

**课程设计报告**

( 2024--2025年度 第 一学期 )

课程名称: python高级应用

院 系: 软件学院

专业班级: 2022级软件工程6班

学 号: 2203260111

学生姓名: 吴同宇

**猫眼电影年度票房排行榜和中国国家法定节假日爬取和分析**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **一** | **二** | **三** | **四** | **五** | **总分** |
|  |  |  |  |  |  |

**一、****选题的背景介绍（10分）**

为什么要选择此选题？要达到的数据分析的预期目标是什么？ 从社会、经济、技术、数据来源等方面进行描述（200 字以内）

随着互联网的发展，电影行业逐渐成为人们娱乐生活的重要组成部分。电影票房数据不仅反映了观众的观影偏好，还对电影制作、宣传策略以及市场预测具有重要指导意义。通过爬取猫眼电影票房排行榜的数据，我们可以深入分析不同电影的票房表现，从而为电影行业的决策提供数据支持。

本项目旨在通过网络爬虫技术获取猫眼电影票房排行榜的数据，进一步分析电影的票房、平均票价、场均人次等指标，探索影响电影票房的关键因素，为电影制作方、发行方和院线提供有价值的参考。

社会：电影是大众文化的重要载体，了解观众的观影习惯有助于提升电影质量和观影体验。

经济：电影票房直接影响电影产业的经济效益，数据分析可以帮助优化资源配置。

技术：网络爬虫技术可以高效地获取大量数据，Python 的 requests 和 BeautifulSoup 库提供了强大的网页抓取和解析功能。

数据来源：猫眼电影票房排行榜提供了丰富的电影票房数据，包括片名、上映日期、票房、平均票价和场均人次等信息。（https://piaofang.maoyan.com/rankings/year）

**二、网络爬虫设计方案（15 分）**

1. 网络爬虫爬取的内容与数据特征分析

内容：猫眼电影票房排行榜的电影名称、上映日期、票房（万元）、平均票价、场均人次。

数据特征：

电影名称：字符串类型，唯一标识每部电影。

上映日期：日期类型，记录电影的上映时间。

票房（万元）：数值类型，表示电影的总票房。

平均票价：数值类型，表示平均每张票的价格。

场均人次：数值类型，表示每场电影的平均观众人数。

2. 网络爬虫设计方案概述

技术选型：使用 Python 的 requests 库发送 HTTP 请求，获取网页内容；使用 BeautifulSoup 库解析 HTML 页面，提取所需数据。

流程：

发送 HTTP 请求，获取猫眼电影票房排行榜的 HTML 页面。

使用 BeautifulSoup 解析 HTML 页面，提取电影名称、上映日期、票房、平均票价和场均人次。

将提取的数据写入 CSV 文件，方便后续的数据分析。

**三、网络爬虫具体步骤（55 分）**

1. 主题页面的结构特征分析、HTML 页面解析（10分）

页面结构：

主页 URL：https://piaofang.maoyan.com/rankings/year

目标数据位于 div 标签中，id 为 ranks-list。

每个电影的详细信息分布在多个 ul 标签中，每个 ul 标签包含多个 li 标签，分别对应不同的数据字段。

解析步骤：

使用 requests 库发送 GET 请求，获取网页内容。

使用 BeautifulSoup 解析 HTML 页面，找到 id 为 ranks-list 的 div 标签。

在 div 标签中找到所有的 ul 标签，每个 ul 标签对应一部电影。

在每个 ul 标签中找到不同类名的 li 标签，提取相应的数据字段。

2. 数据爬取、采集与处理（20分）

数据爬取：

遍历 div 标签中的所有 ul 标签，提取每部电影的详细信息。

使用 find\_all 方法找到不同类名的 li 标签，提取相应的数据字段。

数据处理：

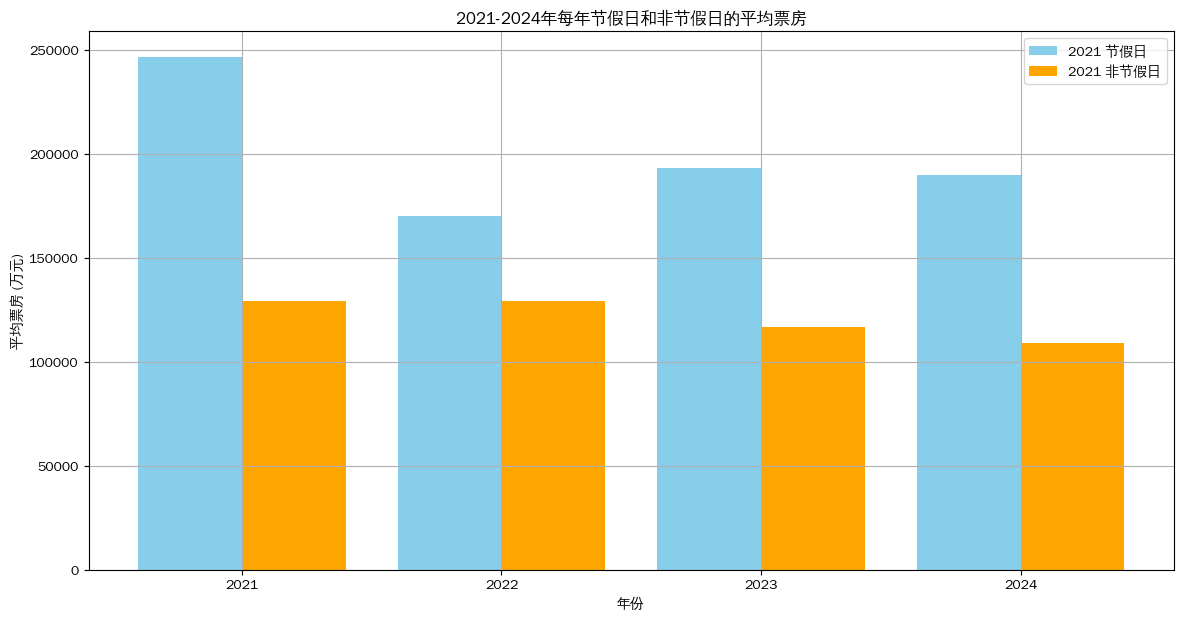
去除数据中的多余空白字符。

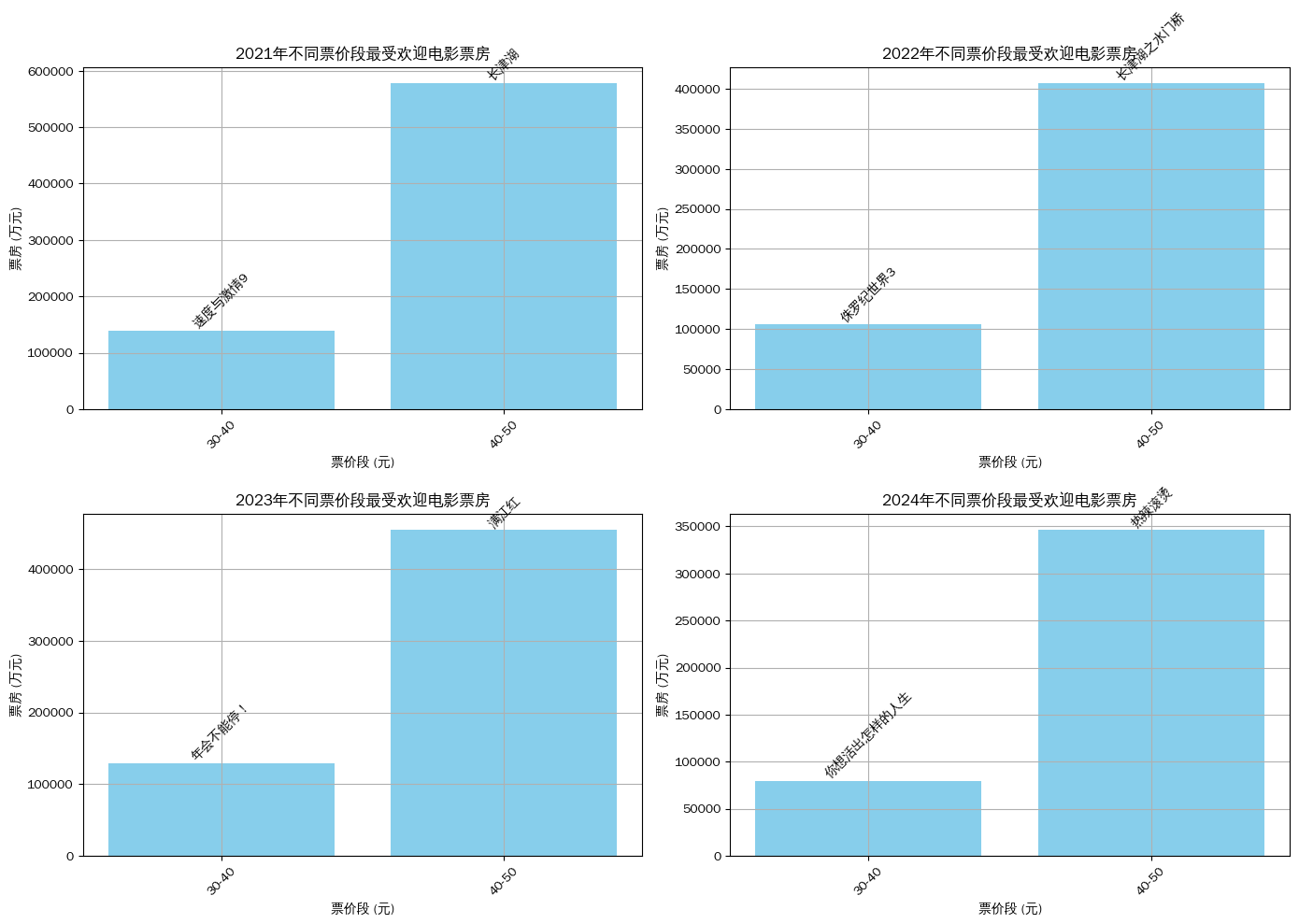
将提取的数据写入 CSV 文件，方便后续的数据分析。

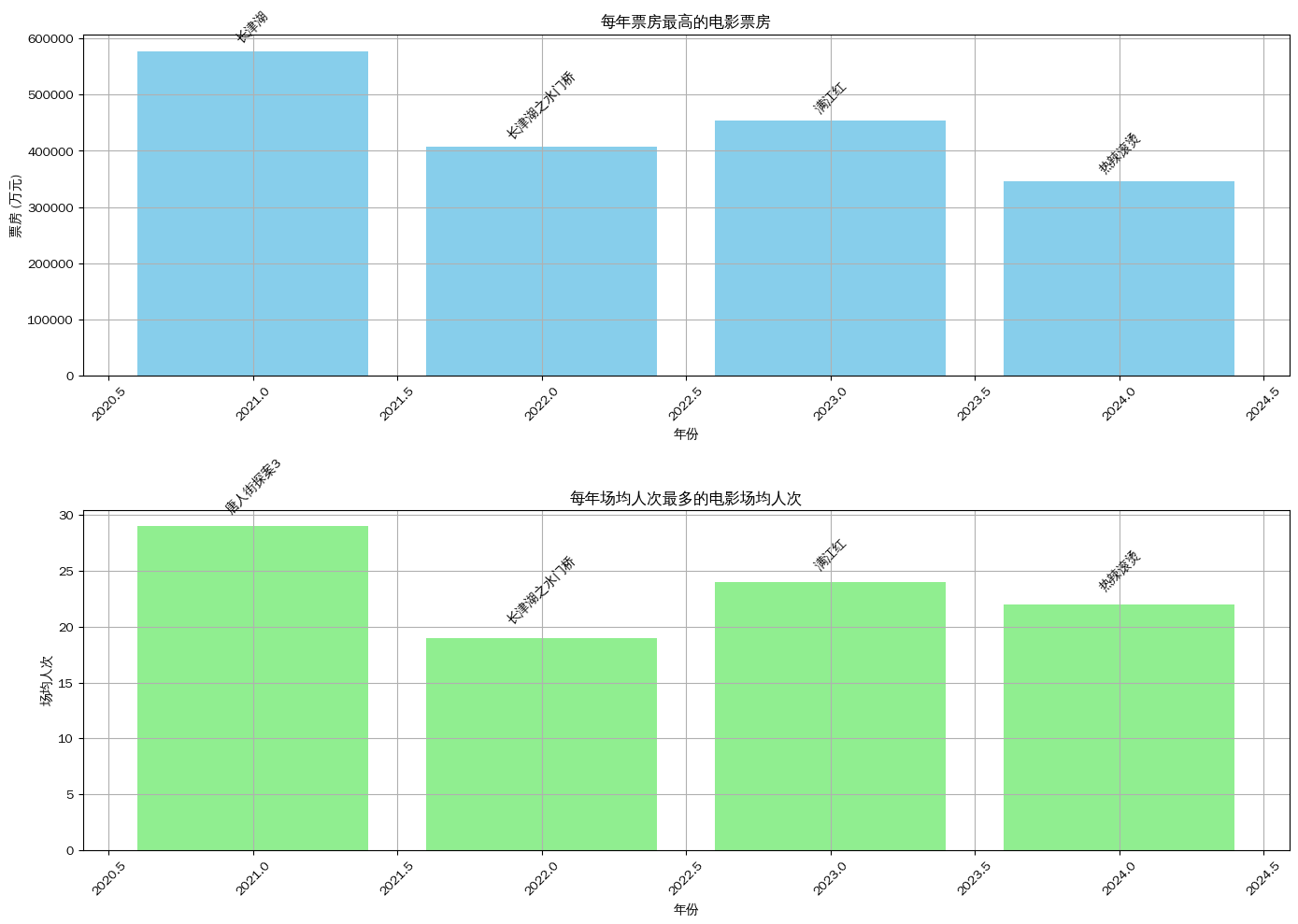
**四、总结（10分）**

通过前面各个图表的分析，得出数据集综合分析的结论，是否达到预期的目标，结论必需和选题的背景介绍要达到的数据分析目标相对应。（不少于150字）。

图表





****

**五、****附录（10分）**

完整源代码（代码要做好注释，说明步骤中的过程，要与前面的数据可视化分析图表对应。代码不少于180行，注释部分不计入代码的行数）。

import requests

from bs4 import BeautifulSoup

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.42"

} # 将程序发出的请求伪装成浏览器

text = requests.get("https://piaofang.maoyan.com/rankings/year", headers=headers).text

# 通过requests请求到电影票房的网页

main\_page = BeautifulSoup(text, "html.parser") # 后面这个是html的解析器

# 使用BeautifulSoup对text进行解析

div = main\_page.find("div", attrs={"id": "ranks-list"})

# 找到名为div且id为ranks-list的所有标签

with open("MovieData.csv", mode="w", encoding="utf-8", newline='') as f: # 创建一个名为MovieData.csv的文件，放在变量f中，且可以在其后面直接追加文字

f.write("片名,上映日期,票房(万元),平均票价,场均人次\n") # 写入表头

uls = div.find\_all('ul')

# 找到div元素中所有名为ul的元素，并存储在uls变量中

for ul in uls: # 对uls中的元素进行遍历

lst1 = ul.find\_all("li", attrs={"class": "col1"}) # 找到ul中所有类名为col1的Li元素，并储存在lst1

lst2 = ul.find\_all("li", attrs={"class": "col2"}) # 找到ul中所有类名为col2的Li元素，并储存在lst2

lst3 = ul.find\_all("li", attrs={"class": "col3"}) # 找到ul中所有类名为col3的Li元素，并储存在lst3

lst4 = ul.find\_all("li", attrs={"class": "col4"}) # 找到ul中所有类名为col4的Li元素，并储存在lst4

row\_data = [] # 用于存储当前行的数据

for li in lst1: # 对lst1中的元素进行遍历

ps = li.find\_all("p") # 找到ps中所有p元素，并储存在ps

for p in ps: # 对ps中的元素进行遍历

row\_data.append(p.text.strip()) # 将p标签中的文本去除分隔符后添加到row\_data中

for li in lst2: # 对lst2中的元素进行遍历

row\_data.append(li.text.strip()) # 将li标签中的文本去除分隔符后添加到row\_data中

for li in lst3: # 对lst3中的元素进行遍历

row\_data.append(li.text.strip()) # 将li标签中的文本去除分隔符后添加到row\_data中

for li in lst4: # 对lst4中的元素进行遍历

row\_data.append(li.text.strip()) # 将li标签中的文本去除分隔符后添加到row\_data中

f.write(",".join(row\_data) + "\n") # 使用join方法将row\_data中的元素连接成一个字符串，并写入文件

# 筛选掉MovieData.csv中，上映日期在2021年之前的数据

import pandas as pd

# 读取CSV文件

df = pd.read\_csv('MovieData.csv')

# 将'上映日期'列转换为日期格式

df['上映日期'] = pd.to\_datetime(df['上映日期'].str.extract(r'(\d{4}-\d{2}-\d{2})')[0])

# 筛选出2021年及之后上映的电影

filtered\_df = df[df['上映日期'] >= '2021-01-01']

# 将数据条目按照上映日期先后排序

sorted\_df = filtered\_df.sort\_values('上映日期')

# 输出到MovieAfter2021.csv

sorted\_df.to\_csv('MovieAfter2021.csv', index=False)

print(sorted\_df)

import requests

import re

import importlib

import sys

import pandas as pd

import datetime

importlib.reload(sys)

class Spider(object):

def \_\_init\_\_(self):

print(u'开始爬取...')

# 获取网页源码，返回页面所有信息

def get\_source(self, url):

html = requests.get(url)

return html.text

# 改变url实现抓取所有年份的假日，得所有url列表

def change\_page(self, url, now\_page, total\_page):

page\_group = []

for i in range(now\_page, total\_page + 1):

link = url[:26] + str(i) + url[30:]

page\_group.append(link)

return page\_group

# 把页面上假日的table截取出来，返回字符串

def get\_every\_table(self, source):

everytable = re.findall('(<table style=.\*?</table>)', source, re.S)

return everytable

# 取出放假日期范围，返回日期列表

def get\_holiday(self, year, string):

string = "".join(re.findall('\d.\*', string, re.S))

if string.find('年') >= 0:

string = string[5:]

if string.find('~') == -1:

string = string + '~' + string

str1 = string.split('~')

m1 = "".join(re.findall('\d.\*(?=月\d)', str1[0], re.S))

d1 = "".join(re.findall('(?<=月).\*\d', str1[0], re.S))

m2 = "".join(re.findall('\d.\*(?=月\d)', str1[1], re.S))

d2 = "".join(re.findall('(?<=月).\*\d', str1[1], re.S))

# 规范化日期格式为yyyy mm dd

if int(m1) < 10:

m1 = '0' + m1

if int(d1) < 10:

d1 = '0' + d1

if int(m2) < 10:

m2 = '0' + m2

if int(d2) < 10:

d2 = '0' + d2

if int(m1) == 12:

y1 = str(int(year) - 1) # 如果元旦前是12月则月份要取少一年

else:

y1 = year

# 根据日期范围获取所有连续日期

date\_list = []

begin\_date = datetime.datetime.strptime(y1 + m1 + d1, "%Y%m%d")

end\_date = datetime.datetime.strptime(year + m2 + d2, "%Y%m%d")

while begin\_date <= end\_date:

date\_str = begin\_date.strftime("%Y%m%d")

date\_list.append(date\_str)

begin\_date += datetime.timedelta(days=1)

return date\_list

# 取出调休上班日期，返回日期列表

def get\_workday(self, year, string):

string = "".join(re.findall('\d.\*', string, re.S))

if string.find('年') >= 0:

string = string[5:]

list1 = re.split('[，、]', string)

list2 = []

for a in list1:

m1 = "".join(re.findall('\d.\*(?=月\d)', a, re.S))

d1 = "".join(re.findall('(?<=月).\*\d', a, re.S))

if int(m1) == 12:

y1 = str(int(year) - 1)

else:

y1 = year

if int(m1) < 10:

m1 = '0' + m1

if int(d1) < 10:

d1 = '0' + d1

list2.append(y1 + m1 + d1)

return list2

# 分割源码获得每年所有假日表：日期、节日名称、是否放假

def get\_info(self, eachyear):

info = {}

ho = re.findall('jiad/">(.\*?)</a>', eachyear, re.S) # 节日名称列表

year = re.search('<a href="/(.\*?)\_', eachyear, re.S).group(1) # 年份字符串

a = re.findall('<td>(.\*?)</td>', eachyear, re.S) # 除了节日名称列的其他数据列表，包括放假时间、调休上班日期、放假天数

j = len(a)

# 得出放假日期

i = 0

k = 0

df = pd.DataFrame()

while i < j:

if i > j:

continue

holidayname = ''.join(ho[k])

holiday\_list = list(map(int, self.get\_holiday(year, a[i])))

df1 = pd.DataFrame(holiday\_list, columns=['period\_id'])

df1['holiday\_name'], df1['is\_holiday'] = [holidayname, 1]

df = pd.concat([df, df1])

i += 3

k += 1

# 得出调休上班日期

m = 1

n = 0

df2 = pd.DataFrame()

while m < j:

if m > j:

continue

holidayname = ''.join(ho[n])

data = a[m]

checkdata = a[m].find('休') # 是否包含休

checkdata2 = a[m].find('无') # 是否包含无

m += 3

n += 1

if checkdata >= 0 or checkdata2 >= 0: # 如果该节日没有特殊调休安排则跳过循环

continue

work\_list = list(map(int, self.get\_workday(year, data)))

df\_temp = pd.DataFrame(work\_list, columns=['period\_id'])

df\_temp['holiday\_name'], df\_temp['is\_holiday'] = [holidayname, 0]

df2 = pd.concat([df2, df\_temp])

return pd.concat([df, df2])

# 将结果写入csv

def write2csv(self, filepath, data):

data.to\_csv(filepath, index=False)

# 将结果追加到csv

def add2csv(self, filepath, data):

data.to\_csv(filepath, mode='a', index=False, header=False)

if \_\_name\_\_ == '\_\_main\_\_':

# 初始化参数

url = 'https://fangjia.51240.com/2024\_\_fangjia/' # 抓取的页面

start\_page = 2021 # 抓取开始年份

end\_page = 2024 # 抓取结束年份

filepath = 'holiday.csv' # 写入csv文件路径

print(u'初始化类..')

myspider = Spider()

all\_links = myspider.change\_page(url, start\_page, end\_page)

df = pd.DataFrame()

for link in all\_links:

print(u'正在处理页面..' + link)

html = myspider.get\_source(link)

everyyear = myspider.get\_every\_table(html)

for each in everyyear:

df = pd.concat([df, myspider.get\_info(each)])

df.index = range(len(df))

df.sort\_values(by='period\_id', inplace=True)

# 生成csv

myspider.write2csv(filepath, df)

# csv中增加内容

# myspider.add2csv(filepath, df) # 写入csv，并追加内容

print('写入CSV完成！')

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib import rcParams

from datetime import datetime

# 设置中文字体

rcParams['font.sans-serif'] = ['WenQuanYi Zen Hei']

rcParams['axes.unicode\_minus'] = False # 解决负号显示问题

# 1. 数据加载

movie\_df = pd.read\_csv('MovieAfter2021.csv')

holiday\_df = pd.read\_csv('holiday.csv')

# 2. 数据预处理

# 将上映日期转换为datetime格式

movie\_df['上映日期'] = pd.to\_datetime(movie\_df['上映日期'])

# 将period\_id转换为datetime格式

holiday\_df['period\_id'] = pd.to\_datetime(holiday\_df['period\_id'], format='%Y%m%d')

# 3. 数据分析

# 创建一个列来标记电影上映日期是否为法定节假日

movie\_df['is\_holiday'] = movie\_df['上映日期'].apply(lambda x: 1 if x in holiday\_df[holiday\_df['is\_holiday'] == 1]['period\_id'].values else 0)

# 按年份和是否为法定节假日分组，计算平均票房

grouped = movie\_df.groupby([movie\_df['上映日期'].dt.year, 'is\_holiday'])['票房(万元)'].mean().reset\_index()

# 4. 数据可视化

# 设置图形大小

plt.figure(figsize=(14, 7))

# 绘制每年节假日和非节假日的平均票房

for year in grouped['上映日期'].unique():

holiday\_data = grouped[(grouped['上映日期'] == year) & (grouped['is\_holiday'] == 1)]

non\_holiday\_data = grouped[(grouped['上映日期'] == year) & (grouped['is\_holiday'] == 0)]

plt.bar(year - 0.2, holiday\_data['票房(万元)'].values[0], width=0.4, label=f'{year} 节假日' if year == 2021 else '', color='skyblue')

plt.bar(year + 0.2, non\_holiday\_data['票房(万元)'].values[0], width=0.4, label=f'{year} 非节假日' if year == 2021 else '', color='orange')

# 添加标题和标签

plt.title('2021-2024年每年节假日和非节假日的平均票房')

plt.xlabel('年份')

plt.ylabel('平均票房 (万元)')

plt.xticks(range(2021, 2025))

plt.legend()

plt.grid(True)

plt.show()

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

# 1. 数据加载

movie\_df = pd.read\_csv('MovieAfter2021.csv')

# 设置中文字体

rcParams['font.sans-serif'] = ['WenQuanYi Zen Hei'] # 使用文泉驿正黑

rcParams['axes.unicode\_minus'] = False # 解决负号显示问题

# 2. 数据预处理

# 将上映日期转换为datetime格式

movie\_df['上映日期'] = pd.to\_datetime(movie\_df['上映日期'])

# 提取年份

movie\_df['年份'] = movie\_df['上映日期'].dt.year

# 3. 票价分段

# 定义票价分段区间

price\_bins = range(0, 60, 10) # 0-10, 10-20, ..., 50-60

price\_labels = [f'{i}-{i+10}' for i in price\_bins[:-1]]

# 将平均票价分段

movie\_df['票价段'] = pd.cut(movie\_df['平均票价'], bins=price\_bins, labels=price\_labels, right=False)

# 4. 数据分析

# 创建一个字典来存储每年每个票价段内票房最高的电影

top\_movies\_by\_year\_and\_price\_range = {}

for year in range(2021, 2025):

top\_movies\_by\_year\_and\_price\_range[year] = {}

yearly\_data = movie\_df[movie\_df['年份'] == year]

for price\_range in price\_labels:

range\_data = yearly\_data[yearly\_data['票价段'] == price\_range]

if not range\_data.empty:

top\_movie = range\_data.loc[range\_data['票房(万元)'].idxmax()]

top\_movies\_by\_year\_and\_price\_range[year][price\_range] = {

'片名': top\_movie['片名'],

'票房(万元)': top\_movie['票房(万元)'],

'平均票价': top\_movie['平均票价']

}

# 5. 数据可视化

# 设置图形大小

plt.figure(figsize=(14, 10))

# 绘制每年不同票价段最受欢迎的电影票房

for year in range(2021, 2025):

data = top\_movies\_by\_year\_and\_price\_range[year]

prices = list(data.keys())

box\_offices = [data[price]['票房(万元)'] for price in prices]

plt.subplot(2, 2, year - 2020)

plt.bar(prices, box\_offices, color='skyblue')

plt.title(f'{year}年不同票价段最受欢迎电影票房')

plt.xlabel('票价段 (元)')

plt.ylabel('票房 (万元)')

plt.xticks(rotation=45)

plt.grid(True)

# 添加每个票价段最受欢迎电影的名称

for i, price in enumerate(prices):

plt.text(i, box\_offices[i] + 1000, data[price]['片名'], ha='center', va='bottom', rotation=45)

plt.tight\_layout()

plt.show()

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime

# 设置中文字体

rcParams['font.sans-serif'] = ['WenQuanYi Zen Hei'] # 使用文泉驿正黑

rcParams['axes.unicode\_minus'] = False # 解决负号显示问题

# 1. 数据加载

movie\_df = pd.read\_csv('MovieAfter2021.csv')

# 2. 数据预处理

# 将上映日期转换为datetime格式

movie\_df['上映日期'] = pd.to\_datetime(movie\_df['上映日期'])

# 提取年份

movie\_df['年份'] = movie\_df['上映日期'].dt.year

# 3. 数据分析

# 创建字典来存储每年票房最高的电影和场均人次最多的电影

top\_movies\_by\_year = {}

for year in range(2021, 2025):

yearly\_data = movie\_df[movie\_df['年份'] == year]

# 找到票房最高的电影

highest\_box\_office\_movie = yearly\_data.loc[yearly\_data['票房(万元)'].idxmax()]

highest\_avg\_audience\_movie = yearly\_data.loc[yearly\_data['场均人次'].idxmax()]

top\_movies\_by\_year[year] = {

'票房最高': {

'片名': highest\_box\_office\_movie['片名'],

'票房(万元)': highest\_box\_office\_movie['票房(万元)'],

'平均票价': highest\_box\_office\_movie['平均票价'],

'场均人次': highest\_box\_office\_movie['场均人次']

},

'场均人次最高': {

'片名': highest\_avg\_audience\_movie['片名'],

'票房(万元)': highest\_avg\_audience\_movie['票房(万元)'],

'平均票价': highest\_avg\_audience\_movie['平均票价'],

'场均人次': highest\_avg\_audience\_movie['场均人次']

}

}

# 4. 数据可视化

# 设置图形大小

plt.figure(figsize=(14, 10))

# 绘制每年票房最高的电影票房和场均人次

years = list(top\_movies\_by\_year.keys())

box\_office\_movies = [top\_movies\_by\_year[year]['票房最高']['片名'] for year in years]

box\_office\_values = [top\_movies\_by\_year[year]['票房最高']['票房(万元)'] for year in years]

avg\_audience\_movies = [top\_movies\_by\_year[year]['场均人次最高']['片名'] for year in years]

avg\_audience\_values = [top\_movies\_by\_year[year]['场均人次最高']['场均人次'] for year in years]

# 票房最高的电影

plt.subplot(2, 1, 1)

plt.bar(years, box\_office\_values, color='skyblue')

plt.title('每年票房最高的电影票房')

plt.xlabel('年份')

plt.ylabel('票房 (万元)')

plt.xticks(rotation=45)

plt.grid(True)

# 在每个柱状图上标注电影名称

for i, year in enumerate(years):

plt.text(year, box\_office\_values[i] + 10000, box\_office\_movies[i], ha='center', va='bottom', rotation=45)

# 场均人次最高的电影

plt.subplot(2, 1, 2)

plt.bar(years, avg\_audience\_values, color='lightgreen')

plt.title('每年场均人次最多的电影场均人次')

plt.xlabel('年份')

plt.ylabel('场均人次')

plt.xticks(rotation=45)

plt.grid(True)

# 在每个柱状图上标注电影名称

for i, year in enumerate(years):

plt.text(year, avg\_audience\_values[i] + 1, avg\_audience\_movies[i], ha='center', va='bottom', rotation=45)

plt.tight\_layout()

plt.show()

# 打印结果

for year in years:

print(f"Year: {year}")

print(f"票房最高的电影: {top\_movies\_by\_year[year]['票房最高']['片名']} (票房: {top\_movies\_by\_year[year]['票房最高']['票房(万元)']} 万元)")

print(f"场均人次最多的电影: {top\_movies\_by\_year[year]['场均人次最高']['片名']} (场均人次: {top\_movies\_by\_year[year]['场均人次最高']['场均人次']})")

print("-" \* 50)