=5, aspect=1.5)
plt.gca().yaxis.set_major_locator(matplotlib.ticker.MaxNLocator(integer
=True))
plt.grid(True, alpha=0.3)
    plt.title('豆瓣电影前 250 国家分布')
```

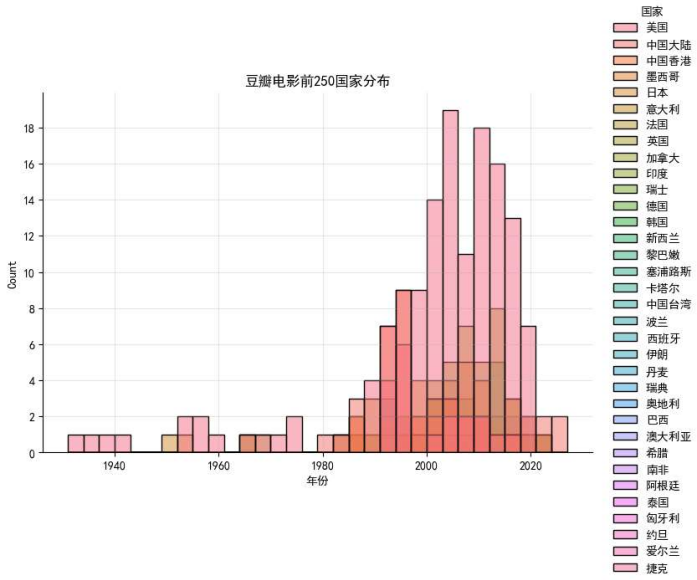

图 5：电影前 250 国家分布