

**Python程序设计**

**学 院：＿计算机科学与技术**

**题 目： 链家新房爬虫及结果分析**

**专 业：＿ 信息安全＿＿＿**

**班 级：＿ 信息2101＿＿**

**学 号： 2100170042**

**姓 名： 吴江林**

**指导老师： 段晓旗**

2023年 11月 28日

**目录**

1背景和意义……………………………………………………………………………………3

1.1背景…………………………………………………………………………………………3

1.2意义…………………………………………………………………………………………3

2方法介绍及相关工作介绍……………………………………………………………………4

2.1方法介绍…………………………………………………………………………………4

2.1.1请求头信息…………………………………………………………………………4

2.1.2发送http请求………………………………………………………………………5

2.1.3解析HTML……………………………………………………………………………5

2.1.4数据写入……………………………………………………………………………5

2.1.5提取信息并写入csv文件…………………………………………………………6

2.1.6异常处理……………………………………………………………………………7

2.1.7睡眠随机时间………………………………………………………………………7

2.2相关工作介绍……………………………………………………………………………8

2.3总结………………………………………………………………………………………8

3结果及分析……………………………………………………………………………………9

3.1 爬取数据结果……………………………………………………………………………9

3.2结果分析及可视化………………………………………………………………………10

3.2.1辖区分布占比………………………………………………………………………10

3.2.2楼盘类型占比………………………………………………………………………11

3.2.3各辖区均价排序……………………………………………………………………13

3.2.4各标签数量统计……………………………………………………………………14

3.2.5最小面积和最大面积的分布情况…………………………………………………18

3.2.6不同楼盘类型和不同辖区的均价分布情况………………………………………19

4讨论……………………………………………………………………………………………20

4.1 数据更新频率……………………………………………………………………………20

4.2 标签信息补充……………………………………………………………………………21

4.3 防止爬虫被封……………………………………………………………………………21

4.4 异常处理细化……………………………………………………………………………21

4.5 数据可视化平台…………………………………………………………………………21

5伪代码或关键代码……………………………………………………………………………22

5.1 爬虫代码…………………………………………………………………………………22

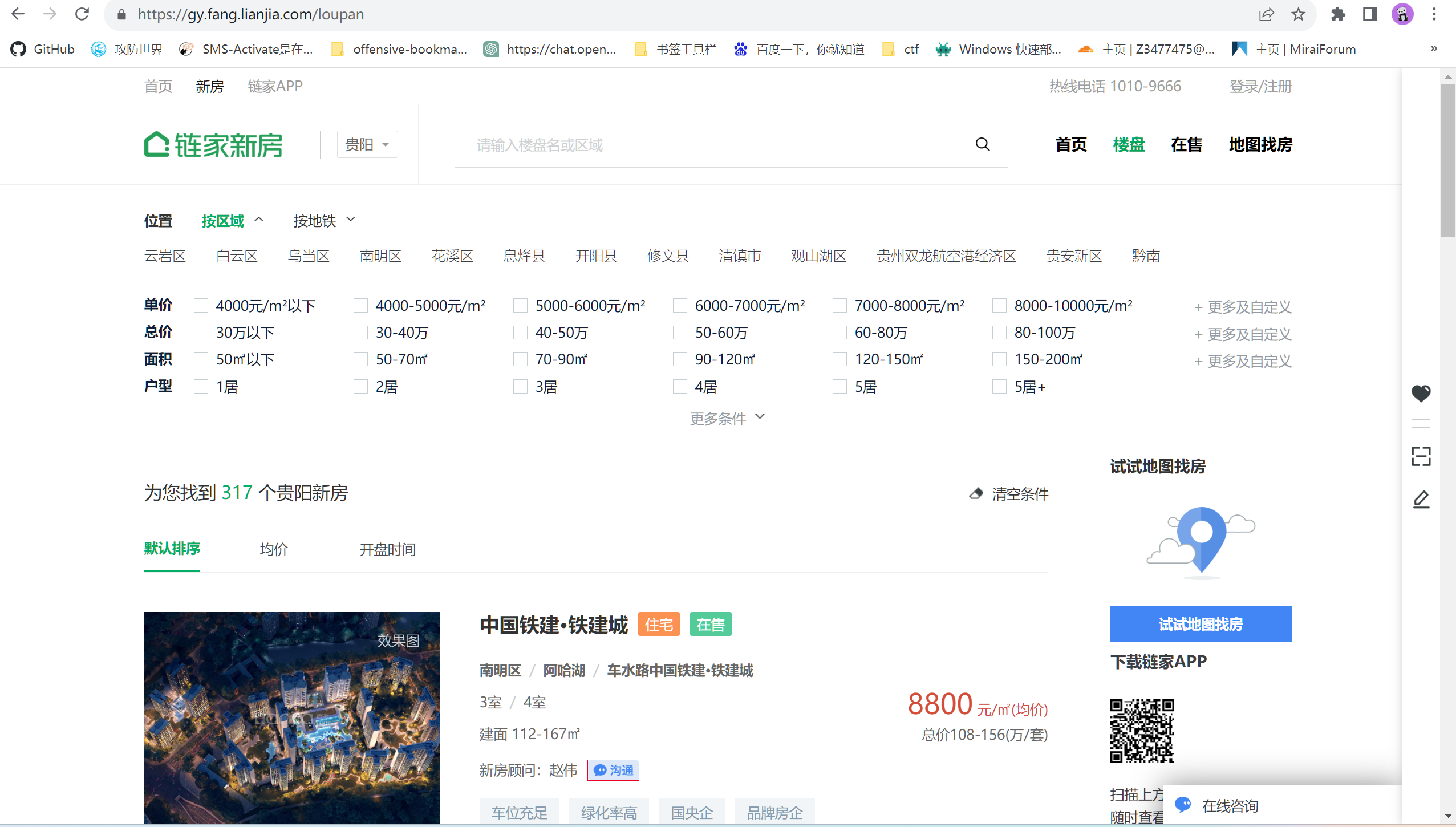
5.2 可视化代码………………………………………………………………………………25

# 1背景和意义

## 1.1背景

随着城市化的不断推进，房地产市场成为社会经济中一项重要的支柱产业。城市的扩张和人口的流动使得购房需求愈加迫切，而购房决策往往关系到家庭的未来生活质量和财务状况。在这个背景下，购房者对于准确、及时的新房信息需求急剧增加，以便能够做出明智的购房决策。

## 1.2意义



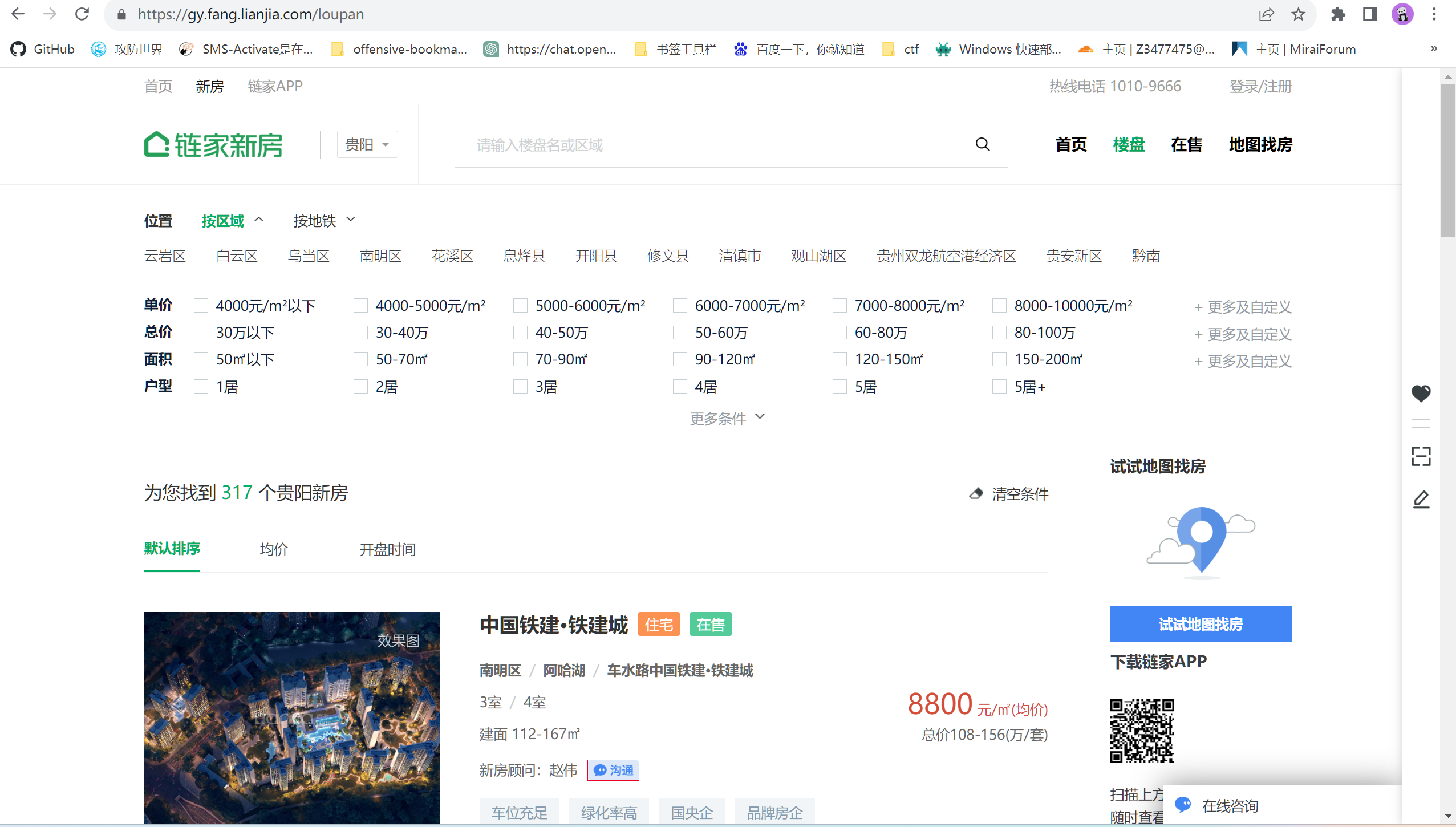
链家作为一家提供全产业链房产服务的平台，汇聚了大量的楼盘数据，包括新房的基本信息、售价、位置等。这使得链家成为购房者获取房产信息的重要渠道之一。为了更好地满足购房者的需求，开展链家新房爬虫并进行结果分析具有重要的实际意义。

首先，新房信息的获取对购房者而言是至关重要的。购房者可以通过了解市场上新房的价格走势、楼盘的位置特点、户型布局等方面的信息，更好地选择符合自身需求和经济状况的房产。因此，通过对链家新房数据进行爬取，我们能够提供购房者全面而准确的新房信息，帮助他们做出更明智的决策。

其次，对链家新房数据的结果分析有助于洞察房地产市场的动态。通过深入挖掘新房数据，我们可以了解不同城市、不同地段的楼盘热度、价格水平等特征，为市场研究者提供宝贵的数据支持。这有助于制定更精准的市场策略、预测市场走势，对于政府、开发商和投资者都有着重要的参考价值。

# 2方法介绍及相关工作介绍

## 2.1方法介绍



这是链家网站贵阳地区的房子信息，我们需要在这个页面提取楼盘的信息，比如一平方米的均价，它的总价为多少，它在那个区等等。要提取信息并保存在csv文件中。

### 2.1.1 请求头信息

headers = {  
 'User-Agent': UserAgent().random  
}

这部分代码使用fake\_useragent库生成一个随机的User-Agent头，以模拟浏览器请求，防止被网站识别为爬虫而被拒绝访问。

### 2.1.2 发送HTTP请求

res = requests.get(url, headers=headers, verify=False)

使用requests库向链家网站发送HTTP请求，获取网页内容。verify=False参数表示不验证SSL证书，这是因为有些网站的证书可能无法通过验证。

### 2.1.3 解析HTML

context = html.etree.HTML(res.text)

使用lxml库中的html.etree模块解析HTML文本，方便后续从网页中提取信息。



### 2.1.4 数据写入

with open('data.csv', 'a', newline='', encoding='utf-8-sig') as csvfile:  
 fieldnames = ['辖区', '楼盘名称', '位置', '详细地址', '类型', '楼盘状态', '最小面积','最大面积', '均价', '最低总价','最高总价', '标签1', '标签2', '标签3', '标签4']  
 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  
 # 检查文件是否为空，为空就写入标题  
 if csvfile.tell() == 0:  
 writer.writeheader()

在这一部分，脚本打开一个CSV文件用于存储爬取到的楼盘信息，并使用csv.DictWriter初始化一个写入器。如果文件为空，首先写入CSV文件的标题。

### 2.1.5 提取信息并写入CSV文件

ul = context.xpath('//ul[@class="resblock-list-wrapper"]')  
for i in ul:  
 li = i.xpath('.//li[@class="resblock-list post\_ulog\_exposure\_scroll has-results"]')  
 for j in li:  
 # ...  
 # 提取楼盘信息  
 writer.writerow({  
 '辖区': area[0],  
 '楼盘名称': name[0],  
 # ...  
 })  
 time.sleep(random.randint(1, 5))



这一部分是主要的信息提取和写入CSV文件的过程。通过XPath表达式选择特定的HTML元素，提取楼盘的相关信息，然后将这些信息写入CSV文件中。

### 2.1.6 异常处理

except Exception as e:  
 print('提取信息发生错误', e)  
 # ...

这部分是异常处理，当提取信息出现异常时，会输出错误信息，并记录一个空白的信息行到CSV文件中。

### 2.1.7 睡眠随机时间

sleeptime = random.randint(30, 35)  
print("睡眠%d秒后继续" % sleeptime)  
time.sleep(sleeptime)

为了模拟人的行为，爬虫在每次爬取一页后会随机休眠一段时间，以减缓请求频率，避免对服务器造成过大的压力。

## 2.2相关工作介绍

在爬虫工作中，使用requests库发起HTTP请求，并使用lxml库进行HTML解析是很常见的做法。fake\_useragent库的使用则可以有效防止被网站封锁，提高爬虫的稳定性。

脚本中的睡眠时间设置也是一种良好的爬虫策略，通过随机休眠时间，可以更好地模拟真实用户的行为，减少被服务器封锁的概率。

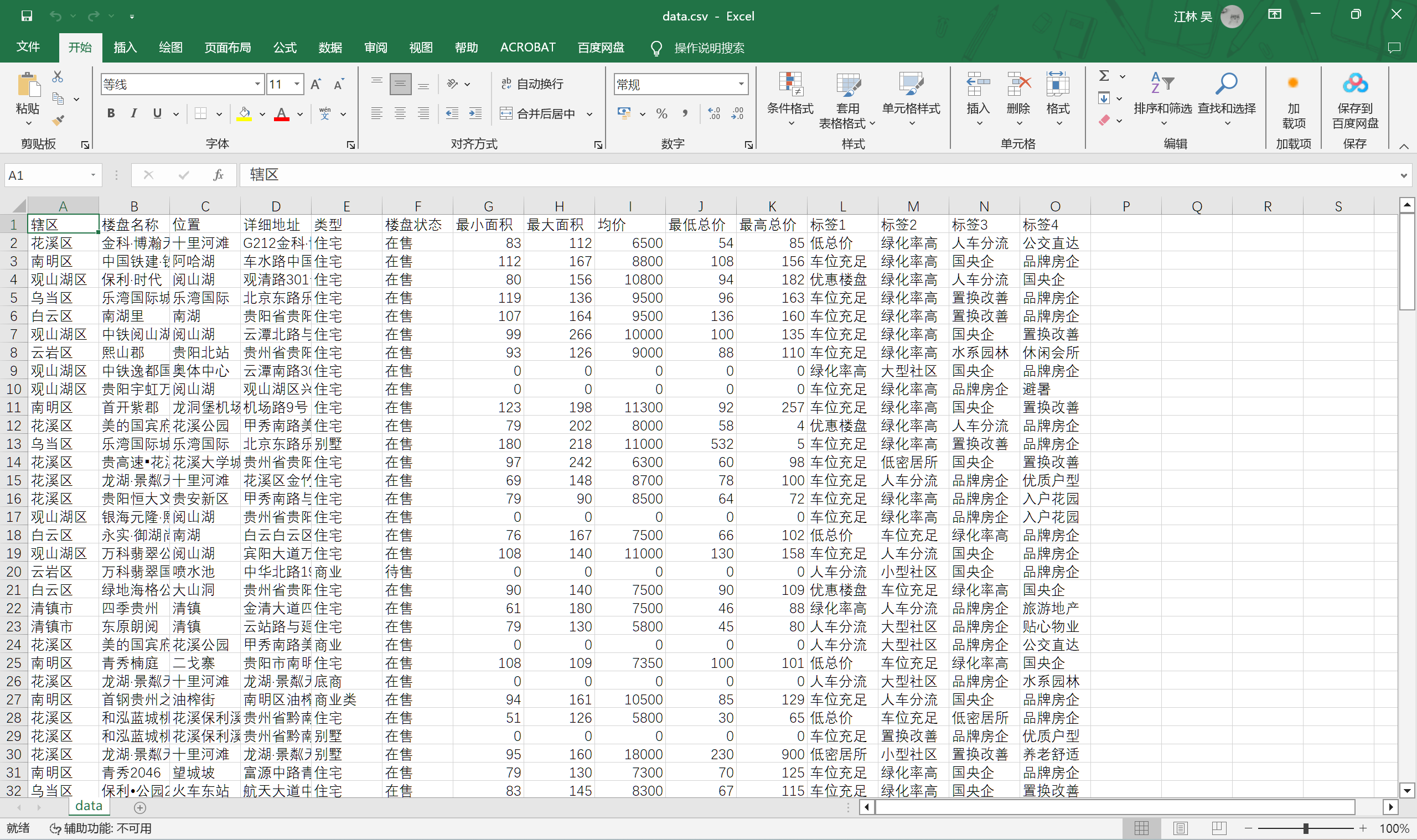
值得注意的是，异常处理的部分需要更加详细的处理，具体根据实际情况添加适当的异常类型处理，以保证爬虫的健壮性。在这个脚本中，异常处理部分包括了两次写入CSV文件的逻辑。

## 2.3总结

该爬虫脚本通过模拟浏览器行为，成功地爬取了链家网站上贵阳楼盘的信息，并将其存储在CSV文件中。爬虫中使用了一些常见的Python库，如requests、lxml、fake\_useragent等，通过这些工具，实现了对网页信息的提取和存储。在实际应用中，需要不断优化和维护，以适应目标网站的变化。

# 3结果及分析

## 3.1爬取数据结果



该爬虫脚本成功爬取了链家网站上关于贵阳楼盘的丰富信息，并将这些信息存储在CSV文件中。以下是主要的爬取结果字段：

辖区

楼盘名称

位置

详细地址

类型

楼盘状态

最小面积

最大面积

均价

最低总价

最高总价

标签1

标签2

标签3

标签4

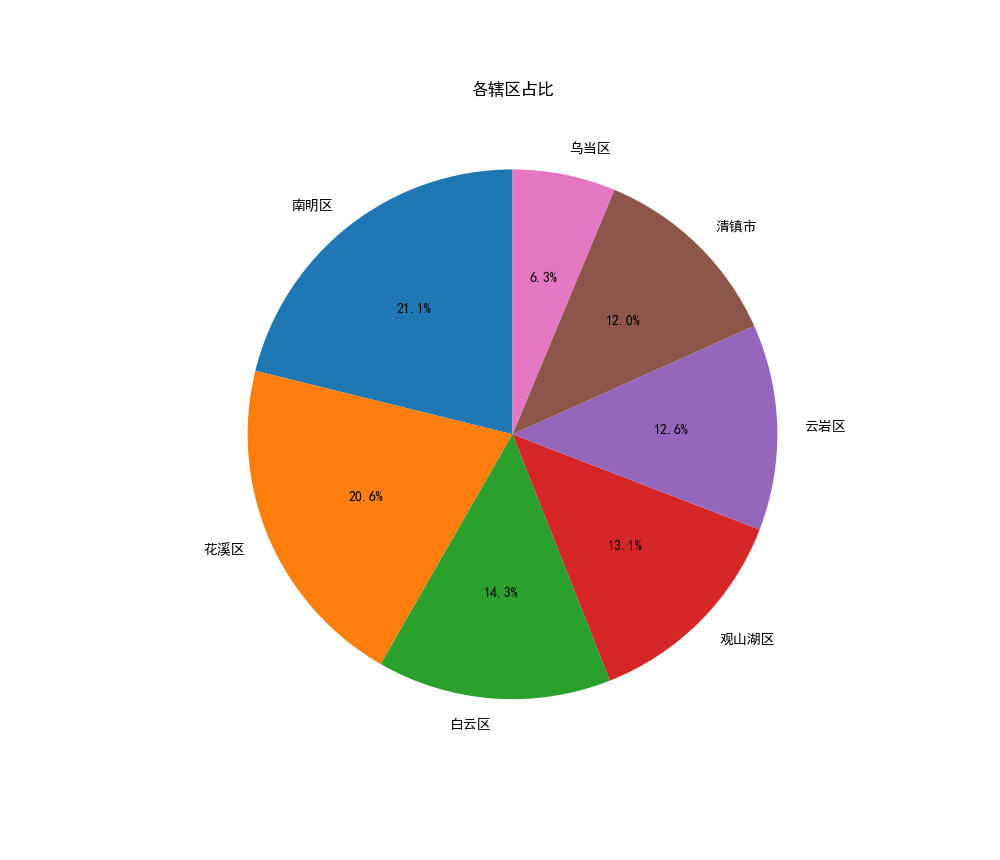
这些字段涵盖了楼盘的基本信息、地理位置、面积、价格以及一些标签信息，为购房者提供了全面而详细的楼盘数据。

## 3.2 结果分析及可视化

随着链家新房爬虫的成功运行并获取了大量的贵阳楼盘数据，接下来我们将对这些数据进行分析和可视化，以便更好地理解贵阳楼市的现状和趋势。

### 3.2.1 辖区分布占比

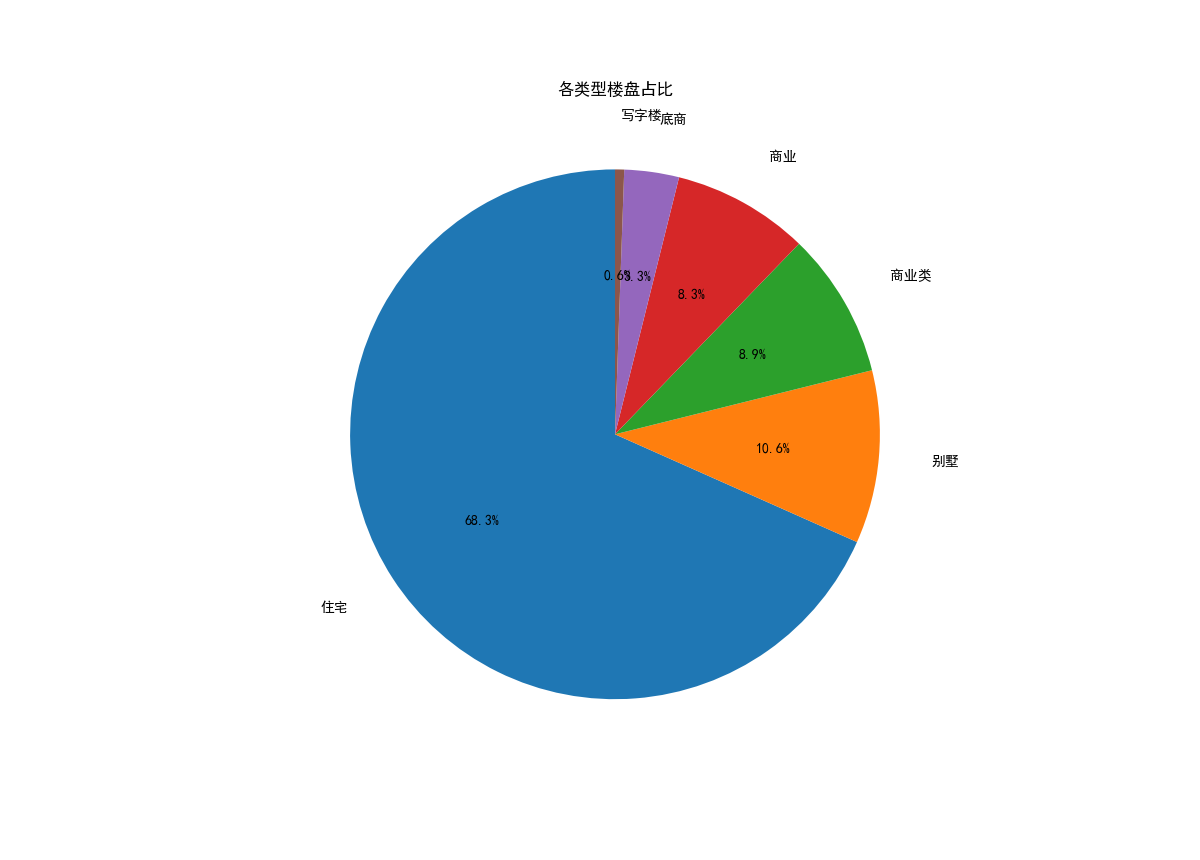
首先，我们对不同辖区内楼盘的分布情况进行了分析。通过扇形图展示了各个辖区楼盘数量的占比。如图所示：



在扇形图中清晰呈现出贵阳市各辖区楼盘的分布占比，尤其突出的是南明区和花溪区，它们占据了整个市场的相当大比例。对于购房者而言，这是一份有益的参考资料，因为这两个区域可能呈现出更为繁荣的发展态势，拥有更多的楼盘选择。南明区可能因其独特的地理位置和文化底蕴成为购房热点，而花溪区可能因其良好的生态环境和城市规划而备受青睐。购房者可以结合自身需求和偏好，在这些热门辖区中寻找更适合的房产，从而实现对高品质生活的向往。此外，对于投资者和市场分析者而言，这种分布情况也提供了有力的市场指引，有助于更精准地把握贵阳市楼市的发展趋势。

### 3.2.2 楼盘类型占比

其次，我们对不同类型的楼盘进行了占比分析。通过扇形图展示了各个类型楼盘数量的占比。如图所示：



通过楼盘类型占比的扇形图，我们清晰地看到了贵阳市各种类型楼盘的分布情况，为购房者提供了有趣的观察点。

住宅类楼盘呈现最高的占比，说明购房者对于居住需求的关注度最高。这可能与人们对于舒适住宅环境的追求、家庭生活的需求紧密相连。

其次是别墅，这表明一部分购房者更加注重独立、豪华的生活方式，对于独立空间和高品质生活环境有较高的期待。

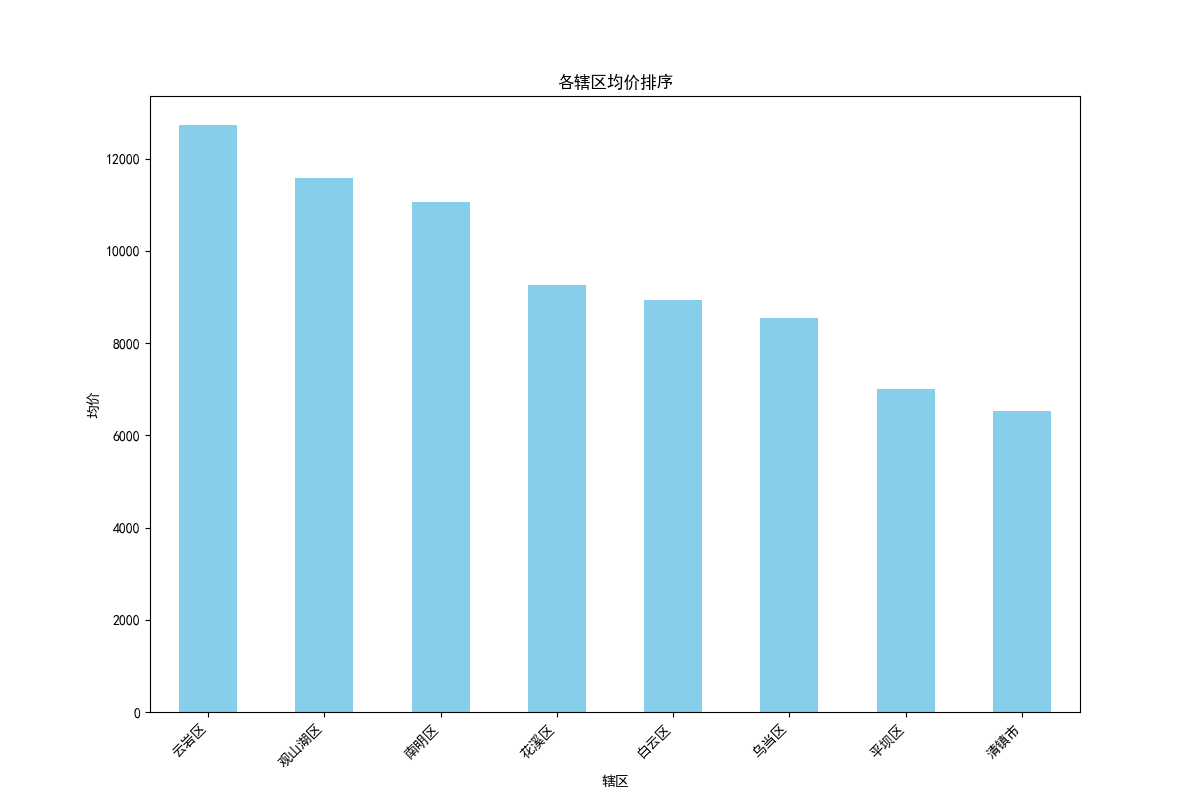
商业楼盘排在第三位，这反映了城市商业发展的重要性，对于购房者来说，商业楼盘的选择可能受到商业配套、交通便捷等因素的影响。

最后是写字楼，虽然在占比中排在末位，但在城市经济和商务中起到了重要作用。

这一观察点为购房者提供了更深入的思考，使他们能够更加清晰地根据个人需求和生活方式选择更符合心意的楼盘类型。此外，对于房地产开发商和市场分析者，这种类型分布的了解有助于更好地调整市场策略，满足市场需求，提高项目的竞争力。因此，这样的数据观察不仅在个人购房决策中有指导作用，同时也对整个房地产市场的发展趋势提供了有益的参考。

### 3.2.3 各辖区均价排序

进一步地，我们对各个辖区的楼盘均价进行了排序和排名。通过条形图展示了各辖区楼盘均价的情况。如图所示：



通过辖区均价排序的条形图，我们明显观察到贵阳市各辖区的楼盘均价存在显著差异，为购房者提供了关键的指标，有助于更好地了解各辖区的房价水平。均价在购房决策中是一个极其重要的参考因素，直接关系到购房者的财务承受能力和投资回报预期。

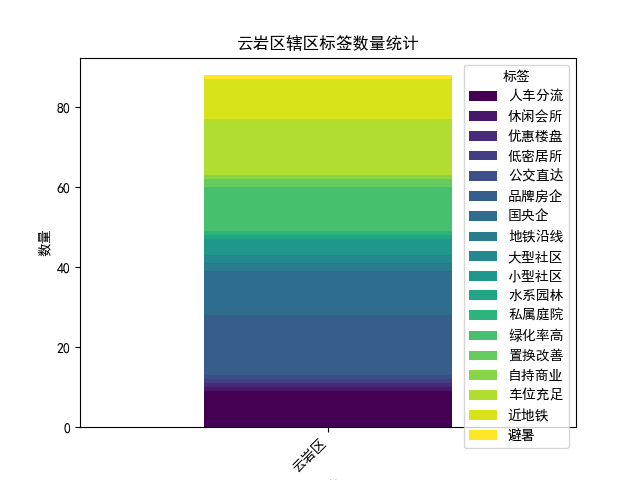
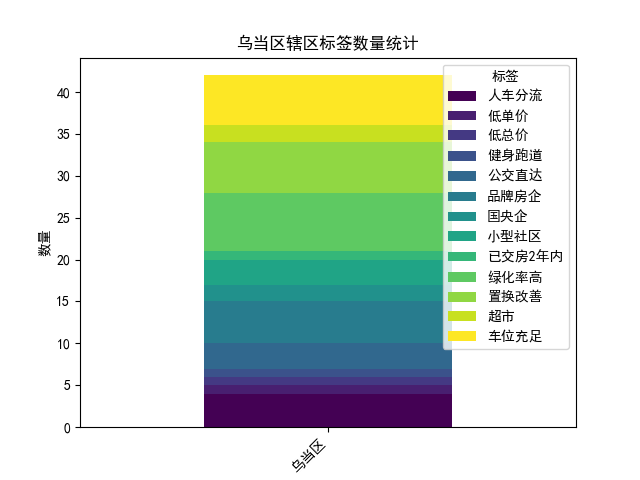
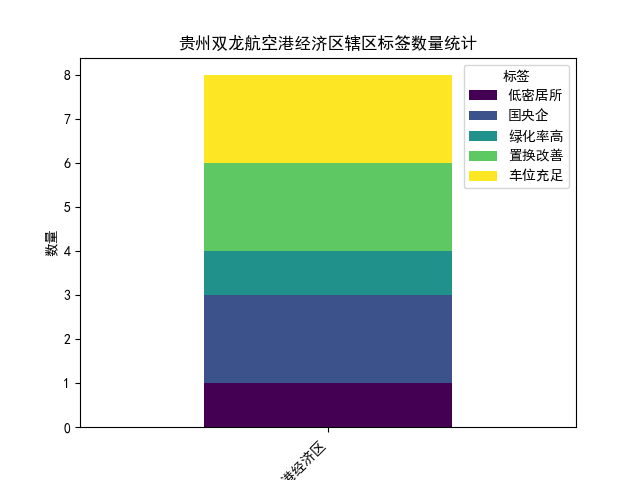
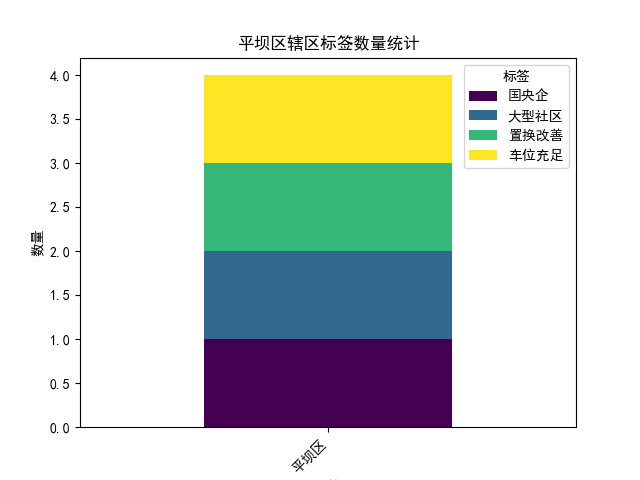
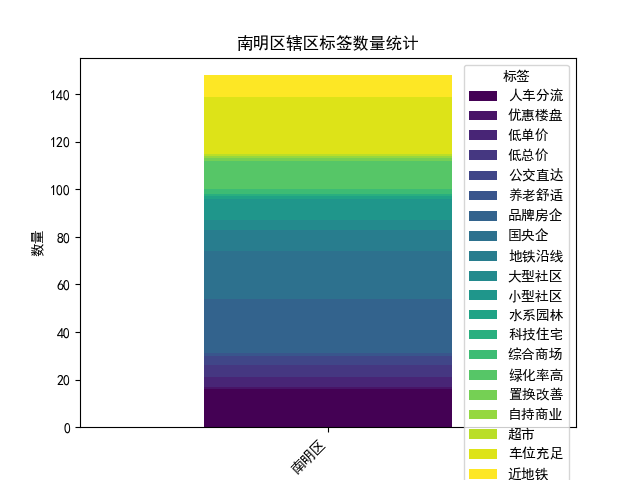
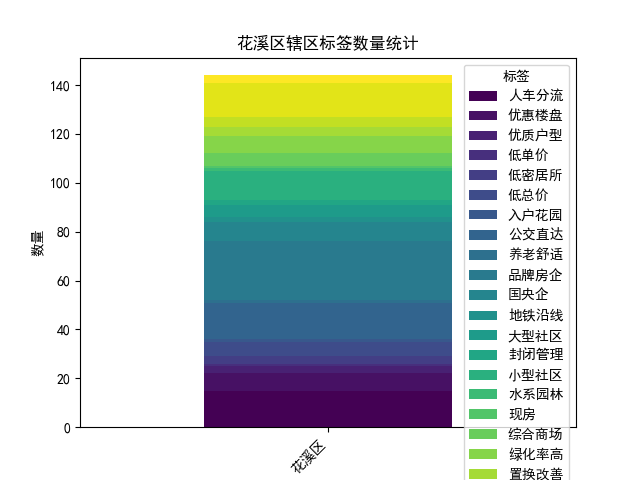
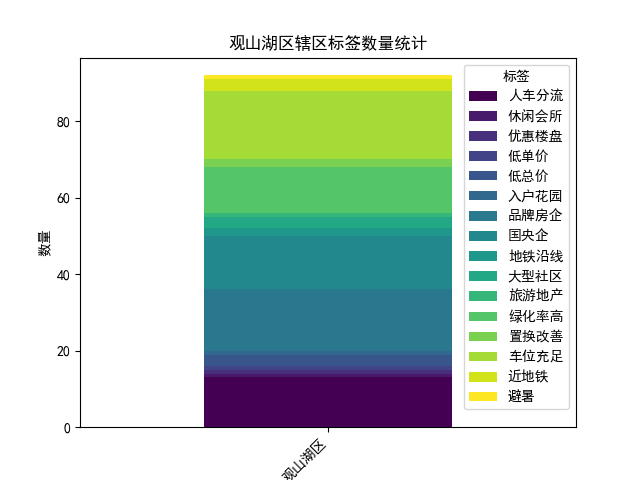
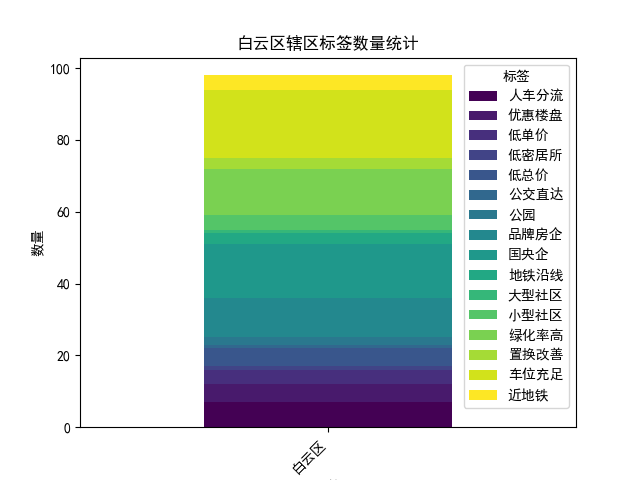
在这一观察中，云岩区的楼盘均价位居第一，这可能反映了云岩区的地理位置、基础设施以及社区配套等方面的优势，使得该区域成为高价值投资的热门选择。观山湖区紧随其后，显示出该区域的楼市也具备较高的投资价值。南明区和花溪区分列第三和第四，说明这两个区域在价格上相对平衡，适合不同层次的购房者。

白云区、乌当区、平坝区和清镇市分别位居第五至第八位，这或许反映了这些区域的楼市相对稳定，价格更加亲民，适合追求性价比的购房者。

这一指标不仅为购房者提供了关于各辖区房价水平的直观认知，也为他们在众多选择中更明智地进行投资决策提供了重要依据。对于房地产开发商和市场分析者而言，了解各辖区的均价分布有助于更精准地定位目标市场，提升项目的市场竞争力。因此，这一观察点不仅关系到个体购房者的利益，也涉及到整个房地产市场的健康发展。

### 3.2.4 辖区标签数量统计

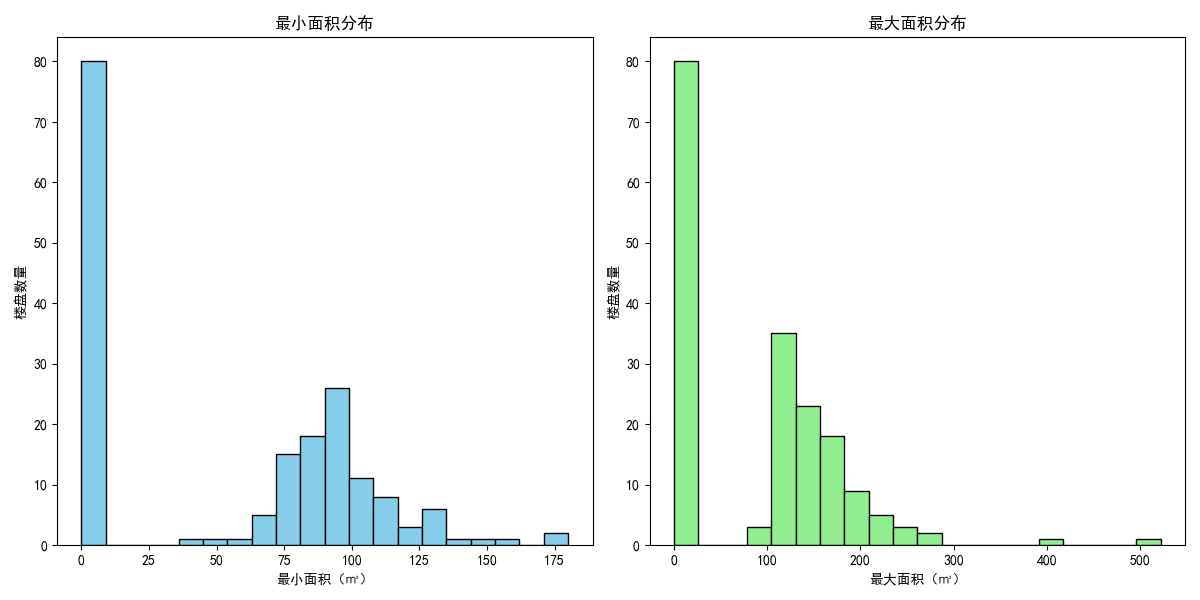
我们对各个辖区内楼盘的标签进行了数量统计，并通过堆叠柱状图展示了标签的分布情况。如图所示：



从图中可以观察到每个辖区内标签的分布情况。这对购房者来说是一个观察点，可以根据自己对楼盘特色的需求选择更符合个人口味的楼盘。

### 3.2.5 最小面积和最大面积的分布情况

我们对最小面积和最大面积的分析可以揭示不同楼盘的户型结构，了解市场对不同户型需求的分布情况，为购房者提供更多选择。如图所示：



首先，我们通过图表分析了不同楼盘的最小面积和最大面积分布情况，以揭示市场对不同户型需求的分布情况。这有助于购房者更全面地了解不同户型的供应情况。

图1：最小面积分布

从图1可以看出，绝大多数楼盘的最小面积集中在75~100㎡之间，说明市场主要以中小户型为主。同时，也有一部分楼盘提供了更小面积的房源，适应了对小户型需求的购房者。此外，还有一些楼盘提供了较大面积的房源，满足了对大户型的需求。

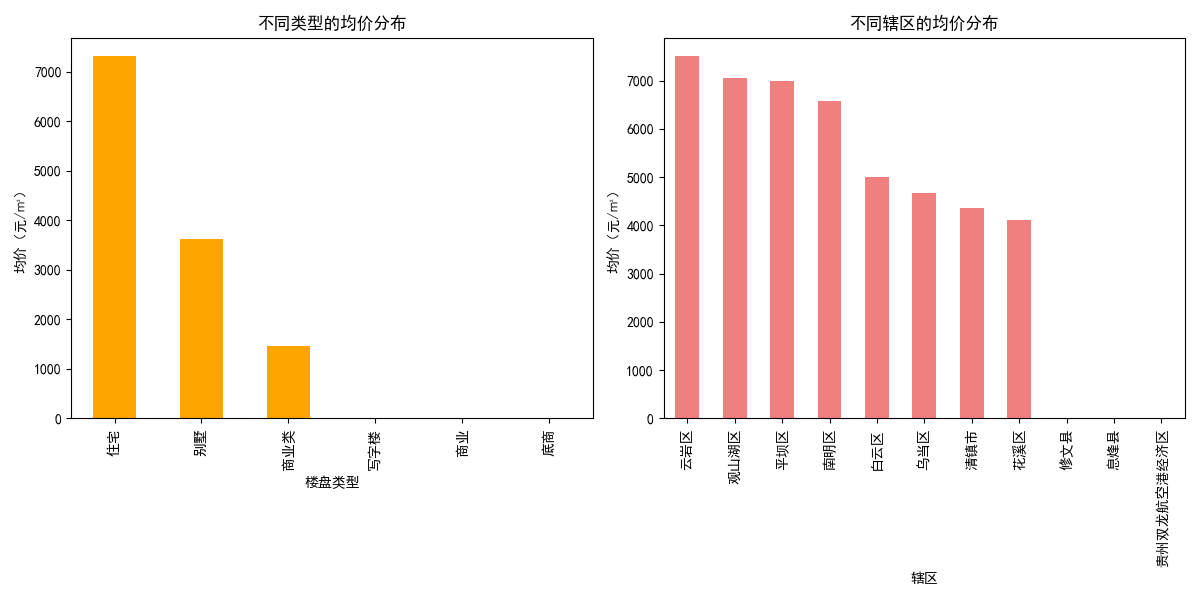
图2：最大面积分布

图2展示了不同楼盘的最大面积分布情况。大部分楼盘的最大面积也集中在100~200㎡之间，符合中小户型的市场趋势。同时，一些楼盘提供了更大面积的房源，适应了高端住宅需求。

这两张图帮助购房者更好地理解市场对不同户型需求的响应，为购房决策提供了重要的参考。

### 3.2.6 不同楼盘类型和不同辖区的均价分布情况

我们计算不同类型和辖区的均价和总价，分析价格的波动情况，了解市场的价格水平和趋势。如图所示：



我们通过图表分析了不同楼盘类型和不同辖区的均价分布情况，以帮助购房者更好地了解市场的价格水平和趋势。

图3：不同类型的均价分布

图3展示了不同楼盘类型的均价分布情况。可以看到，住宅的均价相对较高，别墅楼盘次之。

图4：不同辖区的均价分布

图4展示了不同辖区的均价分布情况。云岩区的均价最高，观山湖区次之，南明区、花溪区等辖区也有各自的价格水平。购房者可以根据个人预算和需求，更有针对性地选择合适的辖区。

这两张图为购房者提供了直观的价格分布信息，帮助他们更好地了解市场的价格趋势，从而做出明智的购房决策。

# 4讨论

## 4.1 数据更新频率

由于房地产市场的动态性，爬取的数据需要保持实时性以反映市场的最新状况。购房者在做决策时需要考虑到楼市状况的变化，因此定期执行爬虫脚本以更新数据至关重要。建议实施自动化爬虫定时任务，确保数据的及时更新，例如每日或每周执行爬取操作，以提供购房者准确而及时的信息。

## 4.2 标签信息补充

虽然标签信息提供了楼盘的一些特色，但在满足购房者需求的同时，更详细的信息能够提供更全面的楼盘画像。考虑补充其他关键信息，如周边交通、教育、医疗等配套设施情况，以便购房者能够更全面地了解楼盘所在区域的生活便利性，从而更好地做出购房决策。

## 4.3 防止爬虫被封

在爬虫执行过程中，通过设置随机睡眠时间模拟人类访问的方式是一种合理的策略。然而，可以进一步优化爬虫的稳定性和效率。考虑根据网站的访问频率限制动态调整爬取速度，以避免被网站封锁，并提高爬虫的效率。此外，可以通过使用代理IP池等方式来降低被封禁的风险。

## 4.4 异常处理细化

当前的异常处理逻辑主要输出错误信息并记录空白信息行，可以进一步细化异常处理策略。将不同类型的异常分别处理，例如网络连接异常、HTML解析异常等，以提高对不同异常情况的应对能力。详细的异常处理将有助于提高脚本的稳定性和可靠性。

## 4.5 数据可视化平台

为了更好地呈现爬取到的数据，考虑使用数据可视化平台。通过构建交互式的可视化图表，购房者可以更直观地理解市场趋势和楼盘特征。可考虑使用工具如Tableau、Power BI等，或将数据导入到Web应用程序中，提供用户友好的查询和筛选功能，从而提升用户体验。

# 5伪代码或关键代码

## 5.1爬虫代码

# -\*- coding: utf-8 -\*  
import time  
import random  
import requests  
from lxml import html  
from fake\_useragent import UserAgent  
import csv  
import re  
  
def get\_info(page):  
 # 随机头信息  
 headers = {  
 'User-Agent': UserAgent().random  
 }  
 url = 'https://gy.fang.lianjia.com/loupan/pg' + page  
 try:  
 requests.packages.urllib3.disable\_warnings()  
 res = requests.get(url, headers=headers, verify=False)  
 context = html.etree.HTML(res.text)  
 #print(context)  
 with open('data.csv', 'a', newline='', encoding='utf-8-sig') as csvfile:  
 fieldnames = ['辖区', '楼盘名称', '位置', '详细地址', '类型', '楼盘状态', '最小面积','最大面积', '均价', '最低总价','最高总价', '标签1', '标签2', '标签3', '标签4']  
 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  
 # 检查文件是否为空，为空就写入标题  
 if csvfile.tell() == 0:  
 writer.writeheader()  
  
 ul = context.xpath('//ul[@class="resblock-list-wrapper"]') # 获取标签,按页遍历  
 for i in ul:# 遍历每一个ul[@class="resblock-list-wrapper"下的每一个li标签  
 li = i.xpath('.//li[@class="resblock-list post\_ulog\_exposure\_scroll has-results"]')  
 for j in li:  
 try:  
 con= j.xpath('.//div[@class="resblock-desc-wrapper"]')  
 for a in con:#遍历con  
 area = a.xpath(".//div[@class='resblock-location']/span[1]/text()")  
 print("辖区：",area[0])  
 name =a.xpath(".//div[@class='resblock-name']/a/text()")  
 print("楼盘名称：",name[0])  
 area\_b =a.xpath(".//div[@class='resblock-location']/span[2]/text()")  
 print("位置：",area\_b[0])  
 area\_code = a.xpath(".//div[@class='resblock-location']/a/text()")#  
 print("详细地址：",area\_code[0])  
 type = a.xpath(".//div[@class='resblock-name']/span[1]/text()")  
 print("类型：",type[0])  
 stat = a.xpath(".//div[@class='resblock-name']/span[2]/text()")  
 print("房子状态：",stat[0])  
 square = a.xpath('.//div[@class="resblock-area"]//span//text()')  
 #正则提取数字  
 square = re.findall(r'\d+', square[0])  
 print("面积：", square)  
 avg\_price=a.xpath('.//div[@class="resblock-price"]//div[@class="main-price"]//span//text()')  
 print("均价：",avg\_price[0],"元/㎡")  
 price = a.xpath('.//div[@class="resblock-price"]//div[@class="second"]//text()')  
 # 正则提取数字  
 price=re.findall(r'\d+', price[0])  
 print("总价：", price,"万/套")  
 #遍历楼盘标签  
 tag=[]  
 for i in range(5):  
 tag.append(a.xpath(f'.//div[@class="resblock-tag"]/span[{i}]/text()')[0] if a.xpath(f'.//div[@class="resblock-tag"]/span[{i}]/text()') else '')  
 #写入csv文件  
 writer.writerow({  
 '辖区': area[0],  
 '楼盘名称': name[0],  
 '位置': area\_b[0],  
 '详细地址': area\_code[0],  
 '类型': type[0],  
 '楼盘状态': stat[0],  
 '最小面积': square[0],  
 '最大面积':square[1],  
 '均价': avg\_price[0],  
 '最低总价': price[0],  
 '最高总价':price[1],  
 '标签1': tag[1],  
 '标签2': tag[2],  
 '标签3': tag[3],  
 '标签4': tag[4],  
 })  
 time.sleep(random.randint(1, 5))  
 except Exception as e:  
 print('提取信息发生错误',e)  
 # 遍历楼盘标签  
 tag = []  
 for i in range(5):  
 tag.append(a.xpath(f'.//div[@class="resblock-tag"]/span[{i}]/text()')[0] if a.xpath(f'.//div[@class="resblock-tag"]/span[{i}]/text()') else '')  
 # 写入csv文件  
 writer.writerow({  
 '辖区': area[0],  
 '楼盘名称': name[0],  
 '位置': area\_b[0] if area\_b else area[0],  
 '详细地址': area\_code[0],  
 '类型': type[0],  
 '楼盘状态': stat[0],  
 '最小面积': 0,  
 '最大面积': 0,  
 '均价': 0,  
 '最低总价': 0,  
 '最高总价': 0,  
 '标签1': tag[1],  
 '标签2': tag[2],  
 '标签3': tag[3],  
 '标签4': tag[4],  
 })  
 time.sleep(random.randint(1, 5))  
  
 sleeptime = random.randint(30, 35)  
 print("睡眠%d秒后继续" %sleeptime)  
 time.sleep(sleeptime)  
 except requests.exceptions as e:  
 print("发送数据包发生错误",e)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 遍历的页数  
 for i in range(11, 19):  
 print("正在爬取第%d页" % i)  
 get\_info(str(i))  
 print("任务完成")  
 print("爬完")

## 5.2可视化代码

import pandas as pd  
import matplotlib.pyplot as plt  
  
plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文显示  
  
# 读取CSV文件  
df = pd.read\_csv('data.csv')  
  
  
# 统计每个辖区的数量  
area\_counts = df['辖区'].value\_counts()  
threshold = 0.03 # 设置一个占比的阈值  
small\_parts = area\_counts[area\_counts / area\_counts.sum() < threshold]  
area\_counts['其他'] = small\_parts.sum()  
area\_counts = area\_counts[area\_counts / area\_counts.sum() >= threshold]  
# 创建扇形图  
plt.figure(figsize=(10, 10))  
plt.pie(area\_counts, labels=area\_counts.index, autopct='%1.1f%%', startangle=90, labeldistance=1.1, textprops={'rotation': 'horizontal'})  
plt.title('各辖区占比')  
plt.show()  
  
# 统计每种类型的数量  
type\_counts = df['类型'].value\_counts()  
# 创建扇形图  
plt.figure(figsize=(12, 12))  
plt.pie(type\_counts, labels=type\_counts.index, autopct='%1.1f%%', startangle=90, labeldistance=1.2, textprops={'rotation': 'horizontal'})  
plt.title('各类型楼盘占比')  
plt.show()  
  
  
# 将均价数据转换为数值类型  
df['均价'] = pd.to\_numeric(df['均价'], errors='coerce')  
# 过滤掉均价为0的数据  
df = df[df['均价'] > 0]  
# 按辖区分组，计算均价的总和和平均值  
average\_prices = df.groupby('辖区')['均价'].agg(['sum', 'mean'])  
# 按平均值排序  
sorted\_avg\_prices = average\_prices.sort\_values(by='mean', ascending=False)  
# 可视化输出  
plt.figure(figsize=(12, 8))  
sorted\_avg\_prices['mean'].plot(kind='bar', color='skyblue')  
plt.title('各辖区均价排序')  
plt.xlabel('辖区')  
plt.ylabel('均价')  
plt.xticks(rotation=45, ha='right') # 使x轴标签斜着显示  
plt.show()  
  
# 读取CSV文件  
df = pd.read\_csv('data.csv')  
  
# 获取唯一的辖区列表  
unique\_areas = df['辖区'].unique()  
  
# 遍历每个辖区，绘制独立的可视化图  
for area in unique\_areas:  
 # 筛选出当前辖区的数据  
 area\_data = df[df['辖区'] == area]  
  
 # 将标签列转换为长格式  
 area\_tags = area\_data[['辖区', '标签1', '标签2', '标签3', '标签4']]  
 area\_tags\_melted = area\_tags.melt(id\_vars=['辖区'], value\_vars=['标签1', '标签2', '标签3', '标签4'],value\_name='标签').dropna()  
  
 # 统计当前辖区中标签的数量  
 tag\_counts = area\_tags\_melted.groupby(['辖区', '标签']).size().unstack().fillna(0)  
  
 # 可视化输出  
 plt.figure(figsize=(10, 8))  
 tag\_counts.plot(kind='bar', stacked=True, colormap='viridis')  
 plt.title(f'{area}辖区标签数量统计')  
 plt.xlabel('标签')  
 plt.ylabel('数量')  
 plt.xticks(rotation=45, ha='right') # 使x轴标签斜着显示  
 plt.legend(title='标签')  
 plt.show()  
  
  
  
import pandas as pd  
import matplotlib.pyplot as plt  
plt.rcParams['font.sans-serif'] = ['SimHei'] # 设置中文显示  
# 读取CSV文件  
df = pd.read\_csv('data.csv')  
  
# 面积分布分析  
plt.figure(figsize=(12, 6))  
  
# 最小面积分布  
plt.subplot(1, 2, 1)  
plt.hist(df['最小面积'], bins=20, color='skyblue', edgecolor='black')  
plt.title('最小面积分布')  
plt.xlabel('最小面积（㎡）')  
plt.ylabel('楼盘数量')  
  
# 最大面积分布  
plt.subplot(1, 2, 2)  
plt.hist(df['最大面积'], bins=20, color='lightgreen', edgecolor='black')  
plt.title('最大面积分布')  
plt.xlabel('最大面积（㎡）')  
plt.ylabel('楼盘数量')  
  
plt.tight\_layout()  
plt.show()  
  
# 均价和总价分析  
plt.figure(figsize=(12, 6))  
  
# 不同类型的均价分布  
plt.subplot(1, 2, 1)  
df\_avg\_price = df.groupby('类型')['均价'].mean().sort\_values(ascending=False)  
df\_avg\_price.plot(kind='bar', color='orange')  
plt.title('不同类型的均价分布')  
plt.xlabel('楼盘类型')  
plt.ylabel('均价（元/㎡）')  
  
# 不同辖区的总价分布  
plt.subplot(1, 2, 2)  
df\_total\_price = df.groupby('辖区')['均价'].mean().sort\_values(ascending=False)  
df\_total\_price.plot(kind='bar', color='lightcoral')  
plt.title('不同辖区的均价分布')  
plt.xlabel('辖区')  
plt.ylabel('均价（元/㎡）')  
  
plt.tight\_layout()  
plt.show()